Universidad de las Ciencias Informáticas.

Facultad 3

Título: Diseño e implementación de los procesos de Distribución de productos.

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

Autor: Isabel Torres Abad.

Tutores: Ing. Pedro Julio Rodríguez Paredes.

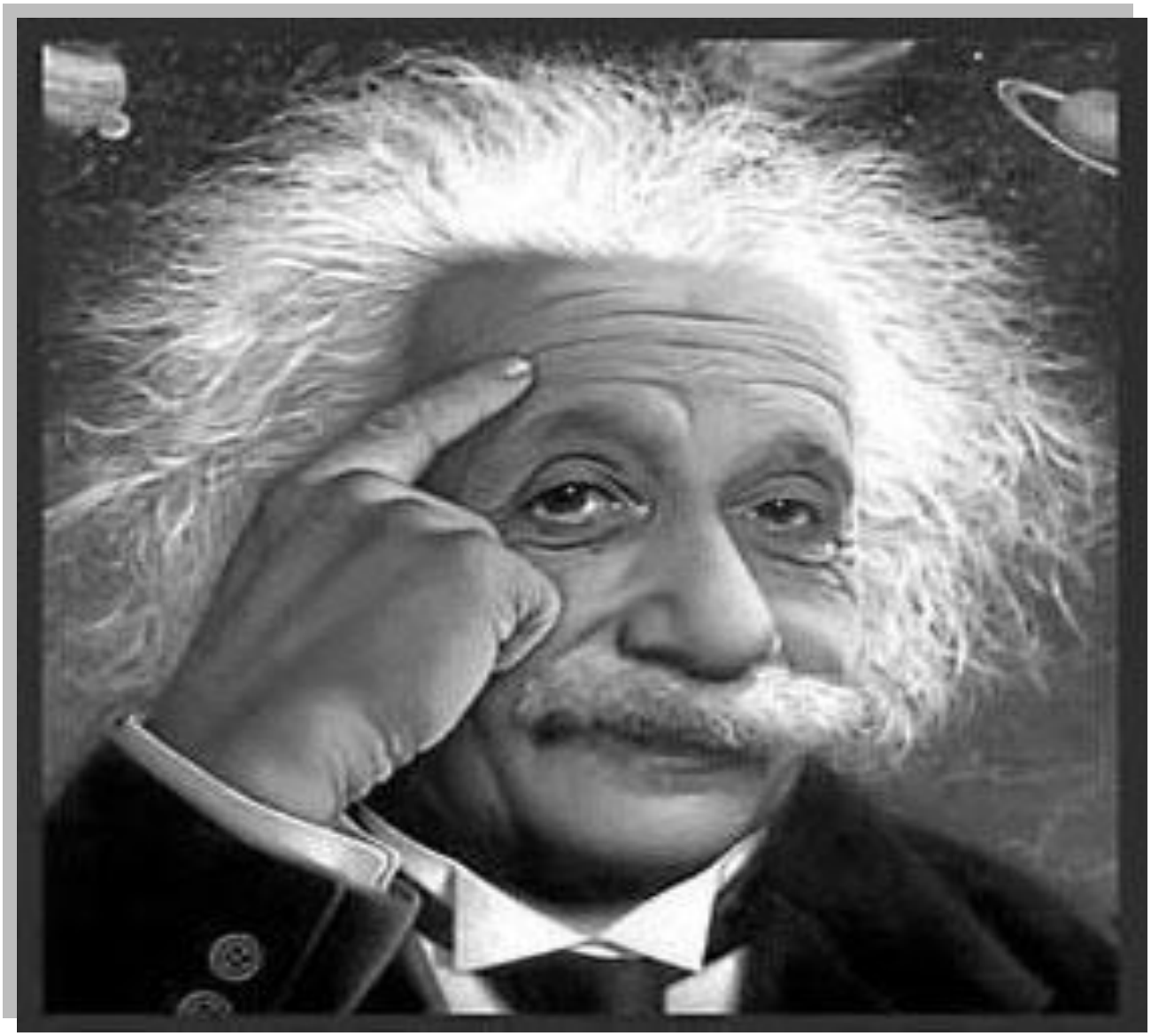
Ing. Osnier Ramírez Alea.

Ing. Alien Fernández Fuentes

.

La Habana, junio 2011

"Año 53 de la Revolución"



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

Albert Einstein

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ___ días del mes _____ del año 2011.

Isabel Torres Abad Ing.

_____

Autor

Pedro Julio Rodríguez Paredes

_____

Tutor

Ing. Osnier Ramírez Alea

_____

Tutor

Ing. Alien Fernández Fuentes

_____

Tutor

AGRADECIMIENTOS

A mis padres por el amor y la preocupación, por lo consejos y las palabras de aliento, por confiar en mí y apoyarme en todo.

A mi abuela Tata, aunque no tengo palabras para describir lo maravillosa que es y todo lo que ha hecho por mí.

A la familia tan maravillosa que tengo que nunca dejaron de confiar en mí, que han ayudado a lograr que mis ilusiones se conviertan en realidad, a tía Aileen y Leonardo, a mis primas que son un orgullo para mí, Elbita, Yadira, Ailencita y Yoanna. A Yaimi por la preocupación y el cariño. A mi prima Aracelis y a Bárbara por toda su ayuda, por cumplir la función de mi mamá aquí en La Habana. Y gracias a toda esa otra parte de mi familia que aunque no las he mencionado porque no me alcanzarían todas las páginas de la tesis siempre han estado pendientes de mí, brindándome su amor.

A mis amigos que son una parte importante de mi vida:

A Mayara, que desde aquella vez que me vio triste se hizo mi amiga para siempre. A Lisandra, que siempre me ha dado energías para seguir adelante. A Ruby, Grettel y Anay, mis amigas de alegrías y penas. A Yordanis y Hector Luis por ser buenos amigos, Yordy gracias por todo. A Juan Pablo, que sin su ayuda en las programaciones nunca hubiera llegado hasta aquí. A Leisy, una buena amiga y la mejor compañera de cuarto que existe. A Katia, mi compañera de tesis que me dejó sola a mitad de camino. A Yadelis, Mavis, Yandy, Gretter (La titi), Lino y Ailyn.

Y tantos otros amigos que de una forma u otra han estado presentes en estos 5 años compartiendo juntos experiencias inolvidables.

A Uds. nunca los olvidaré y los llevaré en mi corazón para siempre.

A mis tutores, en especial a Pedro.

DEDICATORIA

A mis padres, abuela, familiares y amigos.

Ing. Pedro Julio Rodríguez

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de experiencia en el tema: 4

Email: pjrodriguez@uci.cu

Ing. Osnier Ramírez Alea

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de experiencia en el tema: 2

Email: oramirez@uci.cu

Ing. Alien Fernández Fuentes

Especialidad de graduación: Ingeniero en Ciencias Informáticas

Años de experiencia en el tema: 2

Email: affuentes@uci.cu

RESUMEN

Actualmente la situación general de los procesos de gestión de las empresas está afectada por la existencia de sistemas informáticos que no cumplen con las expectativas de las nuevas tecnologías y la misión del país en el desarrollo de sus empresas. En la Universidad de las Ciencias Informáticas se está desarrollando el sistema integral de gestión Cedrux, como parte del fortalecimiento de la gestión de las entidades y de la informatización de la sociedad.

En el presente trabajo de diploma se expone la implementación de un sistema, que integrado al sistema Cedrux, será capaz de gestionar las operaciones relacionadas con la gestión de distribución de productos en una empresa.

Una vez finalizada la implementación de los procesos de distribución se realiza la validación, a partir de la utilización de métricas y pruebas para validar el diseño y la implementación.

Palabras Claves:

Gestión de distribución, Productos.

INTRODUCCION

Por lo general, hoy en día, todos los sectores administrativos de una empresa requieren llevar a cabo un trabajo de gestión a corto y largo plazo, que ofrezca resultados confiables, con el fin de mejorar la capacidad organizativa y rendimiento de la empresa. Para ello, en gran parte de las empresas se utilizan diferentes sistemas informáticos, desarrollados para optimizar la gestión empresarial en diversos campos. Por esta razón, surgen los Sistemas de Planificación de Recursos Empresariales (ERP: Enterprises Resources Planning), para alcanzar la optimización de procesos, como eslabón fundamental de la integración de la información en las diferentes áreas o departamentos de las entidades. (1)

Cuba ha decido no quedarse al margen de la revolución tecnológica y la informatización de procesos y se ha enfrascado en la tarea de construir un sistema de planificación de recursos empresariales cubano denominado Cedrux[1], que es fácilmente aplicable a cualquier empresa dedicada a la producción de bienes o servicios.

La incorporación de este sistema informático también se centra en beneficiar las empresas distribuidoras de productos, las cuales tratan de satisfacer las necesidades de sus clientes, utilizando métodos de distribución que implican una logística planificada y centralizada de acuerdo a la especialidad y aplicaciones de los productos.(2)

Estas empresas se ven afectadas por la planificación manual de la distribución y la utilización de software propietarios, lo cual no brinda una solución completa a las actividades planificadas y reguladas para garantizar que la calidad de los productos se mantenga durante las etapas de recepción, almacenamiento y transportación, lo que trae consigo que se realice un ineficiente proceso de distribución de productos. Por lo que surge la necesidad de realizar un sistema que controle, agilice y facilite la distribución de los productos. Para esto se realizó un estudio de los sistemas informáticos existentes dedicados a la distribución de productos , corroborando que la utilización de estos software no se adecuan a las empresas cubanas, las cuales no tienen semejanzas con las del mundo, pues sus fines no son de rentabilidad, sino que resuelven las necesidades del pueblo. Además el sistema que se desea realizar brindará dos alternativas para conformar la gestión de la distribución, la *distribución por pedidos,* que se realiza a partir de los productos solicitados por los clientes; y la *distribución sin pedidos* se realiza de una forma más rápida sin necesidad de que el cliente envíe alguna solicitud a la empresa, eliminando de este modo la dependencia al pedido. La correcta implementación y utilización de este sistema contribuirá a

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

minimizar los retrasos en la entrega de inventario a proveedores, fabricantes, mayoristas y minoristas y la falta de visibilidad del inventario en la red de distribución.

La gestión de distribución de productos juega un papel fundamental para el buen funcionamiento de las empresas, esa es la principal motivación que ha impulsado al equipo de desarrollo para enfrascarse en la implementación de los requisitos obtenidos por el equipo de análisis para realizar los procesos de Distribución.

Con el propósito de dar solución a la situación descrita con anterioridad se ha planteado el siguiente **Problema a resolver:**

¿Cómo optimizar la ejecución actual de los procesos de distribución de productos en las empresas, para minimizar las pérdidas económicas por errores en el procesamiento de la información?

A partir del problema planteado, el **objeto de estudio** se enfocará hacia los sistemas para la gestión de los procesos de distribución. Consecuentemente el **campo de acción** lo constituyen los sistemas para la gestión de los procesos de distribución de productos con pedidos y sin pedidos.

El **objetivo general** trazado para darle solución al problema identificado es: Realizar el diseño e implementación de los procesos de distribución de productos, para minimizar las pérdidas económicas por errores en el procesamiento de la información.

Los **Objetivos Específicos** definidos para esta investigación son**:**

- Construir el marco teórico-referencial de la investigación a partir del estudio de las particularidades de los sistemas de pedidos existentes.
- Diseñar e implementar la solución para la gestión de los procesos de distribución de productos.
- Validar la solución propuesta.

Para dar cumplimiento a los objetivos se han trazado las siguientes **tareas**:

- Análisis de los procesos de distribución de productos.
- Análisis de los sistemas existentes para la distribución de productos.
- Análisis de las metodologías, tecnologías, lenguajes y herramientas propuestas para el desarrollo de la aplicación.
- Análisis de los artefactos entregados por los analistas para los procesos de distribución de productos.
- Diseño de los artefactos para los procesos de distribución de productos. Implementación del diseño generado para los procesos de distribución de productos.
- Implementación del diseño generado para los procesos de distribución de productos.
- Realizar pruebas de unidad a los componentes obtenidos.
- Aplicación de métricas para validar los resultados del producto obtenido.

Se puede enmarcar como **Idea a Defender*:*** El desarrollo de una solución para el control de los procesos de distribución que se llevan a cabo en las empresas distribuidoras de productos, lo cual contribuirá a minimizar las pérdidas económicas por errores en el procesamiento de la información.
Como **Posibles resultados:** Una herramienta que gestione la información referente al proceso de distribución de productos de forma informatizada, a partir de los requerimientos analizados y priorizados por los usuarios funcionales del mismo.

Los métodos teóricos sustentan la investigación son:

- Histórico-Lógico
- Analítico-Sintético
- Modelación

Estructura del documento:

Capítulo 1: FUNDAMENTACIÓN TEÓRICA: En este capítulo se realiza un análisis de la gestión de la distribución dentro de los sistemas de planificación de recursos empresariales. Se mencionan y describen algunos sistemas de distribución que se utilizan en Cuba y el mundo. Por último, se justifican las herramientas y tecnologías a utilizar para el diseño e implementación del componente.

Capítulo 2: DISEÑO E IMPLEMENTACIÓN: Durante la realización de este capítulo se analizan los artefactos entregados por los analistas, y a partir del resultado obtenido se diseña la solución a implementar. Se modela el diagrama de componentes de la solución propuesta. Se identifican los patrones de diseño a utilizar, se modela el diagrama de clases del diseño permitiendo dar paso a la implementación del sistema, se representa además el diagrama de datos para mostrar las tablas vinculadas al sistema y sus relaciones. Se implementa la solución y se realizan las integraciones entre los componentes que están relacionados con el sistema.

Capítulo 3: VALIDACIÓN DE LA SOLUCIÓN: Aborda las pruebas realizadas al componente, para garantizar el buen funcionamiento del mismo.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

El mundo está en un proceso de constante cambio donde los avances tecnológicos dan pasos agigantados en todas las esferas de la sociedad con el objetivo de elevar el nivel económico de las empresas, aumentando así la producción y mejorando la gestión de sus procesos. El surgimiento del Software de Distribución de Productos está relacionado en gran medida con estos avances.

El presente capítulo estará centrado en la realización de un profundo análisis para obtener información actualizada de algunos sistemas de distribución de productos que son utilizados nacional e internacionalmente, analizando las ventajas y desventajas de su uso en las empresas del país. Además se ofrece una visión del modelo de desarrollo, las tecnologías y herramientas que se utilizarán durante el desarrollo del trabajo.

1.2 ¿Qué es distribución?

La distribución es el proceso logístico que comprende las actividades de:

- Servicio al cliente.
- Transporte.
- Gestión de Inventarios.
- Procesamiento de pedidos.

La distribución se encarga de satisfacer las funciones de servicio al cliente, expresado en términos de disponibilidad en cuanto a tiempo y lugar, con la calidad y en las cantidades requeridas, a partir de la satisfacción de los ciclos de pedido y entrega con los niveles de inventario necesarios.

El proceso de distribución debe estar en correspondencia con la cantidad y ubicación geográfica de los puntos a los cuales se debe suministrar, ya que la demanda generada por los clientes caracteriza los pedidos de los mismos en cuanto a tamaño, frecuencia, surtidos y ciclo de pedido y entrega. Además la dispersión geográfica del segmento de mercado a servir determina la selección de los medios de transporte en cuanto a tipo y cantidades, incidiendo en dos procesos de la gestión de distribución (preparación de pedidos y transporte). (2)

1.3 Sistemas Estudiados

1.3.1 Sistemas Utilizados a nivel internacional.

Cumulus Software

Cumulus Software utilizado para el control de almacenes, pedidos y distribuciones es una solución poderosa y económica para almacenadoras, centros de distribución y empresas que manejan su propio almacén.

