Universidad de las Ciencias Informáticas

Facultad 3

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

Título:

“Diseño e implementación del sistema de informatización para oficinas comerciales del Ministerio de Informática y las Comunicaciones.”

Autor: Norlan Capote Díaz

Tutor: MSc. Osmar Leyet Fernández

Ciudad de la Habana, Junio de 2012.

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo al CEIGE de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

_____	_____
Norlan Capote Díaz	**MSc. Osmar Leyet Fernández**
Autor	**Tutor**

Datos de contacto

Síntesis del Tutor:

- Miembro del equipo de implementación del sistema Saren, del proyecto Registros y Notarías.
- Analista Principal del sistema de Comercio electrónico B2B para la empresa Cubalse de la República de Cuba.
- Arquitecto de Tecnología del sistema Cedrux, del proyecto ERP Cubano.
- Jefe de la línea de Contabilidad y Finanzas del sistema Cedrux, del proyecto ERP Cubano.
- Jefe de departamento “Desarrollo de Productos” del centro CEIGE.

Resumen

En un mundo tan cambiante y globalizado, las empresas requieren de herramientas tecnológicas que contribuyan en la estrategia global de negocio a maximizar la rentabilidad y mejores prácticas operativas en las diferentes disciplinas de la organización. El país avanza con amplios pasos dentro del proceso de utilización ordenada y masiva de las tecnologías de la información y las comunicaciones, siendo este un factor decisivo para el desarrollo de las empresas, la economía y la sociedad cubana.

El presente trabajo tiene como objetivo desarrollar una aplicación para gestionar la información que se genera en el proceso de contratación y ventas en las oficinas comerciales del Ministerio de Informática y las Comunicaciones; ya que este proceso se realiza de forma manual provocando pérdida de información, duplicación de la misma y que los datos estadísticos no sean confiables, lo que conlleva a que el proceso se torne engorroso y complejo. Para el desarrollo del sistema se utilizaron como lenguajes de programación, PHP por el lado del servidor y JavaScript por el lado del cliente. Como gestor de base de datos se utilizó PostgreSQL. Para la estructuración y construcción de la solución se trabajó con el modelo de desarrollo del marco de trabajo Sauxe. Se empleó el Lenguaje Unificado de Modelado (UML) para la modelación del diseño aplicándose mediante la herramienta Visual Paradigm para UML.

Como resultado se obtuvo una solución informática capaz de gestionar la información generada en el proceso de contratación en las oficinas comerciales del MIC, validada mediante métricas de software y probada mediantes pruebas de caja blanca y pruebas de caja negra desarrolladas por el grupo de aseguramiento de la calidad del Centro de Informatización de la Gestión de Entidades (CEIGE).

Palabras clave

Contratación, Contrato, Venta, Proveedor.

Tabla de contenido

Introducción

La era de la información avanza velozmente, obligándonos a mantener un margen de desarrollo tecnológico acorde a los nuevos tiempos. Actualmente, las empresas cubanas se ven en la necesidad de contar con información de alta calidad que les permita agilizar los procesos económicos basándose en las tecnologías y la información para mejorar la toma de decisiones.

En un entorno muy competitivo con unas exigencias de los clientes cada vez mayores y más diversificadas, las organizaciones prestan más atención a los contratos como fuentes de flujos de ingresos predecibles y renovables.

La gestión de los contratos le ayuda a garantizar una previsión precisa de los ingresos y los servicios necesarios, proporcionando transparencia del rendimiento y posibilitando la rentabilidad de los contratos.

El Ministerio de Informática y las Comunicaciones (**MIC**) cuenta con oficinas comerciales en el exterior que son las encargadas de establecer contratos con empresas extranjeras que comercializan gran cantidad de productos informáticos y electrónicos.

Actualmente en las oficinas comerciales del MIC ubicadas en el exterior, el proceso se efectúa utilizando herramientas ofimáticas de apoyo, como las hojas de cálculo de Microsoft Office Excel y el sistema de gestión de bases de datos relacionales Microsoft Access, que aunque son útiles, han demostrado ser ineficientes, debido a que requieren un alto nivel operacional por parte del usuario, lo que significa que el personal debe tener conocimientos avanzados sobre las herramientas ofimáticas provocando un mayor tiempo de duración del proceso.

El uso de estas herramientas trae consigo pérdida de la información y que los reportes estadísticos no sean los más confiables en las entidades donde la gestión de estos procesos se lleva a cabo de forma manual, convirtiéndose en un proceso lento y engorroso que incrementa la pérdida de datos. No se puede hacer consultas de forma remota, ya que los datos están guardados en una base de datos de Microsoft Access y para obtener cualquier dato tiene que ser desde el puesto de trabajo donde se encuentra la base de datos. Posee una capacidad limitada para almacenar información, y no garantiza la consistencia, integridad y seguridad de la misma, debido a que los datos pueden ser modificados fácilmente aunque se pueden usar protecciones, pero no son suficientemente seguras. Además, se tiene que guardar la

información cada cierto tiempo, para garantizar que ante cualquier falla no se pierdan los datos que se están gestionando.

No se cuenta con una buena organización, no permite personalizar filtros de seguimiento y búsqueda, y proveer acceso de información del contrato a todas las partes involucradas para reducir el envío por correo electrónico.

Se tiene como **problema a resolver**: El nivel de informatización actual de las oficinas comerciales del Ministerio de Informática y las Comunicaciones es insuficiente para las actividades de procesamiento y consulta de la información correspondiente.

El objetivo general: Desarrollar un sistema informático que gestione de manera suficiente las actividades de procesamiento y consulta de la información correspondiente en las oficinas comerciales del Ministerio de Informática y las Comunicaciones.

Objetivos Específicos:

- Elaborar un marco teórico relativo a los sistemas para la gestión de la actividad comercial empresarial.
- Diseñar el sistema para la informatización de las actividades de procesamiento y consulta de la información.
- Implementar el sistema para la informatización de las actividades de procesamiento y consulta de la información.
- Realizar pruebas al sistema desarrollado que demuestren la calidad y cumplimiento de los requisitos del mismo.

Idea a defender: Con la realización de esta herramienta para la gestión de la información que se genera en el proceso de contratación y ventas en las oficinas comerciales del Ministerio de Informática y las Comunicaciones (MIC) en el exterior, se agilizarán las actividades de procesamiento y consulta de la información permitiendo una mayor disponibilidad, integridad y seguridad de la información.

Posible resultado:

Contar con un sistema para la informatización de las actividades de procesamiento y consulta de la información en las oficinas comerciales del Ministerio de Informática y las Comunicaciones.

Objeto de estudio:

- Sistemas de gestión empresarial.

Campo de acción:

- Sistemas de contratación empresarial.

Para llevar a cabo las tareas se emplearon los siguientes métodos científicos de investigación:

Métodos teóricos:

- **Método analítico sintético:** Fue útil para el autor dividir la información encontrada de los sistemas de información estudiados, para facilitar su estudio; y luego, unir las partes previamente analizadas, para descubrir las características generales y las relaciones esenciales de estos sistemas.
- **Método inductivo deductivo:** La investigación de la realidad ofreció al autor datos que le permitieron, mediante la inducción, elaborar la idea a defender y a partir de la misma deducir el resultado.
- **Método de la modelación:** Se evidencia cuando el autor creó diagramas con el objetivo de explicar la complejidad del negocio.
- **Método dialéctico:** Fue necesario el conocimiento de las relaciones entre las entidades del negocio para resolver el problema que dio inicio a la investigación.

Métodos empíricos:

- **Entrevista:** las entrevistas realizadas al cliente permitieron profundizar el conocimiento sobre el negocio.

Capítulo 1: Fundamentación teórica

1.1 Introducción

En este capítulo se abarcan un grupo de conceptos y teorías relacionadas con el negocio. Se realiza una investigación en base a sistemas de información, además se analiza el modelo de desarrollo, las tecnologías y las herramientas a utilizar en el desarrollo del sistema.

1.2 Marco conceptual

1.2.1 Contrato

Un acuerdo voluntario, deliberado y jurídicamente vinculado entre dos o más partes competentes. Los contratos son generalmente por escrito, pero pueden ser orales o de forma implícita, y por lo general tienen que ver con el empleo, la venta o alquiler. La relación contractual se evidencia por una oferta, la aceptación de la oferta y considerarla válida de una manera legal. Cada parte de un contrato adquiere los derechos y deberes en relación con los derechos y deberes de los demás partidos. Sin embargo, mientras todas las partes pueden esperar un beneficio justo del contrato no se puede decir que cada parte se beneficiará en igual medida. Los contratos son generalmente aplicables o no en forma escrita, a pesar de que un contrato por escrito protege a todas las partes involucradas en el mismo. (1)

El contrato no es el documento que lo recoge, y que constituye únicamente una exigencia de prueba. El contrato existe desde que una o varias personas consienten en obligarse, respecto de otra u otras, a dar alguna cosa o prestar algún servicio. (2)

1.2.2 Venta

La venta es una de las actividades más pretendidas por empresas, organizaciones o personas que ofrecen algo (productos, servicios u otros) en su mercado meta, debido a que su éxito depende directamente de la cantidad de veces que realicen ésta actividad, de lo bien que lo hagan y de cuán rentable les resulte hacerlo. (3)

La venta también es el contrato a través del cual se transfiere una cosa propia a dominio ajeno por el precio pactado. Puede ser algo potencial (un producto que está a la venta pero que aún no ha sido comprado) o una operación ya concretada (en este caso, implica necesariamente la compra). (3)

Suele hablarse de compra-venta para hacer mención a la operación bilateral donde el vendedor entrega una cosa determinada al comprador, quien paga por ella un precio. Lo habitual es que dicho pago se realice en dinero, ya que si se escoge otro objeto a cambio estamos ante un trueque. (3)

La venta de productos o servicios constituye la base de las operaciones de las empresas. A través de estas ventas, las compañías obtienen ingresos. El hecho de ser rentables dependerá de muchos otros factores, como la gestión de costos. (3)

En síntesis, la definición de venta enfoca la misma desde dos perspectivas diferentes:(4)

- Una perspectiva general, en el que la "venta" es la transferencia de algo (un producto, servicio, idea u otro) a un comprador mediante el pago de un precio convenido.
- Una perspectiva de mercadotecnia, en el que la "venta" es toda actividad que incluye un proceso personal o impersonal mediante el cual, la persona que efectúa la venta: identifica las necesidades y/o deseos del comprador, genera el impulso hacia el intercambio y satisface las necesidades y deseos del comprador (con un producto, servicio u otro) para lograr el beneficio de ambas partes.

1.2.3 Producto

Todo lo que el comprador recibe cuando efectúa un acto de compra: el producto propiamente dicho (bien o servicio), el envase, la garantía y los servicios complementarios. El producto debe responder a las necesidades de los consumidores y no a las preferencias de los ejecutivos y técnicos de la empresa. Un producto comercial es algo más que un bien o servicio que satisface una determinada necesidad. Un producto comercial es, en realidad, una combinación de atributos: diseño, color, calidad, coste, envasado, tamaño, duración, peso, etcétera. Estos atributos, que pueden parecer secundarios desde una óptica meramente utilitarista y no concurrente, son determinantes con frecuencia del éxito o fracaso comercial de muchos productos. (5)

"El producto es el resultado de un esfuerzo creador que tiene un conjunto de atributos tangibles e intangibles (empaque, color, precio, calidad, marca, servicios y la reputación del vendedor) los cuales son percibidos por sus compradores (reales y potenciales) como capaces de satisfacer sus necesidades o deseos. Por tanto, un producto puede ser un bien (una guitarra), un servicio (un examen médico), una idea (los pasos para dejar de fumar), una persona (un político) o un lugar (playas paradisiacas para

vacacionar), y existe para propósitos de intercambio, la satisfacción de necesidades o deseos y para coadyuvar al logro de objetivos de una organización (lucrativa o no lucrativa)". (6)

1.2.4 Cliente

- Según la American Marketing Association (A.M.A. Asociación Americana de Marketing), el cliente es "el comprador potencial o real de los productos o servicios".(7)
- Según The Chartered Institute of Marketing (CIM, del Reino Unido), el cliente es "una persona o empresa que adquiere bienes o servicios (no necesariamente el Consumidor final)".(8)
- En el Diccionario de Marketing, de Cultural S.A., se encuentra que "cliente es un término que define a la persona u organización que realiza una compra. Puede estar comprando en su nombre, y disfrutar personalmente del bien adquirido, o comprar para otro, como el caso de los artículos infantiles. Resulta la parte de la población más importante de la compañía".(9)
- En el libro "Marketing de Clientes ¿Quién se ha llevado a mi cliente?" se menciona lo siguiente: "La palabra cliente proviene del griego antiguo y hace referencia a la «persona que depende de». Es decir, mis clientes son aquellas personas que tienen cierta necesidad de un producto o servicio que mi empresa puede satisfacer". (10)

Por tanto: Cliente es la persona, empresa u organización que adquiere o compra de forma voluntaria productos o servicios que necesita o desea para sí mismo, para otra persona o para una empresa u organización; por lo cual, es el motivo principal por el que se crean, producen, fabrican y comercializan productos y servicios. (11)

1.2.5 Proveedor

Proveedor es la persona o empresa que abastece con algo a otra empresa o a una comunidad. El término procede del verbo proveer, que hace referencia a suministrar lo necesario para un fin.

Los proveedores deben cumplir con los plazos y las condiciones de entrega de sus productos o servicios para evitar conflictos con la empresa a la que abastecen. En muchos casos, estas compañías tienen que tener un departamento de soporte o atención técnica, ya que las interrupciones del servicio causan grandes problemas al cliente. (12)

En informática, un proveedor es una entidad física o virtual que tiene el fin de ofrecer un servicio a otra u otras entidades. Los tipos de proveedores pueden ser tan distintos como una empresa que brinda servicios de Internet a clientes en un país, como un sistema informático que pone aplicaciones y recursos al servicio de otros.

En general, se conoce como proveedores a las empresas o particulares que ofrecen servicios tecnológicos. Estos pueden ser acceso y conexión a Internet, telefonía móvil, hosting[1] de aplicaciones y sitios web, acceso a servicios y cuentas en determinados software o sitios web, etc. (13)

1.3 Sistemas informáticos vinculados a la gestión de contratos y ventas

Planificación de recursos de empresariales ERP (Enterprise Resource Planning, ERP por sus siglas en inglés) es un sistema de información que permite concentrar en su mayoría las operaciones que realiza una empresa como son la producción, ventas, logística, distribución, inventarios, facturas, clientes, contabilidad, entre otras. (14)

Dentro de los objetivos principales de un ERP se puede encontrar:

- Optimización de los procesos empresariales.
- Acceso a toda la información de forma confiable, precisa y oportuna (integridad de datos).
- La posibilidad de compartir información entre todos los componentes de la organización.
- Eliminación de datos y operaciones innecesarias de reingeniería.

El número de empresas que utilizan herramientas informáticas para lograr un mayor rendimiento y desempeño de la misma crece de manera vertiginosa. Ejemplo de estas herramientas son los ERP, algunos de los cuales serán objeto de estudio para analizar sus interfaces de usuario, observando el tipo de información que se maneja y la forma en que se hace, además de poder analizar las funciones que presentan, las cuales podrían ser muy útiles para el desarrollo del sistema a implementar.

[1] Implica el alquiler de un espacio en el disco rígido de un servidor conectado directamente a la red de Internet para alojar su sitio web.

En el mercado existe una gran variedad de proveedores de sistemas ERP, entre los cuales podemos encontrar ERP Openbravo, Compiere, Axapta, SAP, entre otros. Así como sistemas de Gestión que no necesariamente son ERP, por ejemplo Codeka: Sistema de Gestión Empresarial. (15)

1.3.1 Mercanza GESTIÓN

Es un sistema de gestión empresarial que ofrece la facilidad de mantener la información comercial eficientemente organizada tanto de las actividades de ventas como de las de compras, así como los artículos debidamente referenciados, el registro y valoración de existencias, y también los contratos tipificados.(16)

Módulo de Contratos en Mercanza Gestión

El módulo de Contratos de Mercanza Gestión permite realizar la facturación automática de aquellos servicios que tienen una periodicidad concreta en su liquidación. El módulo de contratos se convierte en un sistema idóneo para la gestión y facturación de contratos de alquiler, mantenimiento y asistencia técnica. (17)

Con un par de breves acciones se puede generar automáticamente todas las facturas correspondientes a un período, mensual, trimestral, semestral o anual y enviar a su banco los recibos de cargo correspondientes. (17)

1.3.2 OpenERP

OpenERP es un sistema de gestión de empresas/organizaciones (ERP) de licencia libre que cubre las necesidades de las áreas de contabilidad, ventas, compras, almacén, inventario, proyectos, recursos humanos, tiendas virtuales, entre otros.

