UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 5

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Título:

Módulo de Impresión de recibos para un Sistema POS de código abierto

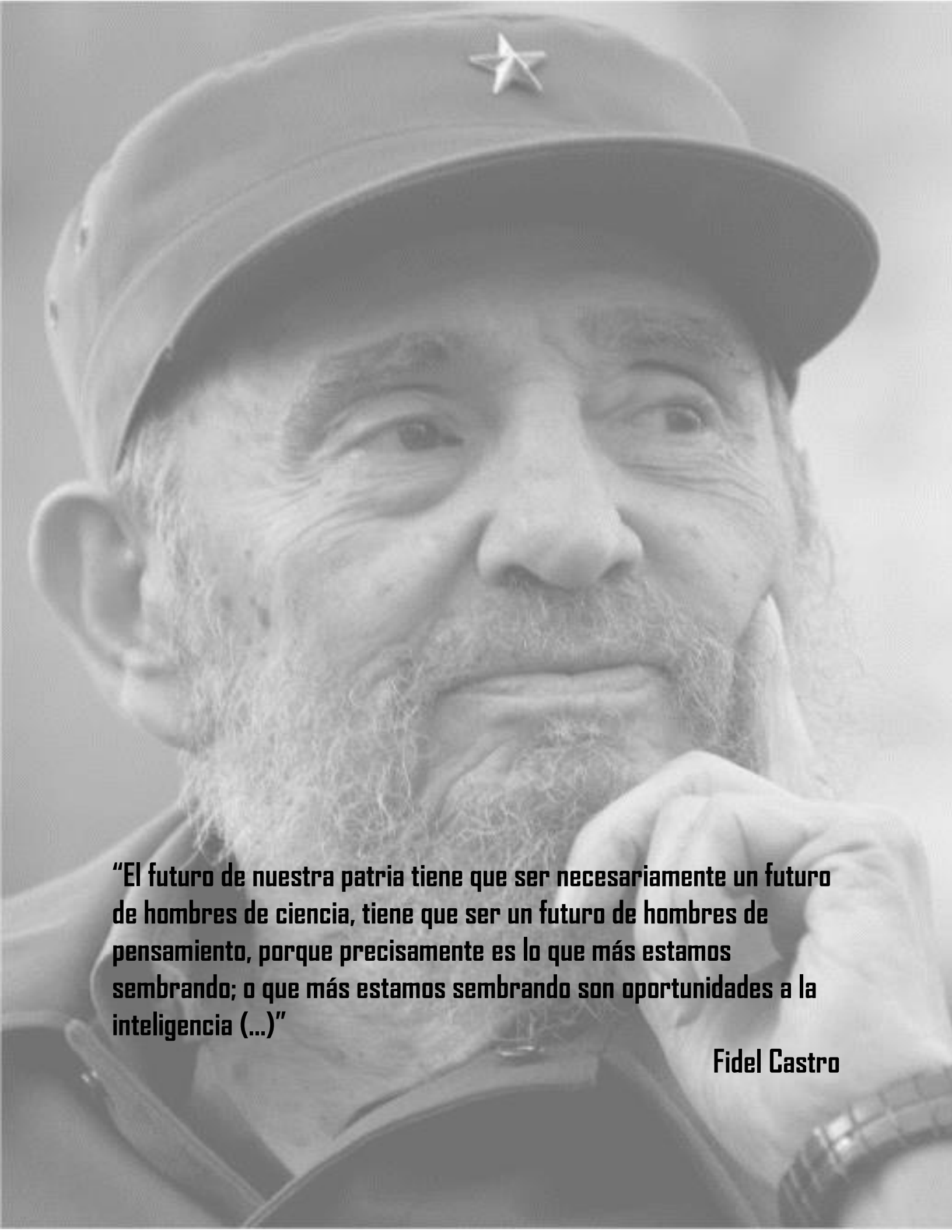
Autor: Lazaro Moreno Ramos.

Tutor: Dr. Antonio Cedeño Pozo.

Cotutores: Ing. Julio Alberto Leyva Durán
Ing. Frank Geiler Vega Duverger

La Habana, Cuba 2016

¨Año 58 de la Revolución¨



“El futuro de nuestra patria tiene que ser necesariamente un futuro de hombres de ciencia, tiene que ser un futuro de hombres de pensamiento, porque precisamente es lo que más estamos sembrando; o que más estamos sembrando son oportunidades a la inteligencia (...)”

Fidel Castro

DECLARACIÓN DE AUTORÍA

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

_____ _____

Firma del autor
Lazaro Moreno Ramos

Firma del Tutor
Dr. Antonio Cedeño Pozo.

_____ _____

Firma del tutor
Ing. Julio Alberto Leyva Durán

Firma del tutor
Ing. Frank Geiler Vega Duverger

DATOS DE CONTACTO

Tutor: Dr. Antonio Cedeño Pozo.

Edad: 31

Ciudadanía: cubana

Institución: Universidad de las Ciencias Informáticas (UCI)

Título: Doctor en Ciencias Técnicas (Informáticas) en el 2015

Categoría Docente: Instructor

E-mail: acedeno@uci.cu

Agradecimientos

Después de cinco años incansables de estudio se hace difícil recordar todas aquellas personas que de una forma u otra han aportado su granito de arena durante todo este tiempo y que merecen ser agradecidas.

A mi mamá, mi papá; ustedes son mi razón de ser.

A mi prima por estar siempre a mi lado y apoyándome en todo.

A mi familia y en especial a mis abuelos y mi tía.

A todos los profesores que aportaron a mi desarrollo como profesional.

A mis amigos de siempre; Eliuvis, Leduan Yassiel, Felipe, Raidel, Jose Carlos, Alexis, Alik.

A todas las amistades que la UCI me dio la oportunidad de tener.

¡Gracias a todos…!

Dedicatoria

A mi mamá, mi papá y mi prima que son mi guía en la vida, que siempre han creído en mí y me han dado fuerzas para seguir adelante.

Resumen

La presente investigación se basó en el desarrollo de un Módulo de Impresión de recibos para un Sistema POS (*Point of Sale* por sus siglas en inglés, Terminal de Punto de Venta en castellano) de código abierto, específicamente para las tiendas cubanas ya que muchas no cuentan con puntos de ventas automatizados (POS).

Con la finalidad de lograr un mejor desempeño y agilizar los procesos desarrollados en las tiendas cubanas que no cuentan con sistemas POS, se implementó una aplicación que gestiona todo el proceso durante una venta, garantizando la impresión de un recibo como constancia de pago, además de permitir la gestión de los productos que están almacenados en la base de datos, logrando así facilitar el trabajo del vendedor.

Para la elaboración del sistema y darles cumplimiento a los objetivos planteados se utilizó la metodología de desarrollo de *software* AUP-UCI, la cual encaminó el proceso; adicionalmente, para el desarrollo de la aplicación se utilizaron diversas tecnologías como el lenguaje de programación JavaScript y HTML, el sistema manejador de base de datos PostgresSQL y el IDE de desarrollo NetBeans.

Palabras Claves: *Point of sale* (POS), procesos de ventas, impresión de recibos, constancia de pago.

Índice de contenido

Índice de Ilustraciones

Índice de Tablas

Introducción

El acelerado desarrollo de la ciencia y la tecnología ha traído consigo un notable incremento del uso de las tecnologías de la información y las comunicaciones (TIC), imponiéndose en todas las ramas de la sociedad. Su notable avance ha generado un extraordinario impacto, convirtiéndola en una fuente vital para el desarrollo de la humanidad.

Las TIC han posibilitado desarrollar un proceso de automatización de las actividades que se realizan en las empresas e instituciones, permitiendo mejorar la productividad, la calidad de los servicios y el control, así como también proporcionar una eficiente comunicación entre los miembros de las mismas.

Cuba no se encuentra exenta de la necesidad de incluir estos procesos de automatización para sus empresas y entidades. Los cuales han tomado auge desde el 2002 con la fundación de la Universidad de las Ciencias Informáticas (UCI), fundada con el objetivo de formar profesionales altamente calificados y comprometidos con la patria, para poder responder ante las necesidades sociales y la informatización del pueblo.

En esta institución se encuentran varios centros de investigación y producción, los cuales se dividen en diversos temas para sus investigaciones y proyectos, uno de ellos es el Centro de Informática Industrial (CEDIN), en él se desarrollan proyectos que surgen de necesidades tanto sociales, como industriales y mercantiles. Con el objetivo de darle respuesta a estas necesidades y así lograr la informatización de la sociedad.

Un ejemplo de estas necesidades se puede observar en los centros de mercado, en estos existen varios puntos de ventas, en los cuales se gestiona todo el proceso de venta de productos, controlando las cantidades almacenadas y precios de los mismos, en estos puntos de ventas después de desarrollar una venta deben generar un recibo como constancia de pago del producto.

Un estudio realizado en diferentes puntos de ventas en la UCI detectó que solo en la tienda por CUC, existe un *software* para punto de venta o POS (*Point of Sale* por sus siglas en inglés, Terminal de Punto de Venta en castellano) implementado, mientras que los que trabajan solamente con moneda nacional, realizan todos estos procesos de manera manual, no se entregan los recibos de constancia de pago, y no existe una base de datos que tenga registrada la cantidad de productos existentes lo cual dificulta el control de los mismos, posibilitando la pérdida de dichos productos y la alteración de los precios, además le consume más tiempo y esfuerzo a los trabajadores. Partiendo de lo analizado anteriormente surge la siguiente **situación problemática**:

- En las tiendas cubanas que trabajan únicamente con moneda nacional no cuentan con sistemas de puntos de ventas automatizados (POS), además los POS de las tiendas por CUC se desarrollan en el extranjero y tienen elevados costos.

Teniendo en cuenta lo anteriormente expuesto, se plantea como **problema de investigación** la siguiente interrogante: ¿Cómo facilitar el diseño y la impresión de recibos personalizados para sistemas POS de código abierto?

Por lo cual se define como **objeto de estudio** los sistemas POS de código abierto y el **campo de acción** se centra en el diseño y la personalización en la impresión de recibos para sistemas POS de código abierto.
Se traza como **objetivo general** desarrollar un módulo de impresión de recibos que permita la personalización de los mismos en un sistema POS de código abierto.

Para darle respuesta al objetivo, el trabajo debe enmarcarse en relación con las siguientes **Tareas de investigación**:

- Conformar de la fundamentación teórica a partir del estado del arte de los sistemas POS.

- Seleccionar las tecnologías para el desarrollo de la solución.
- Definir los requerimientos del módulo de impresión de recibos para un sistema POS de código abierto.
- Diseñar e implementar el módulo de impresión de recibos para un sistema POS de código abierto.
- Diseñar y ejecutar las pruebas del módulo de impresión de recibos para un sistema POS de código abierto.

Métodos de investigación:

La selección de los métodos se realiza abarcando no solo la naturaleza del objeto de estudio, sino también al objetivo de la investigación y a las condiciones bajo las que se produce la misma, por lo cual se utilizaran métodos teóricos y empíricos, mediante los cuales se obtiene una idea más detallada de lo que se quiere lograr.

Métodos Teóricos:

- Histórico-Lógico: Para realizar el estudio del arte, e investigar los sistemas y aplicaciones similares, además de los lenguajes y metodologías que permitan el desarrollo del sistema que se propone.
- Analítico-Sintético: Para arribar a conclusiones de conocimiento sobre el objeto de estudio. Facilitó, dentro de otras cosas, concretar los aspectos teóricos fundamentales y necesarios para la investigación, una vez realizada toda la revisión de la bibliografía.
- Modelación: Para ofrecer una variante a la hora de realizar los diagramas necesarios que respondan a los requisitos funcionales y los no funcionales.

Métodos Empíricos:

- Revisión y Análisis de diferentes documentos: Permite constatar la evolución del problema, las concepciones que en torno al objeto de

estudio se tienen; leyes y normativas que existen en torno a él; investigaciones que se han realizado en esta esfera, desde que perspectivas y cuales han sido los resultados.

- Entrevista: Para entender todo el proceso de funcionamiento del negocio, además de llegar a conocer todas las necesidades y problemas existentes en el mismo.

Posibles resultados: Módulo que garantice la impresión de recibos personalizados para un sistema POS de código abierto.

El documento está estructurado por tres capítulos, los cuales se muestran a continuación:

Capítulo 1. Fundamentación teórica:

En este capítulo se abordan todos los elementos teóricos que sustentan el problema científico y los objetivos de la investigación. También se explican las tecnologías, metodologías y herramientas usadas para el desarrollo del módulo.

Capítulo 2. Propuesta de Solución:

Enmarca la descripción de la solución propuesta. En un primer momento se enuncian las características del sistema para luego modelar el mismo haciendo uso de la metodología de desarrollo de *software* y se presentan los principales artefactos generados en las fases que esta propone.