Cumulus cuenta con 2 versiones que le ayudan a maximizar la eficacia y productividad de la entidad, estas son La Estándar y la Profesional con Radiofrecuencia (WiFi). Ambas versiones constan de funciones básicas como: control de inventarios, administración de inventarios, control de pedidos y control de distribución, recibo de productos y surtido de órdenes. La versión Profesional, además incluye funciones como mejora continua en el nivel de servicio de su operación, línea de surtido, administración de back order[2] y reabasto de áreas de surtido.

Cumulus es completamente escalable en funcionalidad y plataforma de Base de Datos. Gracias a esta flexibilidad, su empresa puede pasar de la Versión Estándar a la Profesional cuando las necesidades de su negocio así lo requieran.

Principales ventajas:

- ✓ Aumentan el nivel de servicio a sus clientes.
- ✓ Reducen gastos operativos.
- ✓ Son amigables con los usuarios.
- ✓ Mejoran el proceso de toma de decisiones.

Los requerimientos de hardware son Pentium III, 256 Mb de RAM, Pantalla de 15", Resolución mínima 1024x768, 16-bit color video, mouse, 25 Mb de espacio en disco duro para Cumulus y 1 Gb de espacio en disco duro para la base de datos y tarjeta de red, los sistemas operativos soportados son Windows 98 ó XP. (3)

EcoSoftCS.net

EcoSoftCS.net es un producto especialmente diseñado para facilitar todas las labores de introducción de datos, haciendo posible que con el menor número de pasos a realizar pueda cargar su pedido y partiendo de este, terminar el ciclo de compra y gestión de stock[3] .Por supuesto, contempla todas las posibilidades

[2] **Back order:** Pedido en espera o pendiente.

[3] **Stock:** Cantidad de producto en inventario.

de facturación de proveedores, tales como agrupación de albaranes[4] y abono de mercancía. Dispone de las mismas herramientas en facturación de ventas.

Este sistema intercambia mercancía entre sus diferentes sucursales y factura a empresas del grupo con distribución asociada, no tiene que facturar y posteriormente hacer la distribución, sino que puede realizar las dos operaciones automáticamente desde la misma opción.

Los requerimientos de hardware son mínimos, pudiéndose utilizar en casi cualquier equipo, los sistemas operativos soportados son Windows® 98, 98Se, ME, XP, 2000, 2003, Windows® Vista y Windows® 7. (4)

Axos Visual

Axos Visual es una solución empresarial capaz de gestionar las diversas áreas de negocio de la empresa de forma integrada, y mejorar la eficiencia de los recursos de la misma, incorporando la última tecnología.

Cada operador tiene acceso a la parte de la información que precisa. No existe duplicidad de datos ni tediosos procesos de intercambio de información o traspasos entre los diferentes módulos de la aplicación.

La plataforma cliente servidor en la cual está desarrollado Axos Visual permite una perfecta conectividad entre los diferentes centros de trabajo, independientemente de la ubicación geográfica de los mismos, y permite al usuario trabajar durante todo el día, desde cualquier parte del mundo, disponiendo siempre de toda la información en tiempo real.

Los requerimientos de hardware son mínimos, pudiéndose utilizar en casi cualquier equipo, los sistemas operativos soportados son según la necesidad del cliente. (5)

1.3.2 Sistemas Utilizados a nivel nacional.

Stock Mistral

Es una herramienta útil para facilitar la toma de decisiones. Está diseñado para disminuir los niveles de stock, reducir los recursos inmovilizados e incrementar la liquidez en la empresa. Posee herramientas que crean, en tiempo real, almacenes virtuales a diferentes niveles, con las existencias y movimientos de sus dependencias, permitiendo organizar el proceso de compras, distribución y pedidos. La consulta de esta información se realiza sobre sitios web ajustándose a los procedimientos internos de cada cliente.

Brinda prestaciones como el completo control de la consignación comercial, generación de ofertas, conciliación con proveedores, así como el tratamiento de los lotes y sus vencimientos (caducidad),

[4] **Albarán:** Nota de entrega que firma la persona que recibe una mercancía.

logrando su total aplicación para productos perecederos como alimentos y medicamentos. Corre sobre plataforma propietaria.

1.3.3 Resultado del análisis de los sistemas utilizados

Después de realizar un análisis de los sistemas de distribución utilizados actualmente en Cuba y en el mundo, no se pueden ignorar las visibles ventajas que estos ofrecen. Sin embargo analizándolos individualmente y desde una perspectiva inmediata y futurista, es importante considerar que sistemas como Cumulus Software, EcoSoftCS.net y Axos Visual, soportan sistemas operativos propietarios, cuando Cuba está en un proceso de migración hacia software libre. Otra desventaja de estos Sistemas de Gestión es que han sido desarrollados para empresas capitalistas que tienen un modelo de gestión y de procesos muy diferente a las empresas cubanas, donde la economía es centralizada. Por otra parte el Stock Mistral que es utilizado en el país, brinda una solución parcial a los problemas presentes en el proceso de distribución que se realiza en el país y está desarrollado sobre plataforma de software propietario, lo cual implica gastos que para la economía nacional significaría el pago de las licencias a los proveedores internacionales. Problema que se soluciona con la informatización de los procesos de distribución. Este sistema brindará entre otras opciones, la de configurar según las especificidades de la empresa el tipo de distribución que se quiere realizar, ajustándose así al negocio de la empresa, además de que soportará plataforma libre que es un factor importante a tener en cuenta por lo expuesto anteriormente.

1.4 Modelo de desarrollo

Modelo de desarrollo orientado a componentes

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto de software que reúna los requisitos del cliente. Concretamente define “quién está haciendo qué, cuándo hacerlo y cómo alcanzar un cierto objetivo”. (6)

El proyecto ERP-Cuba, para el desarrollo de Cedrux definió la utilización del Modelo de Desarrollo Orientado a Componentes. Este es un modelo de desarrollo orientado a las necesidades y artefactos generados durante el proceso de desarrollo del proyecto ERP-Cuba. Es una combinación de diferentes metodologías de las cuales se ha tomado lo que sería más conveniente para llevar a término el proyecto. El Modelo de desarrollo fue definido por la dirección del proyecto, con el objetivo de lograr un modelo estandarizado entre los equipos de desarrollo, que define de forma clara y precisa las responsabilidades de cada uno de los roles que se ven involucrados en el desarrollo de la solución entre ellos se encuentran

jefe de línea, planificador, arquitecto principal, analista principal, especialista de calidad, desarrolladores, analistas y funcionales. (7)

1.5 Tecnologías de Desarrollo

1.5.1 Librerías y Marcos de trabajo

Un marco de trabajo es una estructura de software compuesta por componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework (o marco de trabajo) se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, puede incluir soporte de programas, bibliotecas y un lenguaje de scripting entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

El marco de trabajo que se empleará para la realización de este trabajo es Sauxe, el cual posee lenguajes, frameworks y tecnologías que facilitarán la implementación del sistema que se desea obtener.

Sauxe

El Marco de Trabajo Sauxe se ha estructurado de manera tal que facilite la reutilización de los diferentes componentes y que sea de fácil entendimiento para todos los programadores que desarrollen sobre el mismo. Sauxe utiliza el framework Ext, para implementar la capa de presentación, se apoya en Zend_Ext, una extensión de Zend Framework para el desarrollo de la lógica del negocio y para la gestión de los datos utiliza Doctrine. Este utiliza como patrón arquitectónico Modelo Vista Controlador (MVC). También tiene como propósito insertar la programación orientada a aspectos así como la inversión de controles.

Además Sauxe implementa mecanismos de autenticación y autorización, mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de precondiciones, poscondiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de las funcionalidades de un sistema, entre otras funcionalidades**.** (8)

ExtJS Framework

Es un framework JavaScript del lado del cliente para el desarrollo de aplicaciones Web. ExtJS posee una librería inmensa que permite configurar las interfaces Web de manera semejante a aplicaciones desktop, incluye la mayoría de los controles de los formularios Web basándose en Grids para mostrar datos y elementos semejantes a la programación desktop como los formularios, paneles, barras de herramientas, menús y muchos otros. Dentro de su librería contiene componentes para el manejo de datos, lectura de XML, lectura de datos JSON e implementaciones basadas en AJAX. Usar un motor de render como ExtJS

permite entre otros beneficios un balance entre Cliente – Servidor, la carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo; una comunicación asíncrona. En este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta, además de que facilita eficiencia de la red, el tráfico de red puede disminuir al permitir que la aplicación elija que información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico. A continuación se muestra la descripción de las clases pertenecientes a este framework. (9)

- ***ext-base:*** Encargada del manejo de las solicitudes y respuestas, trabajo con ajax y manejo de componentes de Ext. Está incluida en el paquete original.
- ***ext-all:*** Es la encargada de la creación de los componentes visuales de la vista. Está incluida dentro de las clases que trae ExtJS.
- ***Vista***: Representa la vista que se muestra al usuario.
- ***js_vista:*** Fichero js con las funciones JavaScript asociadas a la vista. Aquí se establece la referencia a las clases de Ext.

Doctrine Framework

Doctrine es un potente y completo sistema ORM (object relational mapper) para PHP 5.2+ con un DBAL (database abstraction layer) incorporado, el mismo cuenta con disimiles funcionalidades, una de ellas es la que brinda la posibilidad de exportar una base de datos existente a sus clases correspondientes y también a la inversa, es decir convertir clases (convenientemente creadas siguiendo las pautas del ORM) a tablas de una base de datos. Su principal ventaja radica en poder acceder a la base de datos utilizando la programación orientada a objetos (POO), debido a que doctrine utiliza el patrón Active Record para manejar la base de datos, tiene su propio lenguaje de consultas y trabaja de manera rápida y eficiente.(10)

Zend Framework

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 Open Source y trabaja con PHP 5. Zend Framework utiliza el estilo MVC como base de su funcionamiento. Es fácilmente integrable a las aplicaciones debido a su composición y a que contiene diferentes clases de gran utilidad, como por ejemplo en la búsqueda dinámica de ficheros a incluir o utilizar. Cuenta con un importante mecanismo de manejo de controladores y vistas. (11)

Dentro de sus principales características están:

- Proporciona los componentes que forma la infraestructura del patrón MVC.
- Proporciona una capa de acceso a base de datos, construida sobre PDO[5] pero ampliándola con diferentes características.
- Proporciona mecanismos de filtrado y validación de entradas de datos.
- Permite convertir estructuras de datos PHP a JSON y viceversa, para su utilización en aplicaciones AJAX.
- Proporciona capacidades de búsqueda sobre documentos y contenidos.
- Permite consumir y proveer servicios web.
- Incluye objetos de las diferentes bases de datos, por lo que es extremadamente simple para consultar la base de datos.
- Completa documentación y pruebas de alta calidad.
- Robustas clases para autenticación y filtrado de entrada.
- Proporciona un sistema de caché dividido en frontend y backend, de forma que se puedan almacenar en caché diferentes datos como resultados de funciones, páginas completas, entre otros. (11)

Zend_Ext Framework

Es un framework open Source, que está diseñado para php 5 y buenas capacidades de ampliación. Es elaborado a partir de Zend Framework cumpliendo con todas sus características. Este trae de novedoso un controlador vertical para el control de las acciones realizadas por las vistas hacia el controlador, un motor de reglas para las validaciones en el servidor. Se le incluyó el IoC para la comunicación entre los módulos o componentes, la integración con el ORM Doctrine Framework para trabajo en la capa de abstracción a base de datos, el ExtJS Framework para el desarrollo de las vistas.

1.5.2 Lenguajes de programación

Lenguajes de programación del lado del cliente

En la actualidad existen numerosos lenguajes de programación para desarrollar la web, tanto lenguajes de lado cliente como de lado servidor. Cuando se habla de lenguajes del lado cliente, son aquellos capaces de ser digeridos directamente por el navegador sin necesidad de un pre-tratamiento, entre los lenguajes

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

utilizados se encuentran los siguientes:

Javascript

Es un lenguaje interpretado creado por Brendan Eich en la empresa Netscape Communications. El código Javascript puede ser integrado dentro de las páginas web. El Consorcio World Wide Web (W3C) para evitar incompatibilidades diseñó un estándar para la modelación de objetos del documento, más conocido como DOM (en inglés Document Object Model). Entre sus ventajas está su scripting seguro y fiable y contradictoriamente su debilidad es su visibilidad al alcance de cualquier usuario. (12)

AJAX

Acrónimo en inglés de Asynchronous JavaScript And XML («JavaScript y XML asíncronos»). Es una técnica de desarrollo Web para crear aplicaciones interactivas mediante la combinación de tres tecnologías ya existentes: HTML (o XHTML) y Hojas de Estilo en Cascada (CSS) para presentar la información. Document Object Model (DOM) y JavaScript, para interactuar dinámicamente con los datos. XML y Extensible Stylesheet Language Transformations (XSLT), para intercambiar y manipular datos de manera no sincronizada con un servidor Web (aunque las aplicaciones AJAX pueden usar otro tipo de tecnologías, incluyendo texto llano, para realizar esta labor). Todos los navegadores Web usados por las aplicaciones AJAX soportan las tres tecnologías mencionadas anteriormente. Entre estos se incluyen: Mozilla Firefox, Internet Explorer, Safari y Opera. (13)

Ventajas:

- Utiliza tecnologías ya existentes.
- Soportada por la mayoría de los navegadores modernos.
- Interactividad: El usuario no tiene que esperar hasta que lleguen los datos del servidor.
- Portabilidad (no requiere plug-in).
- Mayor velocidad, debido a que no hay que retornar toda la página nuevamente.
- La página se asemeja a una aplicación de escritorio.

Lenguaje de programación del lado del servidor

Los lenguajes de lado del servidor son aquellos lenguajes reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él.

PHP

PHP (Hypertext Preprocessor) es un lenguaje interpretado, diseñado especialmente para desarrollo web y puede ser incluido dentro de código HTML. Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida. Puede ser desplegado en la

mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin costo alguno.
A continuación se muestran algunas de las principales ventajas que ofrece PHP.(14)

- **Rendimiento*:*** utilizando un servidor modesto pueden ser atendidas millones de peticiones al día. Para mejorar este rendimiento Zend Technologies, quien patrocina PHP, ha desarrollado versiones especiales con este fin.
- **De código abierto (Open Source):** se tiene acceso al código fuente. Permite agregar o modificar algo para obtener un mejor funcionamiento.
- **Librerías incluidas:** PHP fue diseñada para trabajar sobre la web, por ello trae un conjunto muy amplio de funciones para ser utilizadas en diferentes tareas relacionadas con la web. Se puede conectar con bases de datos, conectar a web services, parsear XML, enviar e-mail, generar PDF, generar imágenes, etc. Basadas en estas librerías existen clases implementadas para facilitar el trabajo de los desarrolladores. Otro punto es que hay desarrolladores que agregan librerías especializadas para extender las funcionalidades de PHP.
- **Portabilidad**: PHP está disponible para la mayoría de sistemas operativos existentes. Desde Unix, Linux, Microsoft Windows, MAC, entre otros. Una vez desarrollado la aplicación PHP esta puede funcionar con cualquiera de estos sistemas operativos sin necesidad de modificar el código.
- **Soporte para Programación Orientada a Objetos (OOP):** La versión 5 de PHP está diseñada para soporte de características de programación orientada a objetos. Características como herencia, métodos y atributos públicos o privados, clases y métodos abstractos, constructores, interfaces y destructores. (14)

1.5.3 Arquitectura

La Arquitectura del Software es el diseño de más alto nivel de la estructura de un sistema, también denominada Arquitectura lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información. Establece los fundamentos para que analistas, diseñadores y programadores, trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades. (15)
La arquitectura que se utiliza es la arquitectura basada en componentes, con una estructura en capas, que en las capas superiores implementa el patrón Modelo Vista Controlador(MVC)

Arquitectura multicapas

1. Capa de presentación: en esta capa se emplea las facilidades que brinda el Marco de Trabajo ExtJS

para la construcción de interfaces amigables a la vista de los usuarios. Ext centra su desarrollo en tres componentes fundamentales JS-File, CSS-File y Client-page y Server-Page. (16)

2. Capa de control o negocio: en esta capa se emplea el patrón de arquitectura Modelo Vista Controlador (MVC). De forma vertical al modelo descrito hasta este momento, estarán los aspectos fundamentales del sistema así como el encargado del tratamiento de la seguridad a nivel de aplicación Web. (17)

3. Capa de acceso a dato: en esta capa estará presente el ORM (Object Relational Mapping) Doctrine, como Marco de Trabajo de persistencia para la comunicación con el servidor de datos mediante el protocolo PDO, también estará un persistidor de configuración que es el encargado de comunicarse vía XML con los ficheros de configuración del sistema denominado FastResponse. (16)

4. Capa de datos: en esta capa estará ubicado como servidor de base de datos PostgreSQL y un conjunto de ficheros de configuración de la Arquitectura Tecnológica. (16)

5. Capa de servicio: en esta última capa se encuentran todos los subsistemas que prestan y consumen servicios entre sí. (17)

Patrón arquitectónico Modelo - Vista - Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de manera paralela e independiente. (17)

- **Modelo:** Es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en un contexto del sistema proveen de información al usuario o a la aplicación misma.
- **Vista:** Es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web la "Vista" es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.
- **Controlador:** Es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

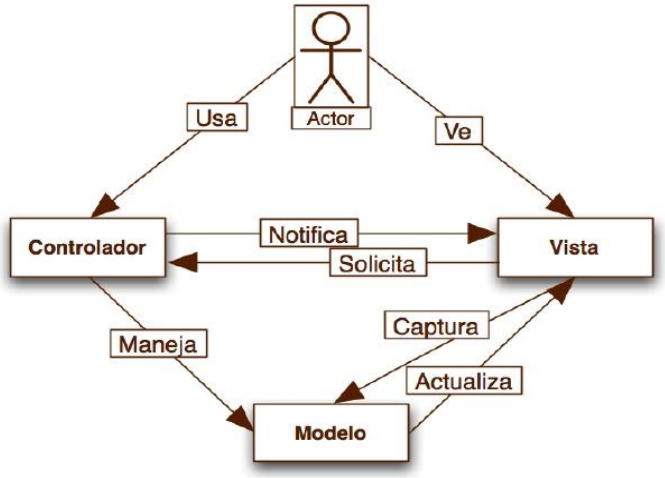


Figura # 1 Patrón Modelo-Vista-Controlador.