Incorpora funcionalidades de gestión de documentos y conectores con otras aplicaciones. Permite trabajar remotamente mediante una interfaz web o aplicación de escritorio multiplataforma (Windows, Linux y Mac) e incluye un entorno modular de programación/adaptación rápida de aplicaciones. Se basa en tecnología Python/XML trabajando sobre una base de datos PostgreSQL. (18)

Módulos de base más importantes de OpenERP

- Facturación, cobros y pagos
- Contabilidad

- Productos
- Gestión de Atención a Clientes y Proveedores
- Gestión de Compras
- Workflow(Flujo de trabajo) de procesos
- Gestión de Ventas
- Gestión de informes
- Gestor documental

Gestión de Atención a Clientes y Proveedores

OpenERP ofrece además de un módulo de atención al Cliente, un módulo de atención al proveedor.

Entre otras la funcionalidad ofrecida por estos módulos es la de Gestión de casos. El concepto de caso, permite gestionar diferentes comunicaciones de los clientes o proveedores que requieran una atención posterior por parte del personal de la empresa que utilice este sistema de planificación de recursos.

Algunos de esos casos pueden ser: reclamaciones de pedidos, problemas de calidad, gestión de llamadas, tickets de soporte y ofertas de trabajo.

Gestión de Compras

El módulo de gestión de compras permite:

- Realizar el seguimiento de tarifas de sus proveedores y convertirlas en órdenes de compra.
- OpenERP tiene varios métodos de monitorizar factura y realizar el seguimiento de la recepción de materiales solicitados.
- Gestionar entregas parciales del proveedor y mercancías faltantes.
- Gestión de reclamaciones a proveedor por retrasos en la entrega.
- Las reglas de reabastecimiento de mercancía del sistema permiten generar borradores de pedidos de compra (necesidades de compra) automáticamente.
- También se pueden configurar para que se ejecute un procedimiento totalmente ajustado a las necesidades de compra marcadas por el área de fabricación.

Productos

En OpenERP producto significa:

- Almacenable
- Consumible
- Servicio

Se puede trabajar con productos concretos o con plantillas que separan la definición del producto y sus variantes. Las variantes de productos son definidas por los atributos que se definan para dicho producto (color, talla, peso, calidad, densidad, etc.)

Asociadas al producto, se definen las listas de precios o tarifas, tanto de compra, como de venta. Los precios se ajustan a los cambios de moneda.

Las listas de precios pueden ser definidas con un precio fijo por producto o se pueden construir definiendo reglas. Esta opción permite definir múltiples descuentos, precios de venta basados en los de compra, reducciones de precio y ofertas en un determinado rango de productos.

Gestión de Ventas

La funcionalidad ofrecida en esta área es similar a la ofrecida en el área de compras, sin embargo hay grandes diferencias en el Workflow (flujo de trabajo) aplicable a cada una.

- Creación de pedidos de venta
- Revisión de los pedidos en sus distintos estados
- Consulta del Workflow de los pedidos de venta
- Confirmación de envío
- Se puede definir una fecha de facturación y las condiciones individualmente en cada pedido

Los gastos de envío pueden asignarse usando una tabla de tarifas donde se pueden consultar los precios de los diferentes proveedores de transporte.

1.3.3 Openbravo

Es un sistema ERP de código abierto ideal para la gestión de pequeñas y medianas empresas. Sobre todo para aquellas empresas que no pueden realizar una gran inversión, pero desean tener una herramienta de excelente calidad que les permita facilitar la administración de sus empresas. (19)

Se encuentra desarrollado en el lenguaje de programación Java, se ejecuta sobre Apache y Tomcat y brinda soporte para bases de datos como los son Oracle y PostgreSQL. (19)

Tiene como principales ventajas que sigue un licenciamiento de software libre lo cual permite adquirirlo sin costo alguno o la necesidad de depender de algún proveedor de servicio. Además la licencia del producto asegura el acceso público al código fuente y la posibilidad de modificar dicho código libremente adaptándolo a las necesidades de la empresa. (19)

El ERP Openbravo ha sido diseñado para ser una solución web (montado sobre un servidor), haciendo posible que sea consultado desde cualquier parte del mundo, proporcionando a su vez una excelente seguridad y una facilidad de navegación a través de cualquier navegador web (Explorer, Firefox o Chrome). (19)

Las grandes áreas que integra actualmente el sistema de gestión son:(19)

- **Gestión de los datos maestros**: productos, componentes, listas de materiales, clientes, proveedores, etc.
- **Gestión de los aprovisionamientos**: tarifas, pedidos de compra, recepción de mercancías, verificación de facturas de proveedores, evaluación de proveedores, etc.
- **Gestión de almacenes**: almacenes y ubicaciones, unidades de almacén, lotes, número de serie, bultos, etiquetas, entradas, salidas, movimientos entre almacenes, inventarios, valoración de existencias, transportes, etc.
- **Gestión de proyectos**: proyectos, fases, presupuestos, gastos, compras asociadas, etc.
- **Gestión de servicios**: recursos, servicios, gastos, gastos refacturables, facturación de servicios, nivel de servicio, etc.
- **Gestión de la producción**: estructura de planta, hojas de ruta, órdenes de fabricación, partes de trabajo, incidencias de trabajo, partes de mantenimiento, etc.
- **Gestión comercial y gestión de las relaciones con clientes**: pedidos de venta, tarifas, albaranes, facturación, comisiones, etc.
- **Gestión económico-financiera**: plan de cuentas, cuentas contables, impuestos, contabilidad general, cuentas a pagar, cuentas a cobrar, contabilidad bancaria, balance, cuenta de resultados, activos fijos, etc.

Puede ser integrado de manera natural con diversas áreas como lo son la gestión de relaciones con clientes, inteligencia de negocios o business intelligence y terminal de punto de venta.

Orden de venta

Documento usado para aprobar, hacer seguimiento y procesar los pedidos del cliente. Las órdenes de venta son tomadas generalmente por los representantes de venta; sin embargo, cada empresa puede generar diferentes fórmulas. (19)

Tipos de órdenes de venta

- **Orden estándar***:* Una orden que automáticamente separa materiales para ser enviados. En este caso el envío y la factura pueden generarse por aparte.
- **Orden en punto de venta***:* Los materiales son recogidos en una factura y pueden pagarse de maneras diferentes. En este caso el envío y la factura se generan automáticamente; y dependiendo de los términos de pago, también se puede generar el pago automáticamente.
- **Cotización***:* Una oferta o venta potencial que automáticamente reserva materiales. Una cotización puede convertirse en una orden, o permanecer como cotización que nunca se convierte en una orden. El segundo caso ocurre si la orden nunca fue confirmada por el cliente.
- **Propuesta***:* Una oferta o potencialmente una orden de venta que puede ser adaptada manualmente de otros documentos de ventas.
- **Orden prepaga***:* La factura es cancelada antes de que los productos en ella sean enviados. El pago puede atarse a otro envío o enviarlo directamente. Después de que se haga el pago, los documentos de factura y envío son generados automáticamente.
- **Orden a crédito***:* Los materiales se relacionan en una factura. En este caso el envío y la factura se generan automáticamente. El pago se ingresa manualmente después de que el cliente efectúe los pagos.

Orden de compra

Documento que especifica los productos solicitados a un proveedor específico, así como el precio, términos y condiciones de esa orden. Las órdenes de compra creadas a partir de las solicitudes que han sido formuladas por diversos departamentos. (19)

Factura de compra

Una factura es una declaración escrita que marca las mercancías enviadas a un comprador por un proveedor. También se indica la cantidad y el precio de cada producto o servicio incluido en la orden. (19)

1.3.4 Codeka

Es una aplicación bajo licencia GPL (General Public License, Licencia Pública General) para controlar la facturación y gestionar el almacén de una pequeña o mediana empresa. Su gran virtud está en la facilidad de uso y en cubrir las necesidades de las pymes.[2] Está desarrollada sobre entorno web, lo que le hace ser muy versátil. Es independiente del sistema operativo y además permite el trabajo a través de una conexión de red. (15)

Funciones:

- Gestión de Interlocutores comerciales [Clientes y proveedores]
- Gestión de Artículos y Familias
- Gestión de Facturas de los clientes
- Gestión de Facturas de los proveedores
- Ventas en mostrador
- Gestión de los cobros y pagos
- Creación y configuración de códigos de barras
- Gestión de copias de seguridad
- Listados en formato PDF

1.3.5 Cedrux

Basado en los principios de independencia tecnológica y en las particularidades de la economía cubana. Esta solución nacional incluye un subsistema para el control de los activos fijos tangibles (AFT), que pretende servir de apoyo a la toma de decisiones en las entidades. El subsistema es capaz de gestionar activos de forma dinámica, tributa mediante operaciones al sistema contable de la empresa y estandariza el tratamiento de los AFT a nivel nacional. Incluye además temas como la multimoneda, la agrupación de

[2]Empresas que empleen, a menos, 250 trabajadores; y que posean un balance general inferior a los 43 millones de euros.

activos y la generación de documentos, prestaciones del sistema que responden a necesidades del país, actualmente no satisfechas de manera conjunta por los sistemas existentes. A partir de la puesta en marcha del sistema, la economía del país se verá favorecida y el control de los AFT será confiable, estable y legalmente correcto. (20)

1.3.6 Resultados del estudio de los sistemas informáticos observados

Durante la investigación y estudio de los sistemas en cuestión, se puede observar que Mercanza GESTIÓN está inclinado hacia la contratación de servicios y no a la contratación de compra de productos por lo cual su solución no brinda información relevante que se pueda utilizar en la construcción de la propuesta de solución del trabajo. No siendo así con Codeka, Openbravo y OpenERP que aportan actividades y funcionalidades a tener en cuenta como:

- **Gestión de casos** del OpenERP que incluye reclamaciones de pedidos, problemas de calidad, gestión de llamadas y tickets de soporte.
- **Gestión de los aprovisionamientos** del Openbravo que incluye tarifas, pedidos de compra, recepción de mercancías, verificación de facturas de proveedores, evaluación de proveedores, etc.
- **Gestión de Artículos y Familias** del Codeka que nos indica que agrupando los productos en familias de productos se obtiene una mejor organización de nuestro sistema.
- **Generar listados en formato PDF** que permiten imprimir los reportes que se obtienen del resultado de una búsqueda efectuada en el sistema para una rápida toma de decisiones.

Aunque algunas no estarán disponibles para esta versión pero representan un elemento importante ya que nos aportan una visión futura de lo que podría llegar a hacer el sistema que se propone.

No se utilizarán estos sistemas para darle solución a nuestra problemática ya que no tienen en cuenta las particularidades de la economía cubana; por ejemplo la dualidad monetaria.

Por lo tanto no existe una solución informática capaz de ejecutar las funcionalidades para la gestión de la información en las oficinas comerciales del MIC. Por lo que es de vital importancia su construcción haciendo enfocándola de la manera que lo hacen los sistemas de gestión empresarial Codeka, Openbravo y OpenERP.

1.4 Modelo de desarrollo de software

Se utilizará un modelo de desarrollo elaborado para el Centro de Informatización de la Gestión de Entidades (CEIGE), el mismo tiene su basamento en los principios y buenas prácticas de las metodologías ágiles estudiadas para garantizar rapidez y un buen funcionamiento en el desarrollo de los procesos. (21)

Este se caracteriza por ser:

Centrado en la arquitectura

La arquitectura determina la línea base, los elementos de software estructurales a partir de los elementos de la arquitectura de negocio. Interviene en la gestión de cambios y diseña la evolución e integración del producto. La arquitectura orienta las prioridades del desarrollo y resuelve las necesidades tecnológicas y de soporte para el desarrollo.

Orientado a componentes

Las iteraciones son orientadas por el nivel de significancia arquitectónicas de los componentes, los mismos son abstracciones arquitectónicas de los procesos de negocio y requisitos asociados que modelan, el componente es la unidad de medición y ordenamiento de las iteraciones.

Iterativo e incremental

Las iteraciones son planificadas y coordinadas con el equipo de arquitectura, los clientes y la alta gerencia. Cada iteración constituye el desarrollo de componentes, los cuales son integrados al término de la iteración, permitiendo de esta manera la evolución incremental del producto.

Ágil y adaptable al cambio

El desarrollo de las partes formaliza solamente las características principales de la solución, priorizando los talleres y las comunicaciones entre las personas. Los clientes y funcionales están involucrados en el proyecto y poseen parte de la responsabilidad del éxito del mismo. Los cambios son conciliados semanalmente, discutidos y aprobados.

En la figura 1 se muestran las actividades de la propuesta de solución del presente trabajo.

Figura 1: Secuencia de actividades del modelo de desarrollo del software.

Artefactos que se generan en el modelo de desarrollo:

Modelación de los procesos de negocio

- Mapa de procesos.

Definición de requisitos de software

- Modelo conceptual.
- Listado de los requisitos funcionales.
- Descripción de los requisitos funcionales identificados.

Diseño de la solución

- Modelo de datos.

- Diagramas de clases del diseño.
- Diagrama de componentes.
- Diagrama de despliegue.

Métricas para validar diseño.

- Resultados de la aplicación de la métrica relación entre clases.
- Resultados de la aplicación de la métrica tamaño operacional de clase.

Pruebas para validar la implementación.

- Casos de pruebas de las pruebas de caja blanca.
- Casos de pruebas de las pruebas de caja negra.

1.5 Arquitectura de software

1.5.1 Patrón de arquitectura

Los patrones de arquitectura están relacionados a la interacción de objetos dentro o entre niveles arquitectónicos. Resuelven problemas de adaptabilidad a requerimientos cambiantes, performance, modularidad y acoplamiento. Dando solución a llamadas entre objetos (similar a los patrones de diseño), decisiones y criterios arquitectónicos y empaquetado de funcionalidades.

Modelo vista controlador (MVC)

El patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC), separa el modelado del dominio, la presentación y las acciones basadas en datos ingresados por el usuario; es decir separa en tres capas diferentes los datos de una aplicación, la interfaz de usuario, y la lógica de control:

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

Vista: Esta capa maneja la visualización de la información, es decir que presenta el modelo en un formato adecuado para interactuar, que usualmente es la interfaz de usuario.

Controlador: Esta capa controla el flujo de datos entre la vista y el modelo; es decir que responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista, tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases. (22)

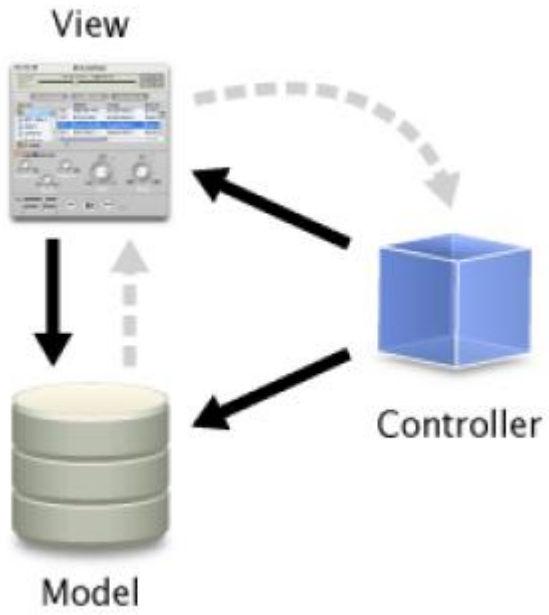

Figura 2: Modelo vista controlador.

1.5.2 Patrones de diseño

Un patrón define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones. (23)

Los patrones de diseño pretenden:

- Proporcionar catálogos de elementos reusables en el diseño de sistemas software.
- Evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente.
- Formalizar un vocabulario común entre diseñadores.
- Estandarizar el modo en que se realiza el diseño.

Patrones Grasp (Patrones generales de software para asignar responsabilidades)

Los patrones Grasp describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Existen nueve patrones Grasp los cuales son: Experto, Creador, Alta Cohesión, Bajo Acoplamiento, Controlador, Polimorfismo, Fabricación Pura, Induración y No Hables con Extraños. De estos se utilizarán Experto, Creador, Alta Cohesión, Bajo Acoplamiento y Controlador con el fin de que contribuya a que el sistema sea más robusto y flexible. (24)

Bajo Acoplamiento: Consiste en tener las clases lo menos ligadas entre sí que se pueda, de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases, potenciando la reutilización y disminuyendo la dependencia entre las clases. Mantener el bajo acoplamiento entre clases más que un patrón en una buena práctica del diseño, dado que al realizar cambios no se afectan otros componentes, es más fácil de entender de manera aislada y de esta forma se pueden reutilizar mejor las clases. (25)

Alta Cohesión: Este patrón nos dice que la información que almacena una clase debe de ser coherente y está en la mayor medida de lo posible 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 y soporta mayor capacidad de reutilización. (25)

Experto: Se pone en práctica con el uso de clases que poseen responsabilidades específicas a cumplir de acuerdo con la información que manejan.

