Universidad de las Ciencias Informáticas

Facultad 3

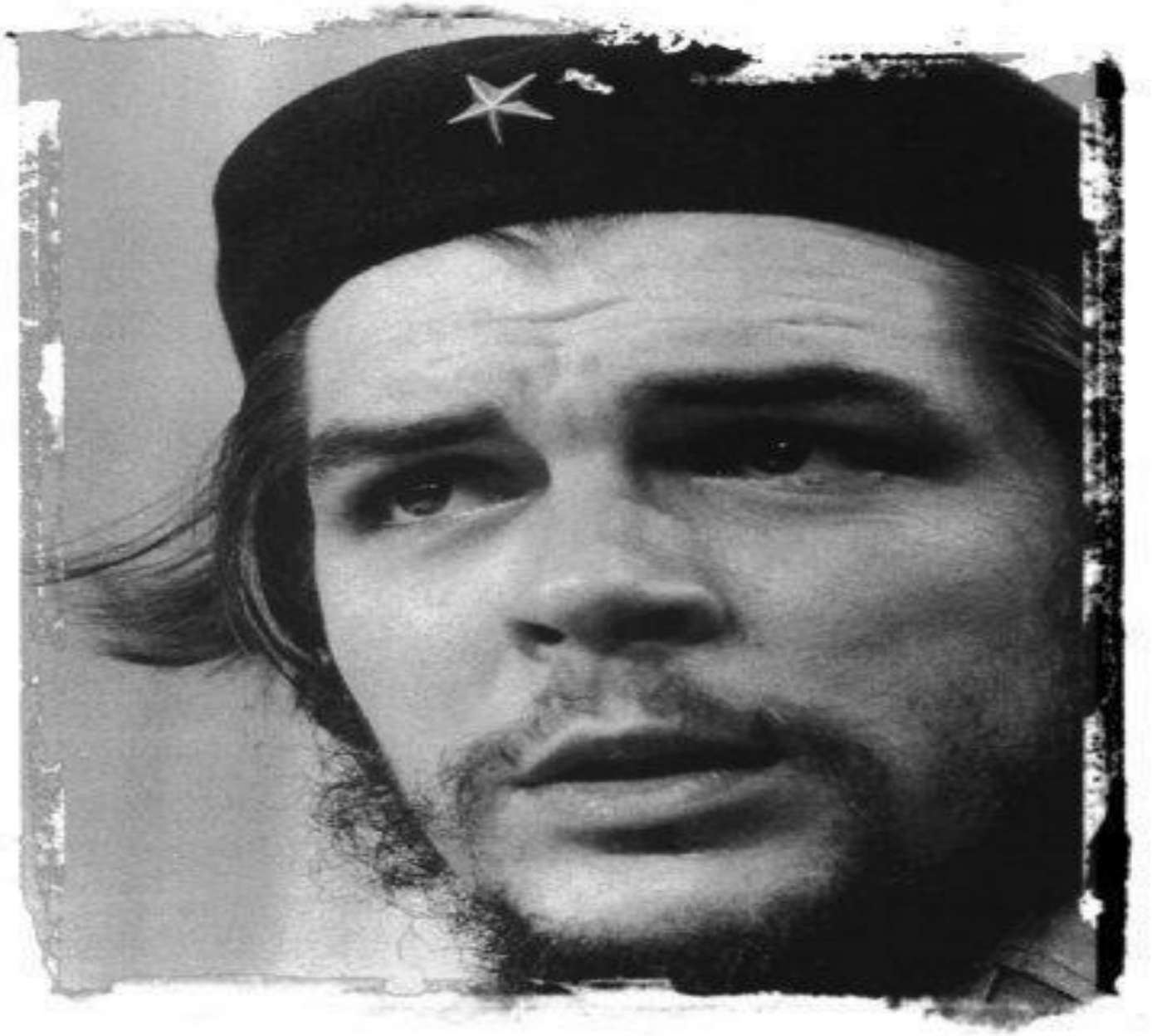
Subsistema Punto de venta del Sistema de Gestión Integral de Organizaciones Xedro-ERP

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

Autor(es): Eddy Nuñez Horta
Idris García Pérez

Tutor(es): Ing. Yisel Niño Benítez
Ing. Saily Oliva Martínez
Lic. Leandro Roberto Reyes Mejías

La Habana, junio de 2014
"Año 56 de la Revolución"



"...aquí está una de las tareas de la juventud: empujar, dirigir con el ejemplo la producción del hombre de mañana. Y en esta producción, en esta dirección, está comprendida la producción de sí mismos..."

Ernesto "Che" Guevara de la Serna.

DECLARACIÓN DE AUTORÍA

Declaramos ser los únicos autores de este trabajo y autorizamos al Centro de Informatización de Entidades (CEIGE) perteneciente a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Idris García Pérez

Eddy Nuñez Horta

_____ _____

Firmas de Autores

Ing. Saily Oliva Martínez

Ing. Yisel Niño Benítez

_____ _____

Firmas de Tutores

Lic. Leandro Roberto Reyes Mejías

_____

Firmas de Co-Tutores

DATOS DE CONTACTO

Ing. Yisel Niño Benítez: Ingeniera en Ciencias Informáticas, graduada en la Universidad de las Ciencias Informáticas en el año 2009. Se desempeña como Asesora de Calidad de Software del centro CEIGE.

Ing. Saily Oliva Martínez: Ingeniera en Ciencias Informáticas, graduada en la Universidad de las Ciencias Informáticas en el año 2009. Se desempeña como analista de la Subdirección de Producción del centro CEIGE.

Lic. Leandro Roberto Reyes Mejías: Licenciado en educación en la especialidad de Informática, Graduado en el ISPETP “Enrique Pineda Zaldívar”, en el año 2008. Actualmente se desempeña como Especialista general, en el grupo de Calidad de Software del centro CEIGE.

AGRADECIMIENTOS

A mi Madre por ser la persona más luchadora que conozco, por ser el mejor ejemplo y la mejor madre que un hijo puede tener, Por todos los sacrificios que ha hecho por mí, por haber dedicado toda su vida a mi felicidad y por estar siempre presente cuando la he necesitado, dios te bendiga mami. A la mejor hermana del mundo, Yuselin, por ser mi motivación, por ayudarme incondicionalmente en estos cinco años de mi carrera a alcanzar todas nuestras metas, por apoyarme en cada paso que voy dando en mi vida, todo agradecimiento para ti es poco mi hermana. A mi luz, mi hijo que cada día me da más fuerzas para vivir por él y por mi familia… te adoro papi. A mi esposa, mujer, amiga y compañera Damarys por darme la familia tan bella que tenemos, por dejarme formar parte de su vida y porque sin ella "nunca nunca" hubiese llegado aquí, nunca olvides que Te amo…
A mi familia, especialmente a mis princesas Yosi y Rosi, a mi tía Belkis, a mi cuñado Santiago, a mi abuela Dora. A mis primos, Yulexis, Yuleidis y Anne porque cada vez que me preguntaban cómo te va la escuela me hacían sentir que a ellos también les interesaba mi carrera. A mis amigos Eiler, Yadian, Javier y Julio Cesar porque siempre estuvieron ahí todas y cada una de las veces que los necesité y no fueron pocas. A mis tutoras, por tener mucha paciencia conmigo y ayudarme, por sus conocimientos y horas dedicadas a mi trabajo, este resultado es también de ustedes, por siempre gracias. En general a todos los que de una forma u otra me han ayudado a avanzar durante estos 5 años. A todos mis compañeros de grupo y de la UCI, porque hicieron estos cinco años inolvidables…
Eddy

A mi madre del alma por ser siempre tan especial y confiar siempre en mí.

A mi abuela por ser un ángel en mi camino y saberme guiar bien y darme la fuerza para seguir en la lucha.

A mi padre por ser el amigo y el consejero que siempre estaba para ayudarme.

A mis tías por ser un talismán en mi camino y darme los mejores consejos.

A mi tío Freedy que es mi inspiración.

A mi adorable hermana por confiar en mí y darme fuerzas para seguir.

A mis colegas Lisvany, Karel, Dian, EL Flaco y Morciego.

A mi amigo Yadrian por ser la persona que más momentos dedico para mí y confió en todo momento de que si se podía lograrlo.

A mi preciosa novia Aliuska que a pesar de sus malcriadeces siempre supo en los momentos más difíciles estar a mi lado y apoyarme y darme fuerzas para continuar, miles de gracias amor.

A mis profesores por saberme inducir por el camino del conocimiento y darse cuenta de quién soy realmente. Principalmente al profe Alain por brindarse y extender su mano en mi auxilio en los momentos difíciles.

A mis tutores por brindarme el apoyo para que esta tesis se realizara.

Idris

DEDICATORIA

A mi abuelo José Elías, porque aunque ya no esté conmigo ni haya podido vivir este momento, él supo siempre que este día llegaría y que yo lograría convertirme en un profesional, aquí te va abuelo donde quiera que estés, este resultado es tuyo… nunca te olvido y sé que estás muy orgulloso de mí.

A mi Madre por darme su apoyo incondicional y guiarme por el camino correcto. Sin sus consejos no hubiese logrado este sueño que teníamos ambos.

A mi hermana y amiga Yuselin porque esta meta es tuya y sin ti no existiera este trabajo, yo solo traté de recompensar todo el amor que me has dado y todas las cosas que haces por mi cada día de mi vida, porque en ocasiones has dejado de vivir tu vida para vivir la mía para hacerme un hombre de bien. Te amo…

Eddy

A las dos personas más importante de mi vida, artífice fundamental del modelo de persona que soy hoy, mi madre y mi abuela. Ellas han sido las personas más importantes en mi vida y en mi formación, sin su amor, apoyo e incondicionalidad no hubiese alcanzado todos mis sueños.

Te amo y siempre los llevaré en mi corazón. Este resultado es para ustedes.

Idris

RESUMEN

El uso de sistemas informáticos para dar solución a diferentes problemas y situaciones relacionadas a los procesos que se realizan en las empresas cubanas se ha convertido en una estrategia habitual del país en la actualidad. En el Centro de Informatización de Entidades (CEIGE), perteneciente a la Universidad de las Ciencias Informáticas (UCI) se desarrolla el producto Xedro-ERP, el cual tiene como objetivo fundamental informatizar, organizar y acelerar los principales procesos que intervienen en una empresa. Xedro-ERP aún no contempla entre sus funcionalidades un subsistema que gestione los procesos que se llevan a cabo en los puntos de venta.

El presente trabajo describe el análisis, diseño y la implementación de un subsistema informático responsable de manejar los procesos relacionados a los puntos de venta, para lo cual se realizó un estudio de sistemas informáticos que informatizan dichos procesos, concluyendo que era necesario construir un subsistema de punto de venta e integrarlo al sistema Xedro-ERP. Las herramientas y tecnologías utilizadas para su desarrollo fueron las definidas por el CEIGE. El subsistema Punto de venta del sistema Xedro-ERP contribuye al cumplimiento de las políticas de migración hacia el software libre, permite llevar a cabo el proceso de compra-venta de mercancías o productos, generar comprobantes de venta y reportes de ventas efectuadas, garantizando de esta manera la integridad de la información que se gestiona en los puntos de venta.

PALABRAS CLAVES: CEIGE, Punto de venta, subsistema, Xedro-ERP.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En el proceso de informatización de la sociedad se encuentran involucradas principalmente las denominadas Tecnologías de la Información y las Comunicaciones (TIC), las cuales ocupan un lugar importante en el desarrollo de un país. Gracias a los cambios continuos en la esfera de la informática, específicamente en la realización de sistemas informáticos, las empresas han logrado mejorar la eficiencia de sus procesos: productivos, logísticos, de ventas, financieros y de administración; logrando así reducir costos y elevar su competitividad, contribuyendo al aumento progresivo de la economía. Además de mejorar la eficiencia de los procesos empresariales, el uso de las TIC tiene especial importancia para el mercadeo y las ventas, pues favorece la búsqueda y comunicación con proveedores y clientes.

En Cuba, empresas o unidades presupuestadas como Havanatur, Peugeot Cuba, ETECSA, TRD Caribe, Tiendas Panamericanas, Librerías, Restaurantes, Agencias de viajes, Centros turísticos, Tiendas minoristas de Comercio Interno presentan Puntos de ventas (PV) o Terminales de Punto de Venta (TPV) como parte fundamental de su objeto social. En los puntos de venta se llevan a cabo las transacciones e información de pago de los clientes por la venta de bienes, productos y/o servicios; se lleva un control de los precios de artículos y descuentos ofrecidos, además de elaborarse los comprobantes de ventas y facturas por las ventas realizadas.

Muchas de estas empresas o entidades comerciales realizan dichas actividades de forma manual, lo que trae como consecuencia que se cometan errores en las anotaciones de las ventas realizadas, se pierda la información que allí se genera o se dupliquen comprobantes de venta, provocando desorganización y pérdida de la integridad en la información que se procesa. Además del consumo excesivo del tiempo y recursos en dichas tareas (entendiéndose como recursos: financieros, humanos y materiales), muchas veces no se lleva un histórico de las ventas por vendedor o artículo, proporcionando desvíos monetarios que se reflejan en los estados financieros, imposibilitando una correcta toma de decisiones de la dirección de la empresa.

Existen otras empresas que poseen aplicaciones informáticas con el objetivo de controlar estos procesos de forma digital, pero muchas de estas soluciones se caracterizan por abordar solamente partes del problema de la gestión en las entidades, otras son desarrolladas con tecnologías privativas a un costo muy elevado para la adquisición de las licencias y mantenimiento del software. Estos sistemas que en ocasiones son aplicados en los puntos de venta cubanos en su mayoría no se ajustan a los requisitos y normativas que rige el Ministerio de Finanzas y Precios.

Los sistemas de Planificación de Recursos Empresariales (Enterprise Resource Planning, ERP) son sistemas informáticos desarrollados para intervenir en las actividades y procesos que se realizan en las empresas, tales como: ventas, compras, entregas, pagos, producción, contabilidad, administración de inventarios y de recursos humanos. Funcionan en todo tipo de empresas, para ello, integran todos los departamentos funcionales de las mismas, herramientas de mercadotecnia y administración estratégica en un solo sistema. (Wailgum, 2008).

Por su parte, el Sistema de Gestión Integral de Organizaciones Xedro-ERP, aún en fase de desarrollo por el Centro de Informatización de Entidades (CEIGE) perteneciente a la Universidad de las Ciencias Informáticas (UCI), cumpliendo con las normativas del Ministerio de Finanzas y Precios, brinda entre sus funcionalidades:

- Controlar los permisos y roles de los usuarios que operan en el sistema.
- Registrar y controlar los clientes asociados a la empresa.
- Generar los comprobantes de las operaciones que se realizan en la entidad.
- Registrar y controlar las monedas que se utilizan en la empresa (incluyendo la moneda contable).
- Generar las facturas comerciales por venta de productos o prestación de servicios a clientes.
- Registrar y controlar los movimientos que se realizan con los productos que se almacenan en los depósitos asociados a la empresa.
- Registrar las cajas, tipos de fondos y fondos financieros de la empresa.

Dichas funcionalidades son necesarias para que los procesos de venta de artículos, generación de ticket o comprobantes de venta, registro de los impuestos y descuentos de productos, que se realizan en los PV se efectúen íntegramente.

Por todo lo antes expuesto se plantea el siguiente **problema a resolver**:

¿Cómo contribuir a la gestión de procesos de punto de venta de manera que permita la integración con el Sistema de Gestión Integral de Organizaciones Xedro-ERP?

Objeto de estudio:

Procesos de punto de venta.

Objetivo General:

Desarrollar un subsistema integrado al Sistema de Gestión Integral de Organizaciones Xedro-ERP que contribuya a la gestión de los procesos de puntos de ventas en las empresas cubanas.

Objetivos Específicos:

- Elaborar la fundamentación teórica de la investigación para establecer las bases teóricas de la

misma.

- Realizar el análisis del subsistema Punto de venta para lograr un mejor entendimiento en el desarrollo del subsistema.
- Diseñar el subsistema Punto de venta para facilitar la implementación del mismo.
- Realizar la implementación del subsistema Punto de venta para lograr una mejora en la gestión de los procesos de puntos de venta en las empresas cubanas.
- Validar la solución propuesta mediante pruebas de caja blanca y caja negra para aceptar la solución propuesta.

Campo de Acción:

Sistemas informáticos que gestionan procesos de puntos de venta.

Para dar cumplimiento a los objetivos específicos se proponen las siguientes tareas de la investigación:

Tareas de la investigación

Sobre objetivo 1:

- Elaboración de un informe sobre los distintos sistemas informáticos que gestionan punto de venta en el entorno nacional e internacional y las características generales de Xedro-ERP.
- Elaboración de un informe sobre los conceptos a tratarse, tecnologías, herramientas y modelos a utilizar en la investigación.

Sobre objetivo 2:

- Realización del mapa de procesos y del modelo conceptual del subsistema.
- Modelación de los prototipos de interfaz de usuario de la solución propuesta.
- Descripción de los procesos de negocio y de los requisitos funcionales del subsistema.
- Especificación de requisitos de software.

Sobre el objetivo 3:

- Realización del modelo entidad relación del subsistema a desarrollar.
- Validación de los artefactos generados en estas fases.
- Definición de los mecanismos de diseño a utilizar en la solución propuesta.
- Realización del diagrama de clases del diseño del subsistema a desarrollar y descripción de las mismas.
- Realización de los diagramas de secuencia.

Sobre objetivo 4:

- Modelación del diagrama de componentes.
- Codificación del subsistema a través de los estándares de codificación definidos.
- Integración del subsistema con el subsistema de Estructura y Composición.
- Integración del subsistema con el subsistema de Seguridad para el control de los roles, usuarios, funcionalidades y permisos.
- Integración del subsistema con el subsistema de Configuración y Multimoneda.
- Integración del subsistema con el subsistema de Inventario para la obtención del stock[1].
- Realización del modelo de despliegue.
- Integración del subsistema con el subsistema Contabilidad para generación de comprobantes de operaciones y con Finanzas para la interacción con la caja y los fondos asignados.
- Publicación de servicios entre componentes.
- Obtención de los artefactos de estándar de codificación empleada, tratamiento de excepciones.

Sobre objetivo 5:

- Ejecución de pruebas de caja blanca y caja negra al subsistema.

Dando cumplimiento a estos objetivos se espera materializar como **idea a defender:** con el desarrollo de un subsistema de Punto de venta dentro del Sistema de Gestión Integral de Organizaciones Xedro-ERP, se contribuirá a la gestión de los procesos en los puntos de venta de las empresas cubanas.

Entre los métodos de la investigación científica que se utilizarán se encuentran:

Métodos teóricos:

Histórico-Lógico

En la investigación se utiliza este método en la realización de un análisis acerca de la evolución que ha tenido la gestión de punto de venta en sistemas informáticos.

Analítico-Sintético

Fue utilizado en el análisis de los documentos obtenidos para el levantamiento y captura de requisitos, extrayendo y examinando los principales elementos relacionados con el objeto de estudio. También fue empleado en el estudio de soluciones informáticas existentes para determinar características comunes y problemas que estas poseen con el objetivo de determinar los posibles elementos que puedan ser utilizados en la solución que se desea obtener.

[1] Stock: Conjunto de productos que tiene almacenados un comercio y que están destinados a la venta.

Modelación

La modelación es el método mediante el cual se crean abstracciones con el objetivo de explicar la realidad. Este método se evidencia en la modelación de los procesos de negocios identificados y del modelo conceptual.

Método empírico:

Observación

La observación científica es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado. Este método se evidencia en la identificación de procesos de negocio y de los requisitos funcionales y no funcionales.

Método particular:

Entrevista

Se realizarán entrevistas a trabajadores de empresas que contienen puntos de ventas para la comercialización de productos y servicios con el fin de verificar si poseen una aplicación informática que les gestione los procesos de venta y en caso de ser positivo qué características tiene; con el objetivo de analizar el funcionamiento de dichas aplicaciones.

Estructura del trabajo

Capítulo 1: Fundamentación teórica. En este capítulo se describen los conceptos fundamentales que están relacionados con la gestión de punto de venta. Se realiza un estudio del estado del arte revisando algunos de los sistemas informáticos utilizados para la gestión de punto de venta, exponiendo las ventajas y las desventajas de cada uno; haciéndose una valoración crítica basada en el estudio de dichos sistemas. Se justifica el empleo de la metodología, herramientas y tecnologías a utilizar en el trabajo.