Capítulo 3. Implementación y pruebas de la solución propuesta:

Comprende la implementación del sistema, las tareas de ingeniería de *software* que complementaron la misma y finalmente la evaluación de la solución a través de los resultados obtenidos una vez realizadas las pruebas al sistema.

Capítulo 1. Fundamentación teórica:

Introducción

En este capítulo se desarrollan dos puntos fundamentales, primeramente, la descripción de los principales conceptos teóricos vinculados con el desarrollo de la investigación para facilitar el entendimiento a partir de distintos enfoques, definiciones y puntos de vistas. En un segundo momento prevalece la descripción de las herramientas, la metodología y los lenguajes de programación que se utilizaron en el desarrollo de la solución propuesta.

1.1 Conceptos y definiciones relacionados con los Sistemas de Puntos de ventas

La *American Marketing Association*, define la **venta** como "el proceso personal o impersonal por el que el vendedor comprueba, activa y satisface las necesidades del comprador para el mutuo y continuo beneficio de ambos (del vendedor y el comprador)". (1)

El concepto de venta supone que es preciso estimular a los consumidores para que compren. Para ello, las empresas que ponen en práctica este concepto, utilizan todo un arsenal de herramientas de venta y promoción para estimular más compras. (2)

Punto es un término con múltiples significados: en este caso, nos interesa su acepción como un lugar físico o sitio. **Venta**, por su parte, es el proceso y el resultado de vender (entregar la propiedad de un bien a otro sujeto, quien pagará un cierto precio ya acordado para quedarse con el producto en cuestión). Con estas definiciones en claro, podemos referirnos a la noción de punto de venta**,** como un local comercial en el cual se ofrecen diversos productos a la venta. Una persona que desea comprar algo, por lo tanto, puede acercarse a un punto de venta de aquello que pretende adquirir para concretar la operación. (3)

Según el coronel Carlos Edgardo Tejada, del ejército argentino, el término sistema se utiliza para explicar en general al conjunto de medios interconectados (objetos, seres humanos, informaciones) utilizados según un proceso dinámico con el fin de alcanzar los objetivos señalados. Si consideramos a un todo, formado por partes, estas partes son afectadas por factores internos y externos a ese todo. Además, cada parte debe interrelacionarse e interactuar entre sí y con otras provenientes del medio externo. También, a su vez, interactuar como un conjunto lógico para dar respuestas a sus propios requerimientos y a los que el medio externo espera de él. Si cada parte de este todo, actúa e interrelaciona interna y externamente mediante plan, método y orden, estaremos entonces, en presencia de un sistema integrado totalmente. La ausencia de plan, método y orden para actuar e interrelacionarse nos coloca frente al caos. Sistema nos indica orden; lo opuesto a sistema es el caos. (4)

Un *software* para POS es un tipo de *software* de aplicación vertical que se emplea en un negocio de venta de productos o servicios para gestionar el proceso de venta por un vendedor. Este tipo de *software* permite asistir en la creación del registro y de la impresión de los comprobantes de venta como facturas o boletas, aparte de agilizar el proceso de venta en los establecimientos, el *software* para POS ayuda a llevar un mejor control administrativo y de inventarios. (5)

Un *ticket* es el resguardo que contiene datos que acreditan ciertos derechos, en la mayoría de los casos obtenidos mediante un pago. Es decir, el *ticket* es un comprobante de pago que se emite en operaciones que se realizan con consumidores o usuarios finales. (6)

1.2 Análisis de los sistemas POS.

Hay muchas razones por las que un sistema de Punto de Venta es indispensable para negocios de ventas. Ayuda a los dueños a administrar y automatizar inventarios, transacciones, ofertas (como descuentos, cupones y promociones

especiales) y mejorar la eficiencia de los empleados, permitiéndoles pasar más tiempo interactuando con los clientes. Un Punto de Venta consiste de varios componentes, incluyendo *hardware*, sistema operativo, *software*, etc. Una parte importante es precisamente, el *software* de Punto de Venta. En el mercado existen muchos, pero por lo general son muy caros. (7)

Características del *software* punto de venta: *(8)*

- Registra transacciones de ventas de manera rápida y fácil.
- Genera e imprime *ticket* de apariencia muy profesional.
- Compatible con impresoras de recibos que usan rollos de papel.
- Administra precios de artículos y descuentos ofrecidos.
- Genera informes para ayudarle a analizar las ventas por vendedor o artículo.
- Función de restauración/copia de seguridad de datos para mantener su información segura.
- Diseñado para ser muy fácil de utilizar en las actividades cotidianas.

El *software* para POS es uno de los componentes de los terminales de punto de venta, los cuales cuentan generalmente con los siguientes elementos: (9)

- Computadora
- *Software* de sistema
- *Hardware* de terminal de punto de venta
- *Software* para POS

El *hardware* de terminal de punto de venta suele depender del tipo de mercado en el que se va a emplear. Puede tener los siguientes componentes: (9)

- Lectores de códigos de barra
- Lectores de tarjetas de crédito
- Impresora de comprobantes de venta

- Caja registradora electrónica
- Lectores RFID (siglas de *Radio Frequency IDentification*, en español identificación por radiofrecuencia)

1.2.1 Análisis de los sistemas similares

Al analizar el estado del arte se detectan aplicaciones existentes a nivel mundial que pueden representar posibles soluciones, debido a esto se realiza un análisis independiente por cada uno.

Libre POS (Openbravo) nos permite disponer de un sistema de punto de venta minorista de una manera sencilla. Tiene multitud de funciones, soporta impresoras ESC/POS, visores de cliente y lector de códigos de barras. El programa puede ser usado por varios usuarios, cada uno con su nivel de permisos dentro del programa. Además, nos permite la edición de los productos y nos informa de las ventas con gráficos. El sistema está internacionalizado y actualmente soporta español, inglés, alemán, portugués, italiano, etc. El diseño del programa está basado en una pantalla táctil para poder interactuar fácilmente con el sistema sin necesidad de teclado o ratón. Para los restaurantes incluye características muy interesantes como la visualización de las mesas en el bar con su correspondiente información. El propio programa incluye opciones para realizar los cierres de la caja, gestionar los productos y organizarlos en categorías. Se encuentra disponible para Windows, Mac, Solaris y Linux. (7)

Lemon es un *software* de Punto de Venta de código abierto diseñado para las *pymes*. Emplea MySQL como motor de base de datos, y su arquitectura es cliente/servidor ya que cuenta con una base de datos central y varios terminales POS en la red local. Lemon provee una interfaz moderna y personalizable, un panel de búsqueda, una herramienta de verificación de precios, herramientas administrativas para operar nuestro negocio e impresión de reportes. (10)

Ilustración 1: Interfaz principal del POS Lemon. (10)

POSper es un sistema para Punto de Venta diseñado para las *pymes*. Ofrece soporte para una amplia variedad de *hardware* y bases de datos. El sistema está comprometido con los principios de colaboración, comunicación y se adapta a los usuarios que vayan por las líneas antes mencionadas. Cuenta con soporte para *touch screen*, diseño orientado al usuario de negocios minoristas y restaurantes, opciones de tarjetas de cliente, un sistema sofisticado de descuentos con facilidades de listas de precios para clientes, modificadores para descuentos, impuestos sobre cambio, etc. Procesamiento de órdenes con notas de entrega, entre otras ventajas. (10)

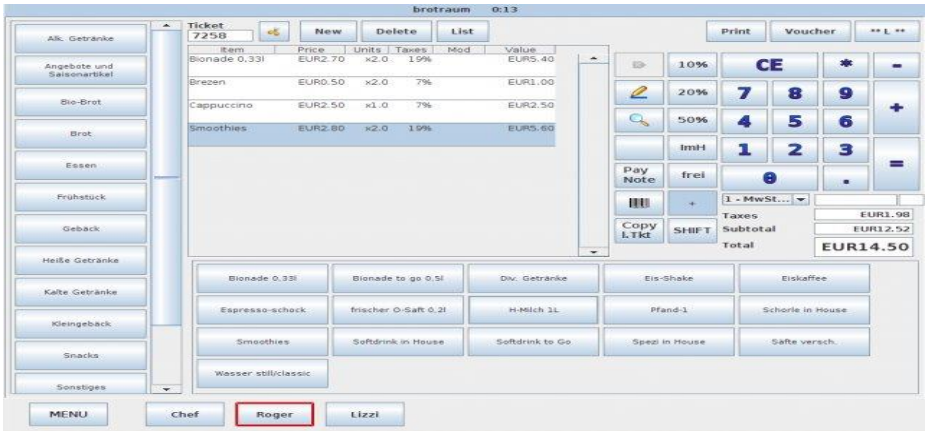


Ilustración 2: Interfaz principal del POSper.

Caja Registradora Quorion QMP Q-2044, este sistema está desplegado en las tiendas cubanas que trabajan con CUC, con el objetivo de gestionar sus procesos de ventas. Este sistema es muy versátil y de fácil manejo, posibilita el control de *stocks*, realiza informes personalizables, posee un teclado programable, cinta de control e impresora de gráficos para logotipos. La Quorion QMP Q-2044 es una registradora adecuada para comercios o restaurantes, conexión a PC, balanzas, lector código de barras o impresoras de cocina. Permite diferentes configuraciones de impresora térmica con anchos de factura de 57 y 80 mm. El sistema utiliza el software QMP POS, este ofrece escalabilidad con las redes de POS y soporte para una amplia gama de dispositivos periféricos, como impresoras, escáneres.

Ilustración 3: Caja Registradora Quorion QMP Q-2044

A continuación, se realiza en esta investigación un estudio de las principales desventajas de los sistemas similares de punto de venta:

Los sistemas POSper y Lemon POS, están desarrollados como aplicaciones de escritorio, este tipo de aplicaciones pueden ser robustas y poseen un tiempo de respuesta rápido, pero presentan varias desventajas, una importante es la escasa portabilidad ya que generalmente se hacen para un Sistema Operativo (SO) especifico y otra desventaja es que requiere instalación en cada cliente. Mientras que una aplicación *web*, en la cual está basada el sistema a desarrollar en esta investigación, es más portable, no requiere instalación y para su funcionamiento solo utiliza un navegador, que está presente en cualquier computadora.

El sistema Libre POS (Openbravo) sigue una arquitectura web basada en el patrón Modelo Vista Controlador (MVC) lo cual permite la reutilización de los elementos de diseño; pero el mismo presenta varias desventajas, una de ellas es que el sistema de soporte y la documentación son insuficientes, además los manejos de los usuarios de administración se realizan de una manera muy engorrosa, dificultando un poco el proceso de venta para un usuario que no tenga el privilegio de administración.

La Caja Registradora Quorion QMP Q-2044 tiene la desventaja de que el proceso de la gestión de los productos para la venta se hace engorroso ya que no se puede realizar la eliminación de un producto de los seleccionados por el cliente, obligando al mismo a realizar un nuevo pedido para satisfacer su demanda; otra desventaja es que esta caja registradora ya viene con el software QMP POS incluido y no permite la actualización del mismo, ni el cambio de este software por otro software pos.

El análisis de los sistemas estudiados sirvió de ayuda y punto de partida para el desarrollo de un módulo de impresión de recibos para un Sistema POS de código abierto, basándose en esta investigación se propone como solución una aplicación *web* que garantice la gestión de los procesos desarrollados durante una venta, la gestión de los productos almacenados, y permita la impresión de un recibo como constancia de pago.

1.3 Metodología de desarrollo de *software*

- Metodología de desarrollo de *software*: es un enfoque estructurado para el desarrollo de *software* que incluye modelos de sistemas, notaciones, reglas, sugerencias de diseño y guías de procesos (11).

No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de *software*. Toda metodología debe ser adaptada al contexto del

proyecto. Existen metodologías ágiles y tradicionales. Las metodologías ágiles ofrecen soluciones que se ajustan a los proyectos que tienen características similares a este. Una de las cualidades más destacables en una metodología ágil es su sencillez, tanto en su aprendizaje como en su aplicación, reduciéndose así los costos de implantación en un equipo de desarrollo. Las metodologías tradicionales de desarrollo de *software* son orientadas por la planificación. Inician el desarrollo de un proyecto con un riguroso proceso de licitación de requerimientos, previo a etapas de análisis y diseño. Con esto tratan de asegurar resultados con alta calidad circunscritos a un calendario (12)

A continuación, se realiza una comparación entre los dos tipos de metodologías las ágiles y las tradicionales.