Creador: Es útil contar con el principio general para la asignación de responsabilidades de creación porque permite que el diseño pueda soportar bajo acoplamiento, mayor claridad, encapsulación y reutilización. Las clases controladoras son responsables de crear el objeto de las modelos y estas a su vez de las entidades.

Controlador: Se utilizan cuando la aplicación es muy extensa, de esta forma, en vez de tener un solo controlador y saturarlo, se tienen clases Controllers, que son controladores más pequeños especializados en las funcionalidades de cada componente.

Patrones Gof (Gang of Four)

Los patrones Gof son patrones de diseño publicados en el libro “Design Patterns: Elements of Reusable Object-Oriented Software” por Gamma, Helm, Jonson y Vlissides conocidos mundialmente por Gang of Four o Pandilla de los cuatro. En este libro se encuentran recopilados un total de 23 patrones clasificados en patrones creacionales, estructurales y de comportamiento. Se utilizará en el diseño de la solución el patrón estructural fachada. (26)

Fachada: Provee de una interfaz unificada simple para el acceso de una interfaz o grupo de interfaces de un subsistema. Un ejemplo de su aplicación es en el uso de los servicios, donde la relación existente entre las clases controladoras y los servicios permite acceder a métodos que no están implementados en el componente y que se encuentran en otros componentes. (27)

1.6 Herramientas y tecnologías

1.6.1 Lenguajes de modelado

A lo largo de los años, durante el desarrollo de los proyectos de software ocurren bastantes confusiones y malas interpretaciones en los requerimientos de los clientes y usuarios, en parte debido a la abundancia de notaciones, metodologías y conceptos que hace que los desarrolladores de sistemas no se pongan de acuerdo en qué es lo que realmente están elaborando. En este sentido, es común encontrar muchos productos de software carentes de diseño formal y entendido sólo por quien lo ha construido.

En cualquier proyecto de ingeniería se requieren etapas de modelado que permitan experimentar y visualizar qué se construirá. Un modelo no es más que una representación simplificada de la realidad que ayuda a entender un sistema grande y complejo que no puede ser comprendido fácilmente en su totalidad.

Un lenguaje de modelado proporciona los elementos básicos con los que escribir un modelo. Estos en las empresas que se dedican a la creación de software han tomado una gran importancia debido a que el modelado es una parte central de todas las actividades que conducen a la producción de buen software.

A través del modelado se consigue:

- Visualizar cómo es que queremos que sea un sistema.
- Especificar la estructura o el comportamiento de un sistema.
- Proporcionar plantillas que guían en la construcción de un sistema.

- Documentar las decisiones que se toman.

1.6.1.1 Lenguaje unificado de modelado (UML)

UML es un lenguaje estándar para escribir planos de software, que se utiliza para visualizar, especificar, construir y documentar los artefactos de un mismo sistema que involucra una gran cantidad de software. (28)

UML sirve para el modelado completo de sistemas complejos, tanto en el diseño de los sistemas software como para la arquitectura hardware donde se ejecuten. Otro objetivo de este modelado visual es que sea independiente del lenguaje de implementación, de tal forma que los diseños realizados usando UML se puedan implementar en cualquier lenguaje que soporte las posibilidades de UML (principalmente lenguajes orientados a objetos).

UML es además un método formal de modelado. Esto aporta las siguientes ventajas:

- Mayor rigor en la especificación.
- Permite realizar una verificación y validación del modelo realizado.
- Se pueden automatizar determinados procesos y permite generar código a partir de los modelos y a la inversa (a partir del código fuente generar los modelos). Esto permite que el modelo y el código estén actualizados, con lo que siempre se puede mantener la visión en el diseño de más alto nivel, de la estructura de un proyecto. (2929)

1.6.2 Herramientas de modelado

En los últimos años el desarrollo del software ha propiciado que surjan herramientas que permitan a los analistas y diseñadores de software realizar estos procesos de forma eficiente, con mayor calidad y fiabilidad. Las herramientas CASE (Computer Aided Software Engineering, Ingeniería de Software Asistida por Ordenador) son un ejemplo de ello.

Las herramientas CASE son un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (30)

En la actualidad existe una gran cantidad de herramientas CASE. Entre las más utilizadas se encuentran Platinum Erwin, EasyCASE, Oracle Designer, System Architect y Visual Paradigm para UML. Para la realización de la propuesta de solución del trabajo se va a utilizar Visual Paradigm para UML ya que este soporta el ciclo de vida completo del desarrollo de software, ayuda a una más rápida construcción de aplicaciones de calidad, permite dibujar todos los tipos de diagramas de clases, generar código desde diagramas entre otras funcionalidades.

1.6.2.1 Visual Paradigm para UML

Visual Paradigm para UML (VP-UML) es una herramienta UML de diseño UML y herramienta CASE diseñada para la ayuda al desarrollo de software. VP-UML soporta estándares claves de la industria, tales como Lenguaje de Modelado Unificado (UML), SysML, BPMN y XMI. Ofrece un completo conjunto de herramientas de los equipos de desarrollo de software necesario para la captura de requisitos, la planificación de programas, la planificación de controles, modelado de clase y modelado de datos. (31)

El uso de esta herramienta trae consigo grandes ventajas, ya que esta soporta el ciclo de vida completo del desarrollo de software, ayuda a una más rápida construcción de aplicaciones de calidad y permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (31)

Es una herramienta colaborativa porque soporta a varios usuarios trabajando en un mismo proyecto, genera la documentación del proyecto automáticamente en varios formatos como son Web o PDF y permite control de versiones. Brinda la posibilidad de generar código a partir de los diagramas, para plataformas como .Net, Java y PHP, así como obtener diagramas a partir de código. Esta es precisamente una gran ventaja puesto que el sistema al que se debe integrar la propuesta de solución está desarrollado en PHP. (31)

VP-UML es multiplataforma, lo cual le permite al usuario utilizar esta herramienta en varios sistemas operativos como Windows, Linux, Unix y otros; además se encuentra disponible en distintas versiones: Enterprise, Professional, Standard, Modeler, Personal y Community. (31)

1.6.3 Lenguajes de programación y frameworks

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Por lo tanto, un lenguaje de programación es un modo práctico para que los seres humanos puedan dar instrucciones a un equipo. (32)

El término framework se refiere a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Entre los objetivos principales que se persigue con su uso están: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. (32)

Para la realización de la propuesta de solución del trabajo se utilizarán los lenguajes de programación y framework definidos en el marco de trabajo de Sauxe. Se utilizarán PHP y JavaScript como lenguajes de programación y los frameworks ExtJS, Doctrine y Zend Framework.

1.6.3.1 JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. Gran parte de su programación está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas, entre otros. Permite la programación de pequeños scripts y de programas más grandes orientados a objetos, con funciones y estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web para poder acceder a ellos y modificarlos dinámicamente. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado, es soportado por Internet Explorer, Netscape, Opera, Mozilla Firefox, entre otros. (33)

1.6.3.2 ExtJS

ExtJS es un conjunto de librerías JavaScript que permite el desarrollo de aplicaciones RIA (Aplicaciones Ricas de Internet) basadas en un navegador. Ofrece al desarrollador un gran conjunto de widgets (componentes como por ejemplo grids, ventanas de diálogo) plenamente integrados y un API (interfaz de programación de aplicaciones) para conseguir interfaces web más dinámicas e interactivas con el usuario.

ExtJS usa el lenguaje JavaScript junto con HTML para la creación de las interfaces de usuario, así como para el manejo de eventos en cada una de las páginas que comprenden una aplicación desarrollada con ExtJS. (34)

Al usar JavaScript, ExtJS trae consigo los beneficios propios del lenguaje tal como:

- La orientación a objetos.
- La manipulación del DOM (ExtJS extiende esta capacidad con su propia implementación para el manejo de DOM).
- El soporte de múltiples navegadores como Internet Explorer, Opera, Safari y Mozilla Firefox. (35)

Esta librería incluye: (35)

- Componentes UI de alto performance y personalizables.
- Modelo de componentes extensibles.
- Un API (interfaz de programación de aplicaciones) fácil de usar.
- Licencias de código abierto y comercial.

La versión 2.2 que es la que se va a utilizar, tiene una alta capacidad de soporte de navegadores, mantiene la misma apariencia independientemente del navegador y la plataforma del cliente. Se integra con aplicaciones desarrolladas utilizando la especificación JEE o con aplicaciones desarrolladas con .Net o PHP.

1.6.3.3 PHP

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran librería de funciones y mucha documentación.

PHP es un lenguaje de script, diseñado para, entre otras cosas, aumentar e incrementar el dinamismo de las páginas web. Sus características han ido creciendo hasta convertirse en un lenguaje de programación completo, capaz de manejar entornos que integran grandes bases de datos. Su popularidad se basa, en gran parte, en su sintaxis similar a la del lenguaje de programación C, su rapidez y simplicidad. (36)

Entre las características que posee este lenguaje y que lo convierten en una potente herramienta están:

- Es un software de código abierto.
- Soporta muchas bases de datos entre las que se encuentran (MySQL y PostgreSQL).
- Integración con varias bibliotecas externas, permite generar documentos en PDF (documentos de Acrobat Reader) hasta analizar código XML.
- Ofrece una solución simple y universal para las paginaciones dinámicas de Web de fácil programación.

1.6.3.4 Doctrine

El Proyecto Doctrine es la casa de un conjunto seleccionado de librerías PHP, se centra principalmente en la prestación de servicios de persistencia y funcionalidad.

El mapeador objeto relacional (ORM) para PHP se encuentra en la parte superior de una capa de abstracción de base de datos de gran alcance (DBAL). Una de sus principales características es la posibilidad de escribir consultas de base de datos en un dialecto orientado a objetos de propiedad SQL llamada Doctrina (Lenguaje de consulta DQL), inspirado en hibernates HQL. Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad sin necesidad de duplicar código innecesario. La potente capa de abstracción de base de datos tiene muchas características de base de datos de la introspección de esquema, la gestión de esquema y la abstracción DOP. (37)

1.6.3.5 Zend Framework

Zend Framework se trata de un framework de código abierto para desarrollo de aplicaciones y servicios Web con PHP5. Utiliza código 100% orientado a objetos y brinda soluciones para construir sitios web modernos, robustos y seguros. La estructura de los componentes de Zend Framework es única; cada componente está construido con una baja dependencia de otros componentes. Esta arquitectura, débilmente acoplada, permite a los desarrolladores utilizar los componentes por separado. (38)

Aunque se pueden utilizar de forma individual, los componentes de la biblioteca estándar de Zend Framework conforman un potente y extensible framework de aplicaciones web al combinarse. Además, ofrecen gran rendimiento y robusta implementación modelo-vista-controlador (MVC), abstracción de base de datos fácil de usar y un componente de formularios que implementa: la prestación de formularios HTML

y validación y filtrado para que los desarrolladores puedan consolidar todas las operaciones usando de una manera sencilla la interfaz orientada a objetos. (38)

1.6.4 Gestor de base de datos

En un ERP es imprescindible contar con una base de datos para la persistencia de los datos. Por lo que es necesario el uso de una herramienta que garantice la gestión de la misma. Para esto se hará uso de un sistema gestor de base de datos.

Un Sistema Gestor de Base de Datos (SGBD) es un conjunto de programas que permiten crear y mantener una base de datos, asegurando su integridad, confidencialidad y seguridad. (39)

Por tanto debe permitir:(39)

- Definir una base de datos: especificar tipos, estructuras y restricciones de datos.
- Construir la base de datos: guardar los datos en algún medio controlado por el mismo SGBD.
- Manipular la base de datos: realizar consultas, actualizarla, generar informes.

Dentro de los numerosos SGBD que se utilizan en la actualidad para la gestión de bases de datos se encuentra PostgreSQL.

1.6.4.1 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, desarrollado en la Universidad de California en Berkeley del Departamento de Ciencias de la Computación. PostgreSQL fue pionera en muchos conceptos que sólo estuvo disponible en algunos sistemas de bases de datos comerciales mucho más tarde.

PostgreSQL es un descendiente de código abierto del código original de Berkeley. Soporta una gran parte del estándar SQL y ofrece muchas características modernas:

- consultas complejas
- claves externas
- desencadenantes
- puntos de vista
- la integridad transaccional

- control de concurrencia multiversión

Debido a la licencia de liberales, PostgreSQL puede ser usado, modificado y distribuido por cualquier persona de forma gratuita para cualquier propósito, ya sea privado, comercial o académico. (40)

1.6.5 Herramientas de base de datos

1.6.5.1 EMS SQL Manager

EMS SQL Manager para PostgreSQL es una herramienta de alto rendimiento para la administración y el desarrollo de bases de datos PostgreSQL. Funciona con cualquier versión de PostgreSQL hasta la más reciente y soporta sus últimas características. (41)

1.6.5.2 pgAdmin III

Es la herramienta Open Source de administración por excelencia para sus bases de datos PostgreSQL. Algunas de sus características son: el soporte completo para UNICODE, edición rápida de consultas y datos multihilo y soporte para todos los tipos de objetos de PostgreSQL.

Incluye una interfaz gráfica de administración, una herramienta para el trabajo con SQL, un editor de código de procedimientos y funciones, y mucho más. pgadmin III está diseñado para darle respuesta de las necesidades de la mayoría de los usuarios, desde la escritura de consultas simples en SQL hasta el desarrollo de bases de datos complejas. La interfaz gráfica soporta todas las características presentes de PostgreSQL y se puede hacer la administración fácilmente. Está disponible en más de 30 lenguajes y para varios sistemas operativos. (42)

1.6.6 Navegador Web

La propuesta de solución al problema expuesto será realizada como una aplicación web, por lo que es necesario contar con un navegador para interactuar con su contenido. Un navegador o navegador web es un programa que permite ver la información que contiene una página web. El navegador interpreta el código en el que está escrita la página web y lo presenta en pantalla permitiendo al usuario interactuar con su contenido y navegar hacia otros lugares mediante enlaces o hipervínculos.

1.6.6.1 Mozilla Firefox

Se utilizará Mozilla Firefox ya que este es un navegador de software libre. Entre sus características se encuentran que presenta una forma rápida y eficiente de navegar por la web y que permite abrir varias páginas en una misma ventana mediante el empleo de pestañas separadas.

1.6.7 Control de versiones

El uso de un control de versiones para el desarrollo del sistema es de vital importancia, debido a que el sistema es programado a la vez por diferentes desarrolladores, por lo cual es necesario contar con una herramienta que guarde todos los cambios hechos a los ficheros y directorios.

Se llama control de versiones a los métodos y herramientas disponibles para controlar todo lo referente a los cambios en el tiempo de un archivo.

Difícilmente un archivo de código o un documento de texto están terminados con la primera escritura; necesita cambios o reescrituras para corregir errores, modificar su contenido, a medida que el documento cambia existen dos opciones, mantener un historial de cambios o dejar que evolucione sin memoria. El control de versiones es un método estándar para mantener esta memoria, haciendo además que sea útil para el desarrollo futuro. (43)

1.6.7.1 Subversion

Subversion es un sistema de control de versiones libre y de código fuente abierto, es decir, maneja ficheros y directorios a través del tiempo. Tiene un árbol de ficheros en un repositorio central. El repositorio es como un servidor de ficheros ordinario, excepto porque recuerda todos los cambios hechos a sus ficheros y directorios. Esto le permite recuperar versiones antiguas de sus datos, o examinar el historial de cambios de los mismos. Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos ordenadores. (43)

Entre las características que tienen estos sistemas están:

- Entendimiento con los lenguajes de programación.
- Suministro de herramientas para la construcción de software.
- Pueden ser usados para administrar cualquier conjunto de ficheros.