Capítulo 2: Modelo de negocio y Requisitos. En este capítulo se analizan y modelan los procesos de negocio relacionados con la gestión de punto de venta. Se aplican técnicas tanto para la captura de los requisitos como para la validación de los mismos.

Capítulo 3: Diseño, Implementación y Pruebas. En este capítulo se exponen los artefactos generados durante el diseño y la implementación de la solución, aplicándose métricas para validar dicho diseño y se le realizan pruebas al software.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1. Introducción

En este capítulo se hace mención a conceptos fundamentales que están relacionados con los procesos que se realizan en un PV, para lograr una mejor comprensión de la investigación. Se realiza un análisis de la gestión de punto de venta en diferentes sistemas informáticos nacionales y extranjeros. Se justifica la utilización de las tecnologías, metodología, técnicas y herramientas definidas para el desarrollo de la investigación.

1.1. Conceptos fundamentales

1.1.1. Punto de venta

El punto de venta (PV) es el lugar que gestiona el proceso de salida y cobro de la mercancía en las tiendas departamentales, comercios, restaurantes y otras instituciones, en las que se organiza el proceso de venta de artículos en mostrador de manera fácil, ágil y cómoda para vendedores y compradores; contando con una caja registradora, un cobrador o cajero y emitiéndose un comprobante de venta como evidencia de la actividad realizada. De poseer un programa para la gestión de la venta facilita la creación e impresión del ticket de venta mediante la información del stock. Para realizar una transacción comercial de compra-venta como el uso de tarjetas bancarias, el PV debe contar con un Terminal de Punto de Venta.

1.1.2. Terminal de punto de venta

Un Terminal de Punto de Venta (TPV) puede definirse como el sistema que está compuesto por dos partes fundamentales, la primera es el hardware (dispositivos físicos), como un escáner de código de barra, una impresora de recibos, lectores de banda magnética y monitores. La parte de software es un programa de gestión de venta de productos o servicios creado específicamente para agilizar los procesos de ventas en los PV. Dicho sistema se encarga de realizar los procesos de la venta de un centro de ofertas públicas a través de una interfaz accesible para los compradores y vendedores relacionados en la ejecución de la actividad, dígase la captura de los productos en la base de datos, imprimir un ticket o gestionar la factura de venta, lectura de la información mediante dispositivos externos para el cobro a través de tarjetas bancarias, emisión de comprobantes de venta, revisión de reportes mensuales así como llevar el control de inventarios y operaciones comerciales determinadas.

1.1.3. Planificación de Recursos Empresariales (ERP)

Los sistemas de Planificación de Recursos Empresariales (Enterprise Resource Planning, ERP) son sistemas de información gerencial que integran y manejan muchos de los negocios asociados con las operaciones de producción y los aspectos de distribución de una compañía comprometida en la producción de bienes o servicios.

Los sistemas ERP típicamente manejan la producción, logística, distribución, inventario, envíos, facturas y contabilidad de una empresa o entidad. Sin embargo, pueden intervenir en el control de muchas actividades de negocios como ventas, entregas, pagos, producción, administración de inventarios, calidad de administración y la administración de recursos humanos. (Cruz Mulet, et al., 2009)

Un sistema puede estar compuesto por:

- Subsistemas: es el mayor nivel de abstracción dentro de un sistema. Agrupación de módulos y/o componentes que responden a una necesidad de negocio.
- Módulos: Es el nivel de abstracción que incluye agrupación de componentes por cuestión lógica o de negocio.
- Componentes: Es el menor nivel de abstracción dentro de un sistema. Agrupación de funcionalidades que responde a una necesidad de negocio y que contribuye a favorecer el mantenimiento, la adaptabilidad y la flexibilidad del sistema.

Por lo tanto la solución que se propone puede definirse como un subsistema que está conformado por componentes que responderán a los procesos de punto de venta y a su vez se integrará a subsistemas que componen un sistema ERP.

1.1.4. Normativas establecidas por el MFP[2]

Uno de los indicadores a tener en cuenta para el análisis de los sistemas informáticos que poseen PV son las normativas establecidas por el Ministerio de Finanzas y Precios de la República de Cuba, las cuales quedaron establecidas en las resoluciones No. 324/94 y No. 12/2007, que describen los procedimientos que se le realizan al efectivo en caja en los establecimientos de venta de productos.

[2] MFP: Ministerio de Finanzas y Precios.

1.2. Sistemas informáticos que poseen módulos de punto de venta

Golden.NET

Golden.NET es la nueva línea de Software de Gestión Empresarial para Pymes[3] utilizado en sitios comerciales de ventas públicas en Cuba como TRD Caribe, Tiendas Panamericanas, Palmares, Artex y Cadena Cubalse.

El sistema incluye los siguientes productos:

- Contabilidad.NET: Contabilidad y fiscalidad para Pymes y Asesorías.
- Facturación.NET: Gestión Comercial y CRM[4] para Pymes y autónomos.
- Autónomos.NET: Programa de Contabilidad y fiscalidad.

Golden.NET es el ERP estándar que integra todas las funciones empresariales de las Pymes (gestión contable, gestión fiscal, gestión comercial de compras y ventas, almacén, CRM y tesorería), ofreciendo en una única aplicación todas las soluciones, evitando engorrosos ajustes independientes y de difícil integración. Es una aplicación diseñada para adaptarse a cualquier tipo de empresa con tiempos y costos de implantación reducidos, también para el control de los inventarios, los cobros y pagos en las Bases Centrales de Almacenes (BCA) o distribuidoras de mercancías, así como para un PV (tiendas, quioscos, uniones de quioscos). (Golden Soft, 2013)

Terminal Punto de Venta en Golden.NET:

El software Terminal Punto de Venta para Comercios es un módulo integrado con Gestión Comercial Golden.NET y ha sido ideado para un amplio abanico de negocios: tiendas de ropa, perfumerías, librerías, zapaterías. Adaptado tanto a monitores con pantalla táctil como con teclado y conexión a los distintos periféricos: lector de código de barras, visor, impresora de tickets.

El software Terminal Punto de Venta para Hostelería es un módulo integrado con Gestión Comercial Golden.NET y ha sido especialmente diseñado para la gestión de bares y negocios de hostelería. Se distingue por una entrada ágil de tickets adaptado a monitores con pantalla táctil y elección de los pedidos seleccionando la imagen del artículo. El usuario podrá configurar la ubicación de salones, terrazas, número de comensales por mesa y si la mesa está libre u ocupada. Además la aplicación conserva la orden del cliente en el cambio de la barra a una mesa.

[3] Pymes: Pequeñas y medianas empresas.

[4] CRM (Customer Relationship Management): Administración de Relaciones con el cliente.

TPV Hostelería y el TPV Comercial de Golden.NET está desarrollado para sistemas de arquitectura de 64 y de 32 bits, confiriéndole como una de las aplicaciones tecnológicamente más avanzadas en el mercado. Esta aplicación permite de manera operativa efectuar las ventas en cadenas de tiendas, el mismo está constituido por diferentes menús y opciones que posibilitan una correcta ejecución, pero dentro de sus desventajas está que necesita una versión actualizada del programa que se utiliza para los reportes comerciales a nivel de División y Cadena y de manera respectiva sus dos tipos de reportes comerciales, los personalizados y los temporales ya que para el caso de los personalizados el sistema guarda los grupos de estudios de las diferentes dimensiones que un usuario utiliza, de manera que sólo tiene que personalizar una vez el sistema, y periódicamente ir variándolo según sus comodidades, por lo que al no estar actualizado imposibilita la eficiencia de la aplicación. (Golden Soft, 2013)

OpenERP

OpenERP es un sistema de planificación de recursos empresariales que cubre las necesidades de las áreas de contabilidad, ventas, compras, almacén e inventario de una empresa. Soporta múltiples monedas, compañías y contabilidades; además incorpora funcionalidades de gestión de documentos para agilizar la colaboración entre departamentos y equipos en la empresa; y permite trabajar remotamente mediante una interfaz web desde una computadora conectada a Internet. Tiene componentes separados en esquema cliente-servidor. Dispone de interfaces XML-RPC (Extensible Markup Language Remote Procedure Call) (protocolo de llamada a procedimiento remoto) y SOAP (Simple Object Access Protocol) (Protocolo de Acceso de simple Objeto). (Hernández, 2010)

Terminal Punto de Venta en OpenERP:

Es uno de los módulos que se incluye en el paquete de OpenERP. Permite gestionar las ventas de una forma muy sencilla, está basado en tecnología web por lo que no se necesita instalar ningún software y todas las tiendas de venta pueden ser fácilmente consolidadas. Permite trabajar con y sin conexión por lo que se puede seguir vendiendo aunque se pierda la conexión a Internet.

Dentro de las distintas opciones para seleccionar productos, posee un lector de código de barras, también logra realizar búsqueda de texto en caso que las otras funcionalidades no estén disponibles.

Se puede realizar en el TPV múltiples tickets sin tener que esperar a realizar una transacción al mismo tiempo. Así mismo, para facilitar los pagos, la aplicación permite múltiples métodos de pago.

El TPV es simple y accesible para el uso en tiendas o restaurantes. El objetivo de ello es una solución empaquetada para mejorar la productividad de los negocio de ventas en las distintas empresas de una forma ágil y segura para el control de las ventas realizadas en los TPV. (openerpsite.com, 2012)

Openbravo

OpenBravo es un sistema de gestión empresarial en software libre, ha sido específicamente diseñado para ayudar a las empresas a mejorar su rendimiento. La cobertura funcional del producto incluye todas las áreas típicas de un sistema de gestión integrado, destacando la solución de PV.

Este sistema posee entre sus módulos: facturación, cobros y pagos, contabilidad, estadísticas, productos, recursos humanos, control de inventario, gestión de atención a clientes y proveedores, gestión de compras, gestión de almacenes, workflow[5] de procesos, gestión de proyectos, planificación de proyectos, gestión de producción/fabricación, gestión de ventas, facturación, gestión de informes, gestor documental y Terminal de punto de venta logrando con ello simplificar y automatizar los procesos empresariales de gestión de entidades. (Gardiner, et al.)

Dentro de sus principales ventajas está su cobertura funcional que incluye la gestión financiera donde enmarca la gestión de caja. Se puede destacar que permite gestionar varias cajas y soporta múltiples monedas.

Además, esta aplicación se integra de manera natural con otras áreas como la gestión de relaciones con clientes, inteligencia de negocio y TPV. (openbravo.com, 2010)

Terminal Punto de Venta en Openbravo:

Se manifiestan las siguientes características:

- Zonas de ventas.
- Pedidos de venta.
- Auto-venta.
- Preventa.
- Tele-venta.
- Reserva de género en almacén para pedidos no servidos.
- Corrección de pedidos.

Donde cada una de estas queda estrechamente relacionada con el módulo sistema de PV denominado Openbravo WEB POS, en el cual se realizan cada una de ellas de forma ágil, flexible y autónoma.

[5] Workflow: flujo de trabajo.

Este sistema de PV es ideal para tiendas, restaurantes o negocios pequeños que desean llevar sus ventas, cobros e inventario todo integrado, tiene capacidad para usarse con pantallas táctiles y códigos de barras. (Openbravo WEB POS, 2010)

Microsoft Dynamics AX

Es una solución de ERP para empresas, que proporciona una base diseñada expresamente para cinco sectores (Fabricación, Distribución, Sector minorista, Sector público, Servicios), junto con funcionalidades de ERP completas y básicas para la gestión de finanzas, recursos humanos y operaciones.

Es un producto creado por Microsoft que se encuentra en múltiples idiomas y permite trabajar con múltiples monedas, pero tiene como desventaja que no permite crear múltiples cajas. (microsoft.com, 2013).

Terminal Punto de Venta en Microsoft Dynamics AX:

La administración del TPV centralizado incluye perfiles visuales y funcionales, diseños de la interfaz de usuario y permisos de empleados. Es la solución ideal para el comercio al por menor y cadenas comerciales. Responde por completo a las funcionalidades más exigentes sin la necesidad de construir, administrar y mantener múltiples aplicaciones e interfaces.

Su modelo de plataforma única evita conectar diferentes sistemas desde TPV a sistemas de tienda o cualquier función de la oficina central. Es una solución de administración completa e integrada, basada en el conocimiento y las tecnologías de Microsoft, diseñada especialmente para administrar y unificar de manera eficiente todos los procesos claves del sector comercial sin importar la industria; abarcando desde finanzas, operaciones, compras, distribución a los PV desde un comercio con un sólo PV, hasta varias sucursales. (microsoft.com, 2013).

1.2.1. Valoración crítica de los sistemas estudiados

Luego de analizar los sistemas informáticos que gestionan el proceso de Punto de venta se determina que no existen sistemas nacionales que informaticen dicho proceso, obteniéndose como resultado una tabla de comparación en la que se puede visualizar con mayor facilidad cada una de las características, ventajas y desventajas de los sistemas a nivel internacional que se analizaron. A continuación la tabla comparativa:

Tabla 1. Comparación de los sistemas de gestión de TPV.

Sistemas Internacionales\ Indicadores	Plataforma tecnológica	Licencia	Sistema Operativo	Normas del MFP	Punto de Venta	Integración con ERP	Multientidad	Soporte y actualización	Uso en Cuba	Multimoneda
Golden.NET	-Gestor de BD: SQL Server Compact 3.5. -Framework.net: 3.5 -Java	Privada	Windows	No	Si	Si	Si	Si	Si	Si
OpenERP	-OpenERP Server 6.0.3 -Open ERP web 6.0.3 -PostgreSQL -Python	AGPL	Windows, Linux, Unix, MacOSX, Android	No	Si	Si	Si	Si	No	Si
Openbravo ERP	-Apache y Tomcat -BD: PostgreSQL y Oracle -Java	Openbravo Public License, GNU (GPL)	Windows, Linux, Unix, Solaris, freeBSD	No	Si	Si	Si	Si	No	Si
Microsoft Dynamics AX	-Microsoft SQL Server 2012 -IDE: MorphX -X++[6]	MS-EULA	Windows	No	Si	Si	No	Si	No	Si

En el análisis a los sistemas informáticos anteriormente comparados se resume que ninguno posee un módulo de Punto de venta que pueda ser integrado al sistema Xedro-ERP ya que:

- Golden es uno de los sistemas que atiende la planificación de recursos empresariales, pero no se ajusta a las normativas del Ministerio de Finanzas y Precios, además su licencia es privada por lo que los límites en la responsabilidad por fallos es determinante al igual que la no reinstalación del

[6] X++: Es un lenguaje de programación híbrido de java y C#, propio de Microsoft Dynamics AX.

programa en equipos distintos al que se instaló originalmente. Su gestor de base de datos (que presenta licencia privativa) y el lenguaje de programación con el que fue desarrollado son incompatibles con las tecnologías y herramientas en que se desarrolla el sistema Xedro-ERP.

- Los sistemas OpenERP y Openbravo ERP son sistemas multientidades, cuentan con funcionalidades que integran a un ERP, pero según la bibliografía consultada no aparecen referencias de su explotación en las entidades cubanas y a pesar de que ambos tienen licencia GPL, no se puede pagar por su soporte y actualización por su elevado costo, ejemplo de ello es el valor de la implementación inicial de Openbravo que se aproxima a 27.000 euros. Los lenguajes de programación en los que fueron implementados los sistemas OpenERP y Openbravo fueron python y java respectivamente, y como el sistema Xedro-ERP está desarrollado con el lenguaje de programación PHP, entonces no permitiría la integración con los módulos TPV de estos sistemas, ya que Xedro-ERP no contiene servicios web para dicha integración, sino que se integra mediante un mecanismo de Inversión de Control, el cual permite que la responsabilidad de cada componente se implemente por separado y después poder integrarlos.
- Microsoft Dynamics AX es un sistema multimoneda, con funcionalidades de ERP completas y básicas para la gestión de finanzas, recursos humanos y operaciones, a pesar de que su licencia es pública, este sistema no es multientidad ni es utilizado en Cuba. Microsoft Dynamics AX fue implementado con un lenguaje de programación propio para su desarrollo, por lo que no permitiría la integración de su módulo TPV con el sistema Xedro-ERP por la ausencia de servicios web.

Por lo tanto se decide como parte de la investigación que ninguno de los módulos Punto de venta de estos sistemas será integrado al Xedro-ERP, sin embargo, sus funcionalidades se tendrán en cuenta para desarrollar un subsistema de Punto de venta propio que logre la integración con Xedro-ERP.

Xedro-ERP es un sistema que se encuentra actualmente en desarrollo, rigiéndose bajo las normativas del Ministerio de Finanzas y Precios. Dentro de sus características resalta que es multientidad, multiusuario y multimoneda; desarrollado sobre plataformas web siguiendo los principios de independencia tecnológica. Entre sus funciones garantiza la integridad de la información que maneja y cuenta con varios subsistemas que atienden diferentes funcionalidades en una misma línea.

El producto Xedro-ERP posee módulos como: Configuración, Estructura y composición, Contabilidad, Inventario, Facturación, Activos Fijos Tangibles, Costo y procesos, Capital Humano (Incluye Nóminas), Generador de reportes y Caja, Banco, Cobros y Pagos.

Con el estudio de los sistemas informáticos se concluye que estos programas no constituyen una elección factible pues representan gastos muy excesivos a instituciones del país por concepto de licencias y mantenimiento, además ninguno posee un TPV que pueda ser integrado al sistema Xedro-ERP y algunos de ellos fueron desarrollados con tecnologías privativas, lo cual está en desacuerdo con el paradigma de independencia tecnológica por el que aboga actualmente Cuba, por lo que se decide desarrollar un subsistema Punto de venta integrado a subsistemas de Xedro-ERP.

1.3. Modelo de desarrollo de software

El modelo de desarrollo a utilizar será el definido por el CEIGE, denominado CEIGE-Modelo de Desarrollo de Software v1.2. Este modelo es el que se utiliza en el desarrollo del Sistema de Gestión Integral de Organizaciones Xedro-ERP, como guía en el proceso de construcción del software, delimitando con precisión los artefactos, roles y actividades involucradas, junto a prácticas y técnicas recomendadas. En el desarrollo del subsistema Punto de venta se realizarán actividades y crearán artefactos que están presentes en las disciplinas Modelado de negocio, Requisitos, Análisis y diseño, Implementación y Pruebas internas, de la fase Desarrollo del modelo definido.

1.4. Notación de modelado de procesos del negocio BPMN

La Notación para el Modelado de Procesos del Negocio (BPMN) es un estándar basado en los diagramas de flujo, adaptado para suministrar una notación gráfica que describe la lógica de los pasos de un proceso de negocio a través de flujos de trabajo. Esta notación ha sido especialmente diseñada para coordinar la secuencia de los procesos y los mensajes que fluyen entre los participantes de las diferentes actividades. (Bizagi, 2014)

En el desarrollo del subsistema se utiliza la BPMN en la modelación de los procesos de negocio identificados, para su posterior análisis e implementación.

1.5. Lenguaje unificado de modelado (UML)

El Lenguaje Unificado de Modelado (Unified Modeling Language, UML) es el lenguaje estándar para el modelado de software que permite la visualización, especificación y documentación de los modelos que se crean durante la construcción y desarrollo de un software. (Jacobson, y otros, 2000)

Se utilizará este lenguaje gráfico que permite además construir los artefactos del sistema de software por poseer diversas ventajas como son:

- Facilita la comunicación entre desarrolladores, permite ahorrar tiempo en el desarrollo del software y hace más sencillas las modificaciones que se vayan a realizar.
- Ofrece varios diagramas para el modelado de sistemas.

- Brinda facilidades para el diseño, documentación, reutilización de código y detecciones de fallas.
- Posibilita la descripción de sistemas, simplificando la complejidad de estos y sin pérdida de información, haciendo posible la comprensión del sistema tanto para usuarios como desarrolladores.

1.6. Arquitectura de software

Una arquitectura es un conjunto de reglas, definiciones, términos y modelos que se emplean para construir un producto. Es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se le percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La arquitectura es el elemento más importante de un sistema, debido a que se encarga de estructurar y establecer las relaciones entre los componentes que lo conforman, representa una abstracción del sistema como un todo. (scribd, 2012)

En el desarrollo del subsistema PV se utilizará la arquitectura definida por el marco de trabajo Sauxe, la cual está basada en componentes, siguiendo el patrón arquitectónico Modelo-Vista-Controlador.