Metodologías ágiles	***Metodologías tradicionales***
Pocos artefactos	Más artefactos
Pocos roles	Más roles
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños y trabajando en el mismo sitio	Grupos grandes
Menos énfasis en la arquitectura	La arquitectura es esencial

Tabla 1: Comparación entre metodologías ágiles y tradicionales. (12)

Después de haber estudiado las dos metodologías, se realiza la propuesta de solución sobre una metodología ágil. Ya que estas están orientadas a proyectos

pequeños y de poca duración, generan pocos artefactos y son sencillas tanto en su aprendizaje como en su aplicación. A continuación, se describen algunas de estas metodologías.

Programación Extrema (*Extreme Programming* o XP**)**

XP es una metodología ágil muy conocida y ampliamente utilizada, centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de *software*, promueve el trabajo en equipo, ayuda al aprendizaje de los desarrolladores, y proporciona un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y flexibilidad para enfrentar los cambios. Se define como una metodología adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

La producción de código está dirigida por las pruebas unitarias. Estas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores). El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Este es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo. La comunicación oral es más efectiva que la escrita. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible (12).

SCRUM

Metodología ágil desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. El desarrollo de *software* se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Las reuniones a lo largo del proyecto, también son muy importantes, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. Algunas características que brinda esta metodología:

- Flexibilidad a cambios: Genera una alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- Reducción del tiempo: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo.
- Mayor calidad del *software*: La forma de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un *software* de calidad superior.
- Mayor productividad: Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- Maximiza el retorno de la inversión : Producción de *software* únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión (11).

AUP

El Proceso Unificado Ágil de Scott Ambler o *Agile Unified Process* (AUP) en inglés es una versión simplificada del *Proceso Unificado de Rational* (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de

software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en RUP. La metodología AUP aplica técnicas ágiles incluyendo: (13)

- Desarrollo Dirigido por Pruebas (*test driven development* - TDD en inglés) Modelado ágil
- Gestión de Cambios ágil
- Refactorización de Base de Datos para mejorar la productividad.

Fases AUP: (13)

1. Inicio: El objetivo de esta fase es obtener una comprensión común cliente-equipo de desarrollo del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.
2. Elaboración: El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.
3. Construcción: Durante la fase de construcción el sistema es desarrollado y probado al completo en el ambiente de desarrollo.
4. Transición: El sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación, aceptación y finalmente se despliega en los sistemas de producción.

AUP-UCI

La metodología AUP-UCI es una variación de la metodología AUP. De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la siguiente Tabla. (13)

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planificación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo, costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Tabla 2: Comparación entre las Fases AUP y AUP-UCI.

Con esta metodología la actividad productiva de la UCI logra estandarizar el proceso de desarrollo de *software*, dando cumplimiento además a las buenas prácticas que define CMMI-DEV v1.3. Se logra hablar un lenguaje común en cuanto a fases, disciplinas, roles y productos de trabajos. Se redujo a 1 la

cantidad de metodologías que se usaban y de más de 20 roles en total que se definían se redujeron a 11. (13)

1.3.1 Selección de la Metodología a utilizar.

Después de realizar un análisis observando las características de las metodologías de desarrollo, se ha llegado a la conclusión de que la metodología a usar para el desarrollo del producto es AUP-UCI ya que ella logra estandarizar el proceso de desarrollo de *software* en la UCI, utiliza prácticas del desarrollo iterativo, se necesita hacer el software en un tiempo mínimo. Además, está enfocada a proyectos de poco tiempo de duración de desarrollo y a equipos pequeños.

1.4 Tecnologías a utilizar.

En el presente epígrafe se describen las herramientas a utilizar para el desarrollo del sistema.

1.4.1 Visual Paradigm

Visual Paradigm ha sido concebida para soportar el ciclo de vida completo del proceso de desarrollo de *software* a través de la representación de todo tipo de diagramas. Constituye una herramienta de *software* libre de probada utilidad para el Analista. Fue diseñado para una amplia gama de usuarios interesados en la construcción de sistemas de *software* de forma fiable a través de la utilización de un enfoque orientado a objetos. (14)

Se caracteriza por:

- Software libre.
- Disponibilidad en múltiples plataformas (Windows, Linux).
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.

- Disponibilidad de múltiples versiones, para cada necesidad.
- Licencia: gratuita y comercial.
- Fácil de instalar y actualizar.
- Soporte de UML versión 2.1." (16)

Esta herramienta automatiza la documentación y ayuda a la reutilización del software, portabilidad y estandarización de la información. La misma es aplicada por política de la universidad, teniendo en cuenta el proceso de migración a *software* libre que permite garantizar la soberanía tecnológica.

Después de analizar un análisis observando las características del Visual Paradigm se decide que esta es la herramienta a utilizar para el desarrollo del producto, ya que permite una perfecta integración con el IDE de desarrollo NetBeans, lo que permitirá invertir código para generar documentación de una manera sencilla y es multiplataforma.

1.4.2 Entorno de Desarrollo Integrado (IDE)

El IDE NetBeans es un producto libre y gratuito sin restricciones de uso, es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Se puede ejecutar en los sistemas operativos Windows, Mac OS, Linux, Solaris y otros. Existe además un número importante de módulos para extender el uso de NetBeans, lo que posibilita que el mismo pueda dar soporte para otros lenguajes de programación.

Entre sus características se destacan:

- Suele dar soporte a casi todas las novedades en el lenguaje Java.
- Asistentes para la creación y configuración de distintos proyectos, incluida la elección de algunos *frameworks*.
- Editor de código, multilenguaje.
- Simplifica la gestión de grandes proyectos con el uso de diferentes vistas, asistentes de ayuda, y estructurando la visualización de manera ordenada.

- Incluye herramientas para depuración de errores.

1.4.3 Framework

La palabra *framework* define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular, que sirve como referencia para enfrentar y resolver nuevos problemas de índole similar. Los objetivos principales que persigue un *framework* son: acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones. (15)

- **Framework AngularJS.**

AngularJS es un *framework* MVC de JavaScript para el Desarrollo *Web Front End* que permite crear aplicaciones **SPA** (*Single-Page Applications*). Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un *framework* para el desarrollo, en este caso sobre el lenguaje Javascript con programación del lado del cliente. Este Javascript pretende que los programadores mejoren el HTML que hacen. Que puedan producir un HTML que, de manera declarativa, genere aplicaciones que sean fáciles de entender incluso para alguien que no tiene conocimientos profundos de informática. El objetivo es producir un HTML altamente semántico, es decir, que cuando lo leas entiendas de manera clara qué es lo que hace o para qué sirve cada cosa. Lógicamente, AngularJS viene cargado con todas las herramientas que los creadores ofrecen para que los desarrolladores sean capaces de crear ese HTML enriquecido. La palabra clave que permite ese HTML declarativo en AngularJS es "directiva", que no es otra cosa que código Javascript que mejora el HTML. Puedes usar el que viene con AngularJS y el que han hecho terceros desarrolladores, puesto que muchas personas están contribuyendo con

pequeños proyectos independientes del propio *framework* para enriquecer el panorama de directivas disponibles. (16)

- **Framework Bootstrap.**

Bootstrap es un *framework* que simplifica el proceso de creación de diseños web combinando CSS y JavaScript. Ha sido desarrollado por Twitter. La mayor ventaja es que podemos crear interfaces que se adapten a los distintos navegadores (*responsive design*) apoyándonos en un *framework* potente con numerosos componentes webs que nos ahorrarán mucho esfuerzo y tiempo. (17)

Características principales de Bootstrap

Bootstrap ofrece una serie de plantillas CSS y ficheros Javascript que nos permiten integrar el *framework* de forma sencilla y potente en nuestros proyectos webs. (17)

- Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tablets y móviles a distintas escalas y resoluciones.
- Se integra perfectamente con las principales bibliotecas Javascript, por ejemplo, JQuery.
- Ofrece un diseño sólido usando LESS (lenguaje de hojas de estilo) y estándares como CSS3/HTML5.
- Es un *framework* ligero que se integra de forma limpia en nuestro proyecto actual.

1.4.4 Hibernate

Es una herramienta de mapeo Objeto/Relacional para emplear en plataformas Java, según el sitio oficial del mismo es “una herramienta que permite a los desarrolladores escribir más fácilmente aplicaciones cuyos datos sobreviven al proceso de solicitud. Como un marco de mapeo objeto relacional (ORM), Hibernate

tiene que ver con la persistencia de datos que se aplica a las bases de datos relacionales. (18)

1.4.5 Gestor de Bases de Datos

Las herramientas encargadas de operar con las Bases de Datos (BD) son las que se conocen como Sistemas de Gestión de Bases de Datos (SGBD), mediante estos se lleva el control de los datos que se guardan una vez que las aplicaciones que se estaban ejecutando se han cerrado, estos sistemas permiten también crear copias de seguridad de la Base de Datos y restaurar otras ya existentes y añaden seguridad a las mismas.

Para la gestión de la Base de Datos perteneciente al SBC se emplea PostgresSQL en su versión 9.2.

PostgresSQL: Es un Sistema de Gestión de Bases de Datos de código abierto, “utiliza un modelo cliente-servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando”. Es capaz de soportar grandes cantidades de datos sin que se afecte su rendimiento además puede operar sobre la mayoría de los sistemas operativos. Soporta operadores, funciones métodos de acceso y tipos de datos definidos por el usuario. Soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos (19).

1.5 Lenguajes de Programación

Un lenguaje de programación es aquel elemento dentro de la informática que permite crear soluciones de *software* mediante un conjunto de instrucciones, operadores y reglas de sintaxis; que pone a disposición del programador para que este pueda comunicarse con los dispositivos de *hardware* y *software* existentes. Los lenguajes de programación son herramientas que posibilitan la creación de programas, están constituidos por un grupo de reglas gramaticales, un grupo de

símbolos utilizables, un grupo de términos con sentido único y una regla principal. Específicamente los de programación *web*, han ido surgiendo según las necesidades de las plataformas, intentando facilitar el trabajo a los desarrolladores de aplicaciones.

1.5.1 Lenguaje de Programación del lado del servidor.

Los lenguajes del lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Un lenguaje de lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo. Por otra parte, los *scripts* son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente (20).

1.5.2 Lenguaje de Programación del lado del cliente.

Los lenguajes del lado del cliente basan su procesamiento en el cliente web, esto quiere decir que se ejecutan en el navegador del usuario y son interpretados por él sin necesitar un pre-tratamiento. Estos lenguajes son completamente independientes del servidor, es por ello que sus páginas pueden guardarse en cualquier sitio (20). Dentro de estos se encuentran HTML (*Hyper Text Markup Language*), Javascript, CSS (*Cascading Style Sheets*) y VBscript (Visual Basic Script).

HTML (*Hyper Text Markup Language*): Es un lenguaje de marcado de elementos para la creación de documentos hipertexto. Básicamente, se trata de un conjunto de etiquetas que sirven para definir el texto y otros elementos que compondrán una página web (21).

Por otro lado, HTML5 es un nuevo concepto de este lenguaje de etiquetas para la construcción de sitios web y aplicaciones en una era que combina dispositivos

móviles, computación en la nube y trabajos en red. HTML5 propone estándares para cada aspecto de la web y también un propósito claro para cada una de las tecnologías involucradas. Con este nuevo concepto HTML provee los elementos estructurales, CSS se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y Javascript tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales. Es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de HTML5 (22).

Conclusiones parciales

1. A lo largo de la investigación realizada se pudo adquirir el conocimiento que permitió identificar los conceptos principales para fundamentar teóricamente la propuesta de solución.
2. El estudio de los sistemas similares permitió conocer las características de estos, y servirá de base para identificar las posibles funcionalidades de la propuesta de solución.
3. Se realizó un estudio de las metodologías y las herramientas más usadas en el desarrollo de sitios *web*. El objetivo principal de este estudio fue seleccionar las metodología y herramientas que permitan darle cumplimiento a los objetivos planteados.

Capítulo 2: Propuesta de solución

Introducción:

Para continuar con la presente investigación en este capítulo se dan conocer las características y el funcionamiento que debe tener el módulo a desarrollar. Con este propósito, se describe la arquitectura propuesta para la solución, así como los requisitos funcionales y no funcionales. También guiados por la metodología AUP-UCI se recogen los resultados de las fases de Inicio, Ejecución y Cierre, como son las historias de usuario, el plan de iteraciones, el plan de entregas y las tareas de ingeniería o programación.

2.1 Recopilación de información para el sistema a desarrollar.

Para la recopilación de información del sistema se realizaron entrevistas a trabajadores vinculados a centros de ventas, donde se entrevistaron los trabajadores, desde los encargados de vender los productos hasta el administrador del centro, para así comprender mejor el funcionamiento del sistema y poder identificar mejor las funcionalidades del mismo.