1.6.7.2 TortoiseSVN

TortoiseSVN es un cliente gratuito de código abierto para el sistema de control de versiones Subversion. Maneja ficheros y directorios a lo largo del tiempo. Es un sistema general que puede ser utilizado para manejar cualquier colección de ficheros, incluyendo código fuente. (44)

1.7 Conclusiones del capítulo

En este capítulo se trataron conceptos muy importantes con el objetivo de tener un mejor entendimiento del negocio. Se realizó un estudio de algunos de los sistemas existentes en la actualidad, teniendo en cuenta sus interfaces y funcionalidades, así como los inconvenientes y buenos enfoques que presentan que nos ayudaran a elaborar una mejor propuesta de solución. Por último se presentan las herramientas y lenguajes a utilizar, los cuales contribuirán al desarrollo de la solución que se quiere construir. Al concluir este capítulo llegamos a la conclusión de que se puede comenzar a desarrollar la propuesta de solución del trabajo.

Capítulo 2: Diseño e implementación

Introducción

En este capítulo se describe la solución teniendo en cuenta los requisitos funcionales y no funcionales, incluye una muestra del prototipo interfaz y el diseño de clases. Además se delimitan varios puntos importantes en el desarrollo de la solución como la descripción de las principales clases utilizadas en la implementación del mismo y la descripción de los estándares de codificación para una mejor legibilidad del código.

2.1 Requisitos

Los requisitos son características que los sistemas deben tener para cubrir las necesidades que lo motivan. Son útiles, ya que denotan una condición o capacidad que necesita un usuario para resolver un problema o lograr un objetivo. Los requisitos funcionales, no funcionales y de interfaz se abordan a continuación.

2.1.1 Funcionales

Los requisitos funcionales que fueron entregados al autor para realizar el diseño y la implementación son los siguientes:

- ✓ R1. Gestionar empresa.
  - R1.1. Adicionar empresa.
  - R1.2. Modificar empresa.
  - R1.3. Eliminar empresa.
- ✓ R2. Gestionar producto.
  - R2.1Adicionar producto.
  - R2.2.Modificar producto.
  - R2.3. Eliminar producto.
- ✓ R3. Gestionar proveedor.

- R3.1. Adicionar proveedor.
- R3.2. Actualizar proveedor.
- R3.3. Buscar proveedor.

✓ R4. Gestionar contrato.

- R4.1Adicionar contrato.
- R4.2. Actualizar contrato.
- R4.3. Eliminar contrato.

✓ R5. Generar reportes de contratos consolidados.

✓ R6. Generar reportes de contratos por proveedor.

✓ R7. Generar reportes de embarques desglosados.

✓ R8. Generar reportes de contratos por carta de crédito.

✓ R9. Generar reportes de contratos consolidados por empresa.

✓ R10. Generar reportes de proveedores.

2.2.1 No funcionales

Usabilidad (USB)

- La solución podrá ser ejecutada por cualquier persona que posea conocimientos básicos en el manejo de la computadora.
- La navegación será intuitiva.
- Las acciones no estarán a más de 3 clics del formulario inicial.
- La aplicación estará disponible las 24 horas del día.

Rendimiento del sistema (REN)

- La aplicación debe requerir un consumo mínimo de recursos.
- La aplicación debe ser ágil al brindarle respuesta a las instrucciones dadas por el usuario.

Requerimientos de apariencia e interfaz externa

- La aplicación debe poseer una interfaz amigable al usuario, basada en paneles con formularios, con una navegabilidad intuitiva para el usuario.

Seguridad (SEG)

- La autenticación será gestionada por el subsistema de seguridad del marco de trabajo Sauxe (contraseña de acceso).
- Proteger la información manejada por el sistema de accesos no autorizados.
- Garantizar que las funcionalidades del sistema se muestren de acuerdo al nivel del usuario que esté activo.
- Verificación sobre acciones irreversibles (eliminaciones).

Portabilidad (POR)

- La solución es multiplataforma, haciendo énfasis en Linux y Windows.
- Navegador Mozilla Firefox.

Soporte (SOP)

- La solución contará, antes de su puesta en marcha, con un período de pruebas.

Políticos – Culturales (CUL)

- La solución solo podrá ser utilizada por las entidades autorizadas del Ministerio de Informática y Comunicaciones (MIC).
- La solución solo debe contener palabras en inglés o español en dependencia del lenguaje seleccionado.
- La solución debe respetar los términos empleados normalmente por los especialistas en el tema de la esfera que se informatiza.

Legales (LEG)

- La solución está avalada por documentos rectores emitidos en el país para la certificación y validación de los sistemas contables.

Software (SFT)

Requerimientos mínimos tecnológicos de software.

Para el cliente:

- Sistema operativo Windows 98 o superior, o Linux.
- Navegador Web Mozilla Firefox 10.0 o superior.

Para el servidor:

- Sistema operativo Windows Advancer Server (2000 o superior) o Linux en cualquiera de sus distribuciones.
- Un servidor Apache 2.0 o superior con módulo PHP 5.0 disponible. Este debe estar configurado con la extensión “pgsql” incluida.
- Un servidor de base de datos PostgreSQL 8.0 o superior.

Hardware (HDW)

Para el cliente:

- Requerimientos mínimos: Procesador Pentium II a 133Mhz con 128 Mb de memoria RAM.
- Tarjeta de red.
- Impresora

Para el servidor:

- Requerimientos mínimos: Procesador Pentium III a 1GHz de velocidad de procesamiento y 1Gb de memoria RAM.
- Al menos 40Gb de espacio libre en disco duro.
- Tarjeta de red.

2.2 Prototipo de interfaz

Se utilizan los prototipos de interfaz para representar las funcionalidades del componente y validarlas con los clientes. En la parte superior estará el nombre de la tarea en ejecución. El color predominante es el azul, el cual permite muy buena visibilidad en las composiciones gráficas en general y en las aplicaciones

web en concreto. Es utilizado ampliamente como color corporativo, por la seriedad y confianza que inspira y admite buenas gradaciones, pudiendo ser el color predominante en una página.

El diseño simétrico de la interfaz sugiere estabilidad y equilibrio, resultando estético, ordenado, atractivo y agradable de contemplar. La simetría usada no es del todo exacta, ya que una simetría demasiado perfecta hace parecer las composiciones artificiales y premeditadas. Pequeñas variaciones en la distribución simétrica dan ese toque de ruptura que hace su contemplación más amena y natural.

Prototipo de interfaz del requisito: R1. Gestionar empresa.

En esta interfaz se mostrará en forma de árbol todas las empresas creadas, permitiendo así, las opciones de agregar una nueva empresa y modificar o eliminar alguna existente; será necesario seleccionar previamente en el árbol de empresas en el nivel donde desea adicionar la empresa (R1.1) o en los otros dos casos, seleccionar la que desea modificar (R1.2) o eliminar (R1.3).

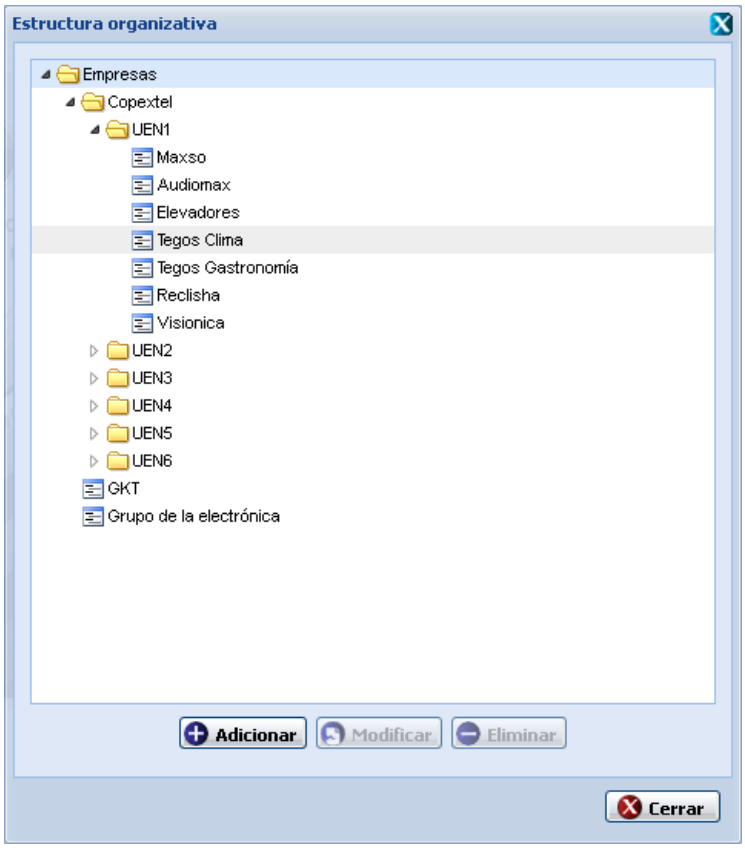

Figura 3: Prototipo de interfaz: R1. Gestionar empresa.

Prototipo de interfaz del requisito: R2. Gestionar producto.

En esta interfaz se muestra una tabla con todas las familias de productos existentes, y en dependencia de la familia que se seleccione en esta se mostrará en la tabla de productos los productos que pertenecen a esta familia. Permite las opciones de adicionar, modificar o eliminar familias de productos y productos. Para adicionar un nuevo producto será necesario seleccionar la familia de productos a la que va a pertenecer. Si desea modificar o eliminar una familia o producto existente será necesario seleccionar previamente en la tabla donde se muestra lo que desea modificar o eliminar.

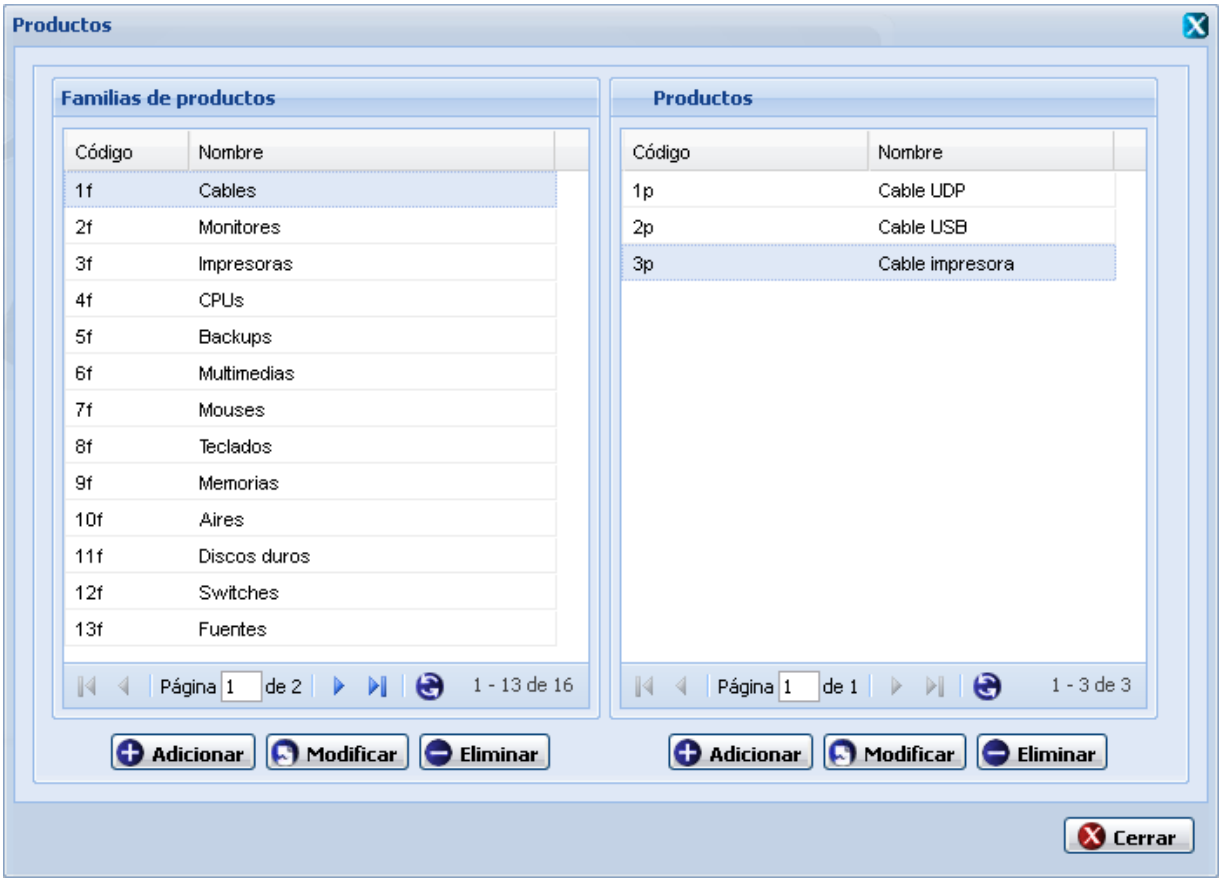

Figura 4: Prototipo de interfaz. R2. Gestionar producto.

Prototipo de interfaz del requisito: R3.Gestionar proveedor.

En esta interfaz se mostrará en dependencia de la selección los proveedores creados, permitiendo así, las opciones de agregar un nuevo proveedor o actualizar alguno existente; será necesario seleccionarlo previamente.

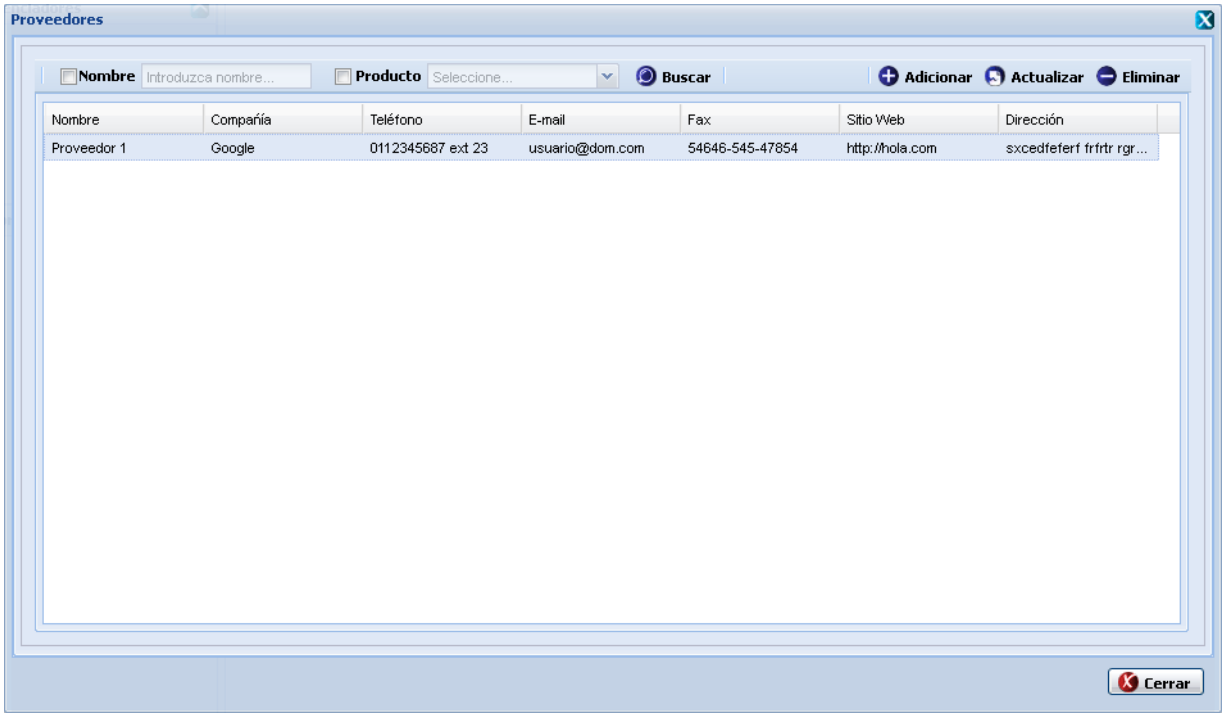

Figura 5: Prototipo de interfaz. R3. Gestionar proveedor.

Prototipo de interfaz del requisito: R4. Gestionar contrato.

En esta interfaz se mostrará en dependencia de la selección los contratos creados, permitiendo así, las opciones de agregar un nuevo contrato o actualizar alguno existente; será necesario seleccionarlo previamente.