1.6 Herramientas

1.6.1 Herramientas CASE

Las herramientas CASE (Computer Aided Software Engineering), Ingeniería de Software Asistida por Ordenador, son el conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores para aumentar la calidad del software reduciendo el esfuerzo, el costo y el tiempo. Además de estructurar la documentación asociada a los artefactos generados, facilitan el desarrollo de software desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas.

Visual Paradigm

Es una herramienta UML[6] profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una rápida construcción de aplicaciones de gran calidad, mejoras y a un menor costo de producción. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Forma parte del IDE de Eclipse. Está diseñado para desarrollar software con Programación Orientada a Objetos, permite reducir la duración del ciclo de desarrollo brindando ayuda a

[6] **UML:** (Unified Modeling Language) Lenguaje Unificado de Modelado

los arquitectos, analistas, diseñadores y desarrolladores. Busca también automatizar tareas tediosas que pueden distraer a los desarrolladores. Dentro de sus características fundamentales están: (18)

- **Multiplataforma:** Soportado en plataformas Java para Sistemas Operativos Windows, Linux y Mac.
- **Interoperabilidad:** Intercambia diagramas UML y modelos con otras herramientas. Soporta exportar e importar a XML y archivos Excel. Importa archivos de proyectos de Rational Rose. Integración con Microsoft Office Visio.
- **Modelamiento de los requisitos:** Captura de requisitos con sus respectivos diagramas, modelamiento de casos de uso y análisis textual.
- **Colaboración de equipo:** Realiza el modelado en colaboración y simultáneamente con el Visual Paradigm TeamWork Server y Subversion.
- **Generación de documentos:** Comparte y genera los diagramas y diseños en formatos como PDF, HTML y Microsoft Word.
- **Editor de detalles de casos de uso:** Entorno que funciona como un todo en uno para la especificación de los detalles de los casos de uso, incluyendo la especificación del modelo general y de las descripciones de los casos de uso.
- **Ingeniería de código:** Permite generación de código e ingeniería inversa en lenguajes como Java, C++, CORBA, IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, ODL, Flash ActionScript, Delphi, Perl y Rugby.
- **Modelado de procesos del negocio:** Visualiza, comprende y mejora los procesos del negocio con herramientas muy completas para estos procesos.
- **Integración con entornos de desarrollo:** Apoyo al ciclo de vida completo de desarrollo del software: análisis, diseño e implementación en IDE como Eclipse, Microsoft Visual Studio, NetBeans, Sun ONE, Oracle JDeveloper, JBuilder y otros.
- **Modelado de bases de datos:** Generación de bases de datos, conversión de diagramas entidad-relación a tablas de base de datos, mapeos de objetos y relaciones, ingeniería inversa desde gestores de bases de datos.(18)

Teniendo en cuenta las características mencionadas se decidió utilizar el Visual Paradigm por ser además un producto de calidad que soporta aplicaciones web en varios idiomas, siendo fácil de instalar y actualizar además de tener compatibilidad entre todas sus ediciones. Otra de sus ventajas es la disponibilidad en múltiples sistemas operativos como Windows, Linux, Unix. Indudablemente esta herramienta posee un grupo de ventajas difíciles de ignorar a la hora de hacer la selección.

1.6.2 Servidor Web

Un servidor Web es un programa que está diseñado para transferir hipertextos, páginas Web o páginas HTML (Hypertext Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música. Además sirve contenido estático a un navegador, carga un archivo y lo sirve a través de la red al navegador de un usuario. Este intercambio es mediado por el navegador y el servidor que hablan el uno con el otro mediante HTTP.

Apache

Entre sus principales características se encuentran:

- Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- Es una tecnología gratuita de código fuente abierto.
- Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor.
- Tiene una alta configurabilidad en la creación y gestión de logs[7]. (19)

1.6.3 Base de Datos

Sistema Gestor de Base Datos

Los sistemas de gestión de bases de datos o SGBD (en inglés Database Management System, abreviado DBMS) son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. Un SGBD permite definir los datos a distintos niveles de abstracción y manipular dichos datos, garantizando la seguridad e integridad de los mismos. (20)

PostgreSQL v8.3

Es un sistema de gestión de bases de datos relacional orientada a objetos, libre y multiplataforma, publicado bajo la licencia BSD (*Berkeley Software Distribution*). Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

Por ser un proyecto de código abierto tiene una gran comunidad de desarrollo en Internet, su código fuente está disponible sin costo alguno. PostgreSQL tiene la extraordinaria potencialidad de permitir que

[7] **Logs:** Registro de errores.

mientras un proceso escribe en una tabla, otros accedan a la misma sin necesidad de bloqueos esto es posible gracias a un sistema denominado Acceso Concurrente Multiversión[8]. (21)

1.6.4 Herramientas de desarrollo colaborativo

SVN (SubVersion)

SVN (SubVersion) es un sistema de control de versiones, que mantiene los registros de todos los cambios que se han realizado a los archivos de un software, lo que permite el trabajo de distintos desarrolladores en un mismo proyecto, esta herramienta es muy usada por los programadores de software libre. Existen dos razones fundamentales para su uso: (22)

- Gestiona las modificaciones durante el desarrollo.
- Permite que varias personas trabajen sobre los mismos ficheros.

1.6.5 IDE de desarrollo

El ambiente de desarrollo (Development Environment) es algo imprescindible en la producción de software. Es donde se definen el conjunto de herramientas y tecnologías (frameworks), versiones a usar y su integración, que intervienen en un proceso de desarrollo de software. Un entorno de desarrollo integrado o Integrated Development Environment (IDE), en inglés, es un programa compuesto por un conjunto de herramientas para un programador. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, poder utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por si solas o pueden ser parte de aplicaciones existentes. Estos proveen un marco de trabajo amigable para la mayoría de los lenguajes de programación tales como C++, Java, C#, Delphi, Visual Basic y Object Pascal. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación. (23)

Zend Studio para Eclipse

El Zend Studio para Eclipse constituye la nueva generación del Zend Studio. Proporciona a las organizaciones que hagan uso del mismo ya que posee un entorno mucho más flexible y profesional para controlar todo el ciclo de vida de un desarrollo. Entre sus funcionalidades, destacan las capacidades de refactorización del código fuente, funcionalidad que permite adecuar el comportamiento externo de una

[8] Acceso Concurrente Multiversión: permite que mientras un proceso escribe en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos.

función clase sin cambiar el funcionamiento interno, que junto a las capacidades de generación de código le facilitaría el trabajo a los desarrolladores. Entre las nuevas características que incluye este IDE están: (23)

- El acceso al ecosistema de plugins de Eclipse.
- Apoyar el desarrollo de múltiples idiomas.
- Mejora el Editor de PHP con el formato avanzado, para listas de tareas y problemas de vista.
- Mejora del soporte de JavaScript.
- Mejora de apoyo, incluyendo HTML, códigos plegables, arrastrar y soltar los componentes.
- Mejora de depuración.
- Mejora de Zend Framework con el apoyo del Proyecto Framework, plantillas y código MVC.

1.6.6 Navegador

Un navegador Web o explorador Web es una aplicación que permite al usuario recuperar y visualizar documentos de hipertexto, comúnmente descritos en HTML, desde servidores Web de todo el mundo a través de Internet. Cualquier navegador actual permite mostrar o ejecutar gráficos, secuencias de vídeo, sonido, animaciones y programas diversos además del texto y los hipervínculos o enlaces. En este caso el explorador usado es Mozilla Firefox

Mozilla Firefox

Mozilla Firefox es un navegador web libre, multiplataforma y está disponible en varias versiones de Microsoft Windows, GNU/Linux y algunos sistemas basados en Unix. Incluye navegación por pestañas, corrector ortográfico, búsqueda progresiva, marcadores dinámicos, un administrador de descargas y un sistema de búsqueda integrado. Seguidamente se mencionan algunas características de Firefox frente a Internet Explorer. (24)

- **Rapidez:** La velocidad de Firefox e Internet Explorer es, en términos generales, casi la misma. Aunque si se manejan muchas páginas web a la vez, se notará una mayor estabilidad con Firefox que con IE.
- **Compatibilidades:** Se ofrecen características DHTML (HTML dinámico) para crear interfaces de usuario eficaces para aplicaciones Web. Se proporciona también integración con la plataforma .NET, simplifica la integración del código de servidor y cliente y permite que las aplicaciones llamen a funciones del servidor de manera asincrónica.
- **Seguridad y control de privacidad:** Las mejoras no son excesivamente significativas en cuanto a la apariencia externa, pero el cambio interno es importante. Una muestra de ello son

las mejoras en seguridad respecto a las versiones anteriores, que tapan gran cantidad de agujeros de seguridad.

1.7 Conclusiones parciales del capítulo

En este capítulo se realizó un estudio de los sistemas existentes en Cuba y el mundo dedicados a la distribución de productos, analizando sus principales funcionalidades y características, concluyendo que brindan una solución parcial a los problemas presentes en la distribución y están desarrolladas sobre plataforma propietaria, lo cual no es factible debido a la política actual del país hacia la soberanía tecnológica. El sistema se desarrolla sobre el marco de trabajo Sauxe, utilizando las tecnologías y herramientas definidas por el mismo.

CAPÍTULO 2: DISEÑO E IMPLEMENTACIÓN.

2.1 Introducción

Para lograr una eficiente gestión de la distribución en las diferentes empresas, resulta de vital importancia contar con una solución informática que asegure el control de estos procesos. En este capítulo se parte de un estudio de los artefactos generados por el análisis para obtener el diseño de la solución. Se llevará a cabo una descripción de la propuesta del sistema, así como de las principales funcionalidades. Se abordarán los patrones de diseño utilizados, se modelarán los diagramas de clases del diseño, modelo de datos y el diagrama de componentes. Comprenderá además la definición de los estándares de codificación, la implementación, y la representación de la estrategia de integración entre los componentes que están relacionados con el sistema.

2.2 Análisis de los artefactos de entrada

Para el diseño de la solución técnica, se debe partir de los artefactos generados por el equipo de analistas de la línea de Logística durante el análisis de los procesos de distribución para las empresas distribuidoras de productos. Para la realización de este trabajo se desarrolló una exhaustiva revisión de los requisitos confirmando que sus definiciones son las que el usuario final espera y necesita. Durante el transcurso de esta actividad se comprobó que los requerimientos identificados están claros, se entienden correctamente y son posibles de probar. Además el estudio de los requisitos demostró que estos cubrían todas las funcionalidades necesarias para la solución, permitiendo así que la interacción del usuario con el sistema facilite su trabajo y lo haga eficiente. Luego de revisados todos los artefactos generados durante el proceso de análisis, y concluyendo que la solución propuesta es factible, se pasa a la realización de las actividades para modelar diseño.

2.3 Objeto de informatización

El proceso de distribución se realiza de dos formas diferentes, distribución por pedidos y distribución sin pedidos.

Por pedidos, el proceso comienza a partir del momento en que el suministrador dispone de los pedidos de los clientes, conoce el ciclo a distribuir según el plan de consumo, verifica la existencia de productos en el almacén y procede a distribuir; si no existen los productos solicitados define si le da seguimiento al pedido.

Sin pedidos, no requiere que el cliente envíe los pedidos al suministrador, pues este dispone de datos

actualizados de los clientes (existencias, plan, consumos, stock máximo y stock mínimos), y así es capaz de reconocer lo que realmente necesita el cliente y en qué momento.

Para realizar la distribución el sistema lleva a cabo una serie de cálculos, después de realizar los cálculos se emite la propuesta de distribución de productos por circuitos o grupo de clientes, que es utilizada para realizar un despacho físico en el almacén. (25)

Finalmente el sistema proporciona la información necesaria para que el módulo de control de inventarios de Cedrux rebaje las existencias del submayor y el módulo de facturación de Cedrux pueda emitir las facturas y guías de expedición.

Las fórmulas que se utilizan para realizar la propuesta de distribución son las siguientes:

Cantidad máxima a distribuir del circuito (CMDC) = Cobertura del producto * (∑ Plan de consumo clientes ***/*** días del período)

Cantidad de bultos = Parte entera (Cantidad a Distribuir al Cliente / Capacidad de bulto)

Resto = Cantidad a distribuir

Porciento de distribución = (Cantidad distribuida / Cantidad a cumplimentar) * 100

Porciento Consumo Cliente = (Consumo de cliente * 100) / (∑ Consumo de todos los clientes de todos los circuitos)

Cantidad Distribuida al Cliente = ((Porciento Consumo Cliente * (Existencia en Inventario del producto en el proveedor + ∑ Existencia del producto en los clientes + ∑ Cantidad total del producto en tránsito + ∑ Cantidad del producto facturado + ∑Cantidad del producto distribuido)) / 100) - (Existencia del producto en el cliente + Cantidad del producto en tránsito hacia el cliente + Cantidad del producto facturado pero no expedido al cliente + Cantidad del producto distribuido pero no facturado al cliente).

Días de diferencia = Cobertura total del cliente - Política de Cobertura.

Cantidad de diferencia = Días de diferencia * (Consumo del período/ días del período)

Coeficiente de distribución = (Cantidad a cumplimentar / ∑ Cantidad a cumplimentar de todos los clientes del circuito)

La cantidad distribuida y la cobertura cliente son fórmulas que tienen diferentes variantes como se puede apreciar a continuación.

Para recalcular la cantidad distribuida.

Cantidad distribuida = Cantidad distribuida anteriormente – cantidad de diferencia

Para recalcular la cantidad distribuida con seguimiento del plan.

Cantidad distribuida = (Plan del periodo del cliente - Cantidad en proceso - Cantidad entregada)

Para calcular la cobertura del cliente.