2.2. Especificación de requisitos

La especificación de requisitos de *software* es una descripción completa del comportamiento del sistema que se va a desarrollar. Incluye un conjunto de historias de usuario que describen todas las interacciones que tendrán los usuarios con el *software*. Los casos de uso también son conocidos como requisitos funcionales. Además de las historias de usuario, la especificación de requisitos de *software* también contiene requisitos no funcionales o complementarios (23)

2.2.1 Descripción de las Historias de Usuario (HU).

Una historia de usuario (HU) es una técnica utilizada en las metodologías ágiles (SCRUM, XP, FDD, ASD, AUP, LD, etc.) para especificar los requisitos del *software*. En ellas, el cliente describe brevemente las características que el sistema debe

poseer. El contenido de las HU debe ser concreto y sencillo, estas son divididas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración. Las HU son planificadas en el tiempo de 1 a 3 semanas de duración para no superar el tamaño de una iteración. Son la base para las pruebas funcionales ya que en la fase de pruebas se utilizan para verificar si el programa cumple con lo que especifica la historia de usuario (24). Cada HU recoge al menos los siguientes aspectos (25):

- **Código:** Posee el número asignado a la HU.
- **Usuario:** El usuario del sistema que utiliza o protagoniza la HU.
- **Nombre de HU:** Atributo que contiene el nombre de la HU.
- **Referencia:** Es el conjunto de identificadores de las HU de las cuales depende historia actual que está en desarrollo.
- **Prioridad:** Evidencia el nivel de prioridad de la HU en el negocio.
- **Puntos estimados:** Este atributo es una estimación hecha por el equipo de desarrollo del tiempo de duración de la HU. Cuando el valor es 1 equivale a una semana ideal de trabajo.
- **Iteración Asignada:** Número de la iteración en la cual se desarrollará la HU.
- **Descripción:** Posee una breve descripción de lo que realizará la HU.
- **Observaciones:** Algunas aclaraciones que es importante señalar acerca de la HU.

A continuación, se muestran las HU que constituyen los procesos de mayor relevancia para el sistema que son la HU1, la HU6 y la HU7 de las 9 historias de usuario que se definieron. De las historias de usuario confeccionadas se tiene 5 con complejidad alta, 2 con complejidad media y 2 con complejidad baja.

Historia de Usuario	
Código: HU1	**Usuario:** Obrero
Nombre historia: Elaborar pedido de la venta	

Referencia:	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada:**
Descripción: El pedido se realiza a través de la interfaz principal donde aparecen los tipos de productos, aquí se escoge el producto deseado por el cliente, el sistema accede automáticamente a la base de datos y satisface el pedido con el stock disponible en el almacén, a continuación, se muestra una tabla con los campos id, nombre de producto, cantidad, precio y fecha de vencimiento.	
Observaciones:	

Tabla 3: HU1-Elaborar pedido de venta

Historia de Usuario	
Código: HU-6	**Usuario:** Obrero
Nombre historia: Crear e imprimir un recibo de pago	
Referencia:1,6	**Prioridad:** Alta
Puntos estimados: 4	**Iteración asignada: 1**
Descripción: Cuando se tengan todos los datos necesarios para realizar la venta, el sistema debe permitir imprimir un recibo como constancia de pago.	
Observaciones:	

Tabla 4: HU6- Crear e imprimir un recibo de pago.

Historia de Usuario

Código: HU-7	**Usuario:** Obrero
Nombre historia: Insertar en el recibo de pago los datos de la venta	
Referencia 1,4,5	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada: 1**
Descripción: Insertar en la plantilla de recibo de venta los datos necesarios de los productos que están en la lista de venta para poder realizar la misma.	
Observaciones:	

Tabla 5: HU7- Insertar en el recibo de pago los datos de la venta.

Las historias de usuarios son estimadas por los programadores para conocer el tiempo que estas pueden demorar en ser implementadas, tomando la semana como unidad de medida. Si la estimación es superior a tres semanas, la HU debe ser dividida en dos o más historias.

Historias de usuario	Puntos de estimación (semanas)
Elaborar pedido de la venta	1.6
Modificar pedido de la venta	0.6
Eliminar pedido de la venta	0.6
Calcular el precio total de la venta	0.4
Verificar si se puede realizar la venta y si hay que devolver algún monto de dinero	0.4
Crear e imprimir un recibo de pago	1.6
Insertar en el recibo de pago los datos de la venta	1.6
Adicionar producto al almacén	1.6

Modificar el producto en el almacén	1.6

Tabla 6: Estimación de esfuerzo por HU

Después de realizar un análisis se pudo estimar que el costo de desarrollo para el proyecto es de 10 semanas. Los valores de las HU están comprendidos entre 0.2 y 1.6 dependiendo de la complejidad correspondiente a cada una.

2.2.2 Plan de Iteraciones

En el plan de iteraciones se seleccionan las Historias de Usuarios que serán implementadas en cada iteración del sistema. Cada HU se traduce en tareas específicas de programación. Al terminar cada iteración se debe realizar un test de aceptación para cada HU verificando así su cumplimiento. (26)

- **Iteración 1:** En esta iteración se implementan las HU: 1, 2, 3 y 4 ya que son las definidas para el desarrollo de la gestión de los pedidos de ventas por el cliente con alto valor en el negocio.
- **Iteración 2:** En esta iteración se implementan las HU: 5, 6 y 7. En esta iteración queda desarrollado todo el proceso de la venta de los productos, desde el pago de los mismos hasta la impresión del recibo de pago.
- **Iteración 3:** En esta iteración se implementan las HU: 8 y 9. En esta última iteración, se realiza el proceso de gestión de los productos ya sea adicionar uno nuevo o modificar alguno ya existente en el inventario.

El plan de duración de las iteraciones se realiza luego de tener el estimado en días que demora implementar cada historia de usuario. (26)

Iteración	Historias de usuario	Puntos de estimación (semanas)
1	Elaborar pedido de la venta	3.2
	Modificar pedido de la venta	
	Eliminar pedido de la venta	

	Calcular el precio total de la venta	
2	Verificar si se puede realizar la venta y si hay que devolver algún monto de dinero	3.6
	Crear e imprimir un recibo de pago	
	Insertar en el recibo de pago los datos de la venta	
3	Adicionar producto al almacén	3.2
	Modificar el producto en el almacén	

Tabla 7: Plan de iteraciones

2.2.3 Plan de Entregas

En el plan de entregas se establecen las HU que serán agrupadas para conformar una entrega, así como el orden de estas. Este plan tiene como finalidad reflejar la duración de cada iteración, lo que ayuda a obtener una idea aproximada del tiempo que durará la confección del sistema en su totalidad. Dicho cronograma será el resultado de una reunión con todo el equipo de desarrollo, incluyendo al cliente, el cual ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo realizadas por los desarrolladores. (26)

Historias de usuario	1ra Iteración 22 de enero del 2014	2da Iteración 21 de marzo del 2014	3ra Iteración 16 de mayo del 2014
Elaborar pedido de la venta	V1.0		
Modificar pedido de la venta			
Eliminar pedido de la venta			
Calcular el precio total de la venta			
Verificar si se puede realizar la venta y si hay que devolver algún monto de dinero		V1.1	
Crear e imprimir un recibo de pago			

Insertar en el recibo de pago los datos de la venta			
Adicionar producto al almacén			V1.2
Modificar el producto en el almacén			

Tabla 8: Plan de Entregas.

2.3 Arquitectura del sistema

El sistema que se implementa está diseñado mediante una arquitectura Modelo Vista Controlador (MVC), esta arquitectura permite mantener separada la lógica de la aplicación, los datos que se manejan y los archivos correspondientes a las vistas. "Se trata de un modelo muy maduro y que ha demostrado su validez a lo largo de los años en todo tipo de aplicaciones, y sobre multitud de lenguajes y plataformas de desarrollo." (27)

- **El Modelo**, contiene una representación de los datos que maneja el sistema, su lógica de negocio, y sus mecanismos de persistencia.
- **La Vista**, o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos interacción con el usuario.
- **El Controlador**, actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

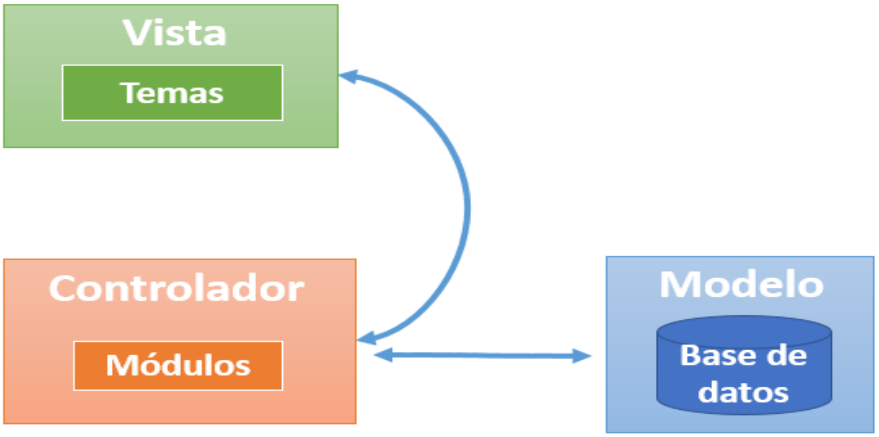


Ilustración 4: Arquitectura MVC

La propuesta de solución se basa en la arquitectura MVC, donde en el modelo se encuentra la clase Producto, en la cual se declaran los atributos del producto, además la implementación del constructor de esta clase y los métodos que definen cada uno de los atributos; en la vista se encuentran agrupadas las interfaces de la propuesta de solución; mientras que en el controlador contiene la clase ProductoController, en la cual se gestiona la información que fluye entre la vista y el modelo.

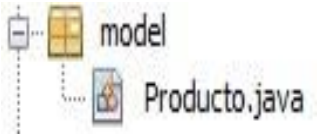


Ilustración 6: Modelo

Ilustración 5: Vista

Ilustración 7: Controlador

2.4 Patrones de diseño

Los patrones de diseño son soluciones acertadas a un problema general, que establecen pautas para definir estructuras de diseño en el desarrollo de un *software*. Estos, están más enfocados a los diseños orientados a objetos pero según dichos patrones a menudo tienen en cuenta características de los objetos tales como la herencia y el polimorfismo que proporciona generalidad. (28)

2.4.1 Patrones GRASP

Bajo Acoplamiento: Este patrón garantiza que no exista demasiada dependencia entre las clases, permite que cada una pueda realizar sus funciones por separado y si en el futuro se desea realizar alguna modificación no hay que efectuar demasiados cambios en el código. El bajo acoplamiento garantiza que la clase controladora pueda operar sin que necesite muchos recursos externos a ella, los modelos solo se gestionan por este controlador y las vistas mediante el controlador que tienen implementado. En la propuesta de solución las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo como la clase producto, la cual no tiene asociación con las de la vista o el controlador, proporcionando esto que la dependencia en este caso sea baja.

Alta Cohesión: En el sistema las clases solo realizan funciones estrechamente relacionadas con su contenido, lo que facilita la organización del proyecto y que estas no estén sobrecargadas de funcionalidades ajenas. Se define que generalmente que una clase que este altamente cohesionada tiene bajo acoplamiento, en la propuesta de solución la clase producto posee la menor cantidad de dependencias posibles con otras clases y a su vez posee responsabilidades moderadas en un área funcional y colabora con las otras para llevar a cabo las tareas.

Controlador: Este patrón se ve reflejado por el modelo arquitectónico MVC utilizado en el desarrollo del *software*, con el mismo se garantiza que estén centralizadas las diferentes actividades que realiza el software, esto ofrece una mejor estructuración y organización durante el desarrollo del producto. En la propuesta de solución

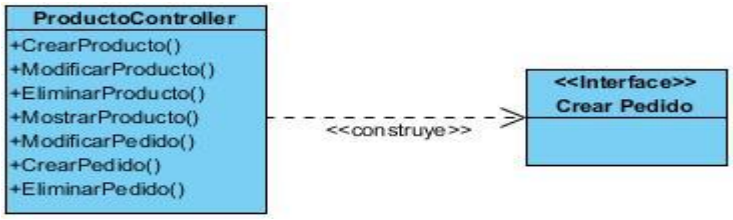


Ilustración 8:Patrón de diseño Controlador

Conclusiones Parciales

1. En este capítulo permitió definir las funcionalidades necesarias para la implementación, siguiendo cada fase de la metodología propuesta.
2. La selección de la arquitectura MVC permitió establecer la estructura lógica de la solución y aplicando los patrones de diseño se definieron la jerarquía de clases, así como sus relaciones y dependencias con lo que se alcanza una vista general del sistema a implementar.

Capítulo 3: Implementación y Pruebas.

Introducción:

El presente capítulo se centra en las fases de implementación y prueba del sistema teniendo en cuenta la metodología seleccionada. Se especifican las tareas de ingeniería tomando como base los artefactos generados en el análisis y diseño. Se realizan las pruebas de aceptación al sistema desarrollado documentando todos los resultados obtenidos para la demostración de su calidad.