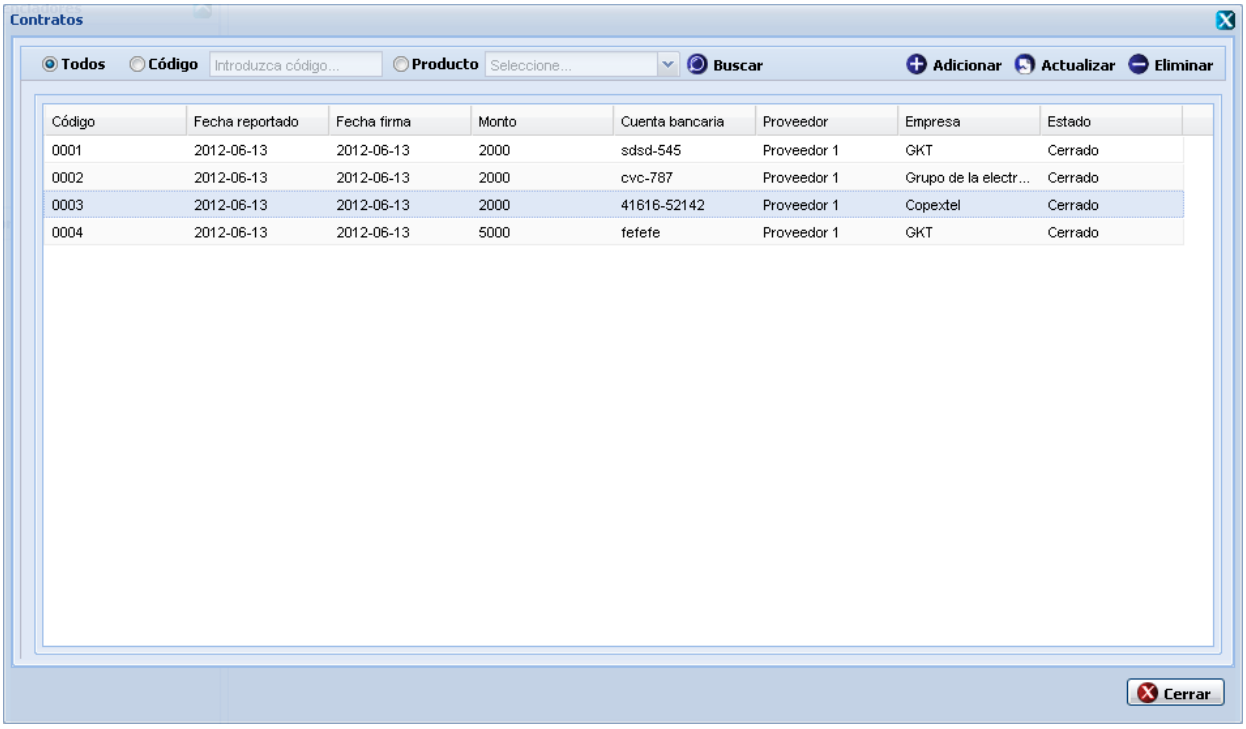

Figura 6: Prototipo de interfaz. R3. Gestionar contrato.

2.3 Arquitectura

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores, programadores, etc. trabajen en una línea común que permita alcanzar los objetivos del sistema de información, cubriendo todas las necesidades.

2.3.1 Estilo híbrido basado en Capas y MVC

El estilo seleccionado para el desarrollo de la arquitectura, es un estilo en capas. Está compuesto básicamente por tres niveles o capas, este presenta algunas ventajas como son:

- Desarrollos paralelos (en cada capa)
- Aplicaciones más robustas debido al encapsulamiento

- Mantenimiento y soporte más sencillo (es más sencillo cambiar un componente que modificar una aplicación monolítica)
- Mayor flexibilidad (se pueden añadir nuevos módulos para dotar al sistema de nuevas funcionalidades)
- Alta escalabilidad. La principal ventaja de una aplicación distribuida bien diseñada es su buen escalado, es decir, que puede manejar muchas peticiones con el mismo rendimiento simplemente añadiendo más hardware. El crecimiento es casi lineal y no es necesario añadir más código para conseguir esta escalabilidad.

Capa Datos: En esta capa se encuentra como servidor de base de datos PostgreSQL.

Capa Acceso a Datos: En esta capa está presente el framework Doctrine, como persistidor para la comunicación con el servidor de datos mediante el protocolo PDO.

Capa Funcionamiento: En esta capa se emplea el patrón Modelo Vista Controlador (MVC), este es uno de los patrones de diseño de arquitectura de software que más se utiliza en la construcción de aplicaciones web, el cual permite la separación del código entre cada una de las capas, ayudando tanto a desarrolladores como a diseñadores a cooperar y mantener el código fuente más fácilmente, además permite que el software sea fácil de modificar.

2.3.2 Estructura del patrón MVC

El **Modelo** es el objeto que representa los datos del programa. Maneja los datos y controla todas sus transformaciones. El Modelo no tiene conocimiento específico de los Controladores o de las Vistas, ni siquiera contiene referencias a ellos. Es el propio sistema el que tiene encomendada la responsabilidad de mantener enlaces entre el Modelo y sus Vistas, y notificar a las Vistas cuando cambia el Modelo. (45)

La **Vista** es el objeto que maneja la presentación visual de los datos representados por el Modelo. Genera una representación visual del Modelo y muestra los datos al usuario. Interactúa con el Modelo a través de una referencia al propio Modelo.

El **Controlador** es el objeto que proporciona significado a las órdenes del usuario, actuando sobre los datos representados por el Modelo. Cuando se realiza algún cambio, entra en acción, bien sea por

cambios en la información del Modelo o por alteraciones de la Vista. Interactúa con el Modelo a través de una referencia al propio Modelo.

2.4 Estrategia de Integración

Entre los componentes se aplica la integración vertical que consiste en como fluyen los datos desde la Vista hacia la capa de datos y viceversa, pasando por los diferentes elementos que componen la arquitectura. Esta consta de cuatros nodos de integración, el que se encuentra entre la Vista y el Controlador, el que está entre el Controlador y el Modelo, el que vincula el Modelo con el marco de trabajo doctrine y el que se encuentra entre el doctrine y la base de datos.

Cada componente tiene su registro de los datos de los módulos en un fichero XML que será mapeado por el framework para el funcionamiento del mismo, dicho fichero tiene por nombre Inversión de control (IOC). La comunicación entre los diferentes módulos y componentes se realiza mediante llamadas a la inversión de control, este registra las funcionalidades que ofrecen los métodos de las clases modelos y entidades del sistema y especifica respuestas deseadas a sucesos o solicitudes de datos concretas, dejando que otro módulo o componente lleve a cabo las acciones de control que se requieran en el orden necesario.

Vista – Controlador: Los datos recogidos en un formulario son enviados al controlador haciendo uso del protocolo de comunicación HTTP a través del método “post” para ser procesados y los resultados son enviados por el controlador a la vista en un JSON a través del método “echo”.

Controlador – Modelo: El controlador toma los datos recibidos desde la vista, instancia una determinada clase del modelo y llama a uno de sus métodos, pasándole como parámetros los datos recibidos.

Modelo – Doctrine: El modelo utiliza llamadas a métodos de doctrine que le permitan crear, modificar, eliminar o actualizar los datos almacenados en las tuplas de la base de datos.

Doctrine – Base de Datos: Doctrine ejecuta las consultas a la Base de Datos utilizando programación orientada a objetos.

2.5 Estándares de código seleccionados

Un estándar de codificación comprende todos los aspectos de la generación de código. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez. Los estándares de codificación permiten una mejor integración entre las líneas de

producción y establece pautas que conllevan a lograr un código más legible y reutilizable, de tal forma que se pueda aumentar su capacidad de mantenimiento a lo largo del tiempo.

2.5.1 Nomenclatura de las clases

Los nombres de las clases comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará notación PascalCasing. Con sólo leerlo se reconoce el propósito de la misma.

Ejemplo: Proveedor

2.5.1.1 Nomenclatura según el tipo de clases

Clases controladoras: Las clases controladoras después del nombre llevan la palabra: “Controller”.

- Controllers (Controladoras)

Ejemplo: ContratacionController

Clases de los modelos:

- Business (Negocio)

Las clases que se encuentran dentro de Business después del nombre llevan la palabra: “Model”. Ejemplo: ContratoModel

- Domain (Dominio)

Las clases que se encuentran dentro de Domain el nombre que reciben es el de la tabla en la Base de Datos. Ejemplo: Empresa

- Generated (Dominio bases)

Las clases que se encuentran dentro de Generated el nombre comienza con la palabra: “Dat” y seguido el nombre de la tabla en la Base de Datos.

Ejemplo: DatProveedor

2.5.2 Nomenclatura de las funciones

El nombre a emplear para las funciones se escribe con la primera palabra en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCasing, y con sólo leerlo se reconoce el propósito de la misma. Ejemplo: adicionarEmpresa En caso de ser una acción de la clase controladora se le pone el nombre y seguida la palabra: “Action”

Ejemplo: adicionarEmpresaAction

2.5.3 Nomenclatura de los comentarios

Los comentarios deben ser lo bastante claros y precisos de forma tal que se entienda el propósito de lo que se está desarrollando. En caso de ser una función complicada se comentaría dentro de la misma para lograr una mejor comprensión del código.

Ejemplo: //adicionamos un contrato

2.6 Diagrama de clases del diseño

Un diagrama de clases sirve para visualizar las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, de uso y de agregación, ya que una clase es una descripción de conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica; mostrando un conjunto de elementos que son estáticos, como las clases y tipos junto con sus contenidos y relaciones. Un diagrama de clases está compuesto por los siguientes elementos:

Clase: atributos, métodos y visibilidad.
Relaciones: Herencia, Composición, Agregación, Asociación y Uso. (46)

En la figura 7 se muestra el diagrama de clases propuesto, que fue concebido teniendo en cuenta las características del patrón arquitectónico Modelo-Vista-Controlador (MVC).

El diagrama muestra como las páginas clientes Contratacion y Producto envían los datos a las clases controladoras, las clases controladoras ContratacionController y ProductoController son las encargadas de capturar la información proveniente de las vistas y envían esta información a las clases del modelo ContratoModel y ProductoModel que son las que realizan toda la lógica del sistema.

En él se evidencia el uso de patrones Grasp como por ejemplo el Controlador ya que en vez de tener un sólo controlador y saturarlo, se tienen clases Controllers, que son controladores más pequeños especializados en las funcionalidades de cada componente y el creador ya que las clases controladoras son responsables de crear el objeto de las modelos y estas a su vez de las entidades.

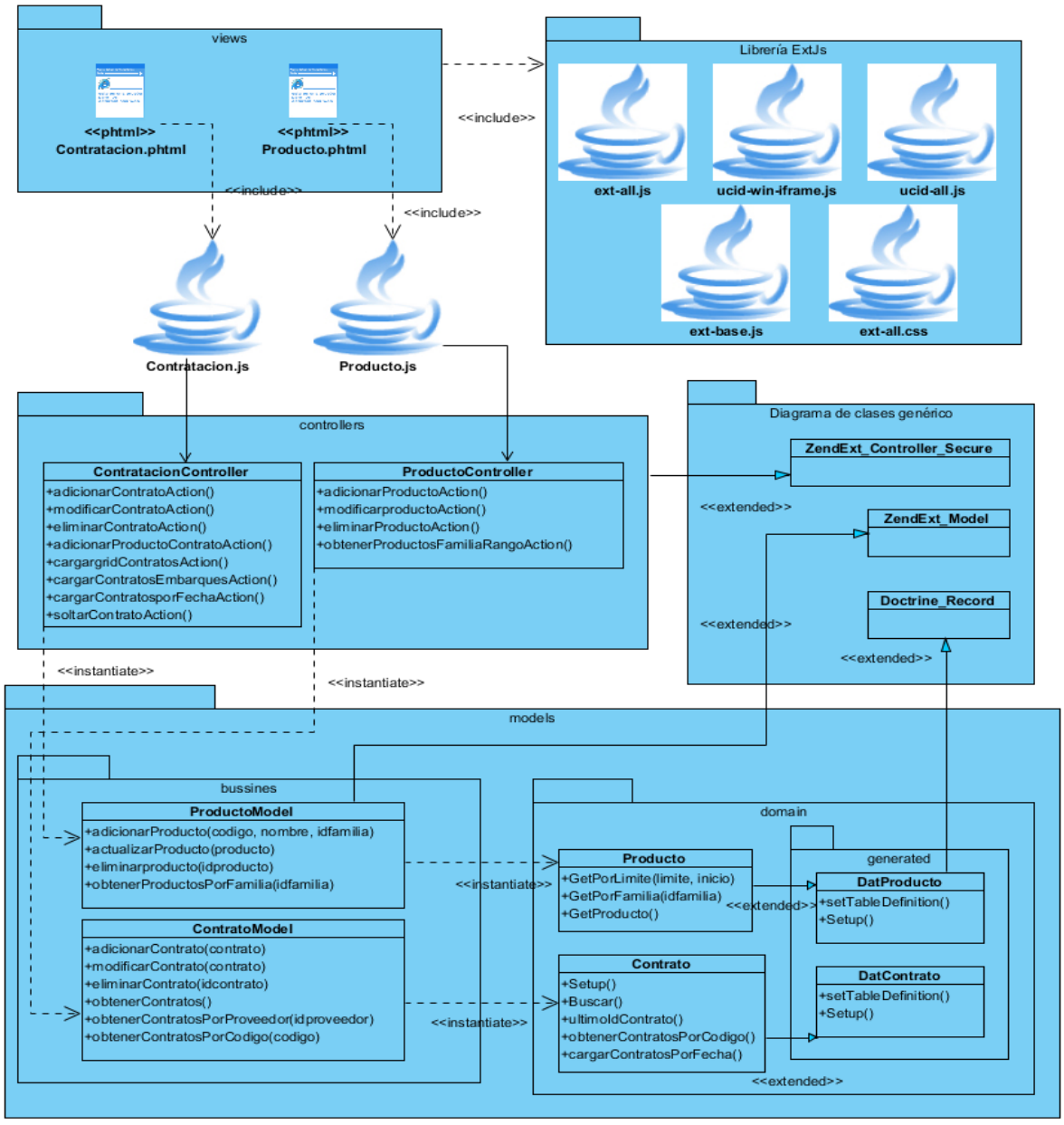

Figura 7: Diagrama de clases del diseño requisitos: Gestionar producto y Gestionar contrato.

2.7 Estructura dentro del marco de trabajo

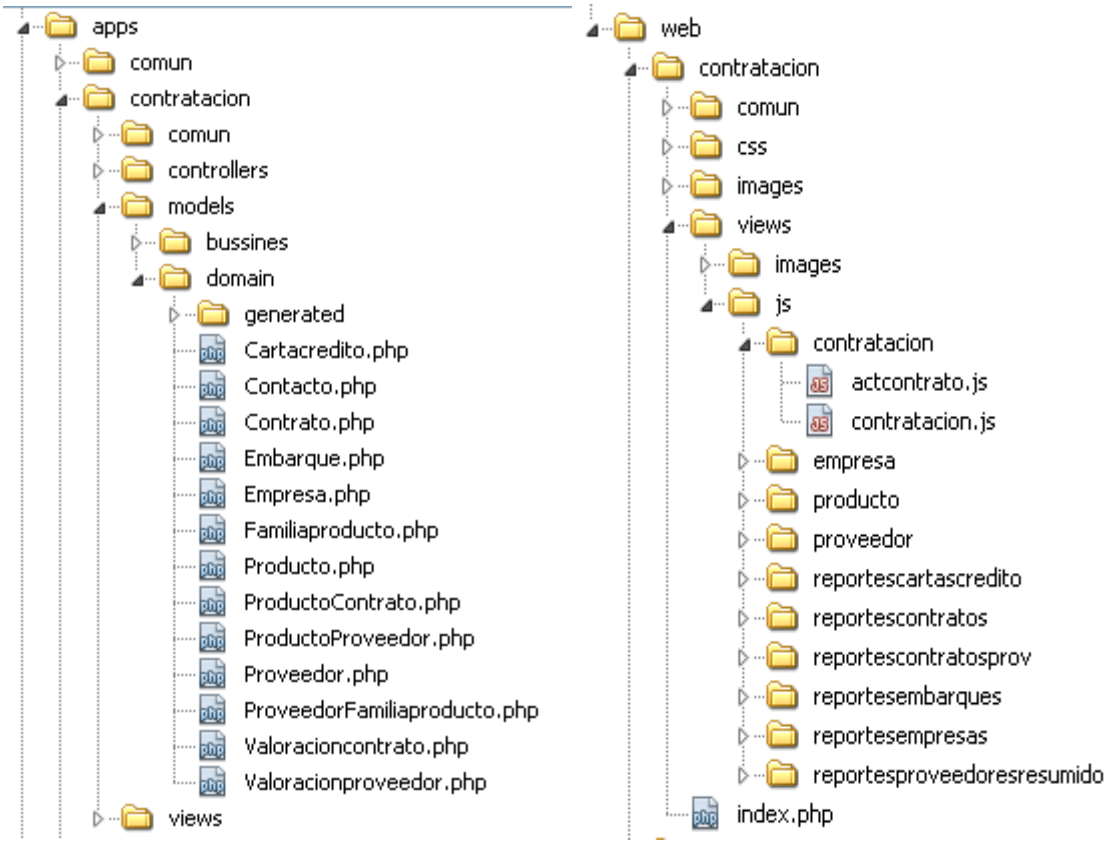

Figura 8: Estructura dentro del marco de trabajo Sauxe.