CoberturaCliente = ((Existencia del producto en el cliente + Cantidad del producto en tránsito + Cantidad del producto facturado + Cantidad del producto distribuido) / Consumo / días del período)

Para calcular la cantidad distribuida al cliente.

CoberturaCliente = ((Cantidad distribuida al cliente) / (Consumo/días del período))

Estas fórmulas, surgen a raíz de un estudio realizado según las necesidades que presentaban las empresas de distribución de productos. Se definieron en conjunto con los usuarios funcionales Quimefa y el equipo de desarrollo de la línea logística. Para esto se tomó como antecedente fórmulas que tenía el sistema stock mistral que es el utilizado en el país y otras fórmulas que facilitó el MINFAR. Estas fórmulas están siendo validadas, para ver su correcto funcionamiento.

2.4 Propuesta del sistema

El sistema propuesto deberá informatizar los procesos para la distribución de productos que se realizan actualmente en las diferentes empresas o entidades del país. El sistema brinda dos alternativas principales para realizar la preparación de la distribución, distribución sin pedidos y distribución por pedidos. Para la realización de la distribución sin pedidos, el sistema debe ser capaz de efectuar la propuesta de distribución sin un pedido previo, como su nombre lo indica. En este caso la existencia de los productos en el almacén, así como los planes de consumo y varios datos de los clientes, serán utilizados para llevar a cabo una serie de cálculos que jugarán un papel fundamental para el éxito o fracaso de la distribución final. El sistema ofrecerá también, la opción de proponer una distribución, pero esta vez basada en un pedido realizado por el cliente. Este dará a la aplicación los datos suficientes para satisfacer si es posible, las necesidades de los proveedores en el período de tiempo para el que se realice el pedido.

Inicialmente el usuario deberá autenticarse en la aplicación, el sistema verificará sus privilegios permitiéndole acceder solo a las áreas que le están asignadas según el rol que desempeñe. El sistema brindará la opción de configurar el tipo de distribución que se desea realizar, distribución por pedidos o distribución sin pedidos. El sistema realiza los cálculos necesarios para emitir la propuesta de distribución de productos por circuitos o grupo de clientes.

El sistema debe realizar, modificar, y eliminar distribuciones, además de listar todas las distribuciones realizadas hasta el momento, mostrar los productos de las distribuciones, imprimir y buscar distribuciones anteriormente realizadas.

2.5 Principales funcionalidades

Los requisitos son la condición o capacidad que tiene que ser alcanzada o poseída por un sistema o componente de software para satisfacer un contrato, estándar, u otro documento impuesto formalmente. A continuación se exponen los requisitos funcionales y no funcionales que debe cumplir la aplicación a desarrollar. (26)

Requisitos funcionales:

Los requisitos funcionales se realizan a partir del análisis de las principales necesidades de los clientes y las características que el sistema debe cumplir, así se definieron un conjunto de requisitos funcionales que se muestran a continuación.

RF1: Gestionar Distribución.

- RF1.1: Adicionar distribución por pedidos.
- RF1.2: Adicionar distribución sin pedidos.
- RF1.3: Modificar distribución sin pedidos.
- RF1.4: Modificar distribución por pedidos.
- RF1.5: Eliminar distribución.
- RF1.6: Listar distribuciones.
- RF1.7: Ajustar distribución al bulto.
- RF1.8 : Buscar distribución.
- RF1.9: Imprimir distribución.

RF2: Calcular propuesta de distribución.

- RF2.1: Calcular propuesta de distribución por pedido.
- RF2.2: Calcular propuesta de distribución sin pedido.

RF3: Realizar distribución por pedido

RF4: Gestionar productos de la distribución sin pedidos

- RF4.1: Adicionar productos a la distribución sin pedidos
- RF4.2: Eliminar productos de la distribución sin pedidos.
- RF4.3: Listar productos de la distribución sin pedidos.

RF5: Realizar distribución sin pedido.

RF6: Confirmar distribución.

RF7: Cancelar estado de distribución

Requisitos no funcionales

Los requisitos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto.

Portabilidad:

- La aplicación debe funcionar en varias plataformas, principalmente (Linux y Windows), siendo posible su acceso a través de un navegador Web.

Seguridad:

- Autenticación (Contraseña de acceso).
- Mostrar opción de advertencia antes de borrar cualquier elemento o información que pueda existir.

Rendimiento:

- Los tiempos de respuesta y velocidad de procesamiento de la información tendrán una espera de 20 a 30 segundos para representar el mapa y 10 a 20 segundos para la representación de la información.

Software:

- Para el cliente:
  - ✓ Navegador Mozilla Firefox.
  - ✓ Sistema operativo Linux o Windows.
- Para el servidor:
  - ✓ Sistema operativo Windows o Linux.
  - ✓ Servidor Apache 2.0 o superior con módulo PHP 5.0 disponible, este debe estar configurado con las extensiones PDO y PDO_pgsql.
  - ✓ Servidor de base de datos PostgreSQL v4.1.0.1 o superior.

Hardware:

- Para el cliente:
  - ✓ Procesador Pentium IV a 3.00Ghz.
  - ✓ 512 MB de memoria RAM.
  - ✓ Tarjeta de red.
  - ✓ Impresora
- Para el servidor:
  - ✓ Procesador Pentium IV a 1GHz.
  - ✓ 1Gb de memoria RAM.
  - ✓ Tarjeta de red.

2.6 Modelo de Diseño

Una entrada esencial en el diseño es el resultado del análisis, o sea el modelo de análisis, que proporciona una comprensión detallada de los requisitos. El modelo de diseño tiene el propósito de formular los modelos que se centran en los requisitos no funcionales y en el dominio de la solución, y que prepara para la implementación y prueba del sistema. (26)

2.6.1 Modelo de dominio

El Modelo de Dominio (o Modelo Conceptual) es una representación visual de los conceptos u objetos del mundo real significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema, se muestran los conceptos del dominio, sus atributos y relaciones. (26)

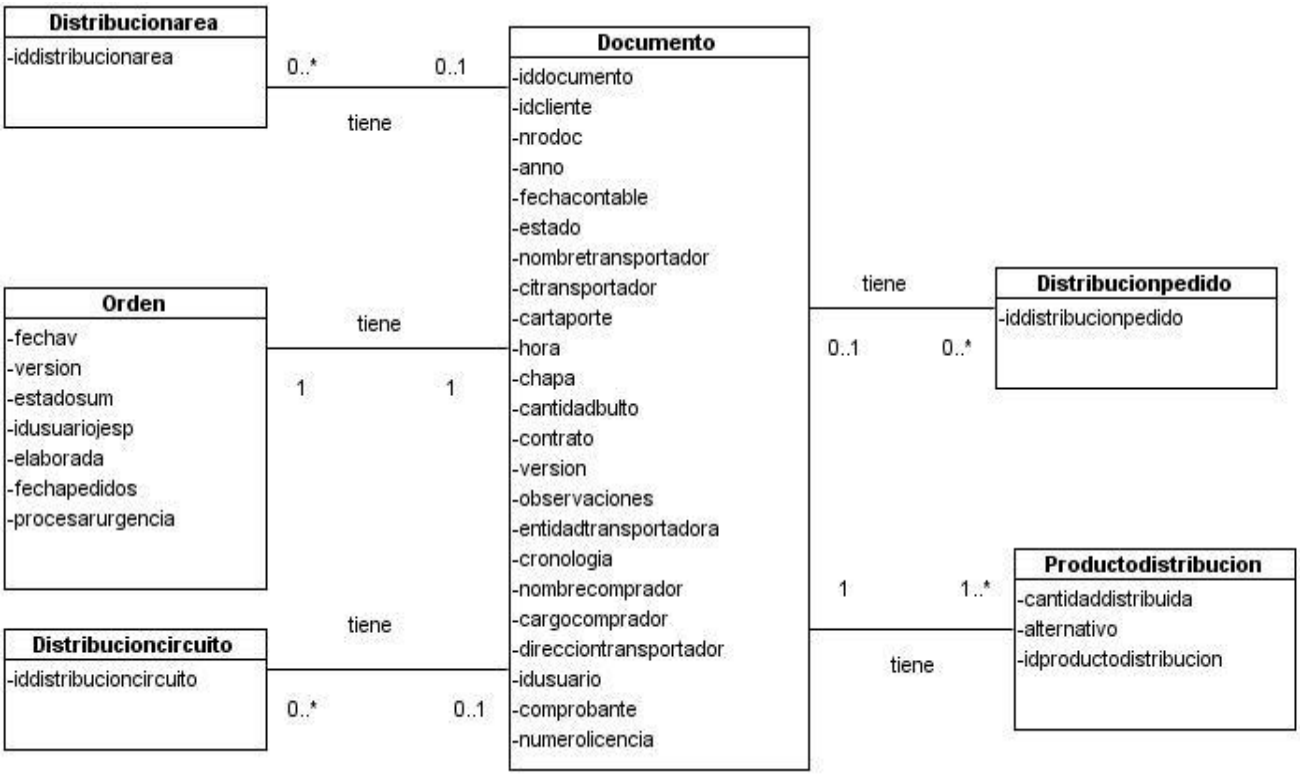


Figura # 2 Modelo Conceptual

2.6.2 Diagramas de clases del diseño

Un diagrama de clase del diseño es un diagrama estático que describe la estructura de un sistema mostrando sus clases, atributos, operaciones y las relaciones existentes entre ellos. (26)

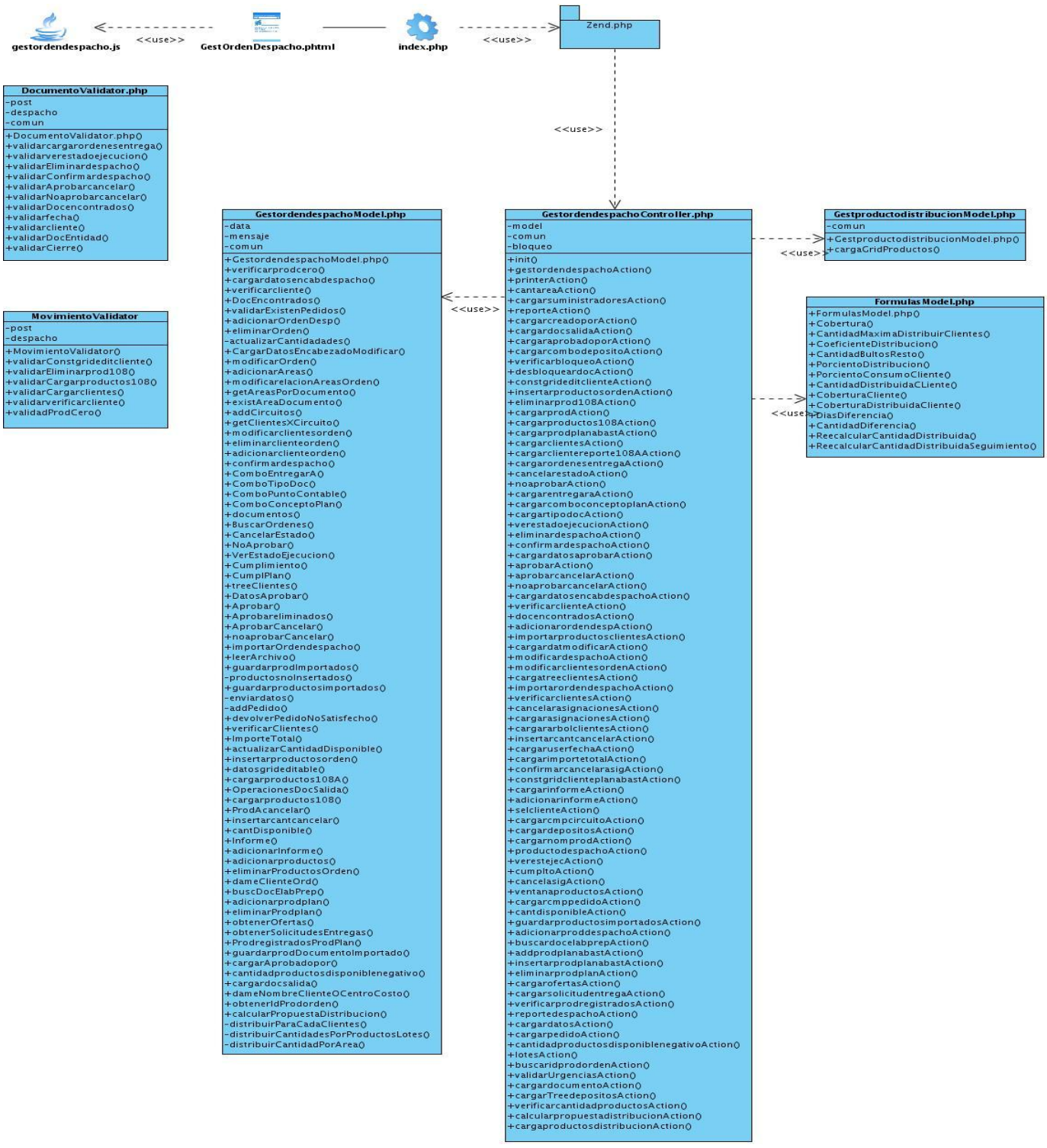


Figura # 3 Diagrama de clases de diseño

2.6.3 Modelo de datos

Un modelo de datos aporta la base conceptual para el diseño de aplicaciones que hacen un uso intensivo de datos, este modelo es por tanto una colección de conceptos definidos que permiten expresar las propiedades estáticas y dinámicas de una aplicación. (26)

El modelo de datos propuesto es una representación de las tablas existentes en la base de datos así como las relaciones entre ellas. Este modelo cuenta con 31 tablas.

- nom_conceptoplan: Nomenclador que guarda los conceptos de un plan.
- nom_formafarmaceutica: Es la disposición individualizada a que se adaptan las sustancias medicinales (principios activos) y excipientes(materia farmacológica inactiva) para constituir un medicamento
- dat_pedido: Tabla que almacena los datos de los pedidos.
- nom_tipoproducto: Agrupación de productos por las características y el uso que tendrá para el cliente.
- nom_grupofarmacologico: Son las características que tienen los medicamentos por el efecto o la acción que producen contra las enfermedades.
- nom_grupoproducto: Nomenclador que contiene los grupos a los cuales pertenecen los productos.
- nom_lineaproducto: En esta tabla se realiza la clasificación del producto de acuerdo a su naturaleza y uso.
- nom_categoriaproducto: Nomenclador que contiene todas las categorías de los productos. (forma de clasificar los productos por determinado concepto.)
- nom_tipoarea: Clasificación del área de acuerdo a las características del producto.
- nom_circuitodistribucion: Nomenclador que contiene los datos del circuito de distribución.
- cfg_distribucion: Tabla que guarda la previa configuración de la distribución.
- dat_orden: Tabla que guarda los datos de las órdenes que son documentos de despacho.
- dat_documento: Tabla que almacena los datos de todos los documentos.
- dat_distribucioncircuito: Tabla que guarda los datos de la distribución por circuito.
- dat_area: Tabla que almacena las áreas.
- dat_circuitodistclienteproveedor: Tabla que guarda los datos del circuito de distribución del cliente y el proveedor.