3.1 Tareas de ingeniería o programación

La planificación de las tareas de ingeniería o programación está relacionada con cada una de las iteraciones, ya que cada HU es desglosada en varias tareas de desarrollo.

La tabla 9 muestra la iteración 1 con sus HU y cada tarea de ingeniería asignada a estas HU. De esta iteración depende el desarrollo de las próximas iteraciones, en esta se realizan actividades vinculadas a el proceso de gestión de los productos para la venta.

Historias de usuario	Tareas por historia de usuario
Elaborar pedido de la venta	1. Desarrollar funcionalidad que permita seleccionar un tipo de producto. 2. Desarrollar funcionalidad que permita vincular cada producto a la base de datos por el id de este producto. 3. Mostrar el producto seleccionado en la tabla de venta de productos.
Modificar pedido de la venta	4. Desarrollar una funcionalidad que permita seleccionar un producto de la tabla de ventas.

	5. Desarrollar una funcionalidad que permita modificar el producto seleccionado. 6. Desarrollar una funcionalidad que permita actualizar la tabla de ventas.
Eliminar pedido de la venta	7. Desarrollar una funcionalidad que permita seleccionar un producto de la tabla de ventas. 8. Desarrollar una funcionalidad que permita eliminar el producto seleccionado. 9. Desarrollar una funcionalidad que permita actualizar la tabla de ventas.
Calcular el precio total de la venta	10. Desarrollar una funcionalidad que permita calcular el precio total de la venta 11. Desarrollar una funcionalidad que permita actualizar este precio total.

Tabla 9: Tareas de ingeniería.

Las tareas de ingeniería se especifican a continuación donde se verán las tareas asignadas a la HU1.

Para la confección de las tareas se utilizaron tablas que contienen los siguientes campos:

- **No. de tarea:** Numeración continua que identifica a la tarea
- **No. de HU:** Número de la HU a la cual pertenece
- **Nombre de la tarea:** Identificación literal de la tarea
- **Tipo de tarea:** Tipo de tarea, dígase diseño, desarrollo, prueba
- **Puntos estimados:** Representación en porciento de la cantidad de tiempo estimada de una semana, que se utilizará para su realización
- **Fecha inicio:** Fecha estimada de inicio de realización

- **Fecha fin:** Fecha estimada de fin de realización
- **Descripción:** Se describe en que consiste la tarea y que elementos deben cumplirse para declarar la tarea terminada.

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 1
Nombre de la tarea: Desarrollar funcionalidad que permita seleccionar un tipo de producto	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la vista principal de la aplicación donde se muestran todos los productos existentes en la base de datos, y el sistema debe permitir la selección del producto deseado por el cliente.	

Tabla 10: Tarea de ingeniería 1.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 1
Nombre de la tarea: Desarrollar funcionalidad que permita vincular cada producto a la base de datos por el id de este producto	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la base de datos, la cual va a contener los datos de los productos. El sistema debe permitir que cuando se escoja un producto en la interfaz, se compare por id y se encuentre el producto deseado por el cliente.	

Tabla 11: Tarea de ingeniería 2.

Tarea de Ingeniería	
No. de tarea: 3	No. de HU: 1

Nombre de la tarea: Mostrar el producto seleccionado en la tabla de venta de productos	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.4
Fecha inicio:	**Fecha fin:**
Descripción: Una vez encontrado el producto seleccionado, el sistema muestra los datos necesarios del producto en una tabla, la cual será la lista de ventas.	

Tabla 12: Tarea de ingeniería 3.

3.2 Estructura del Sistema.

3.2.1 Estructura de Angular JS

AngularJS es un *framework* de JavaScript patrocinado por Google basado en el patrón (Modelo Vista Controlador) que permite el desarrollo de clientes enriquecidos para las aplicaciones *web*. Una de sus más notables características es el *binding*, el cual nos permite sincronizar propiedades entre objetos distintos e interactuar con nuestro html desde nuestro javascript por medio del uso de directivas.

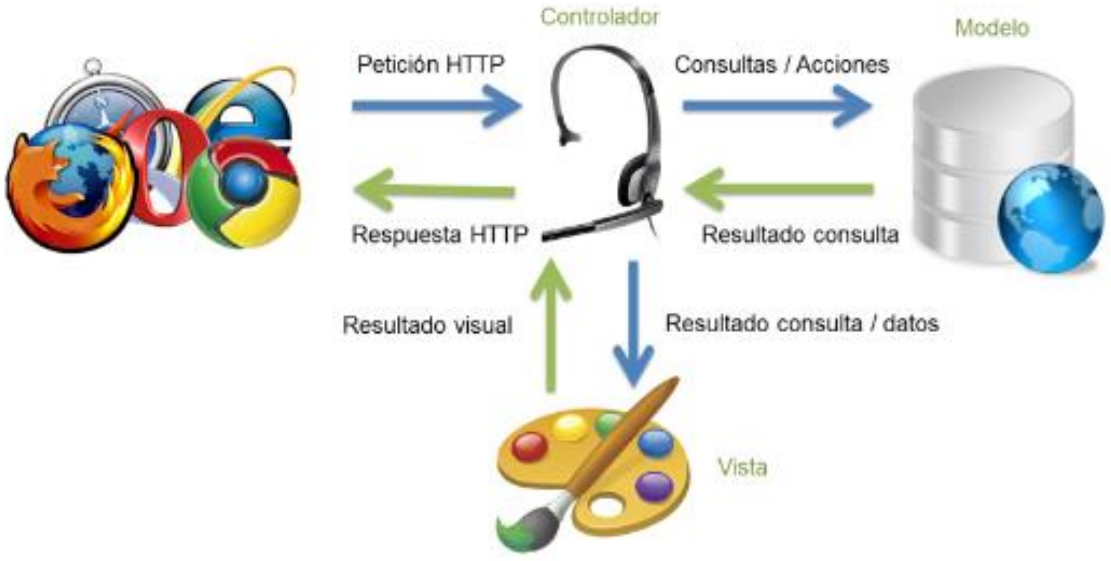


Ilustración 9: Arquitectura MVC en AngularJS

3.2.1 Estructura de Bootstrap

Bootstrap convierte el desarrollo web en más semántico, es un *framework* con base HTML, tiene un sistema de estructura que ordena la información dentro de la web de manera muy ventajosa. Un sitio web con Bootstrap se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como "*responsive design*" o diseño adaptativo.

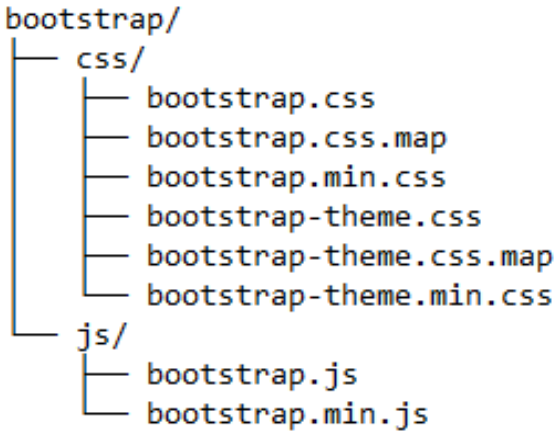


Ilustración 10: Estructura de Bootstrap.

3.2.1 Estructura y descripción de la aplicación

La estructura principal de la aplicación se corresponde con un proyecto Java *Web*, dentro de esta estructura se organizan las clases y archivos del sistema en determinados paquetes de acuerdo a la función que cumplan dentro de la aplicación, la figura 3.3 muestra cómo está estructurado el proyecto.

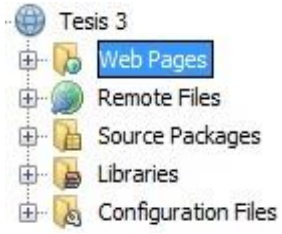


Ilustración 11: Estructura de la aplicación.

A continuación, se muestra una descripción de la estructura de la aplicación haciendo énfasis en el contenido de cada carpeta:

Carpeta Web *Pages*:

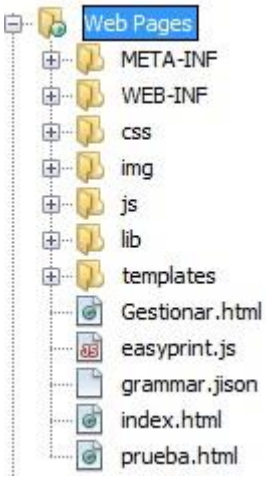


Ilustración 12: Carpeta Web Pages.

Esta carpeta contiene:

- La carpeta de los css los cuales se organizan de acuerdo a la estructura que presenta el *framework* Bootstrap.
- La carpeta de los js donde se encuentran todos lo js necesarios y vinculados a la aplicación.

Ilustración 14: Carpeta CSS

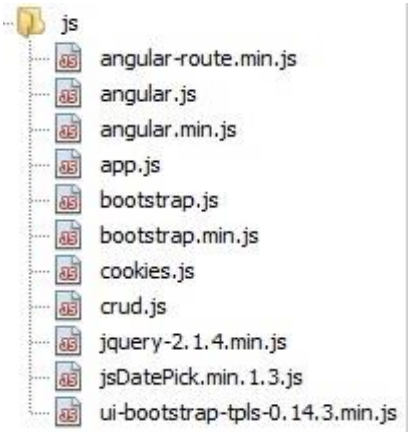


Ilustración 13: Carpeta JS

- Además, también se encuentran otras carpetas que solo tienen información de la aplicación.

Carpeta *Source Packages*:

Ilustración 15: Carpeta Source Packages.

A continuación, se ofrece una descripción del contenido de cada paquete:

1. <*default package*>: Contiene el archivo de configuración del *Framework* Hibernate, ya que este debe aparecer en la raíz del proyecto.
2. *Controller*: Contiene los controladores de las clases relacionadas con los datos de los productos.
3. Modelo: Contiene las clases que conforman el modelo de la aplicación, ya que la misma está estructurada mediante una arquitectura MVC. Dentro de este paquete aparecen separadas las clases en otros paquetes para facilitar la comprensión del modelo.

3.3 Descripción del sistema

El sistema cuenta con varias interfaces, la interfaz principal de gestión de ventas, la interfaz de impresión y la interfaz gestionar los productos, a continuación, se explicará el funcionamiento de cada una de ellas.

3.3.1 Descripción de la interfaz principal de gestión de ventas

En esta interfaz se desarrolla todo el proceso de la venta del producto. Donde el vendedor puede seleccionar el producto deseado por el cliente haciendo clip en las fotos de los productos, de los cuales se mostrará los datos necesarios para la venta en la tabla al final de la vista, de estos productos, la cantidad puede ser variada

haciendo clip en el botón “Modificar” del producto que se desee aumentar o disminuir su valor, el precio total de la venta se actualizara cada vez que se agregue un producto o se modifique la cantidad y se mostrara en la etiqueta “Total”; en el *input* se ingresa el monto de efectivo con el que el cliente va a desarrollar el pago de la venta, el cual se actualizara en tiempo real en la etiqueta “Pagó con”, cuando ingrese este monto el vendedor hace clip en calcular para ver el cambio que se debe entregar al cliente el cual se mostrara en la etiqueta “Cambio”, una vez terminado este proceso se hace clip en el botón “Imprimir” para imprimir el recibo de pago.