En la figura 8se muestra cómo quedarán las clases dentro del marco de trabajo Sauxe para una mejor comprensión de su estructura a la hora de implementar la solución. En la dirección apps/contratación/model/ se encentrarán las clases generadas con el mapeador Doctrine. En la carpeta apps/contratacion/controllers se encontrarán las clases controladoras del sistema y en la carpeta web/contratación/views/js se encontrarán los scripts que contendrán las vistas con la que interactuará el usuario.

2.8 Modelo de datos

Un modelo de datos es el que describe la representación de las clases persistentes de la base de datos. Básicamente consiste en una descripción de algo conocido como tablas de la base de datos (donde se guarda la información), así como los métodos a almacenar y recuperar información de dichas tablas.

Consiste en:

- **Objetos** (entidades que existen y que se manipulan).
- **Atributos** (características básicas de estos objetos).
- **Relaciones** (forma en que se enlazan los distintos objetos entre sí).

En la figura 9 se muestra el modelo de datos físico resultante de las tablas de la base de datos con la que trabaja el componente contratación.

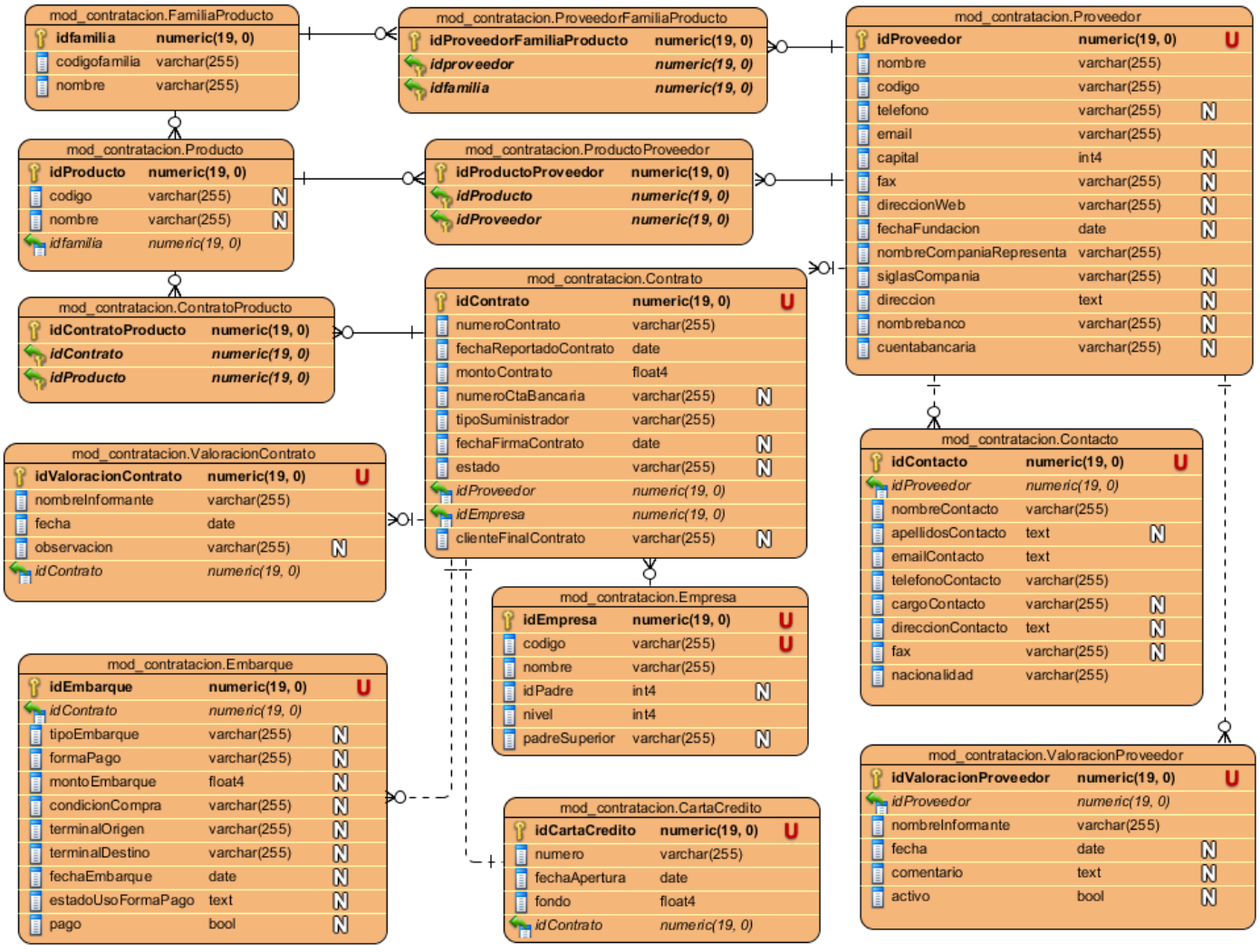

Figura 9: Modelo de datos.

2.9 Conclusiones del capítulo

Entre los aspectos fundamentales incluidos en el proceso se mostraron los artefactos construidos en cada una de las etapas desarrolladas. Se expusieron los mecanismos y patrones de diseño utilizados, también se elaboraron artefactos como los diagramas de clases de diseño, que permitieron desarrollar de manera más eficiente la implementación del proceso de contratación. Se conformó el modelo de datos y fueron explicadas de forma breve algunas de las funcionalidades que tendrá el sistema y los estándares de codificación a seguir. Por lo que podemos llegar a la conclusión de que se puede comenzar a construir la aplicación, tratando de que se cumplan todos los requisitos y las funciones que se han considerado necesarias en este capítulo.

Capítulo 3: Validación y pruebas

3.1. Introducción

La dificultad de construir grandes sistemas de software multiplica la probabilidad de que existan errores aún después de haberse finalizado su construcción. Es imposible asegurar que un software se encuentre completamente libre de errores; sin embargo, existen formas y métodos para acercarse lo más posible a este resultado. En este capítulo se realiza la validación de la solución propuesta. Se evalúa el diseño empleando métricas de software que proporcionan una medida de la complejidad y calidad del software. Se aplican pruebas con el objetivo de verificar la funcionalidad y estructura del sistema desarrollado.

3.2. Métricas para evaluar el diseño propuesto

La realización de métricas son claves en la ingeniería del software orientada a objetos. Estas permiten tener una visión más clara y profunda y proporcionan un mecanismo para la evaluación de la calidad del software.

Los objetivos principales de las métricas orientadas a objetos son:

- Comprender mejor la calidad del producto.
- Estimar la efectividad del proceso.
- Mejorar la calidad del trabajo realizado.

Las métricas que se utilizaron para validar el diseño propuesto aplican los principales atributos de calidad de software, inspiradas en lo propuesto por Pressman (47), y las métricas de tamaño operacional de clase por Lorenz y Kidd (48). Las métricas orientadas a tamaños para una clase se centran en cálculos de atributos y de operaciones para una clase individual y promedian los valores para el sistema en su totalidad. La métrica acoplamiento de clase sugiere que cuanto más acoplamiento se da en una clase, será más difícil de reutilizar. Además, las clases con excesivo acoplamiento dificultan la comprensibilidad y hacen más difícil el mantenimiento por lo que será necesario un mayor esfuerzo.

Atributos de calidad que se abarcan:

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

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

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

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

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 concebidas como instrumento para evaluar la calidad del diseño y su relación con los atributos de calidad definidos son la del Tamaño Operacional de Clase (TOC) y la métrica de Relaciones entre Clases (RC).

Estas métricas son bastante eficientes ya que dan una medida de la calidad del diseño del componente y su utilización es sencilla y fácil. En las tablas 1, 2 y 3 se demuestra la facilidad de uso de la métrica de tamaño operacional de clase, y para la métrica RC es de una manera similar, lo que se tiene en cuenta el número de relaciones de uso de una clase con otras.

3.2.1 Tamaño Operacional de Clase (TOC)

El tamaño operacional de clase (TOC), está dado por el número de métodos asignados a una clase.

Atributo que afecta	Modo en que lo enfoca
Responsabilidad	Un aumento del TOC implica un aumento de la responsabilidad asignada a la clase.
Complejidad de implementación	Un aumento del TOC implica un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC implica una disminución en el grado de reutilización de la clase.

Tabla 1: Tamaño operacional de clase (TOC).

Atributos de calidad	Categorías	Criterio
Responsabilidad	Baja	<= Prom.(9.1)
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2* Prom.
Complejidad implementación	Baja	<= Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2* Prom.
	Media	Entre Prom. y 2 * Prom.
	Alta	<= Prom.

Tabla 2: Rango de valores para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

Tamaño operacional de la clase	Criterio
Pequeña	<= Prom.(9.1)
Media	Entre Prom. y 2 * Prom.
Grande	> 2* Prom.

Tabla 3: Umbrales para el TOC.

Los resultados del instrumento de evaluación de la métrica Tamaño Operacional de Clase (TOC) se muestran en las gráficas que se presentan en las figuras 10, 11, 12 y 13.

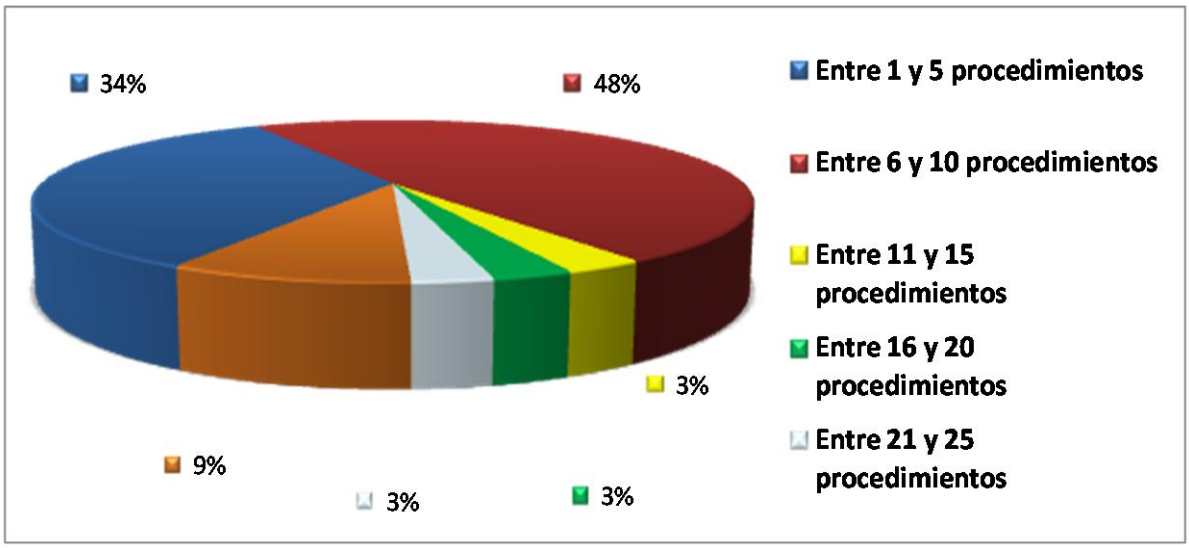

Figura 10: Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos.

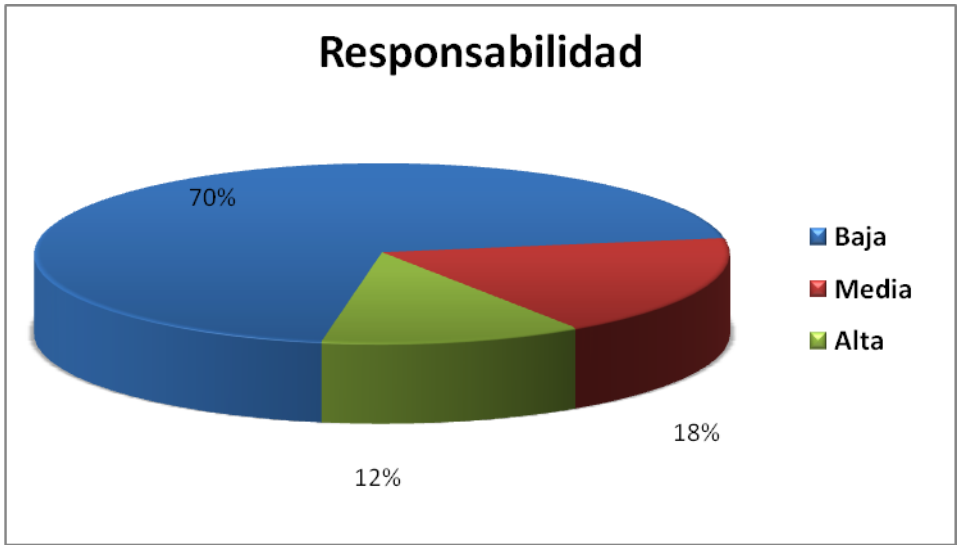

Figura 11: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

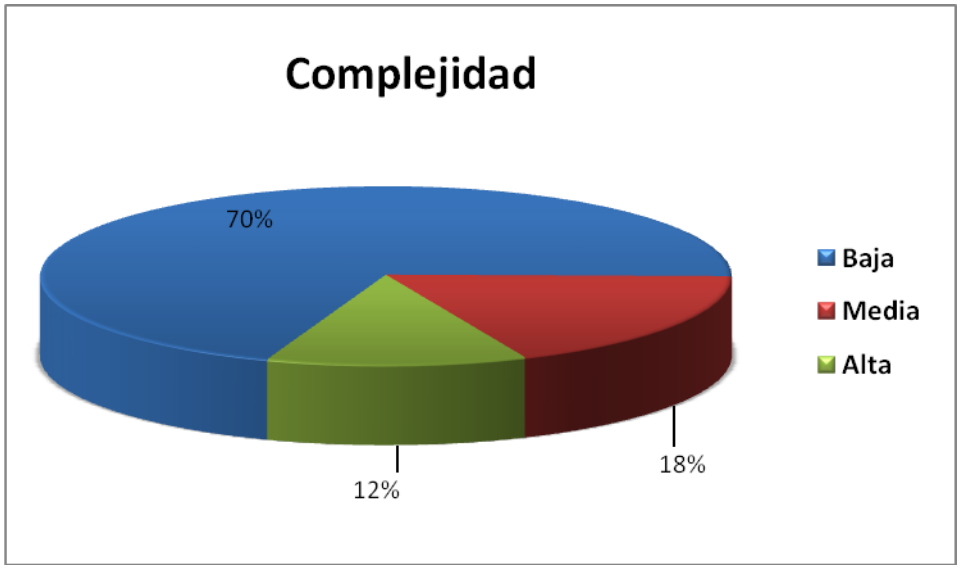

Figura 12: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

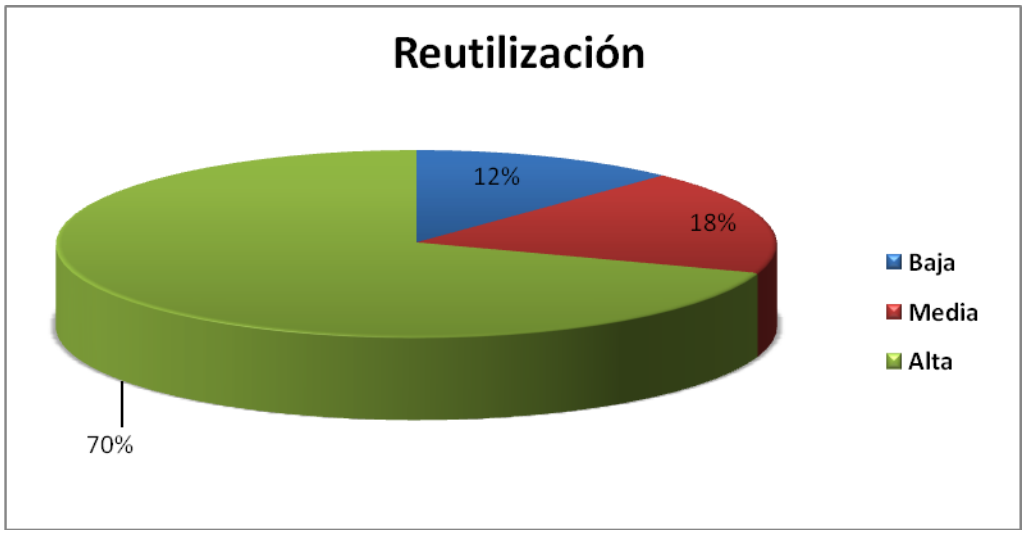

Figura 13: Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Cuando existe un TOC alto se afectan los parámetros de calidad definidos por esta métrica. Se reduce la reutilización de las clases, la implementación se hace más compleja, las pruebas son difíciles de realizar y aumenta la responsabilidad de las clases.