- dat_distribucionarea: En esta tabla se crea de una relación entre dat_distribucion y dat_area.
- dat_distribucionpedido: Tabla que guarda los datos de distribución por pedido.
- nom_productoquimefa: En esta tabla se crea una relación con nom_producto a través del idprod.
- dat_producto: Contiene todos datos de los productos que entran a la entidad.
- dat_areaproducto: Tabla donde se asocian a cada área uno o varios productos.
- nom_producto: Nomenclador que contiene todos los productos con características generales de los mismos.
- dat_valoresdistribucion: Contiene los valores que se van a tener en cuenta para realizar una distribución.
- dat_productodistribucion: Tabla para guardar los datos alternativos de la distribución.
- dat_lote: Almacena los datos correspondientes a un lote el cual contiene el id del movimiento ya que los lotes son de productos y estos a su vez constan en documentos.
- dat_prodorden: Tabla donde se relacionan el producto y la orden asociada a ese producto. Tabla originada de una relación entre tablas de muchos a muchos.
- dat_prodordenlote: Tabla donde se relacionan el lote y la orden asociada a ese producto. Tabla originada de una relación entre tablas de muchos a muchos.
- dat_arealote: Tabla donde se asocian a cada área uno o varios lotes.
- dat_prodordenarea: Tabla donde se relacionan el área y la orden asociada a ese producto. Tabla originada de una relación entre tablas de muchos a muchos.
- dat_prodordenlotearea: Tabla donde se relacionan el lote con el área y la orden asociada a ese producto. Tabla originada de una relación entre tablas de muchos a muchos.

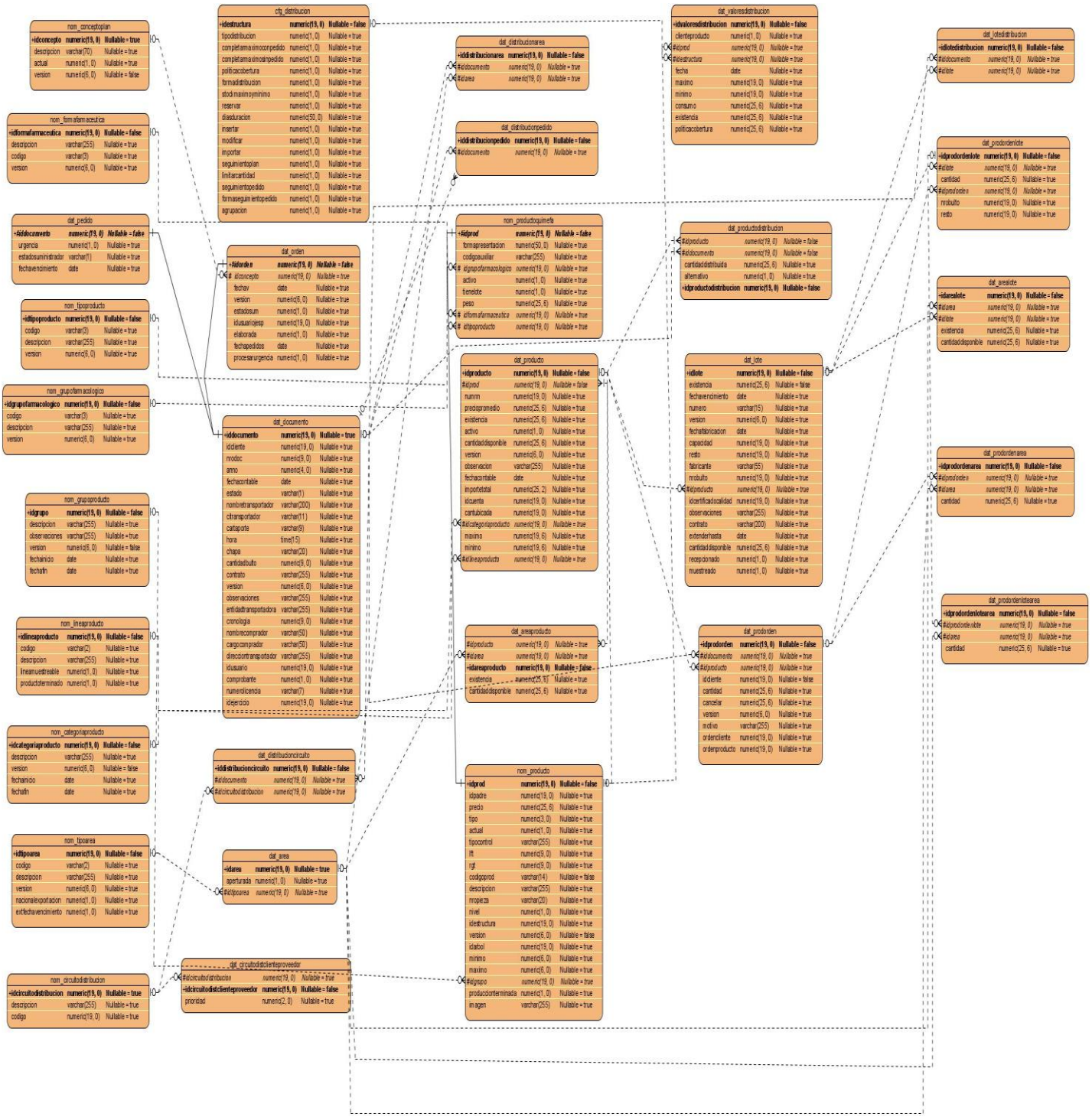


Figura # 4 Modelo de Datos

2.6.4 Diagrama de componentes

Un diagrama de componentes representa gráficamente como un sistema de software es dividido en componentes, mostrando las dependencias existentes entre estos.

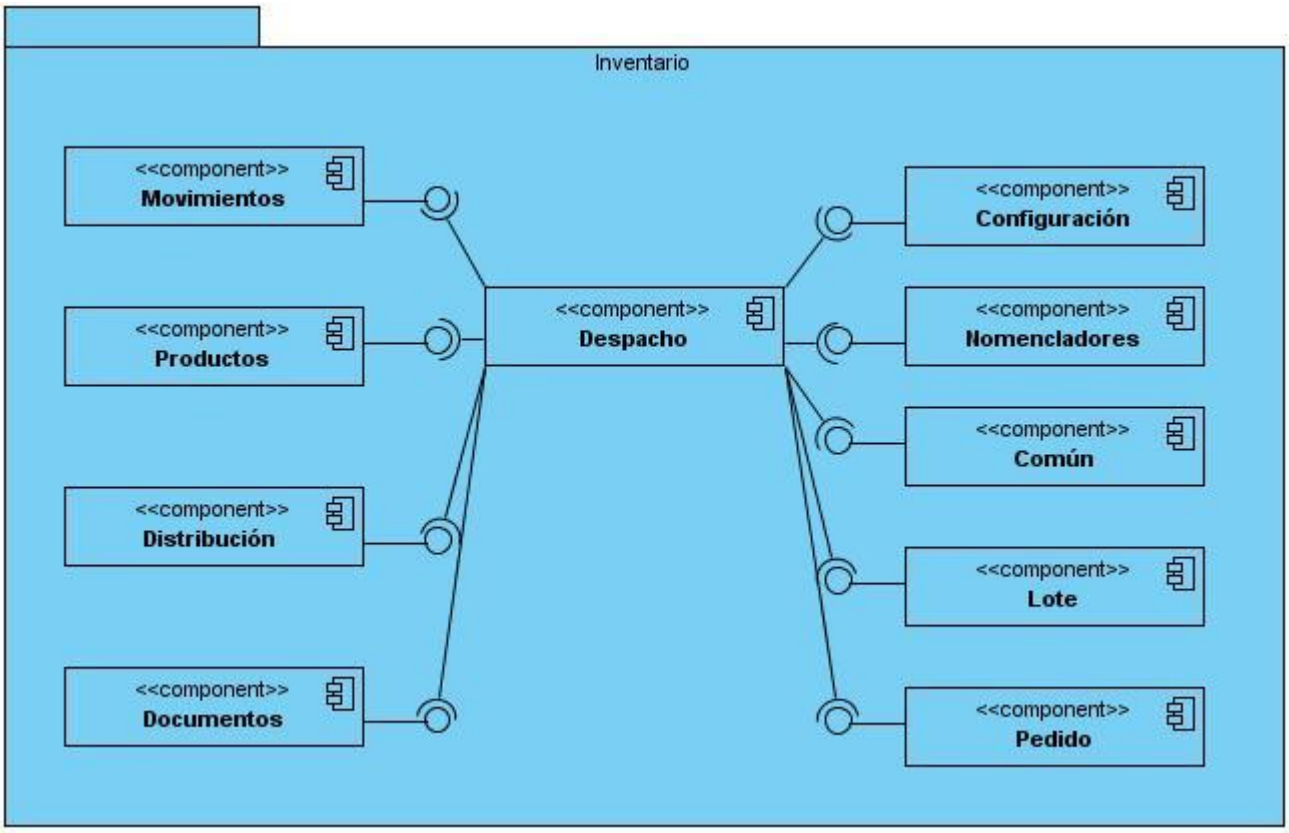


Figura # 5 Diagrama de componentes

El componente despacho dentro del cual se va a efectuar el proceso de distribución se relaciona con componentes como documentos, productos, configuración, nomencladores, común, lote, pedido, movimientos y distribución como se muestra en el anterior diagrama de componentes. A continuación se explica la función de cada uno de estos componentes, haciéndolos necesarios para gestionar la distribución.

- El componente documento tiene la función de gestionar todos los documentos de la distribución.
- El componente producto es el encargado de la gestión de los productos dentro del almacén, ya sea los que están nomenclados, como los nuevos que se incluyen.

- El componente configuración es el encargado de gestionar como su nombre lo indica, la configuración para la distribución con que va a contar cada entidad.
- Nomencladores es un componente que guarda los datos que no deben ser modificados, un ejemplo es el nomenclador de circuito de distribución.
- El componente común brinda un servicio para actualizar la cantidad disponible en las diferentes tablas que se afecten en la base de datos.
- En el componente lote se gestionan los distintos lotes que van a ser utilizados para la distribución.
- El componente pedido gestiona los pedidos enviados por los usuarios para la realización de la distribución con pedido.
- El componente de distribución gestiona existencia, planes de consumo y política de cobertura de los clientes, entre otros valores necesarios para realizar la distribución.
- El componente movimiento tiene una función muy importante, pues es el encargado de la unión entre los documentos y los productos. Dicho componente facilita todas las operaciones que son necesarias para operar con los movimientos de ese documento. Un movimiento es una copia que se le hace al producto para no afectar los valores reales del mismo y una vez contabilizado el documento serán afectados los valores del producto con los del movimiento.

2.6.5 Patrones de diseño

Un patrón de diseño brinda la posibilidad de organizar las clases en estructuras comunes y bien probadas, para facilitar la creación de un software. Los patrones de diseño ayudan a tener un lenguaje común con otros programadores.

Patrones Grasp (Patrones Generales de Asignación de Responsabilidades)

Los Patrones Grasp describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Bajo Acoplamiento

Este patrón es un principio que asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Consiste en tener las clases lo menos ligadas entre sí, permitiendo que al producirse una modificación en alguna de ellas, tenga la mínima repercusión posible en el resto de las clases, potenciando la reutilización, y disminuyendo la dependencia entre las clases. (27)

Alta Cohesión

Este patrón plantea que la información que almacena una clase debe ser coherente y estar estrechamente relacionada con la clase. Realizando un diseño donde las clases mantengan una alta cohesión se mejora la claridad y facilidad con que se entiende el diseño, se simplifica el mantenimiento y las mejoras de funcionalidad, a menudo se genera un bajo acoplamiento, soportando mayor capacidad de reutilización. (27)

Patrón Experto

Este patrón se usa con el objetivo de darle a las clases las responsabilidades necesarias siempre que cuenten con la información para cumplirlas. Logrando así un mejor comportamiento entre las clases y haciendo que éstas fueran más cohesivas, fáciles de comprender y mantener, permitiendo mayores facilidades de soporte. (27)

Este patrón es utilizado en la clase FormulasModel, la cual es especialista en el trabajo con las fórmulas que se utilizan en la implementación del sistema.

Patrón Creador

Este patrón permite tener una política general para la creación de objetos. Se usa en las clases controladoras para crear instancias de las clases del modelo, principalmente las del negocio, así como en las clases del negocio para instanciar las clases del dominio. (27)

Este patrón se manifiesta en gran parte de la solución, al ser creadas instancias de las clases necesarias para ejecutar las funcionalidades.

Patrón Controlador

Es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a clases que representen, un sistema global o la fachada, algo en el mundo real que pueda ejecutar el papel (personas), o un manejador artificial que pueda ejecutar los eventos. (27)

Este patrón se evidencia cuando las clases controladoras convierten un evento generado por el usuario en una llamada a un método del modelo específico, convirtiéndose así en controladoras del flujo de eventos generados en el sistema.

Patrón Singleton

El patrón Singleton es un patrón de diseño, específicamente de creación, es utilizado para garantizar que una clase sólo tenga una instancia y proporcione un punto de acceso global a ella. Su funcionamiento puede resumirse a que oculta el constructor de la clase Singleton, para que no se puedan crear otras instancias. (27)

Un ejemplo del uso de este patrón se puede ver en la clase GestordendespachoModel.

2.7 Modelo de Implementación

El modelo de implementación adquiere gran relevancia, pues describe como los elementos del modelo de diseño se implementan en términos de componentes. Normalmente no se debe crear una aplicación desde cero, reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no necesariamente es la que demanda mayor trabajo ni la más complicada. La complejidad y la duración de esta etapa están relacionadas con el lenguaje de programación utilizado, así como con el diseño previamente realizado. (28)

2.7.1 Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código.

Para la realizar la implementación de los procesos de distribución se tuvo en cuenta los estándares de codificación definidos en el marco de trabajo. (29)

Nomenclatura de las clases según el tipo

Las clases Controladoras se encuentran dentro de la carpeta “controller”. A estas clases se le añade la palabra “Controller”, después de escribir el nombre según su nomenclatura.

- Ejemplo: GestordendespachoController

Las clases Modelos se encuentran dentro de la carpeta “bussines”. A estas clases se le añade la palabra “Model”, después de escribir el nombre según su nomenclatura.

- Ejemplo: GestproductodistribucionModel

Nomenclatura de los métodos o funciones

El nombre de los métodos de una clase comienzan con minúsculas, en caso de que sea compuesto seguido del primer nombre comienza el segundo con la primera letra en mayúscula. Deben describir el propósito del mismo.

- Ejemplos: calcular (), calcularPropuestaDistribucion ().

Si es una función de una clase controladora después del nombre se le agrega la palabra “Action”.

- Ejemplo: calcularPropuestaDistribucionAction ().

Nomenclatura de las variables

El nombre que se le pone a los atributos se escribe con la primera palabra en minúscula, en caso de

que sea un nombre compuesto seguido de la primera palabra se escribe la segunda con inicial mayúscula. Además en caso de ser un objeto se comienza con: “obj” y después se escribe el nombre con la primera letra en mayúscula, en caso de ser un nombre compuesto es seguido del primer nombre y comienza con mayúscula.

- Ejemplo: intAtributo, objDatosDistribucion.