1.6.1. Patrón arquitectónico Modelo-Vista-Controlador (MVC)

Es un patrón de diseño de arquitectura de software usado principalmente en aplicaciones que manejan gran cantidad de datos y transacciones complejas donde se requiere una mejor separación de los conceptos para que el desarrollo esté estructurado de una mejor manera, facilitando la programación en diferentes capas de forma paralela e independiente. MVC sugiere la separación del software en tres estratos:

Modelo: es la representación de la información que maneja la aplicación. El modelo en sí son los datos puros que puestos en un contexto del sistema proveen de información al usuario o a la aplicación misma.

Vista: es la representación del modelo en forma gráfica disponible para la interacción con el usuario. En el caso de una aplicación web la "Vista" es la página HTML con contenido dinámico sobre el cual el usuario puede realizar operaciones.

Controlador: es la capa encargada de manejar y responder las solicitudes del usuario, procesando la información necesaria y modificando el Modelo en caso de ser necesario.

El Modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos. Lo ideal es que el modelo sea independiente del sistema de almacenamiento.
- Definir las reglas de negocio (la funcionalidad del sistema).

- Llevar un registro de las vistas y controladores del sistema.
- Si se encuentra ante un modelo activo, notificará a las vistas los cambios que en los datos pueda producir un agente externo.

El Controlador es responsable de:

- Recibir los eventos de entrada.
- Contiene reglas de gestión de eventos, del tipo "Si evento Z, entonces acción W". Estas acciones pueden suponer peticiones al modelo o a las vistas.

Las Vistas son responsables de:

- Recibir datos del modelo y lo muestra al usuario.
- Tienen un registro de su controlador asociado (normalmente porque además lo instancia).
- Pueden dar el servicio de "actualización", para que sea invocado por el controlador o por el modelo (cuando es un modelo activo que informa de los cambios en los datos producidos por otros agentes).

1.7. Herramientas utilizadas

1.7.1. Herramienta CASE[7]

Visual Paradigm 5.0 es una herramienta CASE para el desarrollo de aplicaciones utilizando modelado UML (Lenguaje Unificado de Modelado) que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de calidad y a un menor costo. Permite dibujar todos los tipos de diagramas de clases, ingeniería inversa, generar código desde diagramas y generar documentación. (Visual Paradigm, 2013)

En el desarrollo del subsistema se utiliza la herramienta Visual Paradigm para la generación de los artefactos descritos en la metodología de desarrollo, tales como, Diagrama de procesos de negocio, Modelo conceptual, Diagramas de clases del diseño, Diagrama de Entidad-Relación (Modelo de datos), Diagrama de componentes.

1.7.2. Entorno de desarrollo integrado (IDE[8])

El **NetBeans** es una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero permite además la implementación de software utilizando

[7] CASE (Computer Aided Software Engineering): Ingeniería de Software Asistida por Computadora.

[8] IDE (Integrated Development Environment): Entorno de desarrollo integrado.

otros lenguajes de programación. Cuenta con una enorme cantidad de extensiones y plugins que pueden incorporarse al sistema para facilitar las tareas de programación. Además, tiene una intuitiva interfaz de usuario, un sistema de depuración de programas y las demás herramientas de ayuda, que facilitan el desarrollo de software. Es una aplicación de código abierto bajo licencias libres por lo que tiene una amplia comunidad de desarrollo en constante crecimiento. También posee una plataforma de aplicación, la cual puede ser usada como una estructura de soporte general (framework) para compilar cualquier tipo de aplicación. (NetBeans, 2013)

En la implementación de las clases controladoras, modelos y de las vistas de la solución se usará NetBeans en la versión 7.4 por poseer un medio de desarrollo integrado en múltiples plataformas, además de ofrecer soporte a otros lenguajes como HTML, PHP[9] 5 y JavaScript, los cuales serán utilizados a lo largo de la implementación de la aplicación. Este entorno de desarrollo fue seleccionado además por integrarse con el servicio de control de versiones Subversion y con los lenguajes de programación a utilizar, brindando el completamiento de código.

1.7.3. Control de versiones

Subversion 1.4.5 (SVN) es un software libre bajo una licencia de tipo Apache/BSD que se utiliza en el control de versiones en la que los desarrolladores puedan trabajar en un mismo proyecto en forma ordenada. Tiene una arquitectura cliente-servidor con controles de concurrencia para cuando varios desarrolladores están trabajando en el mismo archivo, lo cual permite que se tenga un historial o que se recuperen versiones anteriores de los archivos. (www.subversion.apache.org, 2010)

1.7.4. Sistema gestor de Base de datos (SGBD)

PostgreSQL es un potente sistema de gestión de bases de datos objeto – relacional (object- relational data base managament system, ORDBMS) liberado bajo licencia BSD (Berkeley Software Distribution). Posee una arquitectura probada que garantiza una elevada confiabilidad e integridad en los datos, está basado en una arquitectura cliente-servidor. Es multiplataforma y opera en varios sistemas operativos como Unix, Windows y Linux. (PostgreSQL, 2013)

Se usa PostgreSQL por brindar las siguientes características para el desarrollo del subsistema Punto de venta:

- Aislamiento: Asegura que una operación no puede afectar a otras. Esto asegura que dos transacciones sobre la misma información nunca generará ningún tipo de error.

[9] PHP (Hypertext Pre-processor): Pre-procesador de hipertexto.

- Durabilidad: Asegura que una vez realizada la operación, esta persistirá y no se podrá deshacer.
- Documentación: Posee una vasta documentación bien organizada, pública y libre.

1.7.5. Lenguajes de programación

Programación del lado del servidor

PHP 5.2

"PHP es un lenguaje de script incrustado dentro del HTML. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas de sí mismo. La meta del lenguaje es permitir rápidamente a los desarrolladores la generación dinámica de páginas" (Heredia, y otros, 2001)

Dentro de las ventajas que aporta PHP a los desarrolladores están:

- Muy fácil de aprender y de usar.
- Se caracteriza por ser un lenguaje muy rápido.
- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los sistemas de gestión de base de datos: MySQL, PostgreSQL, Oracle, MS SQL Server, entre otros.
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es de código abierto, soportado por una gran comunidad de usuarios.
- Incluye gran cantidad de funciones que permite realizar tareas útiles en la web como: generar imágenes y gráficas, establecer conexiones a servicios de red, enviar correos electrónicos, trabajar con cookies y sesiones y generar documentos PDF. (PHP, 2013)

Programación del lado del cliente

HTML 5.0

Hyper Text Markup Language (Lenguaje de marcado de Hipertexto, HTML) es un lenguaje de programación muy simple que se usa para la elaboración de las páginas web. Está formado por etiquetas que definen que será lo mostrado por el navegador, definiéndose así el formato del texto, los elementos que conforman la página y disposición. Esas etiquetas son leídas y ejecutadas por el navegador, visualizando así la información en pantalla. Una de las características es que además del texto, permiten que se creen enlaces entre distintas partes del documento o entre distintas fuentes de información a través del hiperenlace o hipervínculo. (Alvarez, 2012)

JavaScript 2.2.1

JavaScript es un lenguaje de programación que se utiliza para crear páginas web dinámicas. Se utiliza del lado del cliente logrando incorporar mejoras en las páginas. Trabaja los navegadores más importantes, tales como Internet Explorer, Firefox, Opera y Safari. Es un lenguaje de secuencias de comandos, definiéndose como lenguaje de programación ligero. Además es interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Normalmente, se encuentra incrustado directamente en las páginas HTML, aunque puede ser cargado directamente desde un fichero. (HOEHRMANN, 2006)

Notación de Objetos de JavaScript (Json)

JSON, acrónimo de JavaScript Object Notation es un formato ligero de intercambio de datos. Está basado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto se conoce como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias.

JSON se ha convertido en un estándar en el desarrollo de aplicaciones web. Los servicios web proponen la utilización de JSON en vez de XML para permitir la integración de servicios en el navegador del usuario en vez de en el servidor. El formato JSON es seguramente la mejor opción para el intercambio de información entre el servidor y las funciones JavaScript. (Json, 2011)

CSS 2.1

Las Hojas de Estilo en Cascada (Cascading Style Sheets), son una herramienta usada por las páginas HTML y XML para estructurar de una forma más organizada las propiedades de sus componentes. Permiten el control por parte de los implementadores del estilo y el formato de múltiples páginas web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento, separando el contenido de la presentación. (CSS, 2013)

1.7.6. Tecnologías web

Apache HTTP Server 2.2

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a las últimas versiones de protocolos. Es capaz de funcionar en la mayoría de las plataformas y entornos existentes. Está entre los servidores web más reconocidos a nivel internacional, debido a la capacidad de correr en múltiples sistemas operativos. Apache es una tecnología gratuita de código abierto y altamente configurable, de diseño modular capaz de interpretar diversos lenguajes de programación. (Apache, 2013)

El servidor HTTP Apache es un servidor de código abierto y multiplataforma. Su diseño modular lo convierte en un servidor configurable, de excelencia por su robustez y estabilidad. Incluye una serie de módulos de multiprocesamiento que son responsables de conectar con los puertos de red de la máquina, aceptar las peticiones, y generar los procesos hijos que se encargan de servirlas. Trabaja con diferentes lenguajes, como PERL y PHP.

Permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto. Es altamente configurable en la creación y gestión de ficheros logs. (Apache, 2012)

Mozilla Firefox 4.0

Es un navegador multiplataforma y está disponible en varias versiones de Microsoft Windows, Mac OSX, GNU/Linux y algunos sistemas basados en Unix. Su código fuente es software libre y tiene variedades de plugins para una mejor configuración y desarrollo de aplicaciones web.

Se utiliza Firefox debido a que:

- Las contraseñas son más seguras.
- Tiene extensiones útiles que se pueden descargar y ofrecen funcionalidades adicionales.
- Se actualiza de forma automática.
- Firefox bloquea las ventanas emergentes.
- Incorpora la navegación por pestañas, una forma increíblemente eficaz e intuitiva para tener varias páginas disponibles al mismo tiempo, sin saturar la barra de tareas.

Mozilla Firefox es usado para visualizar páginas web. Incluye corrector ortográfico, búsqueda progresiva, marcadores dinámicos y un sistema de búsqueda integrado que utiliza el motor de búsqueda que desee el usuario. (Mozilla, 2014)

1.7.7. Frameworks[10]

ZendFramework 1.3.7

[10] Framework: marco de trabajo.

Se trata de un marco de trabajo para desarrollo de aplicaciones y servicios web con PHP, brinda soluciones para construir sitios web modernos, robustos y seguros. ZendFramework está basado totalmente en código abierto y se utiliza para el trabajo con el lenguaje web PHP 5. Implementa el patrón arquitectónico MVC. Este framework (marco de trabajo) está formado por una serie de métodos estáticos y componentes. (Zend, 2013)

Doctrine Framework 1.2.1

Es un marco de trabajo que mapea objetos relacionales (ORM por sus siglas en inglés) para PHP con una potente capa de abstracción de bases de datos (dbal por sus siglas en inglés). Uno de sus principales características es la opción de escribir las consultas de base de datos Orientada a Objeto en un lenguaje SQL llamada Doctrine Query Language (DQL por sus siglas en inglés). Esto proporciona a los desarrolladores una poderosa alternativa a SQL que mantiene la flexibilidad, sin necesidad de la duplicación de código innecesaria. (Doctrine, 2010)

ExtJs

ExtJs 2.2 es un framework para construir Aplicaciones de Internet Enriquecidas (RIA por sus siglas en inglés). Está basado en librerías JavaScript de código abierto, ligero y de alto rendimiento, compatible con la mayoría de los navegadores que permiten crear páginas e interfaces webs dinámicas. El mismo incluye tecnologías como Ajax, DHTML, XML, XSLT, JSON, CSS y DOM. Tiene incluidos la mayoría de los 21 controles de los formularios web, incluyendo tablas para mostrar datos y elementos semejantes a la programación de escritorio como los formularios, paneles, barras de herramientas y menús. (ExtJs, 2010)

1.7.8. Sauxe

Sauxe es un marco de trabajo que contiene un conjunto de componentes reutilizables que provee la estructura genérica y el comportamiento para una familia de abstracciones, logrando una mayor estandarización, flexibilidad, integración y agilidad en el proceso de desarrollo. Posee una arquitectura en capas que a su vez contiene en su capa inmediata superior una implementación Modelo-Vista-Controlador (MVC). El mismo da solución a varios escenarios o aspectos arquitectónicos como: gestión y configuración dinámica de cache, integración de componentes de forma distribuida o no distribuida, acceso a bases de datos a través de una capa de abstracción, gestión de concurrencia de recursos, administración centralizada de transacciones, gestión dinámica de las trazas generadas por los sistemas, implementación de mecanismos de autenticación y autorización, implementación de mecanismos de mensajería y control de excepciones, gestión y configuración dinámica de pre condiciones, pos condiciones y validaciones de variables, gestión y configuración de flujos de trabajo, visualización de las

funcionalidades de un sistema, entre otros escenarios de alta complejidad, garantizando los atributos de calidad de los sistemas que se desarrollen con el mismo. (Baryolo, y otros, 2008)

Sauxe fue el marco de trabajo que dirigió el desarrollo del subsistema Punto de venta, este contempla los marcos de trabajo ZendFramework que atiende la lógica de negocio, Doctrine Framework para el mapeo de la base de datos y ExtJs para la implementación de las vistas del sistema.

1.8. Conclusiones parciales

A partir de las necesidades del uso de un sistema que gestione el proceso de PV, se mostraron los Conceptos fundamentales y definiciones asociados a la investigación, los cuales permitieron ampliar el conocimiento y entender el contenido de la investigación. De los sistemas informáticos estudiados que incluyen dicho proceso, se hace un análisis que demuestra y justifica la selección del sistema Xedro-ERP como el sistema apto para integrarle el subsistema PV. En dicho análisis se comparan los sistemas estudiados por indicadores que miden la magnitud de sus funcionalidades en las entidades cubanas, quedando demostrado que el sistema factible para integrarle el subsistema de punto de venta es Xedro-ERP. La descripción de las tecnologías, el modelo de desarrollo, los lenguajes de programación y las herramientas con las que se va a desarrollar el subsistema de PV posibilitan un mayor entendimiento de la solución y garantizan la correcta ejecución de dicho subsistema.

CAPÍTULO 2: MODELO DE NEGOCIO Y REQUISITOS

2. Introducción

En el presente capítulo se realiza la descripción de los procesos de negocio del subsistema Punto de venta. Se utilizaran patrones de control de flujo en el modelado de proceso de negocio. Se modela el mapa de procesos para obtener una comprensión de los pasos que intervienen en un proceso determinado. Para lograr representar los principales conceptos que se manejan en un Punto de venta se elabora el modelo conceptual. Se identifican los requisitos funcionales y no funcionales a partir del uso de técnicas de captura y se realiza la validación de los mismos.

2.1. Modelo de negocio

La modelación de proceso de negocio permite realizar una exploración del dominio del problema, con el fin de lograr comprensión por parte del equipo de desarrollo de los procesos que se realizan actualmente en la entidad y la relación que existe entre estos. (Cybercia, 2011)

El modelado de procesos de negocio es la base para comprender mejor la operación de una organización, documentar y publicar los procesos buscando una estandarización en la organización, buscar eficiencias en la operación e integrar soluciones en arquitecturas orientadas a servicios.

Un proceso de negocio es un conjunto estructurado de actividades, diseñado para producir una salida determinada o lograr un objetivo específico. Los procesos describen como es realizado el trabajo en la organización y se caracterizan por ser observables, medibles, mejorables y repetitivos.

Un modelo de proceso de negocio define los siguientes elementos:

- El objetivo o el motivo del proceso.
- Las entradas específicas.
- Las salidas específicas.
- Los recursos consumidos.
- La secuencia de las actividades.
- Los eventos que dirigen el proceso. (Cybercia, 2011)

2.1.1. Mapa de procesos del negocio

El mapa de procesos de negocio es una representación que señala las actividades que son pertinentes realizar en los procesos de negocio. Además, se identifican las entradas y salidas de cada proceso, o sea, sus actividades relacionadas lógicamente que se llevan a cabo para lograr como resultado un negocio definido. Véase Figura 1.

Para lograr una mayor comprensión de los procesos del negocio de un punto de venta fue necesario el uso de técnicas como entrevistas a trabajadores que ejecutan la actividad de compra-venta en puntos de ventas reales, para obtener una mejor comprensión de dichos procesos. Finalmente como resultado se definieron dos procesos propios del negocio, quedando definidas las principales relaciones existentes entre estos en el mapa de procesos.

Procesos identificados en el subsistema Punto de venta:

- Venta de artículos.
- Cierre de caja.

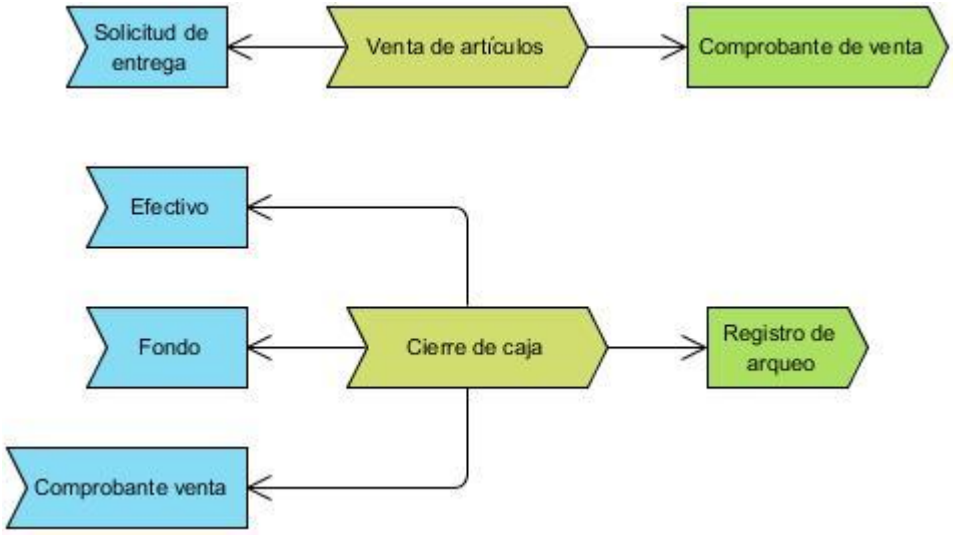


Figura 1: Mapa de procesos de negocio del módulo Punto de venta.

2.1.2. Descripción del proceso de negocio

Venta de artículos: el proceso de negocio Venta de artículos comienza con una solicitud hecha por un cliente para la compra de productos en el Punto de Venta. Los productos que el cliente desea comprar son solicitados al cajero(a) y este lleva a cabo la venta, entregándole al cliente los productos elegidos por él y un comprobante de venta que se genera al finalizar el proceso.

Cierre de caja: al finalizar el horario de venta en el PV, se pasa a realizar la verificación y contabilización de los productos vendidos y del fondo y efectivo de la caja según las ventas realizadas, apoyándose en los comprobantes de ventas.

Para obtener mayor información sobre la descripción de estos procesos.

2.1.3. Modelo Conceptual

El modelo conceptual explica cuáles son y cómo se relacionan los conceptos relevantes y sus atributos en la descripción del problema. Este modelo captura los tipos más importantes de objetos en el contexto del sistema, los cuales representan los elementos que existen o los eventos que suceden en el entorno en el que trabaja el sistema. (López, 2011)

El modelo conceptual se crea con el objetivo de proporcionar un marco analítico, estructurado y claramente definido de los principales conceptos del negocio y las relaciones existentes entre estos.

Teniendo en cuenta que el negocio de Punto de venta adquiere conceptos propios, se procedió a definir las relaciones existentes entre los términos identificados. Véase Figura 2.