La mayoría de las clases que conforman el sistema están dentro de las categorías de pequeña y media, para un 88 % en total, lo que demuestra la elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto. Los resultados obtenidos son positivos según esta métrica.

3.2.2 Relaciones entre Clases (RC)

Las relaciones entre las clases (RC), están dadas por el número de relaciones de uso de una clase con otras.

Atributo que afecta	Modo en que lo enfoca
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 de 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 5: Relaciones entre clases (RC).

Atributos de calidad	Categorías	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2
Complejidad de mantenimiento	Baja	<=Prom.= 5
	Media	Entre Prom. y 2 * Prom.
	Alta	>2 * Prom.
Reutilización	Baja	>2 * Prom.

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

Tabla 6: Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica Relaciones entre clases (RC).

Resultados de la aplicación de la métrica

Al analizar los resultados obtenidos en la evaluación del instrumento de medición de la métrica RC, se puede concluir que el diseño de la aplicación es simple y tiene una calidad aceptable para realizar la implementación, teniendo en cuenta que el 62% de las clases poseen menos de 5 dependencias de otras clases. Por último los atributos de calidad Complejidad de Mantenimiento, Cantidad de Pruebas y Reutilización poseen niveles aceptables en un 87 % de las clases.

3.3 Modelo de Implementación

El modelo de implementación describe cómo los elementos del modelo de diseño, se implementan en términos de componentes. Describe también cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados, y cómo dependen los componentes unos de otros, además de los recursos necesarios para poder ejecutar el sistema desarrollado.

3.3.1 Diagrama de Componentes

La solución propuesta consta de un solo componente llamado Contratación el cual interactúa con otros componentes del marco de trabajo como ZendExt, Zend Framework, Doctrine y ExtJS. A continuación se muestra el diagrama de componentes elaborado.

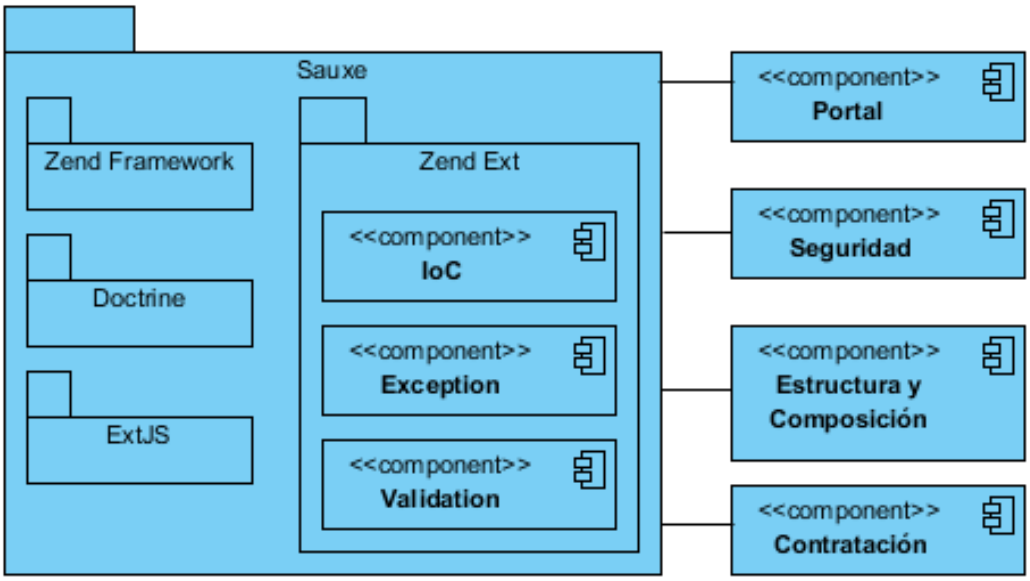

Figura 14: Diagrama de componentes.

3.3.2 Diagrama de Despliegue

Los diagramas de despliegue muestran las relaciones físicas de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos (49).

Un diagrama de despliegue describe la configuración del sistema para su ejecución en un ambiente del mundo real. Representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Para el despliegue se deben tomar decisiones sobre los parámetros de la configuración, funcionamiento, asignación de recursos, distribución y concurrencia. (49)

El usuario desde su puesto de trabajo (conectado a una impresora) podrá acceder al sistema que estará desplegado en el servidor web donde mismo se encuentre ubicada la aplicación. Dicho servidor se conectará al servidor de bases de datos en el cual se almacenará la información de interés para la solución. A continuación se muestra el diagrama de despliegue elaborado.

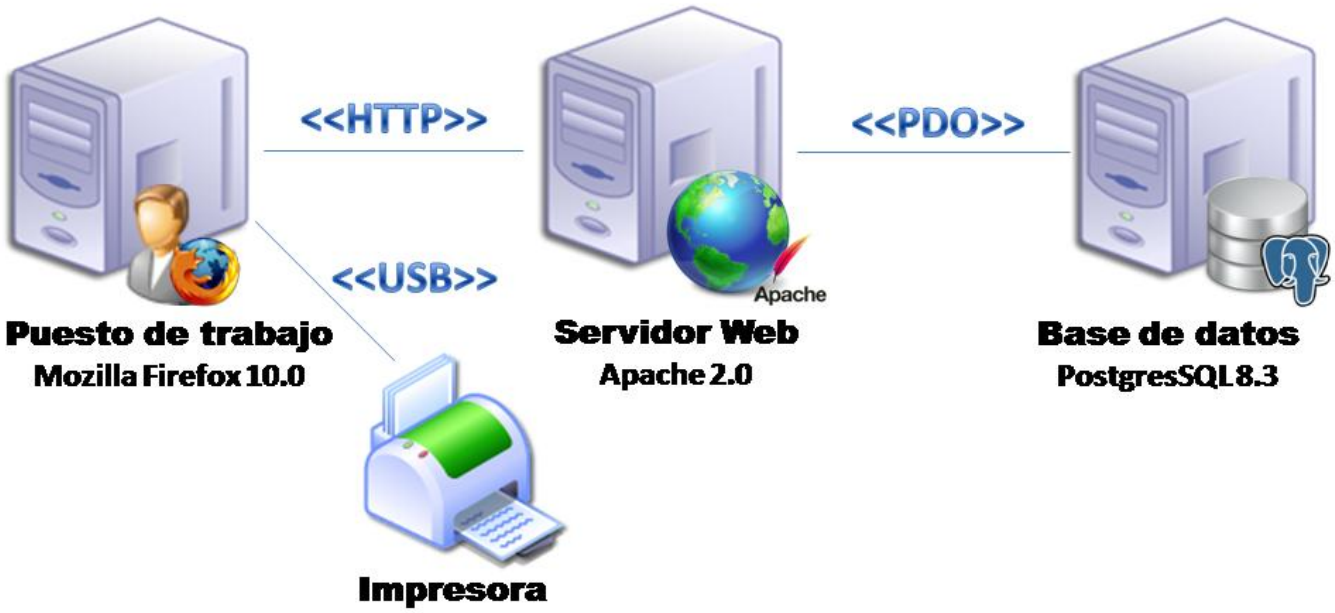

Figura 15: Diagrama de despliegue.

3.4. Modelo de pruebas

Las pruebas del software son un elemento crítico para la garantía de la calidad del software. Las pruebas son realizadas con el objetivo de detectar errores en el sistema, por lo que se llevan a cabo durante todo el ciclo de vida del producto.

Para llevar a cabo el proceso de pruebas al sistema se definen estrategias de pruebas con el propósito de garantizar la calidad del software.

3.4.1 Pruebas de caja negra

Verifican las especificaciones funcionales y no consideran la estructura interna del programa. Es hecha sin el conocimiento interno del producto. No validan funciones ocultas (por ejemplo funciones implementadas pero no descritas en las especificaciones funcionales del diseño) por tanto los errores asociados a ellas no serán encontrados.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: [Pressman, 2000]

- **Técnica de la Partición de Equivalencia**: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- **Técnica del Análisis de Valores Límites**: esta Técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Técnica de Grafos de Causa-Efecto**: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

En otras palabras, la prueba de la 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 las entradas se aceptan de la forma adecuada y que se produce el resultado correcto.
- Así como la integrad de la información externa (por ejemplo archivos de datos) se mantiene.

La prueba de caja negra a utilizar la técnica de la Partición de Equivalencia que es dentro del método de Caja Negra una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así en número de clases de prueba que hay que desarrollar.

Partición de Equivalencia

Partición de equivalencia es una técnica de prueba de Caja Negra que divide el dominio de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. El diseño de estos casos de prueba para la partición equivalente se basa en la evaluación de las clases de equivalencia.

El diseño de casos de prueba para la partición equivalente se basa en una evaluación de las clases de equivalencia para una condición de entrada. Una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada.

Regularmente, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica. (50)

Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:

- Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, en y por encima del rango.
- Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
- Si una entrada es booleana, hay 2 clases: si o no.

Los mismos criterios se aplican a las salidas esperadas: hay que intentar generar resultados en todas y cada una de las clases.

Aplicando estas directrices se ejecutan casos de pruebas para cada elemento de datos del campo de entrada a desarrollar. Los casos se seleccionan de forma que ejerciten el mayor número de atributos de cada clase de equivalencia a la vez.

Para verificar que la aplicación se comporta según los requerimientos establecidos por el cliente, se realizan las pruebas de caja negra aplicando la técnica de Partición de equivalencia utilizando los casos de pruebas elaborados por el autor. En las tablas 7, 8 y 9 se especifica el caso de prueba para el requisito eliminar producto.

Condición de ejecución

- Se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
- Se debe seleccionar el subsistema Contratación.
- Se debe seleccionar al desplegarse el menú, la opción: **Contrasoft**.
- Se debe presionar el botón **Productos** en el menú de **Operaciones/Nomencladores/**.
- Se debe seleccionar en la tabla de productos el que se desea eliminar**.**
- Se debe presionar el botón **Eliminar** de la tabla de productos.

Requisitos a probar

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Eliminar empresa	El sistema debe permitir eliminar un producto, según sea la selección del usuario. Para ello emite un mensaje para obtener la confirmación del usuario, de cancelarse esta operación, no se eliminará el elemento seleccionado. En el caso de que el producto seleccionado se encuentre asociado a uno o varios proveedores, no se podrá efectuar la operación.	EP 1.1: Eliminar producto correctamente.	– Se presiona el botón **Aceptar** del mensaje emitido por el sistema para obtener confirmación del usuario, con el texto: "¿Está seguro que desea eliminar el producto: (nombre del producto)?". – Se presiona el botón **Aceptar** del mensaje de notificación: "El producto fue eliminado correctamente.".
		EP 1.2: Eliminar producto cuando está asociado a uno o más proveedores.	– Se presiona el botón **Eliminar**. – Se muestra el mensaje de error: "Este producto ya está asignado a uno o más proveedores. No se puede eliminar." – Se presiona el botón **Aceptar** de la notificación.
		EP 1.3: Eliminar producto cuando está asociado a uno o más contratos.	– Se presiona el botón **Eliminar**. – Se muestra el mensaje de error: "Este producto ya está asignado a uno o más contratos. No se puede eliminar."

			– Se presiona el botón **Aceptar** de la notificación.
		EP 1.4: Cancelar.	– Se presiona el botón **Eliminar**. – Se presiona el botón **Cancelar** del mensaje emitido por el sistema para obtener confirmación del usuario, con el texto: "¿Está seguro que desea eliminar el producto: (nombre del producto)?"

Tabla 7: Descripción del caso de prueba para el requisito Eliminar producto.

Descripción de variable

No	Nombre de campo	Tipo	Válido	Inválido
1	NA	NA	NA	NA

Tabla 8: Descripción de variables del caso de prueba para el requisito Eliminar producto.

Juegos de datos a probar

Id del escenario	Escenario	Familia de productos	Código	Nombre	Respuesta del sistema
EP 1.1	Eliminar producto correctamente.	NA	NA	NA	El sistema elimina el producto y emite el mensaje de notificación: “El producto fue eliminado correctamente.”.
EP 1.2	Eliminar producto cuando está asociado a uno o más proveedores.	NA	NA	NA	El sistema muestra el siguiente mensaje de error: “Este producto ya está asignado a uno o más proveedores. No se puede eliminar.”.
EP 1.3	Eliminar producto cuando está asociado a uno o más contratos.	NA	NA	NA	El sistema muestra el siguiente mensaje de error: “Este producto ya está asignado a uno o más contratos. No se puede eliminar.”.
EP 1.4	Cancelar.	NA	NA	NA	Se cancela la operación.

Tabla 9: Datos de prueba del caso de prueba para el requisito Eliminar producto.

Las pruebas de caja negra a la solución fueron desarrolladas por la Subdirección de Calidad del Centro de Informatización de la Gestión de Entidades (CEIGE). Estas se realizaron en tres iteraciones de prueba, donde finalmente se comprobó en la tercera iteración que el producto está libre de no conformidades, para así lograr su liberación. La tabla 10 muestra los resultados de cada una de las iteraciones realizadas.

Agrupación de requisitos	No Conformidades		
	Iteración 1	Iteración 2	Iteración 3
Gestionar empresa	2	0	0
Gestionar familia de productos	3	0	0
Gestionar productos	2	1	0
Gestionar proveedor	8	0	0
Gestionar contrato	9	0	0
Reportes de contratos consolidados	1	0	0
Reportes de contratos por proveedor	1	0	0
Reportes de embarques desglosados	1	1	0
Reportes de contratos por carta de crédito	4	0	0
Reportes de contratos consolidados por empresa	1	1	0
Reportes de proveedores	1	0	0
Total	30	3	0

Tabla 10: Resultados de las iteraciones de las pruebas.

3.4.2 Pruebas de caja blanca

La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca el ingeniero del software puede obtener casos de prueba que: (51)

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

Es por ello que se considera a la prueba de Caja Blanca como uno de los tipos de pruebas más importantes que se le aplican a los software, logrando como resultado que disminuya en un gran porciento el número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. [Pressman, 2000]

Dentro de la prueba de caja blanca se incluyen las Técnicas de Pruebas que serán descritas a continuación:

- **Prueba del Camino Básico:** Permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos. (52)
- **Prueba de Condición:** Ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método. (53)
- **Prueba de Flujo de Datos:** Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez. (53)
- **Prueba de Bucles:** Se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.(54)

La prueba de caja blanca empleada en la solución desarrollada fue la prueba del camino básico propuesta por Tom McCabe., la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa. Para realizar esta técnica es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se enumeran las sentencias de código del procedimiento realizado sobre el método modificarEmpresaAction().

```
function modificarEmpresaAction() {
    $empresa = new Empresa(); //1
    $empresa = Doctrine::getTable('Empresa')->find($this->_request->getPost('idempresa')); //1
    $empresa->codigo = $this->_request->getPost('codigo'); //1
    $empresa->nombre = $this->_request->getPost('text'); //1
    $nodo = explode('-', $this->_request->getPost('padre')); //1
    if ($nodo[1] == null) {//2
        $padre = $nodo[0]; //3
    } //4
    else {//5
        $padre = $nodo[1]; //6
        if ($padre == 25) {//7
            $padre = 0; //8
        }//9
    }//10
    $empresa->idpadre = $padre;//10
    $empresa->nivel = $this->_request->getPost('nivel');//10
    $empresa->padresuperior = $this->_request->getPost('padresuperior');//10
    $modelempresa = new EmpresaModel();//10
    $modelempresa->Actualizar($empresa);//10
    echo"{'codMsg':1,'mensaje': 'La empresa fue modificada correctamente.'}";//10
}
```

Figura 16: Método modificarEmpresaAction().

Notación de Grafo de Flujo.

El Grafo de Flujo está formado por 3 componentes fundamentales que ayudan a su elaboración, comprensión y nos brinda información para confirmar que el trabajo se está haciendo adecuadamente.

Los componentes son: (51)

Nodo: Cada círculo representado se denomina nodo del Grafo de Flujo, el cual representa una o más secuencias procedimentales. Un solo nodo puede corresponder a una secuencia de procesos o a una sentencia de decisión. Puede ser también que hayan nodos que no se asocien, se utilizan principalmente al inicio y final del grafo.