Tabla # 1 Prefijos a utilizar en la creación de variables

Tipos de Datos	Prefijos
Arreglos	arr
Objetos	obj
Enteros	int
Cadena	str
Float	flt
Boolean	Boo

Normas de comentario

Es necesario establecer pautas para lograr que el código sea legible y reutilizable, brindando la posibilidad de facilitar su mantenimiento, es por esto que resulta necesario comentar todo lo que se haga dentro del desarrollo. Los comentarios deben ser claros y precisos de forma que se entienda el propósito de lo que se ha desarrollado. Se pueden encontrar comentarios de código donde se explica lo más preciso posible la funcionalidad del mismo, como se muestra en el siguiente ejemplo, figura 6

```
/**
* @name cargarDistribucionProducto
* Funcion para devolver los productos de la propuesta de distribucion
* @param
* @return arreglo
*/
public function cargarDistribucionProducto($filter, $limit, $start){
    $documentos = GestordendespachoModel::documentos();
    $filter->espedido =  ($documentos['distribucionCP']->iddoc == $filter->iddoc);
    $idestructura = $this->comun->BuscarPadrePorSubordinacion();
    $conf = $this->pIntegrator->configuracionInventario->devolverConfiguracion($idestructura);
    $filter->diasperiodo = $conf->diasduracion;
    return $this->pIntegrator->movimientos->cargarPropuestaDistribuciondeProductos($filter, $limit, $start);

}
```

Figura # 6 Comentario del código

Comentarios por líneas

Se utilizan también los comentarios para líneas de código en específico que ayudan a aclarar la complejidad de algunos algoritmos. Para ellos se utiliza // y seguidamente el comentario que describe la línea de código, tal y como se representa a continuación.

```
if($objDatos->tienelote){//flujo 9 , requisito Calcular Propuesta de distribucion
    $productos = DatLote::devolverLotesOrdenadoFechaVencimiento($objDatos);
     $productos[0]['areas'] = $productos[0]['DatArealote'];
     $productos[0]['idestructuraop'] = $productos[0]['DatLote']['DatProducto'][0]['idestructuraop'];
     unset($productos[0]['DatArealote']);
     unset($productos[0]['DatProducto']);
}else{//flujo alterno 9.a
     $productos = $this->pIntegrator->productos->obtenerProducto()->devolverProductoOrdenadoMasViejo($objDatos);
     $productos[0]['areas'] = $productos[0]['DatAreaproducto'];
     unset($productos[0]['DatAreaproducto']);
}
```

Figura # 7 Comentario del código dentro de una función

Estilo del Código

En la implementación cuando se vaya a escribir una sentencia en php la forma de utilizar las etiquetas del mismo es la siguiente:

<? php

//código

¿>

La política de sangría a utilizar en la implementación es por tabulaciones. Las clases se comienzan a declarar pegado al margen izquierdo, después de poner el nombre de la clase se pone un espacio y se abre llave en la misma línea. Se establecieron pautas para utilización de sangrías dentro de las declaraciones dentro de métodos, bloques, al igual que para la apertura y cierre de llaves, el espaciado en blanco en las declaraciones, los controles, las Expresiones y Arreglos. Todo este estándar fue tomado del documento Normas y estándares de codificación de la línea de arquitectura del ERP el cual establece estos y otros conjuntos de normas por las cuales se rigió el desarrollo

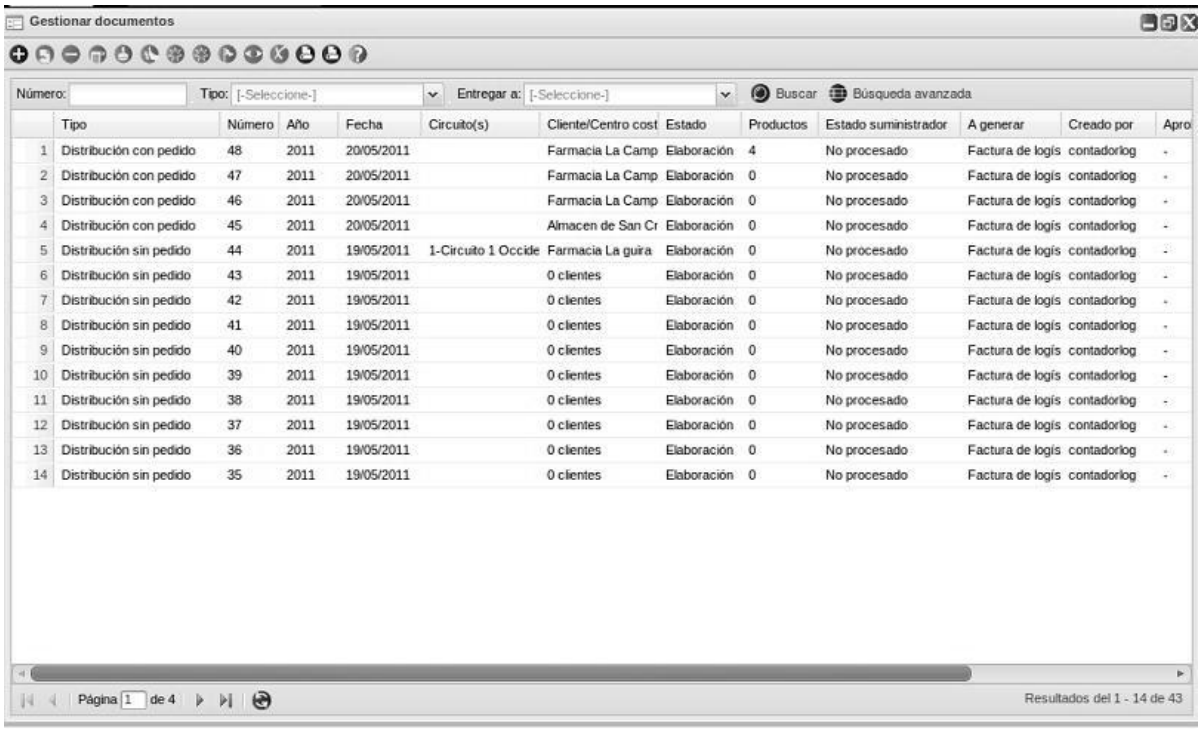
logrando una mejor organización y entendimiento de lo realizado.

2.7.2 Prototipo de interfaz

La interfaz de usuario es una de las partes más importantes de cualquier programa, determina la facilidad con que puede ser ejecutado lo que el usuario desea hacer. Un programa muy poderoso con una interfaz pobremente elaborada tiene poco valor para un usuario no experto. A continuación se muestran algunos prototipos de interfaz, los demás se pueden encontrar en los anexos.

Gestionar documentos

En esta interfaz se encuentran los botones necesarios para acceder a otras interfaces que permiten adicionar, modificar, eliminar distribuciones, entre otras.



	Tipo	Número	Año	Fecha	Circuito(s)	Cliente/Centro cost	Estado	Productos	Estado suministrador	A generar	Creado por	Apro
1	Distribución con pedido	48	2011	20/05/2011		Farmacia La Camp	Elaboración	4	No procesado	Factura de logís	contadorlog	-
2	Distribución con pedido	47	2011	20/05/2011		Farmacia La Camp	Elaboración	0	No procesado	Factura de logís	contadorlog	-
3	Distribución con pedido	46	2011	20/05/2011		Farmacia La Camp	Elaboración	0	No procesado	Factura de logís	contadorlog	-
4	Distribución con pedido	45	2011	20/05/2011		Almacen de San Cr	Elaboración	0	No procesado	Factura de logís	contadorlog	-
5	Distribución sin pedido	44	2011	19/05/2011	1-Circuito 1 Occide	Farmacia La guira	Elaboración	0	No procesado	Factura de logís	contadorlog	-
6	Distribución sin pedido	43	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
7	Distribución sin pedido	42	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
8	Distribución sin pedido	41	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
9	Distribución sin pedido	40	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
10	Distribución sin pedido	39	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
11	Distribución sin pedido	38	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
12	Distribución sin pedido	37	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
13	Distribución sin pedido	36	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-
14	Distribución sin pedido	35	2011	19/05/2011		0 clientes	Elaboración	0	No procesado	Factura de logís	contadorlog	-

Figura # 8 Interfaz Gestionar Documentos

Adicionar autorización de entrega

Esta interfaz permite seleccionar el tipo de distribución que se desea realizar, distribución por pedidos o sin pedidos y los clientes o circuitos a los que se les entregará la distribución. Luego de llenar los campos correctamente según las especificidades se procede a presionar el botón Acepta o Aplicar y se cierra la interfaz.

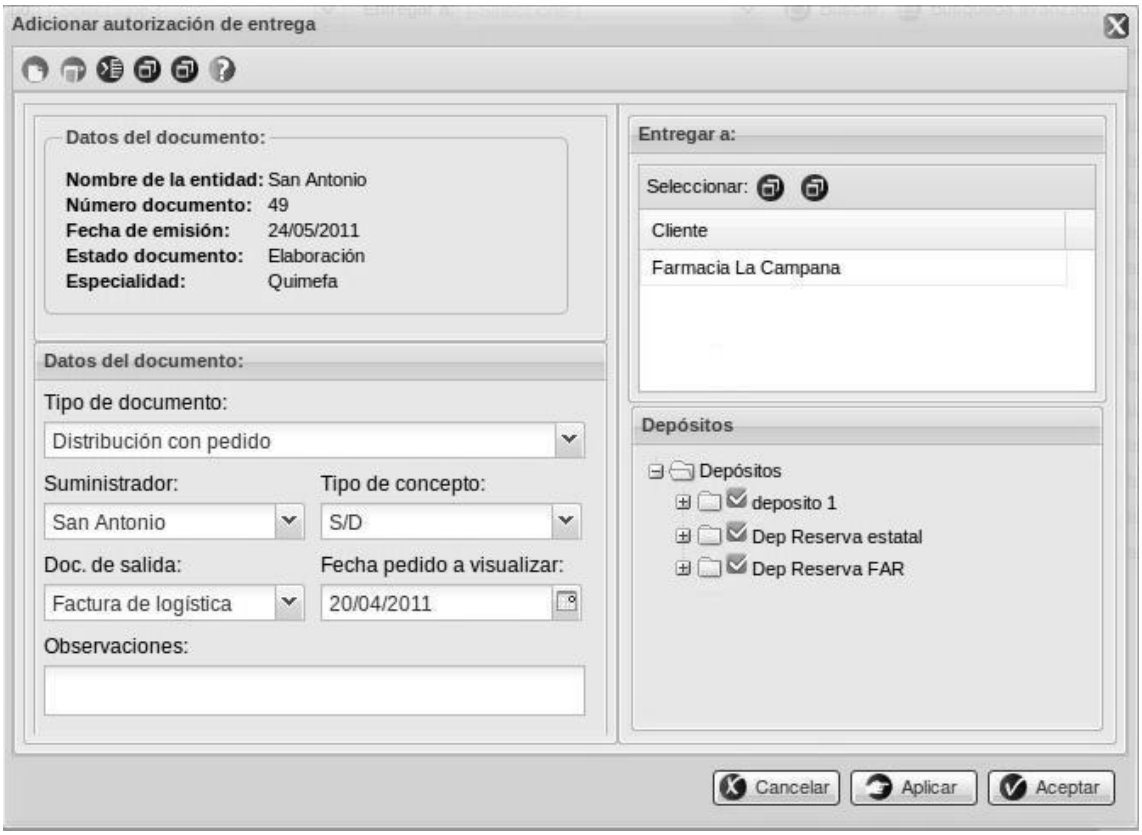


Figura # 9 Interfaz Adicionar autorización de entrega

Productos a distribuir

En esta interfaz se muestran los productos que se desean distribuir, los datos de los clientes a los que se les distribuirá y paralelamente a eso se va mostrando la propuesta de distribución. Además permite mostrar los productos alternativos.

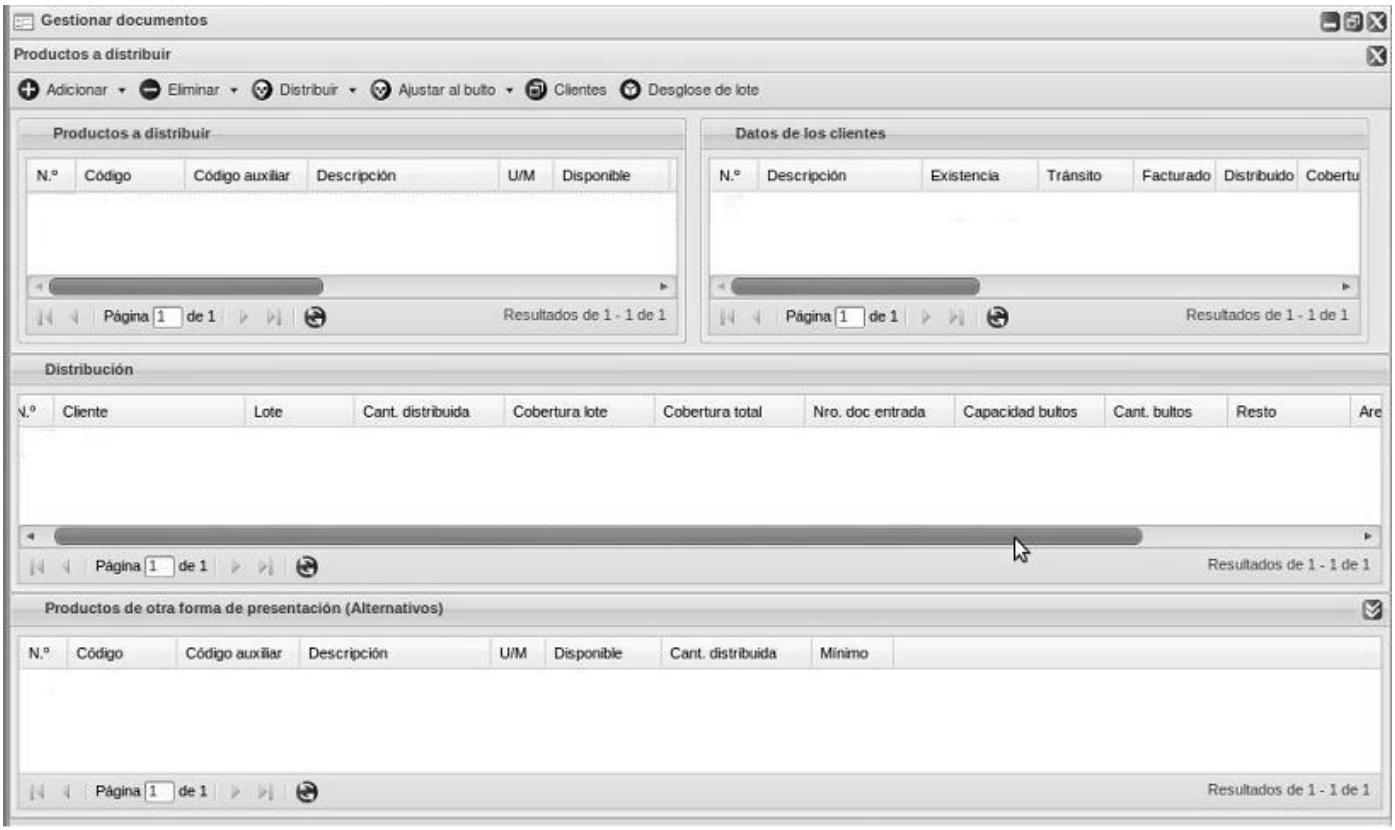


Figura # 10 Interfaz Productos a distribuir

2.7.3 Descripción de las clases principales

A continuación se describen las clases y sus operaciones más importantes, agrupándolas de acuerdo con la clasificación siguiente:

Controladoras

Las clases controladoras tienen la responsabilidad de ejecutar una lógica específica en función de las acciones que se realizan en la aplicación, gestionando todo el flujo de datos y operaciones entre la vista y demás clases del negocio.

Tabla # 2 Descripción de la clase controladora Gestionar orden de despacho.

Nombre: Gestordendespacho	
Tipo de clase : Controller	
Para cada responsabilidad:	
Nombre:	Descripción:

calcularpropuestadistribucionAction()	Acción para calcular la propuesta de distribución por cada producto.
cargaproductosdistribucionAction()	Acción para cargar los productos de distribución.
cargarclientesdistribucionAction()	Acción para cargar clientes de distribución.
modificarprioridadAction()	Acción para modificar la prioridad.

Modelo

Actúan como intermediarias entre las clases controladoras y las clases entidades. Complementan las funcionalidades de las clases controladoras.

Tabla # 3 Descripción de la clase modelo Gestionar producto de distribución.

Nombre: Gestproductodistribucion	
Tipo de clase : Model	
Para cada responsabilidad:	
Nombre:	Descripción:
cargarClientesDistribucion()	Función para devolver los clientes para la distribución.
calcularPropuestaDistribucion()	Función para calcular la propuesta de distribución por cada producto.
distribuirCantidadesPorProductosLotes()	Función para distribuir las cantidades por productos/lotes.
distribuirCantidadPorArea()	Función para distribuir las cantidades por áreas.
cargaGridProductos()	Función para devolver los productos a distribuir.

cargarDistribucionProducto()	Función para devolver los productos de la propuesta de distribución.
cargarProductosDistribuir()	Función para cargar los productos a distribuir.
eliminarProdOrden()	Función para eliminar un producto.
distribuirParaCadaClientes()	Función para distribuir las cantidades a los clientes.