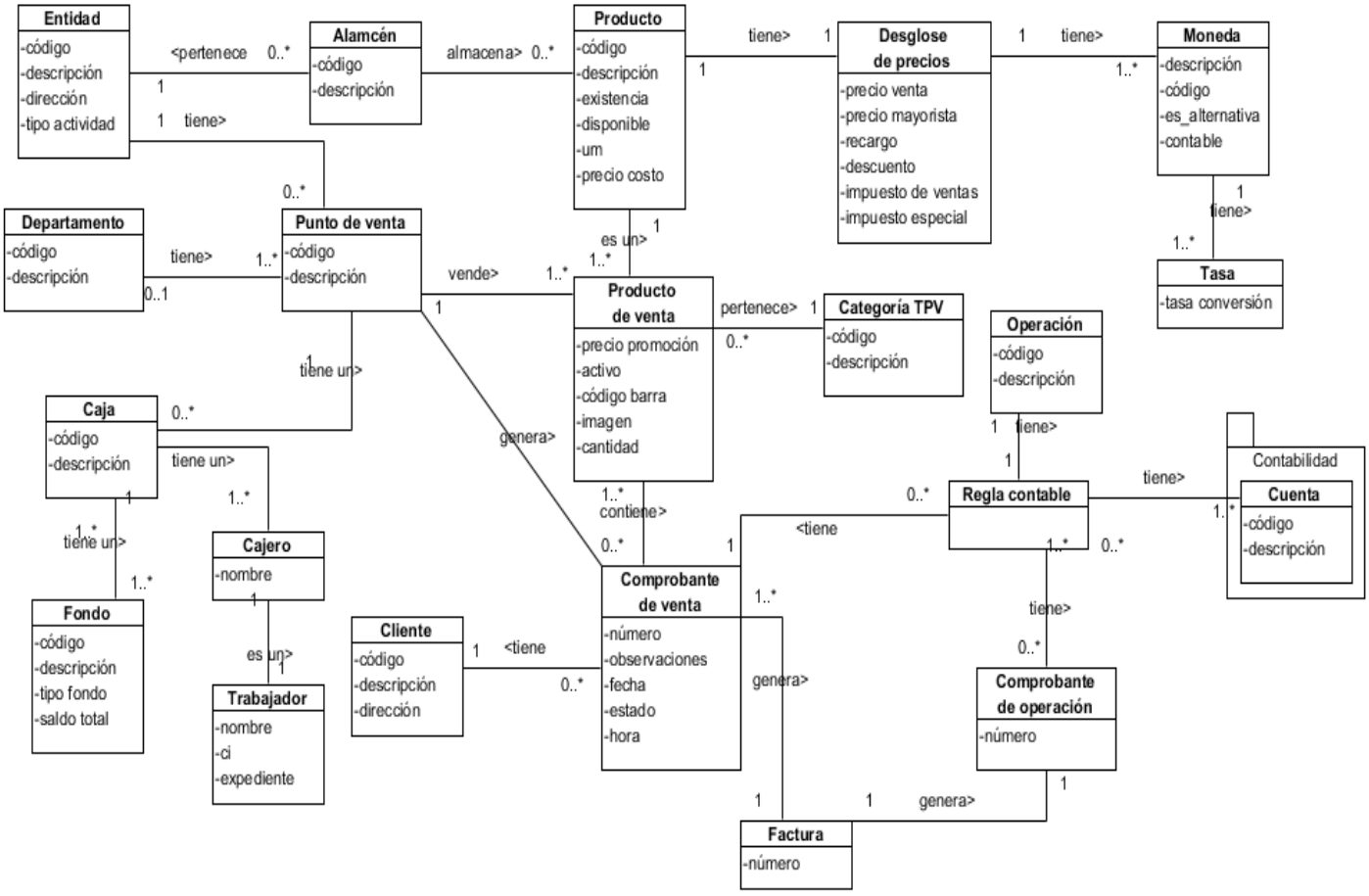


Figura 2: Modelo conceptual del subsistema Punto de venta.

2.1.4. Validación del modelo de negocio

La validación de los artefactos generados como parte de la Modelación del negocio se realizó aplicando la Revisión técnica formal, donde la alta gerencia revisó cada documento generado con el objetivo de validar su correcta elaboración y que el flujo de actividades de cada proceso estuviera en correspondencia con la información brindada. Como constancia de esta revisión se emite un Acta de aceptación que se encuentra en la carpeta de entregables.

2.2. Requisitos

2.2.1. Técnicas para la captura de requisitos

En el mundo se tienen en cuenta diferentes técnicas para la obtención de la información. Las mismas son ejecutadas por un equipo de desarrolladores y clientes para lograr la comprensión del problema a resolver y la magnitud del mismo, proponiendo soluciones, negociando diferentes perspectivas o puntos de vista y especificando un conjunto básico de requisitos de la solución. Las técnicas aplicadas en este trabajo son: entrevista, Open ended Interview y Brainstorming (tormenta de ideas).

Entrevista

Es usada con frecuencia para acercar al cliente y al desarrollador, pues logra una muy buena comunicación entre ambas partes. Permite obtener información sobre el problema en cuestión y comprender los objetivos para su solución. Para realizarla es necesario seleccionar correctamente a los entrevistados, definir previamente las preguntas, pues la forma de redacción puede influir en las respuestas y por último, analizar los resultados. (KOCH, 2002)

Tormenta de ideas

Es una técnica de reuniones que se usa para generar ideas. Su objetivo principal es concebir la mayor cantidad de requisitos para el sistema. Reúne integrantes de varias disciplinas, mientras mayor experiencia y preparación tengan, mayor posibilidad se presenta de generar buenas ideas. En este tipo de reunión no se debe socavar ideas que parezcan locas, ni evaluar o criticar las ideas de los demás, porque puede producir un ambiente hostil que perjudique el dinamismo de la reunión, además de que esa idea puede constituir un gran aporte luego de ser madurada y perfeccionada o relacionada con otras. (ZAPATA, 2004)

2.2.2. Requisitos Funcionales (RF)

Son condiciones o capacidades que el sistema debe desempeñar. Los requisitos funcionales que facilitaron la identificación de las clases y las funcionalidades a desarrollar son:

Tabla 2. Listado de requisitos funcionales identificados.

No.	Requisito funcional	Complejidad
RF 1	Adicionar categoría TPV	Baja
RF 2	Modificar categoría TPV	Baja
RF 3	Eliminar categoría TPV	Baja
RF 4	Listar categorías TPV	Baja
RF 5	Buscar categoría TPV	Baja
RF 6	Configurar punto de venta	Alta
RF 7	Modificar configuración del punto de venta	Alta
RF 8	Eliminar configuración del punto de venta	Media
RF 9	Listar puntos de venta	Media
RF 10	Buscar punto de venta	Media
RF 11	Imprimir listado de puntos de venta	Alta
RF 12	Asociar almacén	Alta
RF 13	Desasociar almacén	Alta
RF 14	Activar punto de venta	Alta
RF 15	Configurar producto del punto de venta	Alta
RF 16	Modificar producto del punto de venta	Media
RF 17	Listar producto del punto de venta	Media
RF 18	Buscar producto del punto de venta	Media
RF 19	Activar producto del punto de venta	Media
RF 20	Adicionar comprobante de venta	Media
RF 21	Modificar comprobante de venta	Media
RF 22	Eliminar comprobante de venta	Media
RF 23	Listar comprobantes de venta	Media
RF 24	Consultar comprobante de venta	Alta
RF 25	Buscar comprobante de venta	Media
RF 26	Realizar búsqueda avanzada del comprobante de venta	Media
RF 27	Imprimir listado de comprobantes de venta	Media
RF 28	Imprimir comprobante de venta	Alta

RF 29	Adicionar producto al comprobante de venta	Alta
RF 30	Modificar producto del comprobante de venta	Alta
RF 31	Eliminar producto del comprobante de venta	Media
RF 32	Listar productos del comprobante de venta	Media
RF 33	Buscar producto del comprobante de venta	Media
RF 34	Confirmar comprobante de venta	Media
RF 35	Cancelar estado del comprobante de venta	Media
RF 36	Cobrar venta	Media
RF 37	Contabilizar comprobante de venta	Alta
RF 38	Mostrar histórico de las ventas realizadas	Alta
RF 39	Mostrar los productos más vendidos	Alta
RF 40	Mostrar productos menos vendidos	Alta
RF 41	Mostrar productos con descuentos en las ventas	Alta
RF 42	Mostrar productos con recargos en las ventas	Alta
RF 43	Apertura del punto de venta	Alta
RF 44	Cierre del punto de venta	Alta

2.2.3. Descripción de los requisitos funcionales

Uno de los artefactos generados y entregables de la investigación, es la descripción de requisitos. Para una mejor organización de los mismos e identificación de futuros componentes en la implementación del subsistema Punto de venta perteneciente a un software de gestión, se crearon agrupaciones que estarían compuestas por los requisitos Adicionar, Modificar, Eliminar, Listar, Consultar, Buscar, Realizar búsqueda avanzada, Imprimir listado e Imprimir modelo. Ejemplo de ello, es la agrupación Gestionar comprobante de venta, que contiene los requisitos:

- Adicionar comprobante de venta
- Modificar comprobante de venta
- Eliminar comprobante de venta
- Listar comprobantes de venta
- Consultar comprobante de venta
- Buscar comprobante de venta

- Realizar búsqueda avanzada del comprobante de venta
- Imprimir listado de comprobantes de venta
- Imprimir comprobante de venta

A continuación se muestra la descripción del requisito Adicionar comprobante de venta con su correspondiente prototipo de interfaz de usuario.

Tabla 3. Descripción del requisito Adicionar comprobante de venta.

Precondiciones	El cliente ha sido validado.
Flujo de eventos	
Flujo básico Adicionar comprobante de venta	
1.	El sistema muestra los datos del comprobante de venta: Nombre entidad Descripción del punto de venta Número de comprobante Nombre del cajero Fecha de emisión Estado del comprobante
2.	Se introducen los datos del comprobante de venta: Observaciones
3.	El sistema valida (ver validación 1, 2, 3) los datos introducidos.
4.	Si los datos son correctos el sistema los registra.
5.	El sistema confirma el registro de los datos.
6.	Concluye el requisito.
Pos-condiciones	
1.	Se registró en el sistema un nuevo comprobante de venta.
Flujos alternativos	
Flujo alternativo 4.a Información errónea	
1	El sistema señala los datos erróneos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo 4.b Información incompleta	
1	El sistema señala los datos vacíos y permite corregirlos.
2	El usuario corrige los datos.
3	Volver al paso 3 del flujo básico.
Pos-condiciones	
1	N/A
Flujo alternativo *.a El usuario cancela la acción	
1	Concluye el requisito.
Pos-condiciones	
1	No se registran los datos.
Validaciones	
1	Se validan los datos según lo establecido en el Modelo conceptual
2	Se valida que el número del comprobante sea único.
3	Se valida que el atributo observaciones es obligatorio.

Relaciones	**Requisitos Incluidos**	N/A
	Extensiones	N/A
Conceptos	**Entidad**	Visibles en la interfaz: descripción Utilizados internamente: código descripción dirección
	Punto de venta	Visibles en la interfaz: descripción Utilizados internamente: código descripción estado
	Departamento	Visibles en la interfaz: descripción Utilizados internamente: código descripción
	Cajero	Visibles en la interfaz: nombre Utilizados internamente: nombre expediente ci
	Comprobante de venta	Visibles en la interfaz: número estado fecha de emisión observaciones Utilizados internamente: Hora
Requisitos especiales	Son los requisitos no funcionales específicos para el requisito.	
Asuntos pendientes	Posibles mejoras al requisito.	

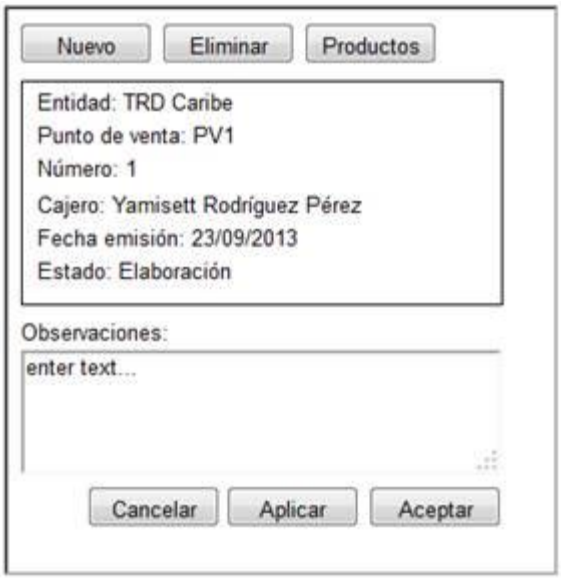


Figura 3: Prototipo de interfaz de usuario de Adicionar comprobante de venta.

2.2.4. Administración de requisitos

Las actividades relacionadas con la definición, clasificación, asignación, seguimiento y control de los requisitos durante todo el ciclo de vida de desarrollo de software están comprendidas dentro de la administración de requisitos. Se utilizó la herramienta Visual Paradigm para la administración de los requisitos, donde se definieron los siguientes elementos de trazabilidad:

- Procesos de negocio.
- Requisitos.
- Modelo conceptual (entidades del negocio).
- Componentes.
- Diseño de casos de prueba

Se realizó el diagrama de requisitos con el objetivo de definir la relación entre los elementos de trazabilidad mencionados anteriormente.

Se realizaron las siguientes matrices de trazabilidad:

- Requisitos-Modelo Conceptual (entidades del negocio).
- Requisitos-Componente.
- Requisitos-Procesos de negocio.
- Requisitos-Diseño de casos de prueba

2.2.5. Priorización de requisitos

Debido a limitaciones de tiempo y personal resulta difícil satisfacer o aplicar todos los requisitos que se han levantado o exigido para la construcción o mantenimiento de un sistema, por lo que se hace necesario definir cuáles son los requerimientos a implementar de acuerdo a una previa priorización y en mutuo acuerdo con el cliente. Uno de los objetivos de dicha estrategia es determinar cuáles de los requisitos son los principales a abordar para su desarrollo durante el ciclo de vida del respectivo producto. (Group, 1997-2000)

Para la priorización de los requisitos identificados se utilizó la plantilla Evaluación de requisitos establecida en el expediente de proyecto del CEIGE, determinando la complejidad de los requisitos, basada en los resultados de los principales indicadores que a continuación se describen:

- **Restricciones de validación:** Complejidad de todas las validaciones que lleve un requisito, tanto las validaciones en el lado del cliente, como en el servidor.
- **Grado de reutilización:** Complejidad de un requisito, para poder ser reutilizado por otros.
- **Lógica de negocio:** Los requisitos pueden presentar diferentes niveles de complejidad para la implementación de la lógica de negocio que contienen. Ejemplo: Operaciones y métodos matemáticos.

Se priorizaron para la implementación aquellos requisitos para los que se obtuvo una complejidad de categoría Alta.

En la siguiente tabla se muestra la evaluación de requisitos del grupo Gestionar comprobante de venta:

Tabla 4. Evaluación de requisitos de Gestionar comprobante de venta.

Requisitos	Restricciones de validación	Grado de reutilización	Lógica de negocio	Complejidad de desarrollo	Prioridad de negocio
RF1 Adicionar comprobante de venta	A	A	A	Alta	Alta
RF2 Modificar comprobante de venta	A	A	A	Media	Media
RF3 Adicionar producto al comprobante de venta	A	A	A	Media	Media
RF4 Modificar producto del comprobante de venta	M	A	M	Alta	Alta

RF5 Buscar producto del comprobante de venta	M	A	M	Baja	Baja
RF6 Cancelar estado del comprobante de venta	M	M	M	Baja	Baja
RF7 Contabilizar comprobante de venta	A	M	A	Media	Media

2.2.6. Validación de requisitos

La validación de los requisitos se realiza con la finalidad de comprobar que los requisitos identificados sean precisos, consistentes, realistas, verificables, definan lo que el usuario desea del producto final, que los errores que hayan sido detectados sean corregidos y el resultado del trabajo cumpla con los estándares establecidos para el proceso. (Pressman, 2005)

Se aplicaron las siguientes técnicas para la validación de requisitos:

Revisión técnica formal (RTF): El mecanismo primario para la validación de requisitos es la Revisión técnica formal. En ella el equipo de revisión (ingenieros de software, clientes, usuarios y otros interesados) examinaron la especificación de requisitos en busca de errores de contenido o de interpretación, inconsistencias, información incompleta y que los requisitos no fuesen discordantes o inalcanzables. Como resultado de la RTF se emitió un Registro de inconsistencias en el que se detallan las no conformidades y la evaluación de la revisión.

Prototipos de interfaz de usuario: Junto a la especificación de requisitos funcionales fueron modelados los prototipos de interfaz de usuario. Los mismos validan que los requisitos están en correlación con las necesidades plasmadas. Al aplicar esta técnica se ofrece como resultado que el cliente tenga una idea de la estructura de la interfaz de usuario teniendo una visión inicial del sistema que se quiere obtener.

2.2.7. Requisitos no funcionales (RNF)

Son propiedades o cualidades que el producto debe tener, que lo hacen atractivo, usable, rápido y/o confiable. Existen múltiples categorías para clasificar a los Requisitos no funcionales.

Atributos de calidad por los que fueron agrupados los requisitos no funcionales:

- **Usabilidad:**

RNF1. Los mensajes de información o de error que se muestran deberán brindarle una explicación básica y entendible al usuario.

RNF2. Lograr la menor cantidad de clic para acceder a las funcionalidades del sistema que requiera el usuario.

RNF3. Los mensajes, títulos y demás textos que aparezcan en las interfaces del sistema deben aparecer en idioma español.

- **Confiabilidad:**

RNF4. El sistema tendrá un respaldo de la información en el centro de datos, permitiendo la recuperación ante la pérdida parcial o total de la información; permitiendo configurar las salvas automáticas de la base de datos cada un período de tiempo determinado.

RNF5. Ante el fallo de una funcionalidad del sistema, el resto de las funcionalidades que no dependen de esta deberán seguir funcionando; en caso que existan funcionalidades que dependan de la funcionalidad que sufrió el fallo, el sistema deberá notificar al usuario la imposibilidad de realización debido a la causa original.

RNF6. El sistema no permitirá la entrada de datos incorrectos.

RNF7. En caso que se requiera el sistema impondrá campos obligatorios para garantizar la integridad de la información que se introduce por el usuario.

- **Eficiencia:**

RNF8. El sistema, para operaciones de inserción, modificación o búsqueda de datos, deberá tener respuestas a peticiones del cliente en un tiempo máximo de 10 minutos.

- **Soporte:**

RNF9. La codificación del sistema será estándar y las funcionalidades serán comentadas.

- **Interfaz:**

Interfaces Hardware

RNF 10. El sistema interactuará con impresoras para imprimir los diferentes documentos que genere la aplicación como respuesta a las funcionalidades del sistema.

RNF 11. El sistema para su instalación en las máquinas clientes requiere: 512MB de RAM, Procesador Intel Pentium 3.0 GHZ y una tarjeta de red.

RNF12. El sistema para su instalación en el servidor de aplicación requiere: 4GB de RAM, Procesador Core2 Duo 2.2 GHz o superior y una tarjeta de red.

RNF13. El sistema para su instalación en el servidor de base de datos requiere: 1GB de RAM (recomendado 4GB), Procesador Intel Pentium 3.0 GHZ, disco duro: 160 GB (recomendado 500GB) y una tarjeta de red.

Interfaces Software

RNF14. Para la PC cliente se usarán navegadores como Internet Explorer, Mozilla Firefox y Google Chrome.

RNF15. Para el servidor de aplicación se deberá instalar Apache 2.2 y para el servidor de la base de datos PostgresSQL 8.3.

- **Portabilidad:**

RNF16. El sistema será multiplataforma (Linux y Windows) respecto al cliente.

- **Seguridad:**

RNF17. Para conservar la seguridad e integridad de la información será necesario que todos los usuarios se autentiquen, permitiéndoles acceder a las funcionalidades habilitadas para rol asociado.

2.3. Conclusiones parciales

Las revisiones realizadas permitieron comprobar que los artefactos generados como parte de la modelación del negocio del subsistema Punto de venta son válidos y que la información brindada coincide con el flujo de actividades de cada proceso. Se identificaron un total de 44 requisitos funcionales mediante técnicas utilizadas, realizándose posteriormente la validación de los mismos. Se describieron los requisitos no funcionales con los que debe contar el subsistema para que se muestre usable y confiable al interactuar con el usuario final. Para tener constancia de las revisiones realizadas se emitió un Acta de aceptación para cada validación ejecutada.