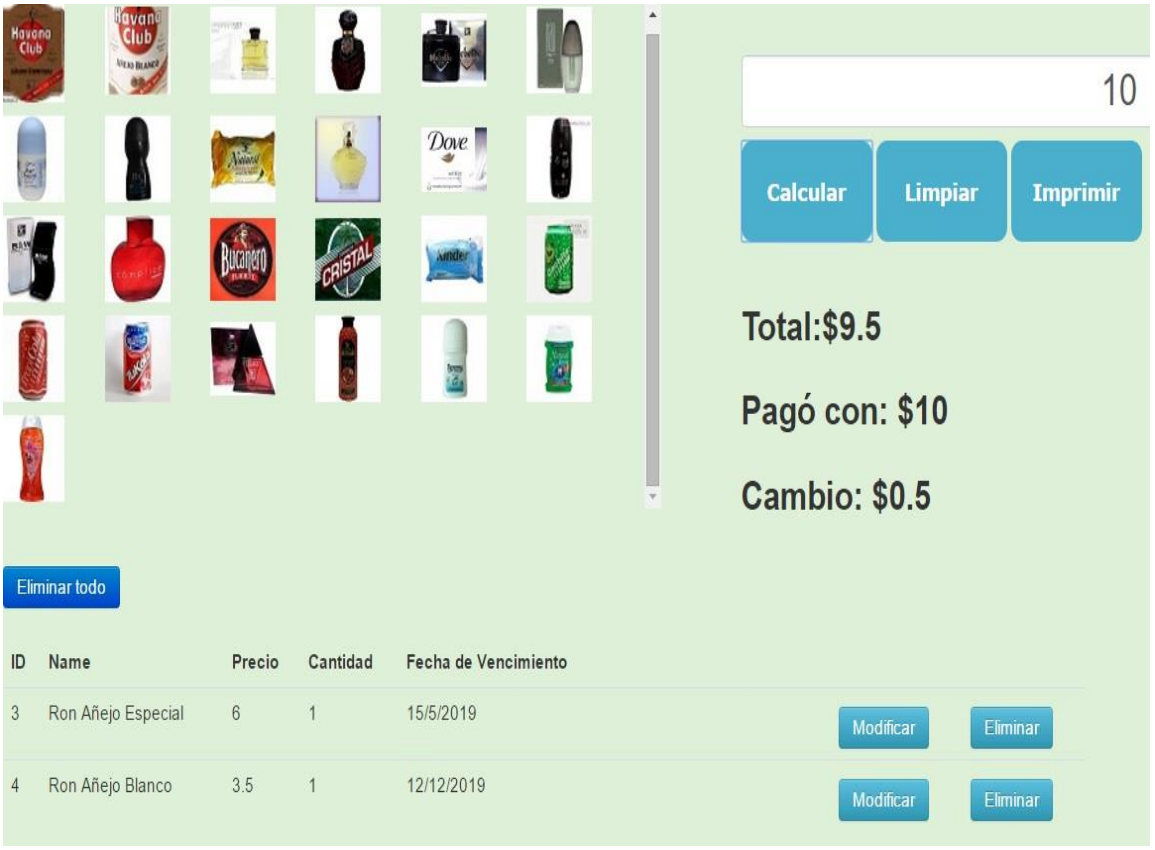


Ilustración 16: Interfaz principal Gestión de ventas

3.3.2 Descripción gestionar los productos

En esta interfaz se gestionan los productos de la base de datos, ya sea adicionando un producto nuevo, llenando los campos nombre, precio cantidad, fecha de vencimiento y seleccionando la foto del producto y luego haciendo clip en el botón “Adicionar”; además se modifican productos ya almacenados haciendo clip en el botón “Editar”, entonces se mostrarán en los *inputs* los datos que están almacenados de ese producto y se puede modificar el campo deseado, también se puede eliminar este producto de la base de datos haciendo clip en el botón “Eliminar”.

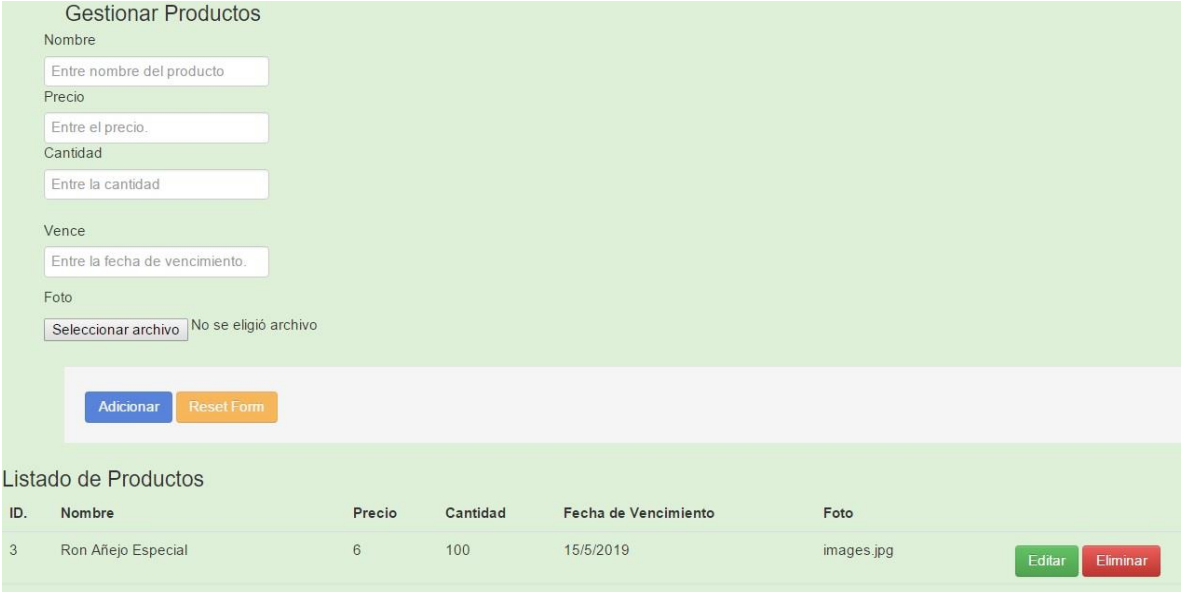


Ilustración 17: Interfaz Gestionar productos.

3.3.3 Descripción de la interfaz de impresión

En esta interfaz se puede modificar la impresión del recibo, haciendo clip en el botón “Cambiar” se puede seleccionar la impresora con que se desea imprimir el recibo, se puede seleccionar el número de copias que se desean imprimir, cambiarle el diseño al papel en horizontal o vertical, el tipo de papel para imprimir el cual cambia en dependencia de la impresora seleccionada, y al hacer clip en el botón “Imprimir” se imprime el recibo de pago.

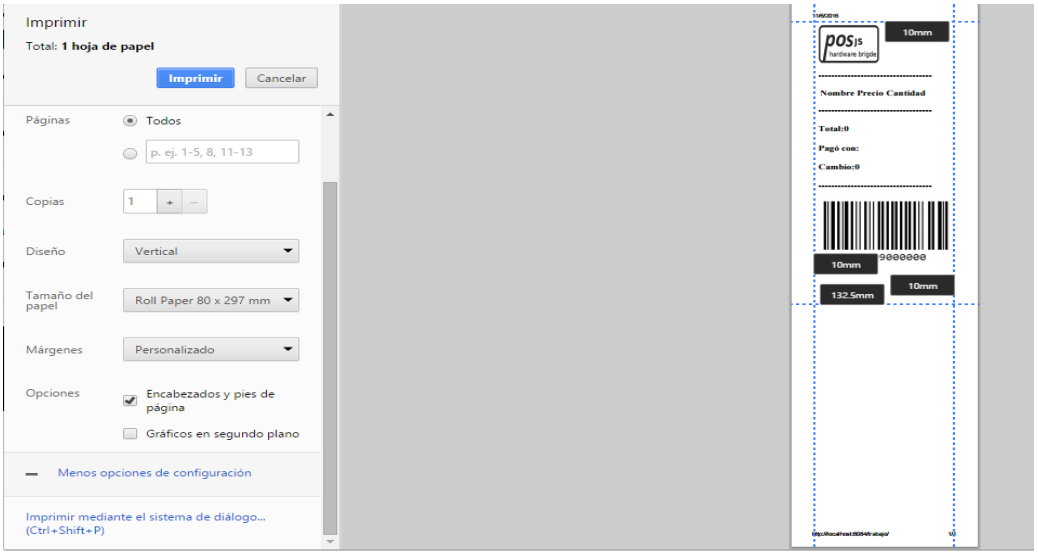


Ilustración 18: Interfaz de impresión.

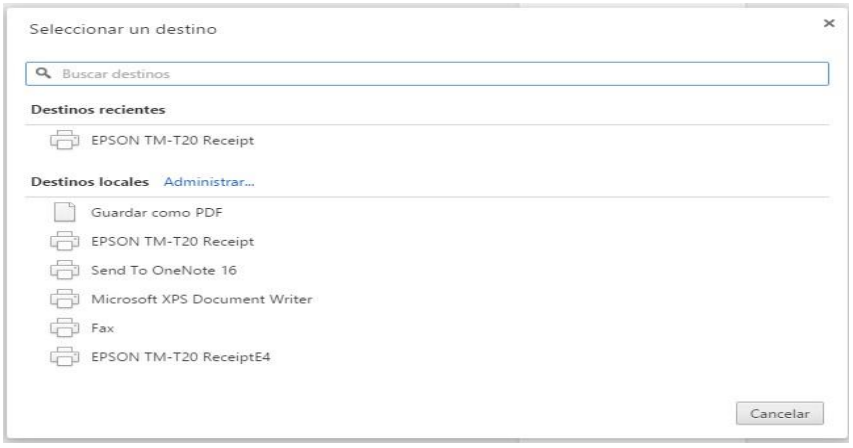


Ilustración 19: Interfaz Selección de la impresora

3.4 Diagrama de despliegue.

El despliegue provee un modelo detallado de la forma en la que los componentes se desplegarán a lo largo de la infraestructura del sistema. Muestran la disposición física de los distintos nodos que componen un sistema y como están repartidos los componentes sobre dichos nodos. Un nodo es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional, como son los procesadores o dispositivos de *hardware*, existen relaciones que representan medios de comunicación entre ellos. (29)

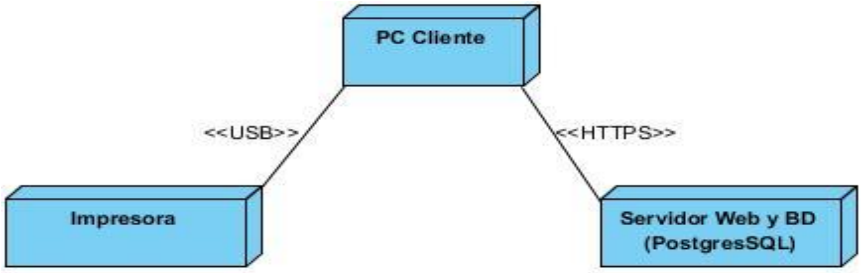


Ilustración 20: Diagrama de despliegue.

PC Cliente: Serán las estaciones de trabajo que el usuario utilizará para acceder a la aplicación *Web* donde se encontraran los *hardware* del terminal de punto de venta.

Servidor: En este nodo están contemplados los servidores web y de base de datos de la aplicación.

- **Servidor *web*:** El servidor de aplicación es utilizado para la publicación de la aplicación. Es la herramienta principal para ejecutar la lógica de negocio en el lado del servidor y es el encargado de ejecutar el código de las páginas servidor.
- **Servidor de Base de Datos:** Se hace referencia al gestor de bases de datos donde se encuentran los datos necesarios para el trabajo con el sistema. El servidor de base de datos elegido es PostgresSQL.

Conexión HTTP: Es el protocolo utilizado entre los navegadores de los clientes y el servidor *Web*.

3.4 Pruebas de Aceptación

Las pruebas de aceptación son aquellas en las que se revisa una versión del sistema cuando se entrega al cliente. Son normalmente pruebas de "caja negra" en las que el equipo de prueba se ocupa simplemente de demostrar que el sistema funciona o no correctamente. Si la entrega es lo suficientemente buena, el cliente puede entonces aceptarla para su uso (6).

Las pruebas de aceptación se describen mediante una tabla llamada Caso de Prueba de Aceptación, creadas sobre la base de las HU y destinadas a evaluar si al final de cada iteración se obtuvo la funcionalidad requerida. El cliente creará los casos de prueba en conjunto con los desarrolladores, según su punto de vista para probar que una HU ha sido implementada correctamente. Por cada HU se elaborarán todas las pruebas de aceptación que se necesiten para asegurar su correcto funcionamiento.

En la confección de la tabla Caso de Prueba de Aceptación se describen los siguientes aspectos:

- **Código:** Identifica la prueba en cuestión, incluye el número de HU, de la prueba y si posee diferentes escenarios.
- **HU:** Número de la HU a la cual pertenece.
- **Nombre:** Nombre del caso de prueba.
- **Descripción**: Acciones que debe realizar el sistema.
- **Condiciones de ejecución:** Describe las características y elementos en general que debe tener el sistema para realizar el caso de prueba.
- **Entrada/Pasos de Ejecución:** Incluye las entradas necesarias del sistema y los pasos necesarios para realizar el caso de prueba.
- **Resultados Esperados:** Respuesta que debe dar el sistema luego de aplicar el caso de prueba.
- **Resultado Obtenido:** Respuesta visual del sistema después de realizar el caso de prueba.
- **Evaluación de la prueba:** Clasificación de la prueba en satisfactoria o insatisfactoria.