Tabla # 4 Descripción de la clase modelo Gestionar orden de despacho

Nombre: Gestordendespacho	
Tipo de clase : Model	
Para cada responsabilidad:	
Nombre:	Descripción:
cargardatosencabdespacho()	Acción para cargar los datos del encabezado.
validarExistenPedidos()	Acción para validar que existan pedidos.

Tabla # 5 Descripción de la clase modelo Fórmulas.

Nombre: Formulas	
Tipo de clase : Model	
Para cada responsabilidad:	
Nombre:	**Descripción:**
Cobertura()	Función para calcular la cobertura del producto.
CantidadMaximaDistribuirClientes()	Función para calcular la cantidad máxima a distribuir a los clientes del circuito seleccionado.
CoeficienteDistribucion()	Función para calcular el coeficiente de distribución para cada cliente.
CantidadBultosResto()	Función para calcular la cantidad de bultos y el resto.

PorcientoDistribucion()	Función para calcular el porciento de distribución.
PorcientoConsumoCliente()	Función para calcular que por ciento representa el plan de consumo de cada cliente con respecto al total de los planes de consumo de los clientes del circuito.
CantidadDistribuidaCLiente()	Función para calcular la cantidad a distribuir por cada cliente.
CoberturaCliente()	Función para calcular la cobertura del producto para el cliente.
CoberturaDistribuidaCliente()	Función para calcular la cobertura del producto para el cliente.
DiasDiferencia()	Función para calcular los días de diferencia entre la cobertura total calculada para cada cliente y la política de cobertura del producto para cada cliente.
CantidadDiferencia()	Función para calcular la cantidad de diferencia.
ReecalcularCantidadDistribuida()	Función para recalcular la cantidad distribuida al cliente.
ReecalcularCantidadDistribuidaSeguimiento()	Función para recalcular la cantidad distribuida al cliente en caso que esté configurado que el seguimiento del plan va a limitar el despacho de los pedidos que excedan su cumplimiento.

2.8 Conclusiones parciales del capítulo

En este capítulo se abordaron los aspectos más significativos del diseño e implementación. Se desarrollaron los artefactos necesarios en cada etapa, brindando una descripción de cada uno de ellos con el objetivo de apoyar junto a la descripción de los tipos de clases definidas y la nomenclatura de sus variables y atributos, la implementación, por parte de los desarrolladores y conformar la

documentación del software para el proyecto. Para lograr la reutilización y optimización del código se planteó el uso de patrones.

CAPÍTULO 3: VALIDACION DE LA SOLUCION.PRUEBAS

3.1 Introducción

Durante el proceso de desarrollo de un software, los errores pueden comenzar a aparecer desde el momento en que fueron definidos los objetivos y estos a su vez especificados de forma errónea e imperfecta; de la misma forma en los posteriores pasos del diseño y desarrollo, por esto el desarrollo del software debe ser acompañado de una actividad que garantice su calidad. Para dar solución a estos problemas surge dentro del ciclo de desarrollo del software el proceso de pruebas. Las pruebas y validación de los resultados no se realizan una vez acabado el software, sino que deben ejecutarse en cada una de las etapas de desarrollo. El uso de las pruebas representa una revisión final de las especificaciones del diseño y de la codificación. (30)

En este capítulo se describe la validación de la solución propuesta, de forma que se pueda comprobar la eficiencia de las clases y operaciones utilizadas para satisfacer las necesidades del cliente.

3.2 Evaluación del modelo de diseño propuesto

Para medir la calidad del diseño se utilizan métricas básicas inspiradas en el estudio de la calidad del diseño orientado a objeto, por ser este un esquema sencillo de implementar y que cubre los principales atributos de calidad de software.

- Responsabilidad: Responsabilidad asignada a una clase en un marco de modelado de un dominio o concepto, de la problemática propuesta.
- Complejidad del diseño: Complejidad que posee una estructura de diseño de clases.
- Complejidad de implementación: Grado de dificultad que tiene implementar un diseño de clases determinado.
- Reutilización: Grado de reutilización presente en una clase o estructura de clase, dentro de un diseño de software.
- Acoplamiento: Grado de dependencia o interconexión de una clase o estructura de clase con otras, está muy ligada a la característica de Reutilización.
- Complejidad del mantenimiento: 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 costos y la planificación del proyecto.

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

Las métricas para evaluar la calidad del diseño y su relación con los atributos de calidad definidos son las siguientes:

- Tamaño Operacional de Clase (TOC)
- Relaciones entre Clases (RC)

3.2.1 Tamaño operacional de la clase (TOC)

El tamaño operacional de clase (TOC), está dado por el número de métodos asignados a una clase, la cual afecta un grupo de atributos como se muestra a continuación:

Tabla # 6 Descripción de los atributos de las métricas TOC.

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.

Para medir el tamaño de la clase se deben tener en cuenta diferentes aspectos como:

- Total de operaciones, ya sean las propias o las heredadas de las clases padres e interfaces que implementen.

- Cantidad de atributos, tanto los de ella, como lo de los padres.
- Promedio general de los dos anteriores para el sistema completo.

El instrumento que se utiliza para efectuar la validación de la métrica TOC se representa en la siguiente tabla.

Tabla # 7 Rango de valores para a evaluación de la métrica TOC

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

3.2.1.1 Resultados de la evaluación de la métrica

Tabla # 8 Resultados de la evaluación de la métrica TOC y su influencia en los atributos de calidad

Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
FormulasModel	13	baja	baja	alta
GestordendespachoController	75	media	media	media
GestproductodistribucionModel	9	baja	baja	alta
GestordendespachoModel	77	media	media	media

Figura # 11 Representación del % que representan los intervalos definidos.

Figura # 12 Representación de los resultados de la métrica TOC en el atributo de responsabilidad.

Figura # 13 Representación de los resultados de la métrica TOC en el atributo de complejidad.

Figura # 14 Representación de los resultados de la métrica TOC en el atributo de reutilización.

Analizando los resultados obtenidos en la evaluación del instrumento de medición de la métrica TOC, se puede concluir que para un total de 4 clases, el 50% de estas se encuentran entre 1 y 10 procedimientos y el otro 50 % tiene más de 26 procedimientos. Además el rango de evaluaciones positivas en los atributos sobre los que incide esta métrica (responsabilidad, complejidad y reutilización), están equilibrados en un 50%.

3.2.2 Relaciones entre clases (RC)

Esta métrica está dada por la cantidad de relaciones de uso que existe entre las distintas clases que forman el diseño propuesto. Se le aplica a las mismas clases que le fue aplicada la métrica TC. Los aspectos de calidad que se miden son: Acoplamiento, Complejidad de mantenimiento, Reutilización y Cantidad de pruebas.

Tabla # 9 Descripción de los atributos de las métricas RC.

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 # 10 Rango de valores para a evaluación de la métrica TOC

Atributo	Categoría	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2

Complejidad de mantenimiento	Alto	>2
	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
Reutilización	Alta	>2*Promedio
	Baja	>2*Promedio
	Media	Entre Promedio y 2*Promedio
Cantidad de pruebas	Baja	<=Promedio
	Media	Entre Promedio y 2*Promedio
	Alta	>2*Promedio

3.2.2.1 Resultados de la evaluación de la métrica

Tabla # 11 Resultados de la evaluación de la métrica RC y su influencia en los atributos de calidad

Clase	Cantidad de Relaciones de Uso	Acoplamiento	Complejidad Mant.	Reutilización	Cantidad de Pruebas
FormulasModel	3	alto	baja	alta	baja
GestordendespachoController	0	ninguno	baja	alta	baja
GestproductodistribucionModel	4	alto	baja	baja	baja
GestordendespachoModel	74	alto	alta	baja	alta

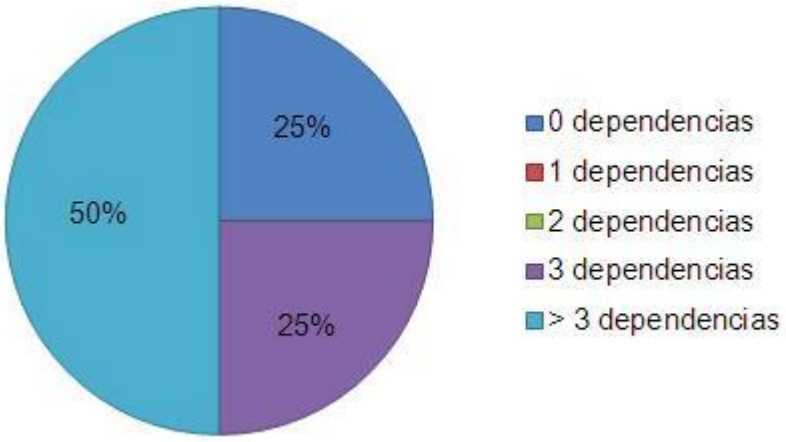


Figura # 15 Representación del % que representan los intervalos definidos.

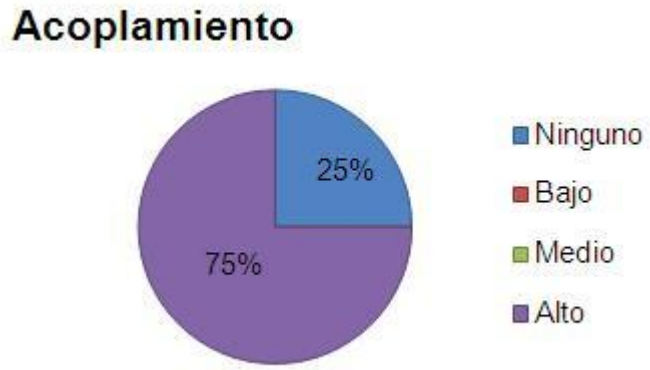


Figura # 16 Representación de los resultados de la métrica RC en el atributo de acoplamiento.

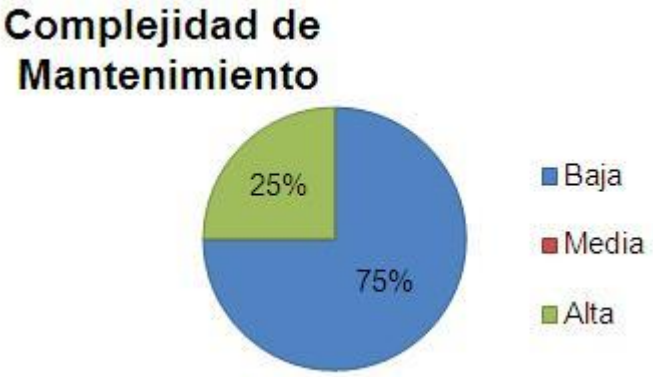


Figura # 17 Representación de los resultados de la métrica RC en el atributo de complejidad de mantenimiento

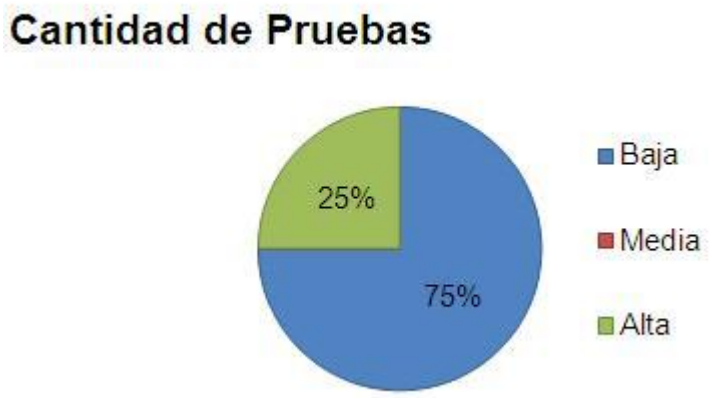


Figura # 18 Representación de los resultados de la métrica RC en el atributo de cantidad de pruebas

Figura # 19 Representación de los resultados de la métrica RC en el atributo de reutilización.

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que para un total de 4 clases, un 50% de estas poseen entre 0 y 3 dependencias y el otro 50% posee más de 3 dependencias. El promedio de acoplamiento que no tiene o posee bajo acoplamiento es del 25%.La complejidad de mantenimiento y la cantidad de pruebas oscilan entre baja y alta y la reutilización entre alta y baja.

3.3 Pruebas de software

Una vez generado el código fuente, el software debe ser probado para descubrir y corregir el máximo de errores posibles antes de ser entregado al cliente. El objetivo es diseñar casos de prueba que tengan una alta probabilidad de encontrar errores. (31)

Las pruebas no pueden asegurar la ausencia de defectos sino que permiten demostrar que existen defectos en el software, que cada prototipo que se quiera entregar al final de una iteración debe ser probado y evaluado. (31)

Al concluir las pruebas se evalúan los resultados obtenidos frente a los resultados esperados. Si se descubren datos erróneos implica que existe un error y hay que corregirlo y empieza el proceso de depuración de errores. Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes afectadas por las modificaciones.

3.3.1 Pruebas de Caja Negra

La prueba de caja negra se refiere a las pruebas que se llevan a cabo sobre la interfaz del software. O sea, los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma adecuada y que se produce un resultado correcto, así como que la integridad de la información externa se mantiene. (31)

Las pruebas de caja negra permiten encontrar errores tales como:

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

Existen varias técnicas para desarrollar las pruebas de caja negra:

Técnica de la Partición de Equivalencia: Divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.

Técnica del Análisis de Valores Límites: Prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.

Técnica de Grafos de Causa-Efecto: Permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

A continuación se aplica la prueba de partición de equivalencia como parte de la realización de la prueba de caja negra sobre la interfaz que responde al requisito funcional Gestionar Distribución. La Partición de Equivalencia divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Las variables de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Se definen dos tipos de variables de equivalencia:

- **Válidas**, representan entradas válidas al programa.
- **No válidas**, representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.

Tabla # 12: Escenarios de prueba.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1- Adicionar distribución.	El sistema debe permitir adicionar la distribución.	EP 1.1: Adicionar distribución introduciendo datos válidos.	– Se introducen los datos de la distribución correctamente. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar**. – Se cierra la interfaz.
		EP 1.2: Adicionar distribución introduciendo datos válidos presionando el botón **Aplicar**.	– Se introducen los datos de la distribución correctamente. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información. – La interfaz permanece abierta.
		EP 1.3: Adicionar distribución dejando campos vacíos.	– Se introducen los datos dejando algún campo en blanco. – Se presiona el botón **Aceptar**. – Se muestra un mensaje informando del error.
		EP 1.4: Se seleccionan los	– Se introducen los datos

		Circuitos de distribución.	correctamente. – Se presiona el botón **Aceptar**. – Se cierra la interfaz.
		EP 1.5: Se seleccionan los clientes.	– Se introducen los datos correctamente. – Se presiona el botón **Aceptar**. – Se cierra la interfaz.
		EP 1.6: Cancelar.	– Se introducen o no los datos de la distribución. – Se presiona el botón **Cancelar**.

Tabla # 13: Descripción de Variables para el caso de prueba

No	Nombre de campo	Tipo	Válido	Inválido	Inválido	Descripción
1	Tipo de documento	Campo de selección (No editable).	letras	Números, caracteres especiales.	Vacío	Se escoge el tipo de documento.
	Suministrador	Campo de selección (No editable).	letras	Números, caracteres especiales.	Vacío	Se escoge el suministrador.
	Tipo concepto	Campo de selección (No editable).	Letras	Números, caracteres especiales.	Vacío	Se escoge el tipo de concepto.
	Doc de Salida	Campo de selección (No editable).	letras	Números, caracteres especiales.	Vacío	Se escoge el documento de salida.
	Fecha pedido a visualizar	Fecha	-	-	Vacío	Se introduce la fecha.

	Observaciones	Campo de texto.	Letras y números.	Caracteres especiales.	Vacío	Se introducen las observaciones.
	Depósito	Tree	-	-	Vacío	Se escoge el depósito.

Ver anexos

Anexo 2: Tabla del juego de datos a probar del requisito Gestionar Distribución.