CAPÍTULO 3: DISEÑO, IMPLEMENTACIÓN Y PRUEBAS

3. Introducción

En el presente capítulo se describe el diseño de la solución propuesta, se muestran los diagramas de clases, los mecanismos de diseño aplicados y los diagramas de secuencia conformados a partir de los patrones empleados en la modelación y los artefactos generados en el análisis. El diseño propuesto fue validado mediante métricas para verificar la correcta elaboración del mismo. Se muestra el modelo de datos, diagrama de componentes y el modelo de despliegue del subsistema Punto de venta. Para un mejor entendimiento del código se definen los estándares de codificación empleados. Para la validación del subsistema Punto de venta se le realizan las pruebas de caja negra y caja blanca, analizando los resultados obtenidos con el objetivo de confirmar que el subsistema desarrollado contenga la calidad necesaria para su liberación.

3.1. Diseño

Dentro de los flujos de trabajo más importantes en el desarrollo del subsistema se encuentra el diseño, que genera varios artefactos como son: los prototipos IU[11] funcionales, los diagramas de clases, diagramas de secuencia, y el inicio del modelo de datos. El diseño es creado en la construcción del software para facilitar varias vistas del sistema y tiene como objetivo: entregar las funciones requeridas por el usuario de manera que responda una especificación funcional dada, o sea, encontrar y describir una forma para implementar los requisitos funcionales del sistema, respetando las restricciones impuestas por los requisitos no funcionales.

3.1.1. Mecanismos de diseño

Los mecanismos de diseño se utilizan con el objetivo de abreviar los diagramas de clases. Cada diseñador establece sus propios mecanismos de diseño, teniendo siempre en cuenta los patrones y estilos seleccionados. (Verdecia, 2013)

En el subsistema Punto de venta se definieron los siguientes mecanismos:

Mecanismo de diseño para las clases controladoras

En el diseño propuesto las clases controladoras definidas, heredaran de la clase ZendExt_Controller_Secure, ya que contiene numerosas funcionalidades para todas de modo común como se muestra en la figura 4.

[11] IU: Interfaz de Usuario.

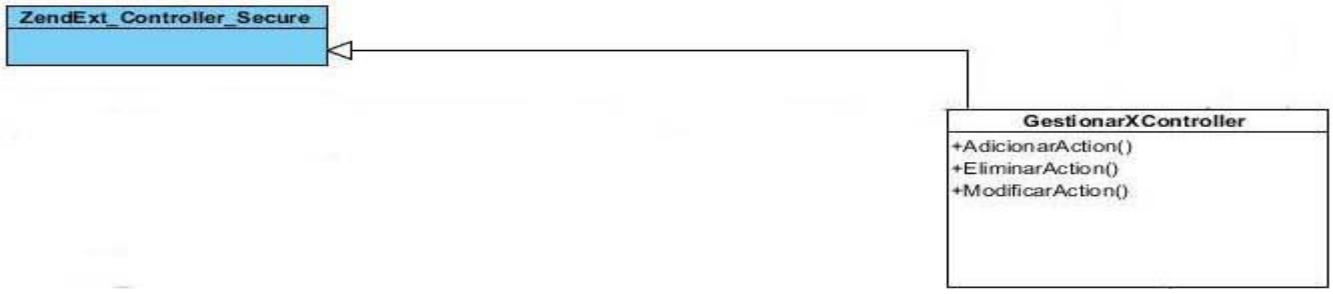


Figura 4: Mecanismo de diseño para las clases controladoras.

Mecanismo de diseño para las clases del dominio

La clase del dominio (Domain) contienen a la clase DatX, que emplean a la clase Doctine-Query del marco de trabajo Doctrine, para lograr la comunicación y acceso a la base de datos utilizando el lenguaje de Doctrine DQL.

Todas estas clases son heredadas de la clase denominada BaseDatX que contiene los atributos mapeados de las tablas de la basa de datos. Las tablas que heredan de Doctrine_Record son las tablas que contienen como inicio el prefijo Base como BaseDatX, que permite agrupar en registros u objetos mapeados los datos de las tablas de la base de datos. Véase figura 5.

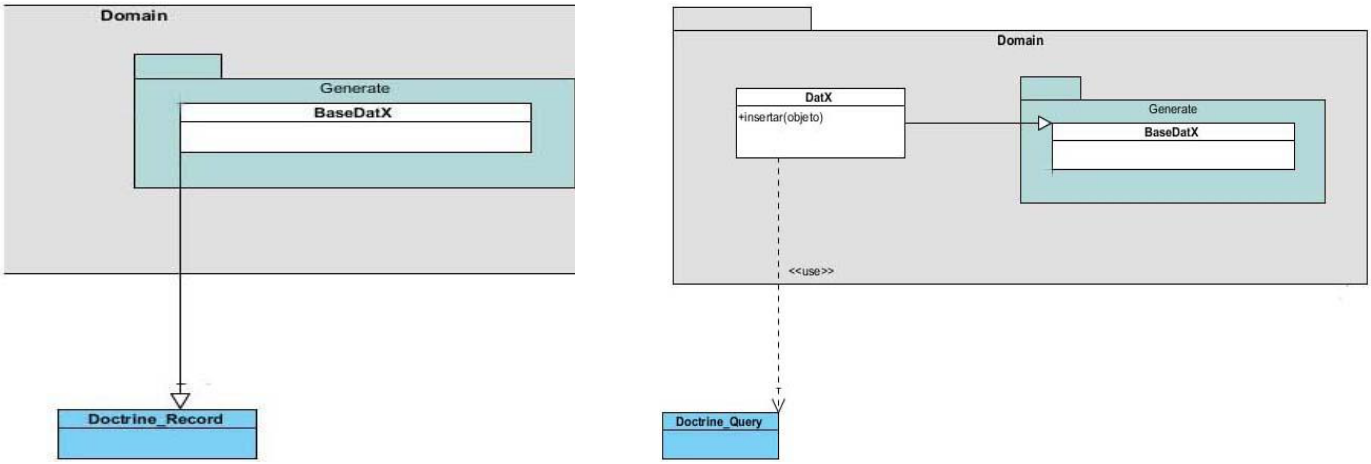


Figura 5: Mecanismo de diseño para las clases del dominio.

3.1.2. Diagrama de clases del diseño

Para cada paquete de requisitos se realiza un diagrama de clases del diseño, estos diagramas son de estructura estática que muestran las clases del sistema con sus atributos, métodos y visibilidad; y las relaciones entre ellas que pueden ser de agregación, asociación y herencia.

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces. Además, tiene como objetivo modelar la vista de diseño estática de un sistema siendo capaz de representar las clases, las relaciones entre ellas y los componentes de cada clase.

En el desarrollo del subsistema se modelaron diagramas de clases del diseño para las agrupaciones de los requisitos funcionales del subsistema Punto de venta. Ver Figura 6.

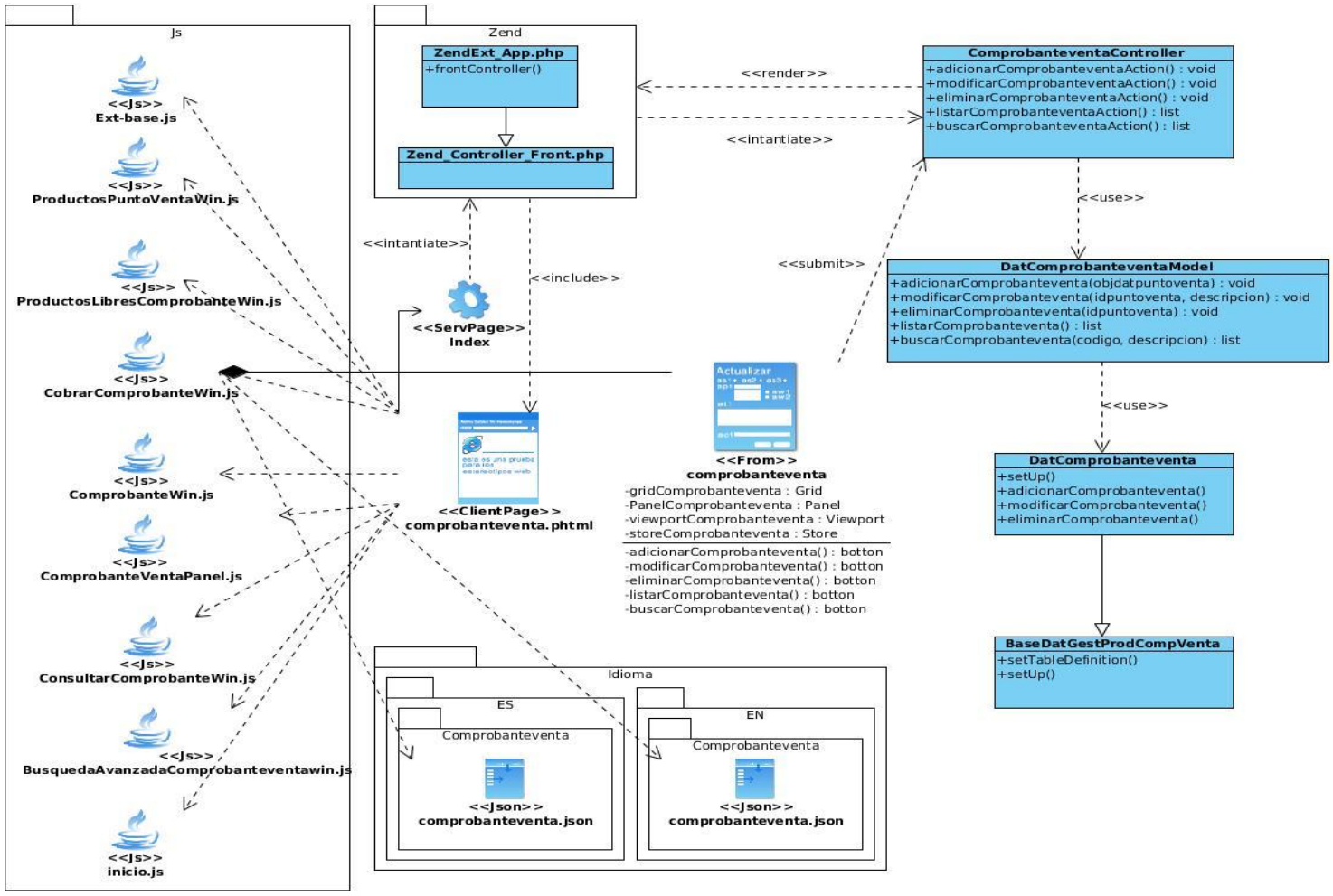


Figura 6: Diagrama de clases del diseño de la agrupación Gestionar productos al comprobante de venta.

3.1.3. Descripción de las clases del diseño

A continuación se presenta la descripción de la clase DatcomprobanteventaModel.

Tabla 5. Descripción de la clase DatcomprobanteventaModel.

Nombre : DatComprobanteventaModel	
Tipo de clase: Modelo	
Nombre	**Descripción**
adicionarComprobanteventa (idcompventa)	A partir de los datos suministrados por el cliente se realiza la inserción de los mismos a la base de datos.
modificarComprobanteventa (idcompventa)	Permite seleccionar el comprobante de venta a modificar y se le introducen los nuevos valores a guardar.
eliminarComprobanteventa (idcompventa)	Permite seleccionar el comprobante de venta a eliminar y se elimina confirmando la acción a realizar.
buscarComprobanteventa (idcompventa)	Permite mediante el id del comprobante de venta realizar una búsqueda por los valores insertados en la tabla dat_comprobanteventa en la base de datos.
listarComprobanteventa ()	Permite listar los comprobantes de venta que se van insertando en la base de datos y mostrarlos.

Tabla 6. Descripción de la clase ComprobanteventaController.

Nombre : ComprobanteventaController	
Tipo de clase: Controlador	
Para cada responsabilidad:	
Nombre	**Descripción**
adicionarComprobanteventaAction()	Adiciona un comprobante de venta al punto de venta.
modificarComprobanteventaAction()	Modifica un comprobante de venta al punto de venta.
eliminarComprobanteventaAction ()	Elimina un comprobante de venta del punto de venta.
buscarComprobanteventaAction ()	Busca un comprobante de venta del punto de venta según diferentes parámetros.
listarComprobanteventaAction ()	Lista los comprobantes de venta del punto de venta.

Tabla 7. Descripción de la clase DatComprobanteventa.

Nombre : DatComprobanteventa	
Tipo de clase: Modelo de dominio	
Para cada responsabilidad:	
Nombre	**Descripción**
setUp()	Constructor de la clase.
adicionarComprobanteventa($idcompventa**)**	Adiciona comprobante de ventas.
modificarComprobanteventa ($idcompventa**)**	Modifica comprobante de ventas.
eliminarComprobanteventa ($idcompventa**)**	Elimina comprobante de ventas.

3.1.4. Diagrama de secuencia

Un diagrama de secuencia es un diagrama de interacción que destaca el orden temporal de los mensajes. Se utiliza para modelar los aspectos dinámicos de un sistema. Gráficamente, es una tabla que representa objetos, colocados a lo largo del eje X, y mensajes, ordenados según suceden en el tiempo, a lo largo del eje Y. (Rojas, 2007)

Se muestra a continuación un ejemplo de diagrama de secuencia.

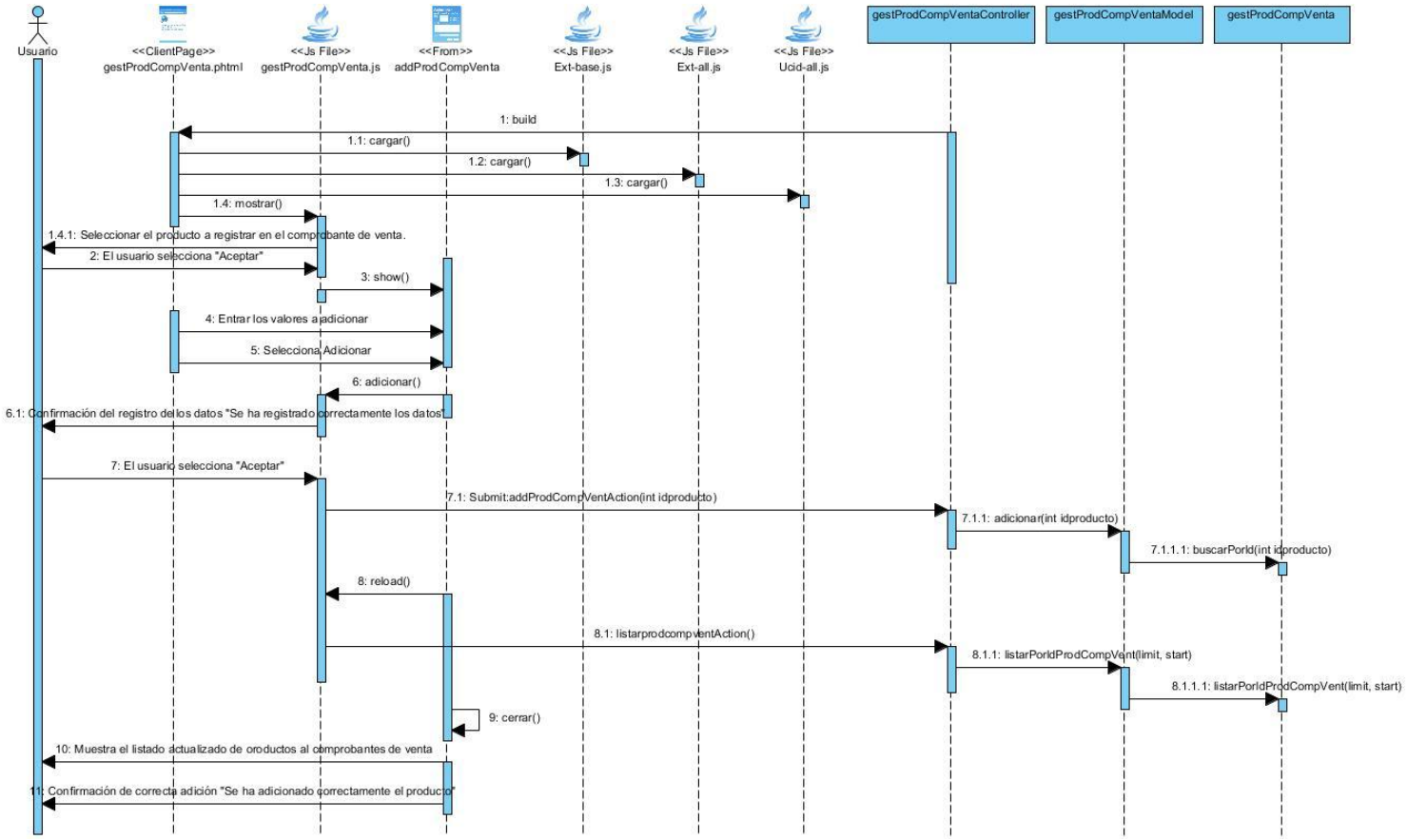


Figura 7: Diagrama de secuencia del requisito Adicionar productos al comprobante de venta.

3.1.5. Patrones de diseño

El diseño es un modelo del sistema, realizado con una serie de principios y técnicas, que permite describir el sistema con el suficiente detalle como para ser implementado. Un patrón es una solución probada que se puede aplicar con éxito a determinados tipos de problemas que aparecen repetidamente en el desarrollo de sistemas software. Puede ser empleado muchas veces, en diferentes contextos, sin tener que duplicar el diseño. Se trata de un elemento de diseño que puede ser reutilizado. (Pimentel, 2007)

Pero los principios y reglas no son suficientes, en el contexto de diseño se puede observar que los buenos ingenieros tienen esquemas y estructuras de solución que usan numerosas veces en función del contexto del problema. Estos esquemas y estructuras son conceptos reusables denominados patrones del diseño. (Larman., 2004)

Patrones GRASP[12]

Los patrones GRASP tienen como objetivo la descripción de los principios fundamentales del diseño para asignarle responsabilidades a los objetos.

Creador: es el patrón responsable de asignarle a una clase la responsabilidad de crear una instancia de otra clase. Se utiliza en el subsistema PV para cada clase controladora que atiende a las peticiones de las funcionalidades de dicho subsistema. Ejemplo: “ConfigurarnomcategoriaController.php”, “ComprobanteventaController.php”, “CfgPuntoVentaController.php”.

Bajo acoplamiento: es el patrón que se encarga de medir el nivel en que una clase depende o está conectada a otra, basándose en la idea de desligar la relación entre las clases de manera que si se tiene que hacer alguna modificación o cambio en una de ellas, se tenga la mínima repercusión posible en las clases que dependen de la clase modificada. Ejemplo: “DatComprobanteventaModel”, “DatProdcompventaModel”, “DatProdventaModel”.

Alta cohesión: es el patrón que mide el grado de responsabilidad que tiene una clase, de manera que las responsabilidades otorgadas a esa clase estén relacionadas, o sea, que toda la información que se almacena en una clase sea coherente y esté relacionada en medida de las posibilidades que se pueda con dicha clase. La aplicación de este patrón se evidencia en las clases modelos. Ejemplo: “NomCategoriatpvModel.php”, “DatProdVentaModel.php”, “DatPuntoventaModel.php”.

Controlador: es el encargado de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase. Permite asignar la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. El controlador delega en otros objetos el trabajo que se necesita hacer; coordina o controla la actividad. Este patrón se aplicó en las clases controladoras del subsistema. Ejemplo: “DatPuntoVentaController.php”, “ComprobanteventaController.php”, “ProdVentaController.php”.

Patrones GOF[13]

Los patrones de diseño GoF se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

[12] GRASP (General Responsibility Assignment Software Patterns): Patrones generales de software para asignación de responsabilidades.

[13] GOF (Gang of Four): La banda de los cuatro.

- Creacionales: tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
- Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
- Comportamiento: Los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Decorator (categoría: Estructurales): es el patrón que se emplea para mejorar la funcionalidad de una clase a la que envuelve, que ya funciona sin él, pero no altera las salidas que ésta produce. Este patrón se emplea en la clase “ViewScript.php”.

Singleton (categoría: Creacionales): patrón que restringe la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. En el desarrollo de la aplicación fue usado para proveer una instancia única de la clase “Conexion.php” usada por Doctrine para acceder a la base de datos.

Builder (categoría: Estructurales): este patrón se aplica al problema de creación de objetos. Es un caso del patrón Strategy ya que determina el uso al realizar el intercambio de mensajes entre diferentes objetos para resolver una tarea. Fue utilizado en la implementación de las interfaces con ExtJS, permitiendo separar la creación de componentes visuales de su utilización. Se implementó en la clase “builder.js”.