  A continuación, se muestran algunos de los casos de pruebas aplicados al sistema:

Caso de Prueba de Aceptación

Código: HU1	**Historia de Usuario:** 1
Nombre: Elaborar pedido de la venta.	
Descripción: El sistema debe permitir escoger el producto deseado por el cliente y mostrar en una tabla los datos necesarios para realizar la venta.	
Condiciones de Ejecución: En la base de datos tiene que existir el producto para satisfacer el pedido.	
Entradas/Pasos de Ejecución: 1. Seleccionar el producto deseado por el cliente.	
Resultado Esperado: El sistema debe mostrar el producto seleccionado, en la tabla de ventas	
Resultado Obtenido: El producto fue mostrado correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 13: Caso de Prueba Elaborar pedido de venta

Caso de Prueba de Aceptación	
Código: HU4	**Historia de Usuario:** 1
Nombre: Calcular el precio total de la venta.	
Descripción: El sistema debe permitir calcular el precio total de la venta	
Condiciones de Ejecución: En el sistema se tiene que haber elaborado un pedido de venta.	
Entradas/Pasos de Ejecución: 1. Seleccionar el producto deseado por el cliente .	
Resultado Esperado: El sistema debe permitir que cada vez que se seleccione un producto se vaya calculando el precio total automáticamente.	

Resultado Obtenido: El precio total fue mostrado correctamente.
Evaluación de la Prueba: Satisfactoria.

Tabla 14: Caso de Prueba Calcular precio total de la venta.

Caso de Prueba de Aceptación	
Código: HU10	**Historia de Usuario:** 1
Nombre: Adicionar producto al almacén.	
Descripción: El sistema debe permitir adicionar un nuevo producto al almacén.	
Condiciones de Ejecución:	
Entradas/Pasos de Ejecución: La entrada corresponde a una instancia que se desea almacenar. **Pasos:** 1. Rellenar correctamente los campos correspondientes. 2. Hacer clic en el botón Add.	
Resultado Esperado: El producto debe almacenarse.	
Resultado Obtenido: El producto se almacenó correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 15: Caso de Prueba Adicionar producto al almacén.

En cada iteración se realizaron las pruebas correspondientes a cada HU desarrollada en ella. Una iteración termina cuando todas las pruebas tengan resultados satisfactorios, cuando la prueba de alguna HU es insatisfactoria se le realiza regresión para analizar e implementar correctamente los errores de la misma. El resultado de las pruebas que se le realizaron a las HU, en cada una de las iteraciones, se muestran en el siguiente gráfico.

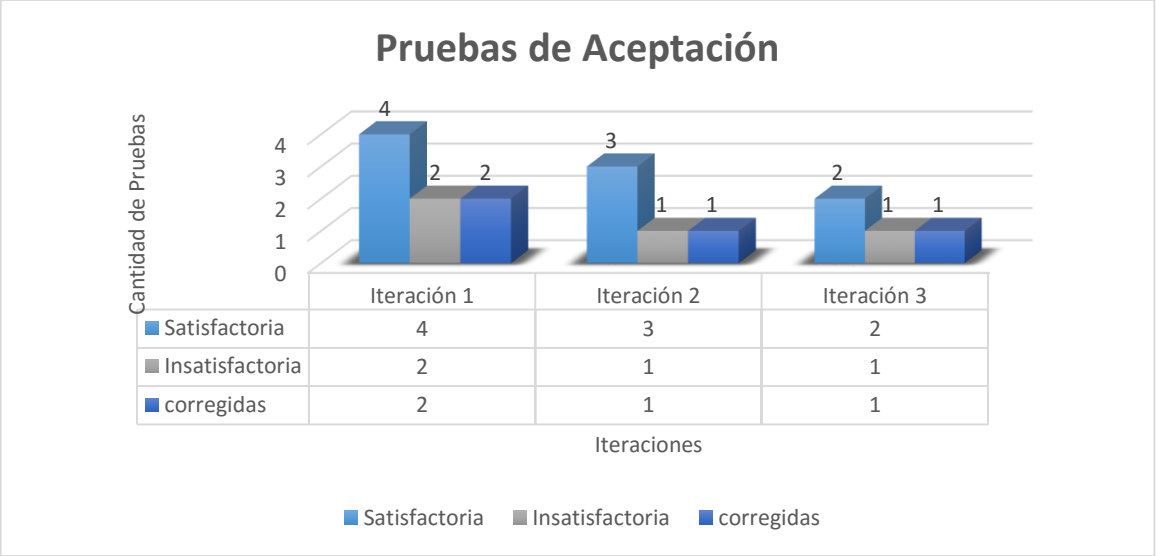


Ilustración 21: Grafico de resultados de las pruebas de aceptación.

Conclusiones Parciales

- Se elaboraron las tareas correspondientes a cada historia de usuario, en las cuales se implementaron las funcionalidades y se describió el código del módulo desarrollado para dar solución a la propuesta.
- Mediante las pruebas de aceptación se pudo documentar las no conformidades detectadas a lo largo de todas las iteraciones del proyecto, las cuales fueron corregidas, permitiendo que el sistema tenga un correcto funcionamiento.

Conclusiones generales

Con el desarrollo de la presente aplicación, se le dio cumplimiento a las tareas y objetivos propuestos inicialmente, por lo que una vez analizados los resultados se arriba a las siguientes conclusiones:

1. Mediante el análisis de las herramientas, tecnologías y aplicaciones similares se logró un mejor entendimiento del proceso de desarrollo del producto
2. El análisis y diseño de la solución propuesta estuvo guiado por la metodología de desarrollo de software AUP-UCI, la cual permitió general los artefactos necesarios para documentar la solución.
3. Mediante las pruebas realizadas se identificaron errores en el sistema, lo que ayudó a dar solución a los mismos para satisfacer el correcto y esperado funcionamiento del software implementado por el usuario final.

Recomendaciones

A partir de la investigación realizada y el desarrollo de la propuesta de solución, surgieron ideas para contribuir a la mejora del sistema desarrollado, por lo que se recomienda:

1. Implementar la vinculación del sistema desarrollado con los *drivers de la impresora en Linux.*
2. Implementar funcionalidades que permitan realizar análisis estadísticos a partir de las ventas realizadas.

Referencias

1. **Rendón, Carlos Augusto.** *La estrategia de ventas.* 2007.

2. **Thompson, Iván.** Promonegocios.net. *CONCEPTO DE VENTA.* [En línea] http://www.promonegocios.net/venta/concepto-de-venta.html

3. **Definicion.de.** Definicion.de. *Definición de punto de venta - Qué es, Significado y Concepto.* [En línea] http://definicion.de/punto-de-venta/, http://www.definicionabc.com/tecnologia/tecnologia.php

4. **Coronel Carlos Edgardo Tejada, Ejército Argentino.** *IDEAS ORIENTADORAS SOBRE EL SISTEMA LOGÍSTICO.*

5. **Lan, Alistair.** Shopify. *POS _ Usos y aplicaciones del software de Punto de Venta.* [En línea] https://es.shopify.com/blog/7978875-usos-y-aplicaciones-del-software-de-punto-de-venta

6. **DefinicionABC.** DefinicionABC. *ticket.* [En línea] http://www.definicionabc.com/economia/ticket.php

7. **DesdeLinux.** DesdeLinux. *El mejor software libre para tu Terminal de Punto de Venta (TPV_POS).* [En línea] http://blog.desdelinux.net/el-mejor-software-libre-para-tu-terminal-de-punto-de-venta-tpvpos/

8. **NCH Software.** Software caja registradora sistema PV de tienda (NCH Software). [En línea] http://www.nchsoftware.com/point-of-sale/es/index.html.

9. **GADAE NETWEB S.L.** Qué es una TPV y para qué sirve conceptos básicos. [En línea] http://www.gadae.com/blog/que-es-tpv-para-que-sirve/

10. **Soluciones Informaticas.** ***LemonPOS el punto de venta que estabas buscando.*** [En línea] https://alcesoluciones.wordpress.com/2010/08/09/lemonpos-el-punto-de-venta-que-estabas-buscando/

11. José Luis Cendejas Valdéz. eumed.net Enciclopedia Virtual. [En línea] http://www.eumed.net/tesis-doctorales/2014/jlcv/software.htm

12. José H. Canós, Patricio Letelier y Mª Carmen Penadés. ***Método logías Ágiles en el Desarrollo de Software.*** Valencia : s.n. s/n 46022.

13. Sánchez, Tamara Rodríguez. *Metodología de desarrollo para la Actividad productiva de la UCI.*

14. Visual Paradigm. [En línea] [Citado el: 23 de Marzo de 2016.] www.visual-paradigm.com

15. Lazo, Niusbel Hernández. *Clasificación Penitenciaria apoyado en Razonamiento Basado en Casos (RBC).* La Habana : s.n., 2013.

16. Alberto Basalo, Miguel Angel Alvarez. Que es AngularJS. *desarrolloweb.com.* [En línea] http://www.desarrolloweb.com/articulos/que-es-angularjs-descripcion-framework-javascript-conceptos.html

17. Mark Otto, Jacob Thornton. Bootstrap 3, el manual oficial. **[En línea] https://librosweb.es/libro/bootstrap_3/**

18. HIBERNATE . [En línea] febrero de 2015. [En línea] http://hibernate.org/orm/

19. Martínez, Rafael. PostgresSQL-es. [En línea] 2013. [En línea] http://www.postgresql.org.es/sobre_postgresql

20. Rubén Alvarez. DesarrolloWeb.com. *Lenguajes de lado servidor o cliente.* [En línea] 01 de Enero de 2001. http://www.desarrolloweb.com/articulos/239.php

21. Miguel Angel Álvarez. DesarrolloWeb.com. *Qué es HTML.* [En línea] 01 de Enero de 2001. http://www.desarrolloweb.com/articulos/que-es-html.html

22. Juan Diego Gauchat. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : MARCOMBO, 2012. 08007.

23. 830-1998., IEEE STD. "Especificaciones de los requisitos del Software".

24. Javier Garderes y Andres Gatto. *Extreme Programming.*

25. Beck, Kent y Andres, Cynthia. *Extreme Programming Explained: Embrace Change. s.l. : U. S.*

26. Joskowicz, José. *Reglas y Prácticas en eXtreme Programming .* 2008.

27. Botella, Andrés Vallés. *ASP.NET.* España : s.n., 2012.

28. Ian Sommerville. *Ingeniería de Software (Parte IV Desarrollo). 7ma.* s.l. : PEARSON Addison-Wesley, 2005. ISBN 84-7829-074-5.

29. Modelo de Implementación: Diagramas de Componentes y Despliegue. [En línea] http://www.dsi.uclm.es/asignaturas/42530/pdf/M2tema12.pdf

30. Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, Jonathan Morales Vélez. *Revisión de metodologías ágiles para el desarrollo de software.* 2012.

31. Aníbal de la Torre. Lenguajes del lado servidor o cliente. [En línea] 2006. http://www.adelat.org/media/docum/nuke_publico/lenguajes_del_lado_servidor_o_cliente.html

32. Enrique González Gutiérrez. APRENDERAPROGRAMAR.COM. *¿QUÉ ES PHP? ¿PARA QUÉ SIRVE PHP? UN POTENTE LENGUAJE DE PROGRAMACION PARA CREAR PÄGINAS WEB.* [En línea] 2006. http://www.aprenderaprogramar.com/index.php?option=com_content&id=492:ique-es-php-y-ipara-que-sirve-un-potente-lenguaje-de-programacion-para-crear-paginas-web-cu00803b&Itemid=193. CU00804B

33. Jorge Villalobos. Código Programación. *Ventajas de usar PHP.* [En línea] 05 de Septiembre de 2010. http://codigoprogramacion.com/articulos/programacionweb/ventajas-de-usar-php.html

34. Líber Batalla; Diego Fontanarossa; Javier Garderes; Andres Gatto. *Extreme Programming (XP).* 2006.

35. Kent Beck; Cynthia Andres. *Extreme Programming Explained: Embrace Change.* s.l. : U. S. Corporate and Government Sales, 1999. ISBN 0-321-27865-8.

36. Definicion abc. Definicion abc. [En línea] [En línea] http://www.definicionabc.com/tecnologia/tecnologia.php

ANEXO

ANEXO 1: Historias de Usuario.

Historia de Usuario	
Código: HU1	**Usuario:** Obrero
Nombre historia: Elaborar pedido de la venta	
Referencia:	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada:** 1
Descripción: El pedido se realiza a través de la interfaz principal donde aparecen los tipos de productos, aquí se escoge el producto deseado por el cliente, se accede automáticamente a la base de datos y se puede satisfacer el pedido con el stock disponible en el almacén, a continuación, se muestra en una tabla los campos necesarios para realizar la venta.	
Observaciones:	

Tabla 16: HU1-Elaborar pedido de venta.

Historia de Usuario	
Código: HU2	**Usuario:** Obrero
Nombre historia: Modificar pedido de la venta	
Referencia:HU-1	**Prioridad:** Media
Puntos estimados: 0.6	**Iteración asignada:** 1
Descripción: De la lista de productos que se muestran en la tabla se puede modificar la cantidad de productos deseados	

Observaciones:

Tabla 17: HU2-Modificar pedido de venta

Historia de Usuario	
Código: HU3	**Usuario:** Obrero
Nombre historia: Eliminar pedido de la venta	
Referencia:HU-1	**Prioridad:** Media
Puntos estimados: 0.6	**Iteración asignada: 1**
Descripción: Cuando se haga un pedido de venta el sistema debe permitir poder eliminar un producto de este pedido o eliminarlo por completo.	
Observaciones:	

Tabla 18: HU3-Eliminar pedido de venta.