Aristas: Las flechas del grafo se denominan aristas y representan el flujo de control, son análogas a las representadas en un diagrama de flujo. Una arista debe terminar en un nodo, incluso aunque el nodo no represente ninguna sentencia procedimental.

Regiones: Las regiones son las áreas delimitadas por las aristas y nodos. También se incluye el área exterior del grafo, contando como una región más. Las regiones se enumeran y la cantidad de regiones es equivalente a la cantidad de caminos independientes del conjunto básico de un programa.

A continuación se muestra el grafo de flujo obtenido al analizar el método modificarEmpresaAction() que se muestra en la figura 16.

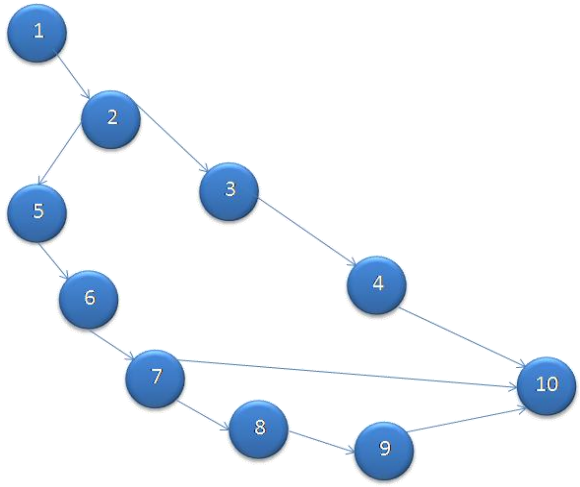

Figura 17: Grafo de flujo asociado al algoritmo modificarEmpresaAction().

Cálculo de la complejidad ciclomática a partir de un segmento de código.

Luego de haber construido el grafo, se realiza el cálculo de la complejidad ciclomática, mediante las tres fórmulas utilizadas para el análisis de complejidad del algoritmo, las cuales tienen que arrojar el mismo resultado para asegurar que el cálculo de la complejidad es correcto.

Fórmulas para calcular complejidad ciclomática:

1. V (G) = (A - N) + 2

   Donde “A” es la cantidad de Aristas y “N” la cantidad de Nodos.

   V (G) = (11 - 10) + 2

   V (G) = 3

2. V (G) = P + 1

Siendo “P” la cantidad de Nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 2 + 1

V (G) = 3

3. V (G) = R Donde “R” representa la cantidad de regiones en el grafo.

V (G) = 3

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de 3, lo que significa que existe esa cantidad posible de caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado. Seguidamente es necesario representar los caminos básicos por los que puede recorrer el flujo.

Número	Camino básico
1	**1-2-3-4-10**
2	**1-2-5-6-7-8-9-10**
3	**1-2-5-6-7-10**

Tabla 11 Caminos básicos del flujo.

Luego se procede a ejecutar los casos de pruebas para cada uno de los caminos básicos determinados en el grafo de flujo.

Caso de prueba para el camino básico 1

Camino: 1-2-3-4-10

Descripción: El dato de entrada importante para que se ejecuten indistintamente los caminos básicos es el campo “padre”, por lo tanto cumplirá con los siguientes requisitos:

El parámetro padre no está vacío y es el identificador de la empresa inmediata superior.

Entrada: 25

Resultados esperados: Se espera que sea modificada la empresa con los datos entrados por parámetro.

Caso de prueba para el camino básico 2

Camino: 1-2-5-6-7-8-9-10

Descripción: El parámetro padre no está vacío y es el identificador por defecto del nodo padre del árbol de empresas. Por lo tanto el valor sería “ynode-25”.

Entrada: ynode-25

Resultados esperados: Se espera que el sistema logre reconocer que el nodo que está seleccionado es un nodo que tiene un identificador asignado por él y que es el nodo padre del árbol de empresas, por lo que el valor del padre de esta empresa va a ser 0.

Caso de prueba para el camino básico 3

Camino: 1-2-5-6-7-10

Descripción: El parámetro padre no está vacío y es el identificador asignado por defecto por el sistema a este nodo.

Entrada: ynode-86

Resultados esperados: Se espera que el sistema logre reconocer que el nodo que está seleccionado es un nodo que tiene un identificador asignado por él y que no es el nodo padre del árbol de empresas, por lo que el valor del padre de esta empresa va a ser 86.

Resultados pruebas de caja blanca

Se le aplico la prueba del camino básico a todos los métodos o funciones del sistema donde se ejecutaron todos los caminos posibles seguir, comprobando en cada uno de ellos que el sistema realice la tarea que debe, corroborando el buen funcionamiento y desempeño del mismo.

3.5 Conclusiones del capítulo

En este capítulo fueron expuestos los artefactos generados como parte del modelo de implementación de la solución propuesta, tales como el diagrama de componentes y el diagrama de despliegue. Se pudo apreciar como el modelo de implementación constituye una entrada fundamental para las pruebas de software. Se describió la prueba de caja blanca realizada que permitió comprobar el correcto funcionamiento de la herramienta. Se validó el diseño mediante la aplicación de métricas para la evaluación de atributos de calidad, las cuales arrojaron resultados satisfactorios, además de las pruebas realizadas a la aplicación por la subdirección de calidad del CEIGE, la cual fue avalada por un acta de liberación llegando a la conclusión de que se encuentra lista para ser liberada.

CONCLUSIONES

Una vez terminado el presente trabajo de diploma se puede concluir que se desarrollaron todas las tareas a fin de cumplir los objetivos propuestos, para esto:

- Se analizaron ventajas y deficiencias de sistemas informáticos tanto nacionales como internacionales vinculados a la gestión de contratos; evidenciándose de esta manera la no existencia de una solución informática capaz de ejecutar las funcionalidades necesarias para esta función.
- Se realizó el diseño y la implementación de un sistema capaz de gestionar la información que se genera en el proceso de contratación y ventas en las oficinas comerciales del Ministerio de Informática y Comunicaciones (MIC) en el exterior, bajo los estándares definidos en el CEIGE, probado y validado mediante métricas y pruebas de software.

La solución propuesta es novedosa, su importancia radica en la gestión de los contratos que se realizan en estas instituciones, la realización de búsquedas y reportes con filtros para proveedores y empresas que están interactuando en el sistema.

Recomendaciones

Ningún sistema es estático en el tiempo debido a que la evolución del negocio y el desarrollo de las tecnologías así lo exigen, es por ello que se recomienda:

- Utilizar el presente trabajo como bibliografía para posibles investigaciones referentes al tema desarrollado en el mismo.
- Se recomienda continuar perfeccionando la herramienta a partir de los nuevos requisitos que puedan surgir como resultado de su explotación.
- Se propone que se utilice el componente como punto de partida para la implementación del componente Contratación del Sistema Integral de Gestión Cedrux.

Referencias bibliográficas

1. [En línea] [Citado el: 2 de diciembre de 2011.][Traducido del Inglés] http://www.businessdictionary.com/definition/contract.html
2. [En línea] [Citado el: 2/12/2011.] www.lexjuridica.com/diccionario/c.htm
3. [En línea] [Citado el: 2/12/2011.] definicion.de/venta/
4. [En línea] [Citado el: 2/12/2011.] www.promonegocios.net/mercadotecnia/definicion-concepto-venta.htm
5. [En línea] [Citado el: 4/12/2011.] www.economia48.com/spa/d/producto/producto.htm
6. [En línea] [Citado el: 4/12/2011.] www.promonegocios.net/mercadotecnia/producto-definicion-concepto.html
7. [En línea] [Citado el: 4/12/2011.] www.marketingpower.com/_layouts/Dictionary.aspx?dLetter=C
8. [En línea] [Citado el: 4/12/2011.] www.cim.co.uk/resources/glossary/home.aspx
9. Del libro: «Diccionario de Marketing», de Cultural S.A., Edición 1999, Pág. 54.
10. Del libro: «Marketing de Clientes ¿Quién se ha llevado a mi cliente?», Segunda Edición, de Barquero José Daniel, Rodríguez de Llauder Carlos, Barquero Mario y Huertas Fernando, McGraw-Hill Interamericana de España, 2007.
11. [En línea] [Citado el: 4/12/2011.] www.promonegocios.net/clientes/cliente-definicion.html
12. [En línea] [Citado el: 4/12/2011.] definicion.de/proveedor/
13. [En línea] [Citado el: 4/12/2011.] www.definicionabc.com/tecnologia/proveedor.php
14. [En línea] [Citado el: 5/12/2011.] www.herramientasparapymes.com/erp-enterprise-resource-planning-planificacion-de-recursos-empresariales
15. [En línea] [Citado el: 5/12/2011.] www.codeka.net/
16. [En línea] [Citado el: 5/12/2011.] www.mercanza.es/productos.html
17. [En línea] [Citado el: 5/12/2011.] www.mercanza.es/index.asp
18. [En línea] [Citado el: 5/12/2011.] www.openerpsite.com/erp-openerp-modulos
19. [En línea] [Citado el: 5/12/2011.] www.herramientasparapymes.com/erp-openbravo
20. «Subsistema de activo fijo tangible del Sistema Integral de Gestión Cedrux», Ing. Lilian Álvarez Almanza, Ing. Osnier Ramírez Alea, Ing. Damián Viltres Ramírez. Ciudad de la Habana, 2012.
21. «Propuesta de modelo de desarrollo de software tecnológico del Centro de Soluciones de Gestión», Mileidy Sarduy, Magalys Pérez, y Sergio Hernández Cisneros. Ciudad de la Habana, 2009.
22. «Arquitectura y Patrones de diseño», Colectivo de Desarrollo.NET, 2008-2009.
23. [En línea] [Citado el: 10/12/2011.] es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o
24. [En línea] [Citado el: 10/12/2011.] www.inf.utfsm.cl/~visconti/ili236/Documentos/08-Patrones.pdf
25. [En línea] [Citado el: 10/12/2011.] tatianamuoz-tatianita.blogspot.com/2010_09_01_archive.html
26. [En línea] [Citado el: 10/12/2011.] migranitodejava.blogspot.com/2011/05/patrones-de-diseno-de-gof.html
27. [En línea] [Citado el: 10/12/2011.] www.grancomo.com/2006/05/02/patrones-para-el-diseno-de-interfaz/
28. [En línea] [Citado el: 6/12/2011.] www.tecnologicocomfenalco.edu.co/iacademica/sistemas/Agora/articulos/UML.pdf

29. [En línea] [Citado el: 6/12/2011.] www.disca.upv.es/enheror/pdf/ActaUML.PDF
30. [En línea] [Citado el: 6/12/2011.] www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE
31. [En línea] [Citado el: 6/12/2011.] www.visual-paradigm.com/product/vpuml/
32. [En línea] [Citado el: 6/12/2011.] es.kioskea.net/contents/langages/langages.php3
33. [En línea] [Citado el: 6/12/2011.] julianaramirezf.blogspot.com/2011/04/javascript.html
34. «Arquitectura tecnológica para el desarrollo de software», Ing. Oiner Gómez Baryolo, Ing. Yoandry Morejón Borbón, Ing. Darien García Tejo. Ciudad de la Habana.
35. [En línea] [Citado el: 6/12/2011.] bibdigital.epn.edu.ec/bitstream/15000/2110/1/CD-2887.pdf
36. [En línea] [Citado el: 6/12/2011.] es.wikibooks.org/wiki/Programaci%C3%B3n_en_PHP
37. [En línea] [Citado el: 6/12/2011.] www.doctrine-project.org/
38. [En línea] [Citado el: 6/12/2011.] www.framework.zend.com/about/overview
39. [En línea] [Citado el: 6/12/2011.] www.error500.net/garbagecollector/archives/categorias/bases_de_datos/sistema_gestor_de_base_de_datos_sgbd.php
40. [En línea] [Citado el: 8/12/2011.] www.postgresql.org/docs/current/static/intro-whatis.html
41. [En línea] [Citado el: 8/12/2011.] www.sqlmanager.net/products/postgresql/manager/
42. [En línea] [Citado el: 22/01/2012.] pgadmin.org/visualtour.php
43. [En línea] [Citado el: 8/12/2011.] www.wikilearning.com/tutorial/control_de_versiones_con_subversion-el_control_de_versiones/19266-1
44. [En línea] [Citado el: 8/12/2011.] forja.rediris.es/docman/view.php/123/117/TortoiseSVN-1.4.1-es.pdf
45. [En línea] [Citado el: 10/12/2011.] sophia.javeriana.edu.co/~cbustaca/DSBP/Patrones_Comparacion/CATALOGO%20DE%20PATRONES.pdf
46. [En línea] [Citado el: 22/01/2012.] www.ecured.cu/index.php/Diagrama_de_Clase
47. Del libro: «Ingeniería de Software, Un enfoque práctico», de Pressman y Roger S, 1998.
48. Del libro: «Object Oriented Software Metrics. Prentice Hall», de Lorenz M., Kidd J. 1994.
49. [En línea] [Citado el: 20/01/2012.] virtual.usalesiana.edu.bo/web/practica/archiv/despliegue.doc
50. [En línea] [Citado el: 22/01/2012.] www.ecured.cu/index.php/Pruebas_de_caja_negra#Partici.C3.B3n_equivalente
51. [En línea] [Citado el: 22/01/2012.] www.ecured.cu/index.php/Pruebas_de_caja_blanca
52. [En línea] [Citado el: 22/01/2012.] indalog.ual.es/mtorres/LP/Prueba.pdf
53. [En línea] [Citado el: 22/01/2012.] gemini.udistrital.edu.co/comunidad/grupos/arquisoft/fileadmin/Estudiantes/Pruebas/HTML - Pruebas de software/node26.html
54. [En línea] [Citado el: 22/01/2012.] www.ecured.cu/index.php/Prueba_de_bucles#Tipos_de_prueba_de_bucles

Anexos

REPÚBLICA DE CUBA

CONTRASOFT

Contratos de la empresa: Grupo de la electrónica

PERÍODO DE ANÁLISIS: 01/01/2012 - 27/06/2012

Código del contrato	Monto del contrato(USD)	Total embarcado en el período	Pendiente por embarcar en el período	Proveedor
0002	$ 2000	$ 2000	$ 0	Proveedor 1
Subtotal	**$ 2000**	**$ 2000**	**$ 0**	

MINISTERIO DE LA INFORMATICA Y COMUNICACIONES

OFICINA COMERCIAL

Anexo 1: Reporte de contratos por empresa.

REPÚBLICA DE CUBA

CONTRA$OFT

Contratos del proveedor: Proveedor 1

PERÍODO DE ANÁLISIS: 01/01/2012 - 27/06/2012

No.	Monto(USD)	Total embarcado en el período	Pendiente por embarcar en el período	Empresa
0001	2000	2000	0	GKT
0002	2000	2000	0	Grupo de la electrónica
0003	2000	2000	0	Copextel
Subtotal	**$ 6000**	**$ 6000**	**$ 0**	

MINISTERIO DE LA INFORMATICA Y COMUNICACIONES

OFICINA COMERCIAL

Anexo 2: Reporte de contratos por proveedor.

GLOSARIO

Componente: Unidad de composición de aplicaciones de software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio.

Eficiencia: Es el logro de los objetivos y metas con el mínimo de los recursos y tiempo. Es el resultado del mejor aprovechamiento de los recursos utilizados para la realización de las actividades que se prevén a fin del cumplimiento de una meta o acción determinadas.

Funcionalidad: Coherencia entre las necesidades detectadas y los resultados que se obtienen con el uso del material. Es lo que un producto puede hacer. Probar la funcionalidad significa asegurar que el producto funciona tal como estaba especificado.

Herramienta: es un objeto elaborado a fin de facilitar la realización de una tarea mecánica.

Método: Conjunto de instrucciones a las que se les da un determinado nombre de tal manera que sea posible ejecutarlas en cualquier momento sin tenerlas que reescribir sino usando sólo su nombre.

Métrica: Medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado.

Proceso: Un proceso se define como un conjunto de tareas, actividades o acciones interrelacionadas entre sí que, a partir de una o varias entradas de información, materiales o de salidas de otros procesos, dan lugar a una o varias salidas también de materiales (productos) o información con un valor añadido.

Reutilización: Acción de volver a utilizar los bienes los bienes o productos.

Sistema: Es un conjunto organizado de objetos o partes interactuantes e interdependientes, que se relacionan formando un todo unitario y complejo.

Software: se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica.

Software libre: Es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente.

Soporte: acciones que permiten mantener disponibles los recursos de hardware o software que necesita un producto de software.

Usuario: El usuario es la persona que consume o usa el producto, bien o servicio.

Validación: Confirmación mediante el suministro de evidencia objetiva de que se han cumplido los requisitos para una utilización o aplicación específica prevista.