3.1.6. Métricas para evaluar el diseño propuesto

Con el objetivo de comprobar que las clases estén bien definidas, se emplean las métricas Tamaño operacional de clases (TOC) y Relaciones entre clases (RC), diseñadas para evaluar los siguientes atributos de calidad:

- **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 costos y la planificación del proyecto.
- **Cantidad de pruebas:** Consiste en el número o el grado de esfuerzo para realizar las pruebas de calidad (unidad) del producto (componente, módulo, clase, conjunto de clases, etc.) diseñado.

Métrica Tamaño Operacional de Clases (TOC):

La métrica Tamaño Operacional de Clases calcula el número de operaciones asignadas a una clase y evalúa los siguientes atributos de calidad:

- **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 del grado de reutilización de la clase.

Para la aplicación de esta métrica se tuvo en cuenta la cantidad de procedimientos que contenían las clases seleccionadas a las que se les aplicará dicha métrica. Después se pasó a realizar el cálculo del promedio de los procedimientos y utilizando un criterio se obtuvo la categoría (Alta, Media, Baja) para la Responsabilidad, Complejidad de implementación y Reutilización.

En la siguiente tabla están las fórmulas por las que se evaluaran el tamaño operacional de las clases del subsistema, además, muestra las categorías que pueden tener las clases respecto a los atributos de calidad que mide la métrica TOC y están diferenciados en otro color la peor categoría que puede tener una clase en un atributo determinado.

Tabla 8. Cálculo del promedio (prom) de los procedimientos mediante la métrica TOC.

	Categoria	Criterio
Responsabilidad	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Complejidad implem	Baja	< =Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	> 2* Prom.
Reutilización	Baja	> 2*Prom.
	Media	Entre Prom. y 2* Pom.
	Alta	<= Prom.

La tabla que se muestra a continuación ofrece las clases del sistema a las que se le aplicó la métrica, la cantidad de procedimientos de cada una y la categoría obtenida para cada atributo evaluado.

Tabla 9. Categoría de las clases según los atributos de la métrica TOC.

No	Subsistema	Clase	Cantidad de Procedimientos	Responsabilidad	Complejidad	Reutilización
1	Punto venta	NomCategoriatpvController	5	Baja	Baja	Alta
2	Punto venta	NomCategoriatpvModel	9	Media	Media	Media
3	Punto venta	DatComprobanteventaController	5	Baja	Baja	Alta
4	Punto venta	DatComprobanteventaModel	9	Media	Media	Media
5	Punto venta	DatCfgpuntoventaModel	11	Media	Media	Media
6	Punto venta	DatCfgpuntoventaController	11	Media	Media	Media
7	Punto venta	DatProdventaController	9	Media	Media	Media
8	Punto venta	DatProdventaModel	9	Media	Media	Media
9	Punto venta	DatProdcompventaModelController	6	Baja	Baja	Alta
10	Punto venta	DatProdcompventaModel	6	Baja	Baja	Alta
11	Punto venta	NomFormadepagoController	1	Baja	Baja	Alta
12	Punto venta	NomFormadepagoModel	1	Baja	Baja	Alta
13	Punto venta	DatDepositotpvController	1	Baja	Baja	Alta
14	Punto venta	DatDepositotpvModel	1	Baja	Baja	Alta

Las gráficas que se muestran a continuación muestran los resultados obtenidos para cada uno de los atributos medidos.

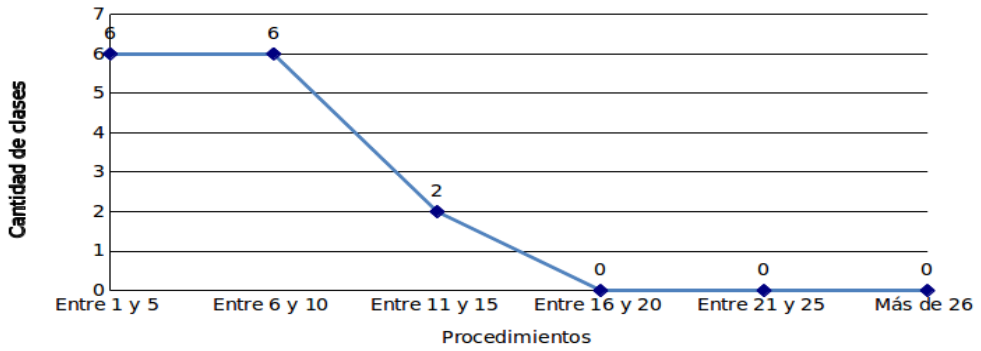


Figura 8: Resultados obtenidos de la aplicación de la métrica TOC por intervalos definidos.

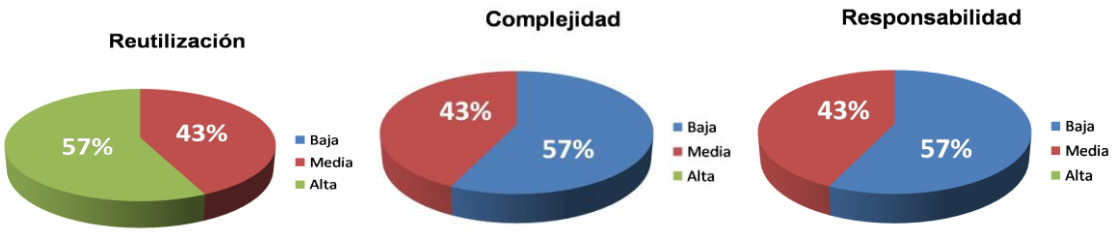


Figura 9: Representación en % de los resultados de la evaluación mediante la métrica TOC en los atributos: Reutilización, Complejidad de implementación y Responsabilidad.

En el análisis de los resultados obtenidos para los atributos de la métrica TOC en la evaluación de las clases del subsistema, se puede observar que todas las clases para los atributos responsabilidad y complejidad están dentro de las categorías Media y Baja, mientras que el atributo reutilización cuenta con

igual evaluación en las categorías Alta y Media demostrando así que el subsistema cuenta con una elevada reutilización, baja complejidad y responsabilidad en el diseño propuesto. Por lo que se concluye que los resultados obtenidos según esta métrica son positivos.

Métrica Relaciones entre clases (RC):

La métrica Relaciones entre clases está dada por el número de relaciones de uso de una clase con otra(s) y evalúa los siguientes atributos de calidad:

- **Acoplamiento:** Un aumento de las RC implica un aumento del acoplamiento de la clase.
- **Complejidad del mantenimiento:** Un aumento de las RC implica un aumento de la complejidad del mantenimiento de la clase.
- **Reutilización:** Un aumento de las RC implica una disminución en el grado de reutilización de la clase.
- **Cantidad de pruebas:** Un aumento de las RC implica un aumento de la Cantidad de pruebas de unidad necesarias para probar una clase.

En la siguiente tabla están las fórmulas por las que se evaluaran las relaciones entre las clases del subsistema, además, muestra las categorías que pueden tener las clases respecto a los atributos de calidad que mide la métrica RC y están diferenciados en otro color la peor categoría que puede tener una clase en un atributo determinado.

	Categoria	Criterio
Acoplamiento	Ninguno	0
	Bajo	1
	Medio	2
	Alto	>2

	Categoria	Criterio
Complejidad Mant	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

	Categoria	Criterio
Reutilización	Baja	>2* Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	<= Prom.

	Categoria	Criterio
Cantidad de Prueb	Baja	<= Prom.
	Media	Entre Prom. y 2*Prom.
	Alta	> 2*Prom.

Figura 10: Cálculo del promedio (prom) de la cantidad de relaciones de una clase mediante la métrica RC.

La figura que se muestra a continuación ofrece las clases del sistema a las que se le aplicó la métrica, la cantidad de relaciones entre cada una y la categoría obtenida para cada atributo evaluado.

No	Subsistema	Clase	Cantidad de Relaciones	Acoplamiento	Complejidad	Reutilización	Cantidad de Pruebas
1	Punto de venta	NomCategoriatpvController	1	Bajo	Media	Media	Media
2	Punto de venta	NomCategoriatpvModel	0	Ninguno	Baja	Alta	Baja
3	Punto de venta	DatComprobanteventaController	2	Medio	Alta	Baja	Alta
4	Punto de venta	DatComprobanteventaModel	0	Ninguno	Baja	Alta	Baja
5	Punto de venta	DatCfgpuntoventaModel	0	Ninguno	Baja	Alta	Baja
6	Punto de venta	DatCfgpuntoventaController	1	Bajo	Media	Media	Media
7	Punto de venta	DatProdventaController	0	Ninguno	Baja	Alta	Baja
8	Punto de venta	DatProdventaModel	2	Medio	Alta	Baja	Alta
9	Punto de venta	DatProdcompventaModelController	2	Medio	Alta	Baja	Alta
10	Punto de venta	DatProdcompventaModel	0	Ninguno	Baja	Alta	Baja
11	Punto de venta	NomFormadepagoController	1	Bajo	Media	Media	Media
12	Punto de venta	NomFormadepagoModel	0	Ninguno	Baja	Alta	Baja
13	Punto de venta	DatDepositotpvController	1	Bajo	Media	Media	Media
14	Punto de venta	DatDepositotpvModel	0	Ninguno	Baja	Alta	Baja

Figura 11: Categoría de las clases según los atributos de la métrica RC.

Las gráficas que se muestran a continuación muestran los resultados obtenidos para cada uno de los atributos medidos.

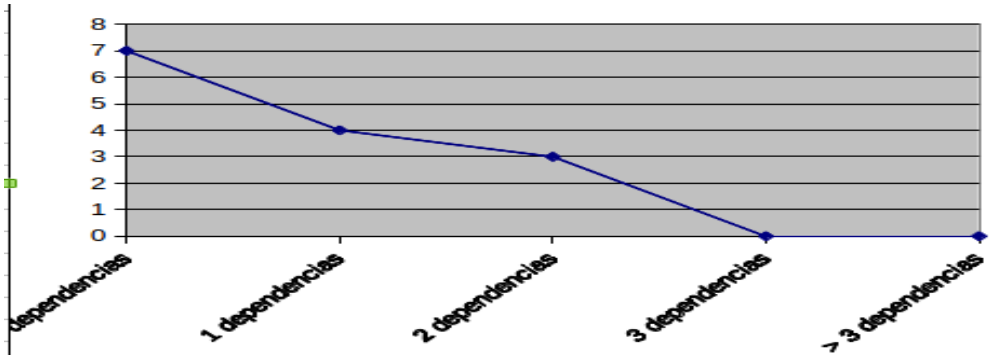


Figura 12: Resultados obtenidos de la aplicación de la métrica RC por intervalos definidos.

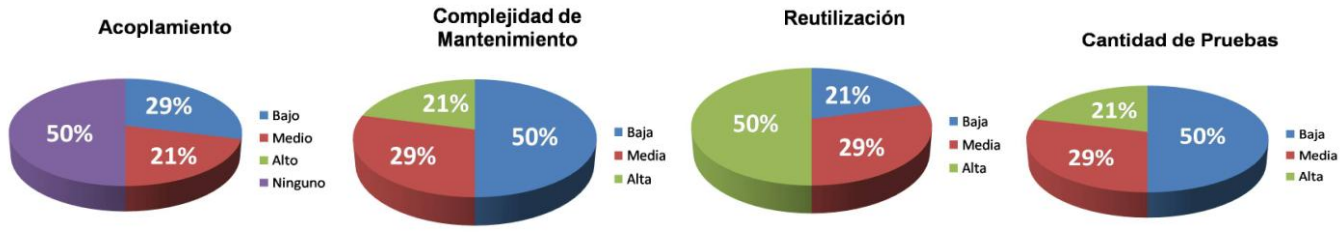


Figura 13: Representación en % de los resultados de la evaluación mediante la métrica RC en los atributos: Acoplamiento, Complejidad de Mantenimiento, Reutilización y Cantidad de pruebas.

Los resultados obtenidos durante la evaluación de las clases y sus respectivas relaciones, mediante los atributos de la métrica RC, demuestran que las clases del diseño poseen un bajo acoplamiento, ya que para este atributo las categorías ninguno y bajo sumaron un 79% del total, mostrando igual por ciento en las categorías alta y media del atributo reutilización. Los atributos complejidad de mantenimiento y cantidad de pruebas, sumaron un 79 % igualmente en las categorías baja y media, lo que demuestra que

no es necesario un elevado esfuerzo en el momento de realizar cambios, rectificaciones y pruebas al software.

3.1.7. Modelo de datos

Con el modelo de datos se especifican los conceptos que se tratan en el sistema y que sirven para describir la estructura de la base de datos diseñada.

Un modelo de datos es la combinación de una colección de estructuras de datos, reglas, y convenciones que pueden utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Constituye un elemento clave en el diseño de la arquitectura de un manejador de base de datos. (Colima, 2010)

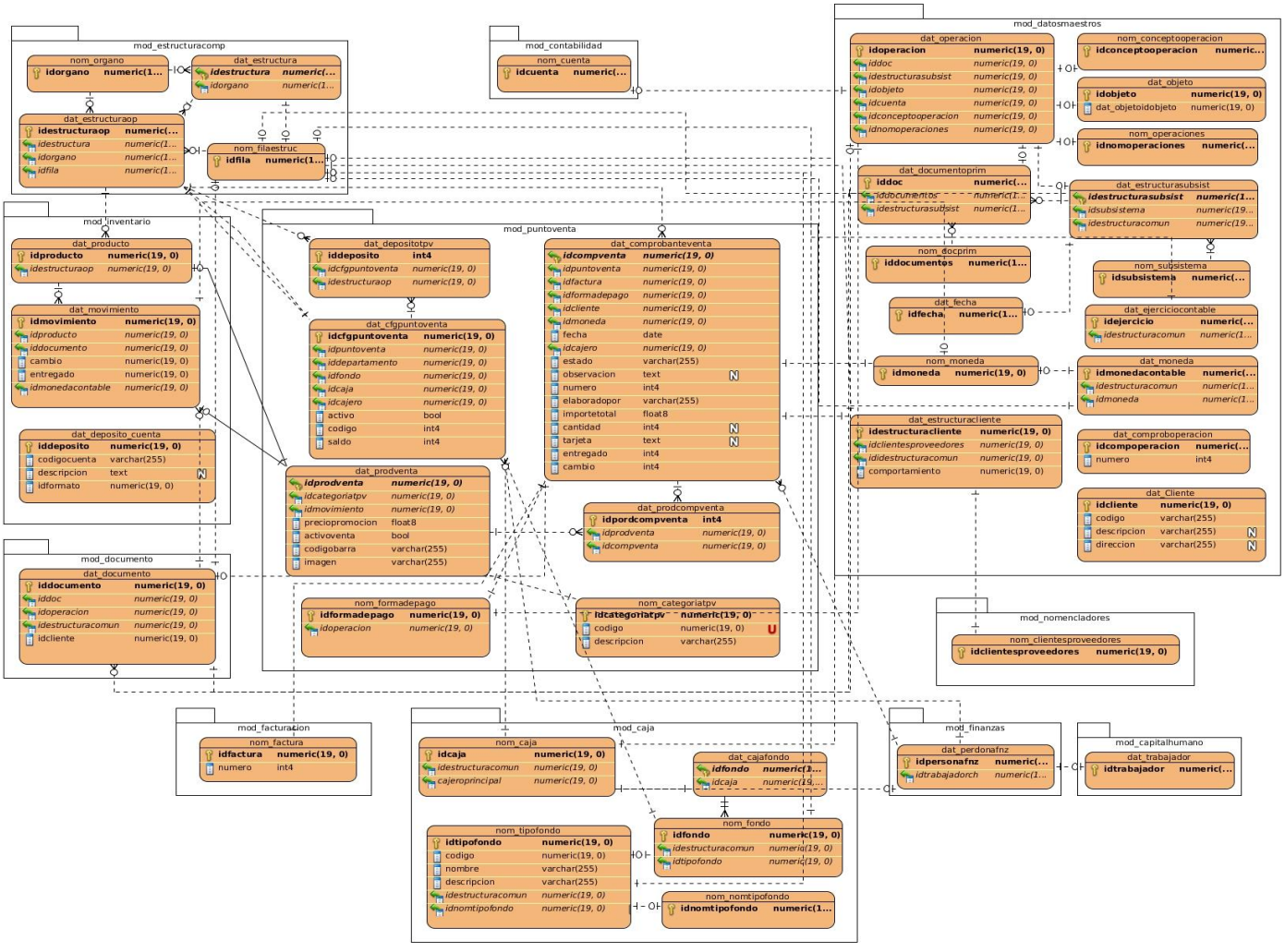


Figura 14: Modelo de Datos del subsistema Punto de venta.

Tabla 10. Descripción de las tablas del modelo de datos.

Nombre	Descripción	Relaciones	Observaciones
dat_puntoventa	Tabla que almacena la información referente al punto de venta.	Se relaciona con el esquema mod_estructuracomp y mod_caja mediante las tablas dat_estructura y dat_cajafondo	La relación se complementa mediante las pk(idestructura y idfondo y idcaja) de dat_estructura y dat_cajafondo que pasa como fk a dat_puntoventa.
cfg_puntoventa	Tabla que almacena la información de la configuración de los puntos de ventas	Se relaciona con el esquema mod_estructura y la tabla dat_estructura	La relación se complementa mediante las pk(idestructura) de dat_estructura que pasa como fk a dat_puntoventa.
dat_prodventa	Tabla que almacena la información referente al producto venta.	Se relaciona con el esquema mod_inventario y las tablas dat_producto y dat_movimiento y del esquema mod_puntoventa con la tabla nom_categoriatpv	La relación se complementa mediante las pk de dat_producto, dat_movimiento y nom_categoriatpv que pasa como fk a dat_prodventa.
nom_categoriatpv	Tabla que almacena la información referente a la categoría del punto de venta.	Se relaciona con el esquema mod_puntoventa y la tabla dat_prodventa	La relación se complementa mediante la pk de dat_pordventa que pasa como fk a nom_categoriatpv.
nom_formadepago	Tabla que almacena la información referente a la forma de pago.	Se relaciona con el esquema mod_caja y la tabla dat_cajafondo	La relación se complementa mediante las pk(idfondo y idcaja) de dat_cajafondo que pasa como fk a nom_formadepago.

3.2.1. Diagrama de componentes

El diagrama de componentes es la representación del sistema de software en componentes y dependencias que existen entre estos, se establecen contratos para la interacción entre componentes, que no son más que servicios o procedimientos que solicitan los componentes.

Mediante el modelo conceptual, las especificaciones de los requisitos funcionales y los prototipos de interfaz de usuarios realizados, se logró definir la estructura interna del módulo, tomando como base las responsabilidades arquitectónicas de cada uno de sus componentes, aumentando la cohesión y disminuyendo el acoplamiento, definiendo los conectores, las relaciones y las restricciones de cada componente y del subsistema en su totalidad.

En la figura se muestra el diagrama de componentes del subsistema Punto de venta donde se puede evidenciar como se integran los componentes a través del mecanismo de Inversión de Control (IoC: Inversion of Control), lo que permite implementar la responsabilidad de cada componente por separado, e integrarlos posteriormente, evitando que estos conozcan el funcionamiento interno de otros componentes y estableciendo la comunicación entre ellos a través de contratos bien definidos.