3.3.1.1 Resultados de las pruebas de caja negra

Para la realización de las pruebas de caja negra al sistema fueron analizados los casos de pruebas de los requisititos del sistema .Para esto se realizó una descripción de diferentes escenarios de prueba, los cuales fueron entre 2 y 5 escenarios a probar, en los mismos se probaron entre 1 y 4 juegos de datos. Atendiendo el comportamiento del sistema ante diferentes situaciones (entradas válidas y no válidas). Se puede concluir que los resultados esperados satisfactorios fueron mayores que los resultados esperados no satisfactorios. Algunos de los errores que arrojaron los resultados de esta prueba fueron, faltas de ortografía en las interfaces y en los mensajes de información o de errores; los cuales fueron corregidos para corroborar que cumplían con los resultados esperados. A continuación se muestra de forma gráfica el resultado de estas pruebas.

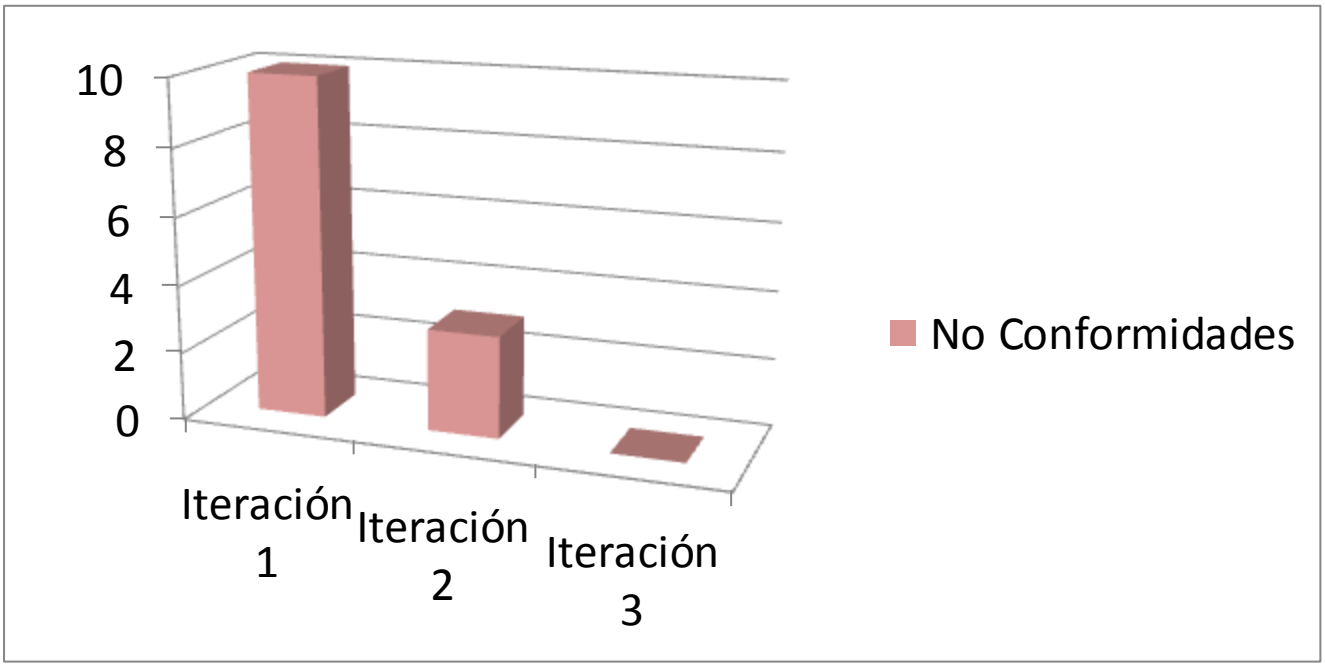


Figura # 20 Pruebas de Caja Negra

3.4 Conclusiones parciales del capítulo

Durante el desarrollo de este capítulo se validó la solución aplicando las métricas TOC y RC dirigidas a evaluar la calidad del diseño del software, donde los indicadores se mantuvieron en un rango de valores satisfactorios. Fueron validadas las funcionalidades implementadas, a través de diferentes pruebas de caja negra, mostrando que respondían adecuadamente a los requisitos funcionales y garantizando la satisfacción plena de las necesidades reales de los usuarios y demandas del cliente.

CONCLUSIONES

A partir del análisis de los procesos de distribución de productos y de los sistemas informáticos vinculados a esta actividad se demostró la necesidad e importancia de desarrollar un sistema para realizar la gestión de distribución de forma eficiente, lo cual significa un paso importante para el mejoramiento de las relaciones entre la entidad suministradora y los clientes. En tal sentido, se realizó un estudio de las herramientas y tecnologías necesarias para implementar la aplicación con la calidad requerida, haciendo uso del marco de trabajo Sauxe. Se realizó el diseño e implementación del sistema, probado y validado mediante métricas y pruebas de software.

La implementación de esta aplicación contribuye al mejoramiento de los procesos de distribución en las empresas y constituye un aporte decisivo a la informatización y economía del país.

RECOMENDACIONES

1. Seguir perfeccionando la aplicación de acuerdo con las nuevas necesidades que van surgiendo en el transcurso del tiempo.

2. Incorporar funcionalidades relacionadas con el empleo de los medios de transporte utilizados en la distribución que permitan optimizar su uso y maximizar el nivel de servicio al cliente.

REFERNCIAS BIBLIOGRAFICAS

1. [En línea] 2007. http://www.cez.com.pe/Sistemas/ERP.html.

2. Manual de Consultas , Gestion de Pedidos y Distribucion . [En línea] http://www.programaempresa.com/empresa/empresa.nsf/0/e88d210e51f9371ac125705b002c66c9/$FILE/pedidos1y2.pdf.

3. Cumulus, Software para Control de Almacenes, Pedidos y Distribución. [En línea] 2005. [Citado el: 22 de enero de 2011.] http://www.supplychain-software.com/index.html.

4. ecoSoftCS.net. ecosoftconsulting. [En línea] 2007. [Citado el: 24 de enero de 2011.] http://www.ecosoftconsulting.net/programasparapymes.aspx.

5. AxosVisual, ERP para Distribución. [En línea] 2007. [Citado el: 24 de enero de 2011.] http://www.axosvisual.com/empresa.php.

6. eva.uci.cu. [En línea] [Citado el: 26 de enero de 2011.] http://eva.uci.cu/mod/resource/view.php?id=11361.

7. Producción, Equipo de. *Modelo de Desarrollo orientado a componentes del proyecto ERP -CUBA.*

8. Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez. Plantilla Registro de la Propiedad intelectual (Sauxe). La Habana : s.n., 2008.

9. ExtJs. [En línea] http://extjs.com/deploy/dev/docs.

10. 2011. Doctrine. [Online] 2011. http://www.doctrine-project.org/

11. **Leopoldo, Carlos.** Zend Framework, una introducción. [En línea] 27 de noviembre de 2007. http://techtastico.com/post/zend-framework-una-introduccion/.

12. Manuales,Tutoriales y Herramientas. Curso de JavaScript. [En línea] 2008. [Citado el: 31 de enero de 2011.] http://max-alva.webs.com/javascript.htm

13 Abartia Team. [En línea] 2006. [Citado el: 31 de enero de 2011.] http://www.abartiateam.com/desarrollo-web/200602_uso-de-la-tecnologia-ajax-en-el-desarrollo-web.

14. otros, Stig Sæther Bakken y. *Manual de PHP.* 2001.

15. Ciclo de Vida Software. [En línea] 2009. [Citado el: 27 de enero de 2011.] http://ciclodevidasoftware.wikispaces.com/arquitectura+de+software.

16 Gómez Baryolo, Oiner, Tenrero Cabrera, Marianela y Nemuris, Silega Martínez. Plantilla Registro de la Propiedad intelectual (Sauxe). La Habana : s.n., 2008.EL DESARROLLO DE SOFTWARE.

17. Model View Controller Pattern. *Best Practice Software Engineering.* [En línea] 16 de Abril de 2010.

http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/mvc.html.

18. Visual Paradigm. [En línea] 2006. [Citado el: 26 de enero de 2011.] http://www.visual-paradigm.com/product/vpuml/.

19. Introducción a Apache. [En línea] 2009. [Citado el: 27 de enero de 2011.] http://linux.ciberaula.com/articulo/linux_apache_intro/..

20. Que es un sistema gestor de bases de datos. [En línea] 2007. [Citado el: 27 de enero de 2011.] http://www.cavsi.com/preguntasrespuestas/que-es-un-sistema-gestor-de-bases-de-datos-o-sgbd/..

21. SQLManager. [En línea] 2006. [Citado el: 28 de enero de 2011.] http://sqlmanager.net/. 9.

22. Fogel, Karl. Subversion. [En línea] http://svnbook.spears.at/nightly/es/svn-book.html#svn-ch-1-sect-1

23. Zend Studio for Eclipse. [En línea] 2007. [Citado el: 28 de enero de 2011.] http://www.zend.com/products/studio/.

24. UML y Patrones. Una introducción al análisis y el diseño orientado a objetos y al proceso unificado2009

25. CIG-UCID-Quimefa, Equipo. *Subsistema de Distribución del ERP Cedrux.* 2010.

26. Eva.uci.cu [Diseño]. [En línea] http://eva.uci.cu/mod/resource/view.php?id=14069.

27. UML y Patrones. Una introducción al análisis y el diseño orientado a objetos y al proceso unificado2009

28. Eva.uci.cu. [En línea] http://eva.uci.cu/mod/resource/view.php?id=14094.

29. ERP, Proyecto. Normas y Estandares de codificación*.* Habana : s.n., 2008.

30. Eva.uci.cu [Disciplina de pruebas]. [En línea] http://eva.uci.cu/mod/resource/view.php?id=14103.

31. Eva.uci.cu. [En línea] http://eva.uci.cu/mod/resource/view.php?id=14105.

ANEXOS 1 Prototipos de Interfaz

Figura # 21 Interfaz Circuitos de distribución

Figura # 22 Interfaz Seleccionar clientes

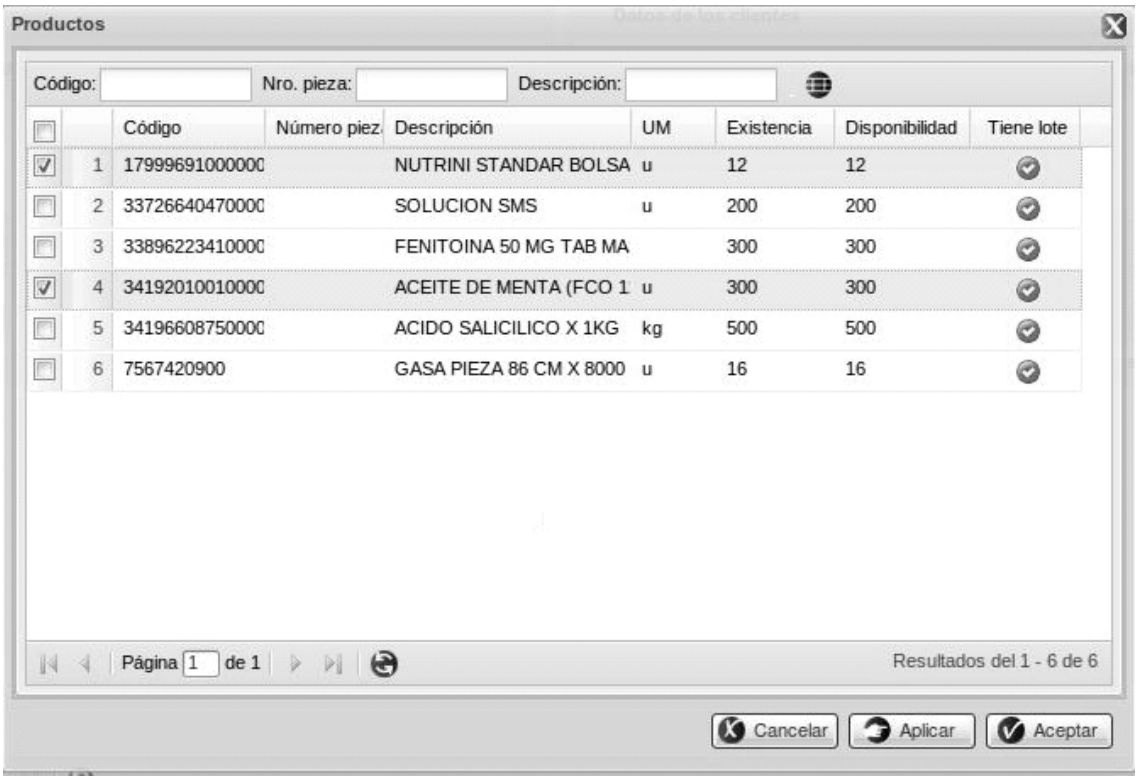


Figura # 23 Interfaz seleccionar productos

ANEXOS 2 Pruebas de Caja Negra

Tabla # 14 Juego de datos a probar

Id del escenario	Escenario	Tipo de documento	Suministrador	Tipo concepto	Doc de Salida	Fecha pedido a visualizar	Observaciones	Entregar a	Depósito	Respuesta del sistema
EP 1.1:	Adicionar distribución introduciendo datos válidos, presionando el botón **Aceptar**	V(Distribución sin pedido)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	V(distribución sin pedidos)	V(Circuito 1 Occidental para lab)	N/A	El sistema registra y muestra un mensaje de información "El documento fue adicionado satisfactoriamente" y se cierra la interfaz.
		V(Distribución sin pedido)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	V(distribución sin pedidos)	V(Farmacia Las Cañas)	V(Reserva estatal 1)	
		V(Distribución por pedidos)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	V(distribución por pedidos)	V(Circuito 1 Occidental para lab)	N/A	
		V(Distribución por pedidos)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	V(distribución por pedidos)	V(Farmacia Las Cañas)	V(Reserva estatal	
EP 1.2	Adicionar distribu	V(Distribución sin pedido)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	V(distribución sin	V(Farmacia Las Cañas)	V(Reserva estatal	El sistema registra y muestra un

	ción introduciendo datos válidos presionando el botón **Aplicar**.	V(Distribución sin pedido) V(Distribución por pedido) V(Distribución por pedido)	V(San Antonio) V(San Antonio) V(San Antonio)	V(S/D) V(S/D) V(S/D)	V(90001) V(90001) V(90001)	20/5/2011 20/5/2011 20/5/2011	pedidos) V(distribución sin pedidos) V(distribución por pedidos) V(distribución por pedidos)	V(Circuito 1 Occidental para lab V(Farmacia Las Cañas) V(Circuito 1 Occidental para lab	1) N/A V(Reserva estatal 1) N/A	mensaje de información "El documento fue adicionado satisfactoriamente"
EP 1.3	Adicionar distribución dejando campos vacíos.	V(Distribución sin pedido) V(Distribución sin pedido)	Vacío V(San Antonio)	V(S/D) Vacío	V(90001) V(90001)	20/5/2011 20/5/2011	V(distribución sin pedidos) V(distribución	V(Circuito 1 Occidental para lab V(Farmacia Las Cañas) V(Circuito 1 Occidental	N/A Reserva estatal 1	El sistema muestra un mensaje de información: "Existen datos requeridos que no han sido especificados."
		V(Distribución sin pedido)	V(San Antonio)	V(S/D)	V(90001)	20/5/2011	n sin pedidos)	para lab Vacío	N/A	
		V(Distribución sin pedido)	o)		V(90001)		Vacío	V(Farmacia Las Cañas)	N/A	
		V(Distribu	V(San Antoni	V(S/D)	V(90	20/5/2011	V(distr			

		ción sin pedido)	o) V(San Antonio)	V(S/D)	001)	20/5/2011	ibución sin pedidos) V(distribución sin pedidos)		Vacío	
EP 1.6	Cancelar.	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	El sistema cierra la interfaz sin realizar ninguna operación.