En la figura 15 se muestra el diagrama de componentes del subsistema de Punto de venta:

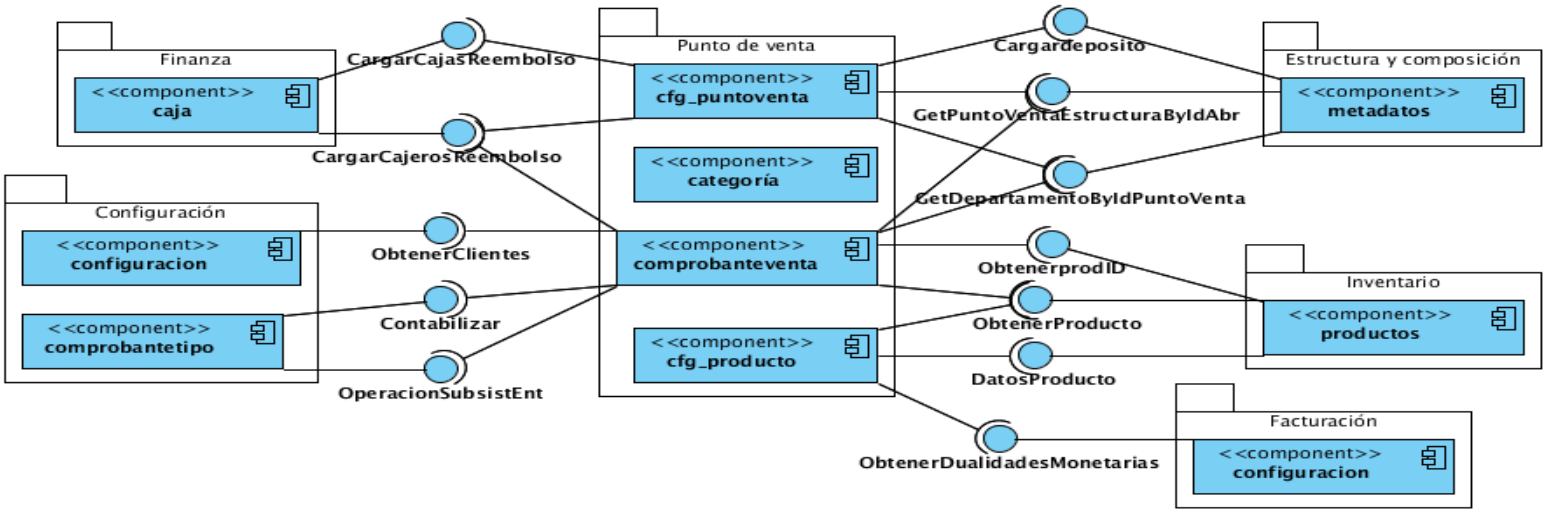


Figura 15: Diagrama de componentes del subsistema Punto de venta.

3.2. Implementación

El flujo de trabajo de implementación describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a la arquitectura definida.

3.2.2. Estructura del componente Comprobante de venta

Dentro de la carpeta raíz del sistema Xedro-ERP se encuentran definidas diferentes carpetas, entre las que se encuentran “apps” y “web”, que contienen la lógica de negocio y las vistas de los componentes de cada subsistema respectivamente. Las carpetas contenidas en Punto de venta llevan el nombre del componente que representan, “comprobanteventa” es una de esas carpetas como se puede observar en la figura 16.

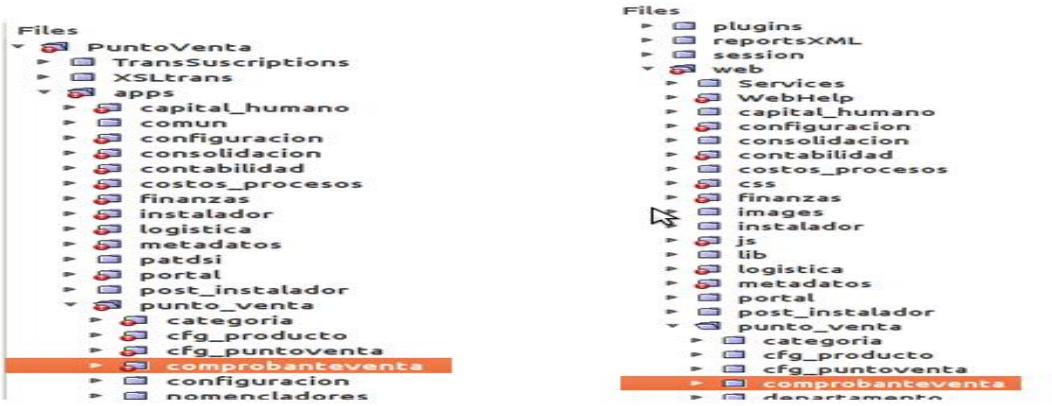


Figura 16: Comprobante de venta en las carpetas apps y web.

La carpeta “apps” contiene la carpeta “común” que incluye el fichero de tipo “.xml”, nombrado ioc en el cual se publican los servicios que brinda cada componente de cada subsistema, de esa manera se tomaron los servicios que consume el subsistema Punto de venta de los demás subsistemas, esto se puede observar en el diagrama de componentes que se muestra más adelante.

Contenido de Comprobante de venta en la carpeta “apps”

El componente Comprobante de venta, muestra la estructura de carpetas que se observa en la siguiente figura en la carpeta apps, dicha estructura es común para el resto de los componentes del subsistema.

Figura 17: Contenido del componente Comprobante de venta en la carpeta apps.

controllers: Es la carpeta que almacena las clases controladoras, que están encargadas de la comunicación entre las interfaces de usuario y la lógica del negocio de la solución.

models: En esta carpeta se encuentran las carpetas bussines y domain encargadas de la lógica del negocio y del acceso a datos del componente respectivamente.

services: Esta carpeta contiene la(s) clase(s) que se utiliza(n) para interactuar con los servicios que brinda el componente, permitiendo el acceso a las funcionalidades desde otros componentes haciendo uso del fichero ioc.xml.

views: En esta carpeta se recopilan los ficheros que van a gestionar la capa de presentación, estos ficheros se agrupan en dos carpetas:

- **idioma:** Contiene ficheros de tipo .json que recopilan etiquetas para la gestión de los mensajes vinculados a la presentación.
- **scripts:** En este directorio se incluyen todas las vistas, para ello se crea una carpeta para cada clase controladora y dentro se incluye la vista o script, archivos de extensión .phtml donde se especifica el título de la página que se gestiona y se carga el archivo .js que mostrará la presentación.

Contenido de Comprobante de venta en la carpeta "web"

El componente Comprobante de venta, muestra la estructura de carpetas que se observa en la siguiente figura en la carpeta web, dicha estructura es común para el resto de los componentes del subsistema.

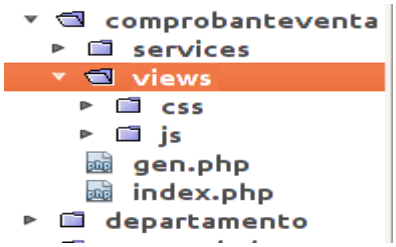


Figura 18: Contenido del componente Comprobante de venta en la carpeta web.

css: En este directorio se encuentran las plantillas y estilos para el diseño del componente.

js: Es la carpeta donde se incluyen las clases JavaScript, ficheros con la extensión .js donde se encuentra el código correspondiente a la capa de presentación (interfaces de usuario).

3.2.3. Estándares de codificación

Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código fuente. Pueden ser utilizados como técnicas a tener en cuenta para la implementación de una aplicación informática, en dependencia del lenguaje de programación que se utilice, permitiendo que a otros programadores se les haga menos complejo entender su código.

A continuación se declaran los estándares de codificación que fueron utilizados para la implementación del subsistema de punto de venta:

Nomenclatura de las clases:

- Las clases controladoras que están presentes en el desarrollo del subsistema comienzan su nombre con la primera letra en mayúscula, tendrán el nombre según la información que gestionan y se le adicionará al final la palabra "Controller". Ejemplo: "ComprobanteventaController.php".
- Para las clases del modelo (carpeta "models") que se encuentran dentro de la carpeta "bussines" el nombre comenzará con la primera letra en mayúscula y el resto en minúscula y al final se le pondrá "Model". Ejemplo: "ProdventaModel.php".
- Las clases del modelo que se encuentran dentro de la carpeta "domain" son nombradas igual que en las tablas de la base de datos. Estas clases son generadas mediante el mapeo a dicha base de datos que se hace con la herramienta Doctrine Generator, la cual fue desarrollada por el CEIGE.

  Ejemplo: "NomCategoriatpv.php".

- Las clases contenidas en la carpeta “generated” que se encuentra dentro de la carpeta “domain” del modelo son generadas por el mapeo a la base de datos, delante del nombre se pone la palabra “Base” y seguido el nombre de la tabla en la Base de Datos. Ejemplo: “BaseProdventa.php”.
- El nombre de las clases de la vista (.js) comenzará con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se empleará la notación PascalCasing, en la que cada vez que se cambie de palabra, la primera letra será en mayúscula.

  Ejemplo: “NomCategoriaTPVGrid.js”.

Nomenclatura de los métodos:

- El nombre de los métodos o funciones de las clase comenzará en minúscula, en caso de que dicho nombre sea compuesto se utilizará la notación CamelCasing, la cual establece que a partir de la segunda palabra, todas comenzarán en mayúscula. En caso de que sea una función de una clase controladora se le agregará al final la palabra “Action”.

  Ejemplo: “obtenerPuntosVentaAction”, “obtenerNomencladorAction”.

Nomenclatura de las variables:

- El nombre de las variables comenzará con minúscula, en caso de que el nombre de estas sea compuesto, se utilizará la notación CamelCasing. En la Vista, las variables que contengan componentes que formen parte de la interfaz, se les colocará delante un sufijo que indique el tipo de componente que contiene. Ejemplo: “btnAdicionar”, “btnModificar”, “btnEliminar”.

3.2.4. Algunas de las funcionalidades implementadas

Mediante las interfaces se muestra la estructura del subsistema Punto de venta entre los demás subsistemas del sistema Xedro-ERP, en PV se encuentran los componentes *Nomencladores, Comprobante de venta*, *Reportes, Apertura, Cierre* y *Configuración*, dentro de este último están: *Productos* y *Punto de venta.* Mediante esta interfaz se pueden desplegar las ventanas de dichos componentes.

En la interfaz del Punto de venta se encuentran opciones como: *Configurar*, *Modificar*, *Eliminar* y *Activar* un PV determinado, además permite *Buscar* por código y descripción y *Limpiar*. En la parte inferior muestra las opciones de *Asociar almacén* y *Desasociar almacén* al PV seleccionado.

Al realizar una venta de un producto en el punto de venta se emite el comprobante de venta como evidencia de la actividad realizada, el cual tiene las opciones de *Adicionar*, *Modificar*, *Eliminar, Confirmar, Cancelar estado, Cobrar venta, Contabilizar, Imprimir* y *Buscar* los comprobantes de un PV determinado.

Modelo de despliegue

El modelo de despliegue se basa fundamentalmente en las relaciones físicas de los distintos nodos que integran un sistema y la dosificación de los componentes del mismo. En las vistas de despliegue se representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación.

Servidores

Servidor de Aplicaciones Web

• Sistema Operativo: Ubuntu Server

• Servidor Web: Apache 2.0

• Librerías Adicionales: PHP 5

Servidor de Base de Datos

• Sistema Operativo: Ubuntu Server

• Sistema Gestor de Base de Datos: PostgreSQL 8.3.8

Servidor de Clientes Ligeros

• Sistema Operativo: Nova Server

• Navegador Web: Mozilla Firefox 2.17 o superior

Clientes

PC Cliente con disco duro

• Sistema Operativo: Linux o Windows

• Navegador Web: Mozilla Firefox v2.2 o superior

• Herramientas Ofimáticas

• Visualizador de ficheros pdf

• Equipo de impresión

PC Cliente sin disco duro

• Todo se instala en el servidor de clientes ligeros

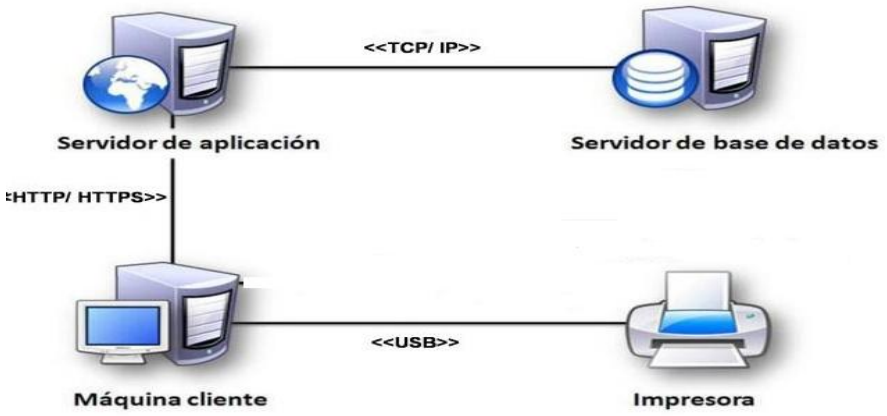


Figura 19: Modelo de despliegue del Terminal punto de venta.

3.3. Pruebas de software

Las pruebas constituyen una etapa imprescindible durante el proceso de desarrollo del software, pues permiten detectar y corregir el máximo de errores posibles antes de la entrega al cliente del software

desarrollado, por lo que el éxito de las mismas puede mejorar la percepción de calidad del usuario final y lograr su satisfacción. El objetivo principal de las pruebas es asegurar que el software cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener. Es importante considerar que las pruebas de software no garantizan que un sistema esté libre de errores, sino que se detecten la mayor cantidad de defectos posibles para su debida corrección.

3.3.1. Pruebas de caja blanca o pruebas estructurales

La prueba de caja blanca, denominada a veces prueba de caja de cristal, es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que:

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

A continuación se citan algunas de las técnicas de prueba de Caja blanca:

- Condición.
- Flujo de Datos.
- Bucles.
- Camino Básico.

Dentro de la prueba de caja blanca, la técnica que se utilizó fue Camino básico. Para aplicar esta técnica se debe introducir la notación para la representación del flujo de control, este puede representarse por un Grafo de flujo en el cual:

- Cada nodo del grafo corresponde a una o más sentencias de código fuente.
- Todo segmento de código de cualquier programa se puede traducir a un Grafo de Flujo.
- Se calcula la complejidad ciclomática del grafo.

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

Componentes del grafo de flujo:

- **Nodo:** Son los círculos representados en el grafo, el cual contiene una o más secuencias del procedimiento, donde un nodo corresponde a una secuencia de procesos o a una sentencia de decisión. Los nodos que no están asociados se utilizan al inicio y final del grafo.
- **Aristas:** Son constituidas por las flechas del grafo, son iguales a las representadas en un diagrama de flujo y constituyen el flujo de control del procedimiento. Las aristas terminan en un nodo, aun cuando el nodo no representa la sentencia de un procedimiento.
- **Regiones:** Son las áreas delimitadas por las aristas y nodos donde se incluye el área exterior del grafo, como una región más. Las regiones se enumeran siendo la cantidad de regiones equivalente a la cantidad de caminos independientes del conjunto básico de un procedimiento.

Para realizar la prueba del Camino básico es preciso calcular la complejidad ciclomática del algoritmo o fragmento de código a analizar. A continuación se muestra el código del método *adicionarComprobanteAction()* encargado de adicionar un comprobante de venta al punto de venta con sus respectivos productos.

```
function adicionarComprobanteAction(){

    //setear todos los campos
    $c = new DatComprobanteventa();  //1
    $c->idpuntoventa = $this->_request->getPost('idpuntoventa');  //1
    $c->idfactura = $this->_request->getPost('idfactura');  //1
    $c->idformapago = $this->_request->getPost('idformapago');  //1
    $c->idcliente = $this->_request->getPost('idcliente');  //1
    $c->idmoneda = $this->_request->getPost('idmoneda');  //1
    $c->fecha = $this->_request->getPost('fecha');  //1
    $c->idcajero = $this->_request->getPost('idcajero');  //1
    $c->estado = $this->_request->getPost('estado');  //1
    $c->observacion = $this->_request->getPost('observacion');  //1
    $c->numero = $this->_request->getPost('numero');  //1
    $c->elaboradopor = $this->_request->getPost('elaboradopor');  //1
    $c->importetotal = $this->_request->getPost('importetotal');  //1
    $c->cantidad = $this->_request->getPost('cantidad');  //1
    $c->tarjeta = $this->_request->getPost('tarjeta');  //1
    $c->entregado = $this->_request->getPost('entregado');  //1
    $c->cambio = $this->_request->getPost('cambio');  //1

    //guardar comprobante en la BD
    $c = DatComprobanteventa::guardar($c);  //1
    if ($c != null) {  //2
        //guardar relación
        $idproductos = json_decode($this->_request->getPost('idproductos'));  //3
        for($i=0; $i < count($idproductos); $i++){  //4
            $p = new DatProdCompVenta();  //5
            $p->idprodventa = $idp;  //5
            $p->idcompventa = $c->idcompventa;  //5
            DatProdCompVenta::guardar($p);  //5
        }
        echo 'Comprobante de venta guardado satisfactoriamente.';  //6
    } else {  //7
        echo 'No se pudo guardar el comprobante de venta.'; //8
    }  //9
}  //10
```

Figura 20: Método para adicionar un Comprobante de venta.

Para el cálculo de la complejidad ciclomática es necesario representar el grafo de flujo asociado al código antes presentado a través de nodos, aristas y regiones, quedando como se muestra en la figura 21.

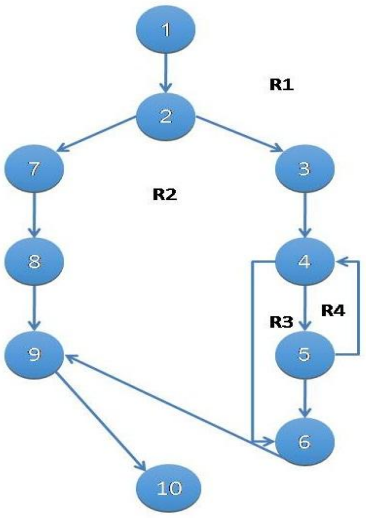


Figura 21: Grafo de flujo de la funcionalidad Adicionar comprobante de venta.

Una vez construido el grafo de flujo asociado al procedimiento anterior se determina la complejidad ciclomática, el cálculo es necesario efectuarlo mediante tres fórmulas, se debe utilizar el mismo grafo en cada caso:

Fórmula 1: V (G) = (A – N) + 2 Donde “A” es la cantidad total de aristas y “N” la cantidad total de nodos.

Resultado:

V (G) = (12-10) + 2 V (G) = 4

Fórmula 2: V (G) = P + 1 Donde “P” es la cantidad total de nodos predicados (son los nodos de los cuales parten dos o más aristas).

Resultado:

V (G) = 3+1 V (G) = 4

Fórmula 3: V (G) = R Donde “R” es la cantidad total de regiones, se incluye el área exterior del grafo, contando como una región más.

Resultado: V (G) = 4

Seguidamente es necesario especificar los caminos básicos que puede tomar el algoritmo durante su ejecución:

Camino básico # 1: 1, 2, 7, 8, 9, 10.

Camino básico # 2: 1, 2, 3, 4, 5, 6, 9, 10.

Camino básico # 3: 1, 2, 3, 4, 6, 9, 10.

Camino básico # 4: 1, 2, 3, 4, 5, 4, 6, 9, 10.

Es necesario aclarar que existen otros caminos en el grafo que no son considerados independientes ya que son combinaciones de los anteriormente especificados.

Después de haber extraído los caminos básicos del flujo, se procede a ejecutar los casos de pruebas para este procedimiento, se debe realizar al menos un caso de prueba por cada camino básico. Para realizarlos es necesario cumplir con las siguientes exigencias:

- **Descripción:** Se hace la entrada de datos necesaria, validando que ningún parámetro obligatorio pase nulo al procedimiento o no se entre algún dato erróneo.
- **Condición de ejecución:** Se especifica cada parámetro para que cumpla una condición deseada para ver el funcionamiento del procedimiento.
- **Entrada:** Se muestran los parámetros que entran al procedimiento.
- **Resultados esperados:** Se expone el resultado que se espera que devuelva el procedimiento.
- **Resultados:** Se muestra el resultado obtenido.
- **Salida:** Se presenta el valor final.

Tabla 11. Caso de prueba para el camino básico 1.

Camino básico # 1: 1, 2, 7, 8, 9, 10.	
Descripción	Se le otorgan valores a un objeto creado y se guarda en la base de datos, si no puede ser guardado se muestra un mensaje de error.
Condición de ejecución	Se debe tener el objeto que tiene los valores asignados de los atributos.
Entrada	$c = new DatComprobanteventa();
Resultados esperados	El PV no tiene comprobantes adicionados por lo que se le adicionan comprobantes de venta.
Resultados	No se adiciona el comprobante de venta.
Salida	-1

Tabla 12. Caso de prueba para el camino básico 2.

Camino básico # 2: 1, 2, 3, 4, 5, 6, 9, 10.	
Descripción	Se le otorgan valores a un objeto creado y se guarda en la base de datos, mostrándose un mensaje de la función realizada.
Condición de ejecución	Se debe tener el objeto que tiene los valores asignados de los atributos.
Entrada	$c = new DatComprobanteventa(); $idproductos = json_decode($this->_request->getPost('idproductos'));
Resultados esperados	El PV no tiene comprobantes adicionados por lo que se le adicionan comprobantes de venta.
Resultados	Se adiciona el comprobante de venta satisfactoriamente.
Salida	1

Tabla 13. Caso de prueba para el camino básico 3.

Camino básico # 3: 1, 2, 3, 4, 6, 9, 10.	
Descripción	Se le otorgan valores a un objeto creado y se guarda en la base de datos, mostrándose un mensaje respecto a la función realizada.
Condición de ejecución	Se debe tener el objeto que tiene los valores asignados de los atributos.
Entrada	$c = new DatComprobanteventa(); $idproductos = json_decode($this->_request->getPost('idproductos'));
Resultados esperados	El PV no tiene comprobantes adicionados por lo que se le adicionan comprobantes de venta.
Resultados	Se adiciona el comprobante de venta satisfactoriamente.
Salida	1

Tabla 14. Caso de prueba para el camino básico 4.

Camino básico # 4: 1, 2, 3, 4, 5, 4, 6, 9, 10.	
Descripción	Se le otorgan valores a un objeto creado y se guarda en la base de datos, mostrándose un mensaje respecto a la función realizada.
Condición de ejecución	Se debe tener el objeto que tiene los valores asignados de los atributos.
Entrada	$c = new DatComprobanteventa(); $idproductos = json_decode($this->_request->getPost('idproductos'));
Resultados esperados	El PV no tiene comprobantes adicionados por lo que se le adicionan comprobantes de venta.
Resultados	Se adiciona el comprobante de venta satisfactoriamente.
Salida	1

Resultado de la ejecución de las pruebas de Caja blanca:

Luego de haber ejecutado el método de Camino básico a las funcionalidades más complejas en el desarrollo del subsistema Punto de venta, se pudo comprobar que el flujo de trabajo de las funcionalidades es correcto, ya que se probó que cada sentencia es ejecutada al menos una vez con los parámetros de entrada y las salidas esperadas. Con las pruebas de caja blanca se comprobó que la implementación se realizó de manera correcta, se examinó el estado de la aplicación en varios puntos de la misma determinando que el estado real coincidía con el esperado.

3.3.2. Pruebas de caja negra

Cuando se utiliza el término pruebas de caja negra se está haciendo referencia a las pruebas que se llevan a cabo sobre la interfaz del software sin tener en cuenta el código, por lo que los casos de prueba pretenden demostrar que las funciones del software son operativas, que la entrada se acepta de forma

adecuada y que se produce una salida correcta, así como que la integridad de la información externa se mantiene.

El diseño de estas pruebas tiene el propósito de detectar:

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

Existen diferentes técnicas de prueba de Caja negra descritas por Pressman para validar la funcionalidad del sistema sin entrar a analizar su ejecución interna:

- Métodos de prueba basados en grafos.
- Análisis de valores límites.
- Tabla ortogonal.
- Partición de equivalencia.

De estas técnicas, la seleccionada fue partición de equivalencia la cual permite examinar los valores válidos e inválidos de las entradas existentes en el software.

Para la aplicación de esta técnica se realizan los diseños de casos de prueba los cuales se basan 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.
- Las clases de equivalencia se pueden definir de acuerdo con las siguientes directrices:
    1. Si un parámetro de entrada debe estar comprendido en un cierto rango, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
    2. Si una entrada requiere un valor concreto, aparecen 3 clases de equivalencia: por debajo, por encima y en el rango.
    3. Si una entrada requiere un valor de entre los de un conjunto, aparecen 2 clases de equivalencia: en el conjunto o fuera de él.
    4. Si una entrada es booleana, hay 2 clases: sí o no.

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

Se realizaron 44 diseños de casos de prueba para validar el sistema atendiendo a la técnica partición de equivalencia. A continuación se especifica el caso de prueba para el requisito “Adicionar comprobante de venta”.

Descripción general:

Dar clic en el botón Inicio del sistema/Punto de venta/Comprobante/Comprobante de venta.

Condiciones de ejecución:

1- El usuario se debe identificar y autenticar ante el sistema y además debe tener los permisos para ejecutar esta acción.
2- Debe de existir al menos un punto de venta y seleccionar uno de ellos.
3- Se debe seleccionar una configuración para el punto de venta seleccionado.

Tabla 15. Posibles escenarios al adicionar un comprobante de venta.

Nombre del requisito	Descripción general	Escenarios de pruebas	Flujo del escenario
1: Adicionar comprobante de venta.	El sistema debe permitir adicionar comprobante de venta con los atributos: departamento y observaciones.	EP 1.1: Adicionar comprobante de venta, introduciendo datos válidos.	– Se presiona el botón **Adicionar.** – Se selecciona el departamento**.** – Se introduce la observación. – Se presiona el botón **Aceptar**. – Se muestra un mensaje de información. – Se presiona el botón **Aceptar.**
		EP 1.2 Aplicar	– Se presiona el botón **Adicionar.** – Se selecciona el departamento**.** – Se introduce la observación. – Se presiona el botón **Aplicar**. – Se muestra un mensaje de información.

			– Se presiona el botón **Aceptar.**
		EP 1.3: Nuevo	– Se presiona el botón **Adicionar.** – Se selecciona el departamento**.** – Se introduce la observación. – Se presiona el botón **Nuevo**.
		EP 1.4: Campos vacios	– Se presiona el botón **Adicionar.** – No se selecciona el departamento a adicionar. – No se introduce la observación a adicionar. – Se presiona el botón **Aceptar.** – Se muestra un mensaje de error. – Se presiona el botón **Aceptar.**
		EP 1.5: Cancelar	– Se presiona el botón **Adicionar.** – Se selecciona el departamento**.** – Se introduce la observación. – Se presiona el botón **Cancelar.**

Tabla 16. Descripción de las variables al adicionar un comprobante de venta.

No	Nombre de campo	Tipo	Válido	Inválido
1	Observación	Campo de texto	Números y letras [0-9 a-z A-Z].	Caracteres especiales.

Tabla 17. Respuestas del sistema para los posibles escenarios al adicionar un comprobante de venta.

Id del escenario	EP 1.1	EP 1.2	EP 1.3	EP 1.4	EP 1.5
Escenario	Adicionar comprobante de venta, introduciendo datos válidos.	Aplicar.	Nuevo.	Campos vacios.	Cancelar.
Observación	V(Estos productos son del área 5)	V(Estos productos son del área 5)	N/A	I( )	N/A
Respuesta del sistema	El sistema registra los valores y muestra el mensaje de información: "El comprobante de venta se ha guardado satisfactoriamente.". El sistema registra los datos del comprobante de venta con los valores seleccionados e introducidos.	El sistema registra los valores y muestra el mensaje de información: "El comprobante de venta se ha guardado satisfactoriamente. ". El sistema registra los datos del comprobante de venta con los valores seleccionados e introducidos. El sistema limpia los campos y permite rellenar los mismos.	El sistema limpia los campos y recarga los datos: entidad, punto de venta, Número, fecha de emisión y estado.	El sistema valida los valores en el formulario y muestra el mensaje de error: "Por favor verifique que no haya dejado campos vacios.". El sistema permite que se seleccione o se rellenen los campos.	Se cancela la operación. El sistema cierra la interfaz sin realizar ninguna operación.

Resultado de la ejecución de las pruebas de Caja negra:

Las pruebas de caja negra a la solución fueron desarrolladas por el Grupo de Aseguramiento de la Calidad del Centro de Informatización de Entidades (CEIGE). Estas se realizaron en dos iteraciones, donde finalmente se comprobó en la segunda iteración que el subsistema estaba libre de no conformidades culminando así la fase de pruebas internas. En el anexo 1 se puede observar el acta de aceptación emitida.

3.4. Conclusiones parciales

A través de la modelación del diseño se identificaron las clases involucradas en la solución, las relaciones entre ellas y los componentes de cada una. La persistencia de los datos en la base de datos pudo observarse a través del modelo de datos, donde se representaron las tablas, sus propiedades y las relaciones entre dichas tablas. A través del diagrama de despliegue se simularon los posibles escenarios en los cuales puede ser desplegado el sistema. Se realizó la validación del diseño a través de las métricas TOC y RC, las cuales arrojaron como resultado que el diseño del subsistema Punto de venta es simple y presenta una calidad aceptable, ya que el mayor porcentaje de las clases que se implementaron poseen un bajo acoplamiento, baja responsabilidad y complejidad (de esta última tanto de mantenimiento como de implementación), además de ser altamente reutilizables, respondiendo de este modo a los patrones de diseño que se describieron y sirvieron de apoyo para que el flujo de trabajo en el que estos intervienen (implementación) sea aceptable. Se aplicaron las pruebas de caja blanca para de esta manera revisar el código, y verificar que el sistema detecte la mayor cantidad de errores que existan. De igual manera se ejecutaron pruebas de caja negra donde se validó que el sistema funciona de forma correcta, respondiendo a los requisitos funcionales aprobados, para obtener una mayor satisfacción ante las demandas del cliente.

CONCLUSIONES GENERALES

Con la realización y culminación del presente trabajo, se logró obtener un subsistema Punto de venta configurable y totalmente funcional, capaz de gestionar de manera integral los principales procesos de compra-venta que se realizan en los puntos de venta de las entidades cubanas. Este subsistema contribuye al proceso de informatización del país y al ser integrado satisfactoriamente al sistema Xedro-ERP, posibilita generar un comprobante de venta a través de dicho sistema, facilitando el manejo de los datos y contribuyendo a la seguridad e integridad de la información en las entidades del país, además, permitirá que el sistema Xedro-ERP no pierda clientes por la ausencia de un subsistema que gestione los procesos que se realizan en un Punto de venta.

Para lograr dicho resultado se realizó un estudio que permitió conocer detalles de los procesos que se realizan en un punto de venta. El análisis de varios sistemas que incluyen dichos procesos, concluyó que era mejor crear una solución que responda a las particularidades de la economía cubana y cumpla con el paradigma de independencia tecnológica por el que aboga el país. Se utilizaron las tecnologías y herramientas definidas por el CEIGE para la obtención del subsistema Punto de venta de modo que permitiera integrarse correctamente al sistema Xedro-ERP. Se diseñó el sistema aplicando patrones, que contribuyeron a un ahorro considerable de tiempo en la construcción del software y que el flujo de trabajo implementación fuese preciso y correcto, además del uso de buenas prácticas de programación para el desarrollo del subsistema. Se realizó la validación del diseño aplicando las métricas TOC y RC donde los resultados arrojados demuestran la presencia de valores positivos en los indicadores de calidad medidos por cada métrica, al obtener un elevado porcentaje de clases que cumplen con las restricciones de dichos atributos.

Se implementó el subsistema Punto de venta a partir del diseño validado haciendo uso del marco de trabajo Sauxe y se logró la integración con el Sistema de Gestión Integral de Organizaciones Xedro-ERP. Como solución al problema planteado y para evitar la ocurrencia de errores del sistema antes de su implantación en las entidades nacionales se realizaron pruebas de caja blanca y caja negra al mismo que demostraron su correcta funcionalidad e influyeron en el tratamiento de errores de las iteraciones realizadas al software.

RECOMENDACIONES

Se recomienda:

- Profundizar en temas referentes a la gestión de los puntos de venta en el país y el resto del mundo para detectar posibles debilidades en el subsistema y agregar mejoras al mismo.
- Desarrollar una funcionalidad para la gestión de garantías de los productos.
- Agregar funcionalidades al subsistema Punto de venta que permitan la gestión de entrega de productos a domicilio.
- Agregar funcionalidades al subsistema Punto de venta que permitan identificar productos mediante un escáner de código de barra.
- Realizar el despliegue del subsistema propuesto como parte del sistema Xedro-ERP.

BIBLIOGRAFÍA

Cruz Mulet, Ing. Henry, Silega Martinez, Ing. Nemury y Vázquez Zambrano., Ing. Donel. 2009. *Modelo de desarrollo orientado a componentes.* 2009.

Alvarez, M.A. 2012. Qué es HTML. [En línea] 2012. [Citado el: 25 de marzo de 2014.] http://www.desarrolloweb.com/articulos/que-es-html.html.

Apache. 2012. Caracteristicas de Apache. [En línea] 2012. [Citado el: 19 de diciembre de 2013.] http://es.scribd.com/doc/52208534/29/caracteristicas-y-ventajas-del-apache.

Apache, 2.2. 2013. Apache. [En línea] 2013. [Citado el: 26 de octubre de 2013.] www.apache.org.

ASSETS. 2004-2013. ASSETS Sistema de Gestion Integral. [En línea] 2004-2013. [Citado el: 24 de octubre de 2013.] http://www.assets.co.cu/index.asp.

Bizagi. 2014. Bizagi. [En línea] Febrero de 2014. [Citado el: 28 de enero de 2014.] http://www.buenastareas.com/ensayos/Bizagi/4235294.html.

Buschmann, Frank, y otros. 1996. A System of Patterns. *Pattern Oriented Software Architecture.* Inglaterra: s.n., 1996.

CSS, Colectivo de. 2013. Introducción a CSS. [En línea] 2013. [Citado el: 10 de febrero de 2014.] http://www.librosweb.es/css/.

Cybercia. 2011. www.cybercia.com. [En línea] 2011. [Citado el: 15 de enero de 2014.] http://www.cybercia.com/index.php?option=com_content&view=article&id=49&Itemid=66.

cybercia.com. 2011. www.cybercia.com. [En línea] 2011. [Citado el: 19 de marzo de 2014.] http://www.cybercia.com/index.php?option=com_content&view=article&id=49&Itemid=66. ..

del Toro Ríos, Dr.C José Carlos y González Brito, Msc.Henry Raúl. 2008. *Documento Rector Visión del ERP Cedrux.* Universidad de las Ciencias Informática. La Habana: s.n., 2008. Rector.

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

Doctrine. 2010. Doctrine-Project. [En línea] 2010. [Citado el: 25 de octubre de 2013.] www.doctrine-project.org.

ExtJs, 2.1. 2010. ExtJs. [En línea] 2010. [Citado el: 30 de octubre de 2013.] www.extjses.com.

Feria González, Liset. 2010. *Vista de arquitectura de seguridad del proyecto ERP-Cuba.* Ciudad de la Habana: Universidad de las Ciencias Informáticas: s.n., 2010.

Gardiner, Geoffrey S y Pinckaers, Fabien. Open ERP a modern approach to integrated business management based on a free Open Source system. *Open ERP a modern approach to integrated business management based on a free Open Source system.*

Gestión, Centro de Soluciones de. 2008. *Modelo de desarrollo orientado a componentes.* CEIGE, Universidad de las Ciencias Informáticas. La Habana: s.n., 2008.

Golden Soft, 2013. 2013. Golden Soft. [En línea] 2013. [Citado el: 15 de noviembre de 2013.] http://www.goldensoft.com.

Heredia, Herminio y Van Der, Cristian. 2001. Introducción al PHP. [En línea] 23 de Mayo de 2001. [Citado el: 2014 de marzo de 28.] http://www.maestrosdelweb.com/editorial/phpintro/.

Hernández, A. Coello. 2002. *El paradigma cuantitativo de la investigación científica.* La Habana: Editorial universitaria, 2002.

Hernández, José Luis García. 2010. *TUTORIAL BASICO OPEN ERP (V. 5.0.7).* Ingeniería de Sistemas, Universidad de Antioquia. Colombia, Medellin: s.n., 2010.

HOEHRMANN, B. 2006. Scripting Media Types. [En línea] Abril de 2006. http://www.apps.ietf.org/rfc/rfc4329.html.

Ivar Jacobson, Grady Booch, James Rumbaugh. 2000. *El proceso Unificado de Desarrollo de Software.* España: Addiso Wesley, 2000.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* España: Addison Wesley, 2000.

Json, Colectivo de. 2011. JSON. [En línea] 2011. [Citado el: 10 de Enero de 2014.] http://www.json.org/json-es.html.

KOCH, NORA Y MARÍA, JOSÉ. 2002. *Ingeniería de Requisitos en Aplicaciones para la Web– Un estudio comparativo.* Lenguajes y Sistemas Informáticos, Universidad de Sevilla, España: s.n., 2002.

KriterERP. 2012. Kriter ERP solutions. [En línea] 2012. [Citado el: 24 de octubre de 2013.] http://www.kritererp.com/es.

López, Daylin Sosa. 2011. eumed.net. [En línea] 2011. [Citado el: 12 de marzo de 2014.] http://www.eumed.net/libros-gratis/2009c/585/Descripcion%20del%20modelo%20del%20dominio.htm.

Microsoft. 2013. Planificación de recursos empresariales (ERP) | Solution ERP | Microsoft Dynamics AX. [En línea] 2013. [Citado el: 24 de octubre de 2013.] http://www.microsoft.com/es-es/dynamics/erp-ax-introduccion.aspx.

microsoft.com. 2013. dynamics.pinpoint.microsoft.com. [En línea] 2013. [Citado el: 24 de octubre de 2013.] http://dynamics.pinpoint.microsoft.com/es-MX/services/it-soluciones-12884939220-4295692200.

Mozilla. 2014. www.mozilla-europe.org. [En línea] 2014. [Citado el: 20 de enero de 2014.] http://www.mozilla-europe.org/es/firefox/features/#personalization.

NetBeans, 7.3. 2013. NetBeans, 7.3. [En línea] 2013. [Citado el: 29 de octubre de 2013.] http://www.netbeans.org/index_es.html.

Obregón Rodríguez, Msc. Guillermo, y otros. 2004. *verSat sarasola.* 2004.

Openbravo WEB POS, Sistema de Punto de Ventas. 2010. Sistema de Punto de Ventas, Openbravo WEB POS. *Sistema de Punto de Venta (TPV) Avanzado.* [En línea] 2010. [Citado el: 24 de octubre de 2013.] http://spocsys.com/es/openbravo/sistema-de-punto-de-venta-tpv-avanzado.html.

openbravo.com. 2010. [En línea] 2010. [Citado el: 24 de octubre de 2013.] http://www.openbravo.com/es/about-us/press-room/news/149/la-gestion-avanzada-de-cobros-y-pagos-disponible-en-openbravo-erp.php

Openbravo.com. 2009. Características OpenBravo. [En línea] 2009. [Citado el: 24 de octubre de 2013.] http://www.openbravo.com/es/product/erp/features/

openerpsite.com. 2012. TPV POS OpenERP. [En línea] 20 de 01 de 2012. [Citado el: 24 de octubre de 2013.] http://www.openerpsite.com/tag/tpv-pos-openerp/.

PHP, 5.0. 2013. PHP 5.0. [En línea] 2013. [Citado el: 30 de octubre de 2013.] www.php.net.

PostgreSQL, 9.1. 2013. PostgreSQL. [En línea] 2013. [Citado el: 25 de octubre de 2013.] www.postgresql.org.es.

scribd. 2012. Arquitectura cliente-servidor. [En línea] 2012. [Citado el: 28 de marzo de 2014.] http://es.scribd.com/doc/49374795/arquitecturacliente-servidor...

ServerGrove. 2010. ServerGrove. [En línea] 2010. [Citado el: 26 de octubre de 2013.] http://www.servergrove.es.

Sosa Marín, Ricardo. 2009. *Herramienta para la generación de código JavaScript para la librería ExtJS.* Ciudad de la Habana, Universidad de las Ciencias Informáticas: s.n., 2009.

Visual Paradigm, 5.0. 2013. Visual Paradigm. [En línea] 2013. [Citado el: 30 de octubre de 2013.] http://www.visual-paradigm.com/.

Wailgum, Thomas. 2008. ABC: An introduction to ERP. [En línea] 2008. [Citado el: 16 de octubre de 2013.] http://www.cio.com/research/erp/edit/erpbasics.html.

wiki.openbravo.com. 2011. wiki.openbravo.com. [En línea] 4 de 06 de 2011. [Citado el: 24 de octubre de 2013.] http://wiki.openbravo.com/wiki/ERP_2.50:User_Manual/es#Caractesiticas_del_nucleo.

www.subversion.apache.org. 2010. www.subversion.apache.org. [En línea] 2010. [Citado el: 12 de diciembre de 2013.] www.subversion.apache.org.

www.zend.com. 2009. www.zend.com. [En línea] 2009. [Citado el: 12 de diciembre de 2013.] www.zend.com/en/community/php.

ZAPATA, CARLOS Y ARANGO J. 2004. *Alineación entre Metas Organizacionales y Elicitación de Requisitos del Software.* Universidad Nacional de Colombia. 2004. Red de Revistas Científicas.

1