Historia de Usuario	
Código: HU 4	**Usuario:** Obrero
Nombre historia: Calcular el precio total de la venta	
Referencia HU-1	**Prioridad:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 1
Descripción: Cuando ya están todos los productos seleccionados por el cliente en la lista es necesario que se calcule el precio total de la venta	

Observaciones:
El precio total se calculará como Σ precio*cantidad

Tabla 19: HU4-Calcular precio total de la venta

Historia de Usuario	
Código: HU5	**Usuario:** Obrero
Nombre historia: Verificar si se puede realizar la venta y si hay que devolver algún monto de dinero	
Referencia: HU-1,HU4	**Prioridad:** Baja
Puntos estimados: 0.4	**Iteración asignada:** 2
Descripción: Cuando ya se calcule el precio total de la venta y el cliente vaya a pagar se debe verificar si se entrega la cantidad de dinero requerida y se debe mostrar, si se tiene que devolver al cliente, la cantidad requerida.	
Observaciones:	

Tabla 20: HU5-Verificar si se puede realizar la venta y si hay que devolver monto de dinero

Historia de Usuario	
Código: HU 6	**Usuario:** Obrero
Nombre historia: Crear e Imprimir un recibo de pago	
Referencia:HU-1, HU-6	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada:** 2
Descripción: Cuando se tengan todos los datos necesarios para realizar la venta, el sistema debe permitir imprimir un recibo como constancia de pago.	

Observaciones:

Tabla 21: HU6 Crear e imprimir un recibo de pago.

Historia de Usuario	
Código: HU 7	**Usuario:** Obrero
Nombre historia: Insertar en el recibo de pago los datos de la venta	
Referencia HU-1, HU-4, HU-5	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada: 2**
Descripción: Inserta en la plantilla de recibo de venta los datos necesarios de los productos que están en la lista de venta para poder realizar la misma.	
Observaciones:	

Tabla 22: HU7 Insertar en el recibo de pago los datos de la venta.

Historia de Usuario	
Código: HU8	**Usuario:** Obrero
Nombre historia: Adicionar producto al almacén	
Referencia	**Prioridad:** Alta Alta
Puntos estimados: 1.6	**Iteración asignada:** 3
Descripción: Adicionar los productos nuevos al stock de la base de datos	
Observaciones:	

Tabla 23: HU8 Adicionar producto al almacén.

Historia de Usuario	
Código: HU 9	**Usuario:** Obrero
Nombre historia: Modificar el producto en el almacén	
Referencia	**Prioridad:** Alta
Puntos estimados: 1.6	**Iteración asignada: 3**
Descripción: Modificar los precios de los productos y las cantidades existentes en el inventario de la base de datos.	
Observaciones:	

Tabla 24: HU9 Modificar producto del almacén.

ANEXO 2: Tareas de ingeniería.

Iteración 1

Tarea de Ingeniería	
No. de tarea: 1	**No. de HU:** 1
Nombre de la tarea: Desarrollar funcionalidad que permita seleccionar un tipo de producto	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la vista principal; de la aplicación. Se muestra cada tipo de producto y esto tienes vinculados los productos de este tipo, debe permitir la selección del producto deseado por el cliente.	

Tabla 25:Seleccionar un tipo de producto.

Tarea de Ingeniería	
No. de tarea: 2	**No. de HU:** 1
Nombre de la tarea: Desarrollar funcionalidad que permita vincular cada producto a la base de datos por el id de este producto	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la base de datos la cual va a contener los datos de los productos. Permitir que cuando se escoja un producto en la interfaz, se recoja sus valores en la base de datos.	

Tabla 26: Vincular cada producto a la base de datos por el id de este producto.

Tarea de Ingeniería	
No. de tarea: 3	**No. de HU:** 1
Nombre de la tarea: Mostrar el producto seleccionado en la tabla de venta de productos	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.4
Fecha inicio:	**Fecha fin:**
Descripción: Una vez recogidos los valores del producto seleccionado, mostrar esto valores en un tabla, la cual será la lista de ventas.	

Tabla 27: Mostrar el producto seleccionado en la tabla de ventas.

Tarea de Ingeniería	
No. de tarea: 4	**No. de HU:** 2
Nombre de la tarea: Desarrollar una funcionalidad que permita seleccionar un producto de la tabla de ventas.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:** 6/3/2014
Descripción: En la tabla de venta de productos poder seleccionar uno para modificarle la cantidad de productos	

Tabla 28: Seleccionar un producto en la tabla de ventas

Tarea de Ingeniería	
No. de tarea: 5	**No. de HU:** 2
Nombre de la tarea: Desarrollar una funcionalidad que permita modificar el producto seleccionado	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la vista que muestre la cantidad de elementos que hay en la tabla de ventas del producto seleccionado. Implementar una funcionalidad que permita poder modificar este valor	

Tabla 29: Modificar un producto seleccionado en la tabla de ventas.

Tarea de Ingeniería	
No. de tarea: 6	**No. de HU:** 2

Nombre de la tarea: Desarrollar una funcionalidad que permita actualizar la tabla de ventas	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: cuando se haya modificado este valor en la interfaz, cambiar el valor de la tabla por el modificado	

Tabla 30: Actualizar tabla de ventas.

Tabla 31: Desarrollar una funcionalidad que permita seleccionar un producto de la tabla de ventasTarea de Ingeniería	
No. de tarea: 7	**No. de HU:** 3
Nombre de la tarea: Desarrollar una funcionalidad que permita eliminar el producto seleccionado.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de la vista que muestre la confirmación de que se desea eliminar el producto seleccionado	

Tabla 31:Eliminar el producto seleccionado en la tabla de ventas.

Tarea de Ingeniería	
No. de tarea: 8	**No. de HU:** 4
Nombre de la tarea: Desarrollar una funcionalidad que permita calcular el precio total de la venta	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación en la vista principal para que se muestre el valor del precio total de la venta. Calcular este valor utilizando los valores de la tabla de ventas .(precio-producto*cantidad)	

Tabla 32: Calcular precio total de la venta.

Tarea de Ingeniería	
No. de tarea: 9	**No. de HU:** 4
Nombre de la tarea: Desarrollar una funcionalidad que permita actualizar este precio total	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: Cada vez que se agregue un producto a la tabla de ventas se debe actualizar el valor del precio total de la venta.	

Tabla 33: Actualizar precio total de la venta.

Iteración 2

Tarea de Ingeniería	
No. de tarea: 10	**No. de HU:** 5
Nombre de la tarea: Desarrollar una funcionalidad que permita entrar la cantidad de efectivo para pagar	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación en la vista principal que permita entrar valores, los cuales van a ser el efectivo con que el cliente va a pagar la venta	

Tabla 34: Entrar cantidad de efectivo para pagar la venta

Tarea de Ingeniería	
No. de tarea: 11	**No. de HU:** 5
Nombre de la tarea: Desarrollar una funcionalidad que permita calcular si esta cantidad es suficiente para el pago.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.1
Fecha inicio:	**Fecha fin:**

Descripción: Comparar el precio total de la venta con el valor entrado y solo si el valor entrado es mayor se puede hacer la venta.

Tabla 35: Calcular si la cantidad entrada es suficiente para el pago.

Tarea de Ingeniería	
No. de tarea: 12	**No. de HU:** 5
Nombre de la tarea: Desarrollar una funcionalidad que permita mostrar la cantidad de efectivo a devolver	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.2
Fecha inicio:	**Fecha fin:**
Descripción: Calcular este valor utilizando los valores de precio total de la venta y el valor entrado (valor entrado – precio total). Se realiza el diseño e implementación en la vista principal para que se muestre el valor del vuelto.	

Tabla 36: Mostrar cantidad de efectivo a devolver

Tarea de Ingeniería	
No. de tarea: 13	**No. de HU:** 6
Nombre de la tarea: Desarrollar una funcionalidad que permita crear una plantilla de recibo de pago.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de una plantilla de recibo de pago.	

Tabla 37: Crear una plantilla del recibo de pago.

Tarea de Ingeniería	
No. de tarea: 14	**No. de HU:** 6
Nombre de la tarea: Desarrollar una funcionalidad que permita imprimir esta plantilla de recibo de pago	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6

Fecha inicio:	**Fecha fin:**
Descripción: implementar un algoritmo que permita la impresión de este recibo de pago	

Tabla 38: Imprimir recibo de pago.

Tarea de Ingeniería	
No. de tarea: 15	**No. de HU:** 7
Nombre de la tarea: Desarrollar una funcionalidad que permita recoger los datos necesarios de la venta.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de una plantilla de recibo de pago.	

Tabla 39: Recoger los datos de la venta.

Tarea de Ingeniería	
No. de tarea: 16	**No. de HU:** 7
Nombre de la tarea: Desarrollar una funcionalidad que permita insertas los datos de la venta en la plantilla de recibo de pago.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de una plantilla de recibo de pago.	

Tabla 40: Insertar los datos de la venta en el recibo de pago.

Iteración 3

Tarea de Ingeniería	
No. de tarea: 17	**No. de HU:** 8
Nombre de la tarea: Desarrollar una interfaz que permita adicionar los datos de un nuevo producto.	

Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.6
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de una interfaz donde se muestren todos los campos necesarios para adicionar un nuevo producto.	

Tabla 41: Adicionar nuevo producto.

Tarea de Ingeniería	
No. de tarea: 18	**No. de HU:** 8
Nombre de la tarea: Desarrollar una funcionalidad que permita almacenar este producto en la base de datos.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 1.0
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza una funcionalidad que permita guardar los datos entrados en la interfaz y adicionarlos en la base de datos como un producto.	

Tabla 42: Almacenar los datos del nuevo producto en la base de datos

Tarea de Ingeniería	
No. de tarea: 19	**No. de HU:** 9
Nombre de la tarea: Desarrollar una interfaz que permita seleccionar un producto en la base de datos.	
Tipo de tarea: Desarrollo.	**Puntos estimados:** 0.4
Fecha inicio:	**Fecha fin:**
Descripción: Se realiza el diseño e implementación de una interfaz donde a partir de un dato de un producto este de muestre en la interfaz.	

Tabla 43: Seleccionar producto de la base de datos.

ANEXO 2: Pruebas de Aceptación.

Caso de Prueba de Aceptación	
Código: HU1	**Historia de Usuario:** 1
Nombre: Elaborar pedido de la venta.	
Descripción: El sistema debe permitir escoger el producto deseado por el cliente y mostrarlo en una tabla los campos necesarios para realizar la venta.	
Condiciones de Ejecución: En la base de datos tiene que existir el producto para satisfacer el pedido.	
Entradas/Pasos de Ejecución: 2. Seleccionar el producto deseado por el cliente.	
Resultado Esperado: El sistema debe mostrar el producto seleccionado, en la tabla de ventas	
Resultado Obtenido: El producto fue mostrado correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 44: Prueba de aceptación a la HU1

Caso de Prueba de Aceptación	
Código: HU2	**Historia de Usuario:** 1
Nombre: Modificar pedido de la venta.	
Descripción: El sistema debe permitir modificar la cantidad de un producto seleccionado.	
Condiciones de Ejecución: : En el sistema se tiene que haber elaborado un pedido de venta.	

Entradas/Pasos de Ejecución: 1. Hacer clic en el botón Modificar. 2. Rellenar el campo que aparece en la ventana con la cantidad deseada. 3. Hacer clic en el botón Guardar.
Resultado Esperado: La cantidad del producto debe modificarse.
Resultado Obtenido: La cantidad del producto se modificó correctamente.
Evaluación de la Prueba: Satisfactoria.

Tabla 45: Prueba de aceptación a la HU2.

Caso de Prueba de Aceptación	
Código: HU3	**Historia de Usuario:** 1
Nombre: Eliminar pedido de la venta.	
Descripción: El sistema debe permitir eliminar del pedido de venta el producto seleccionado.	
Condiciones de Ejecución: En el sistema se tiene que haber elaborado un pedido de venta.	
Entradas/Pasos de Ejecución: 1. Hacer clic en el botón Eliminar .	
Resultado Esperado: El sistema debe eliminar el producto seleccionado en la tabla de ventas	
Resultado Obtenido: El producto fue eliminado correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 46: Prueba de aceptación a la HU3

Caso de Prueba de Aceptación	
Código: HU4	**Historia de Usuario:** 1
Nombre: Calcular el precio total de la venta.	
Descripción: El sistema debe permitir calcular el precio total de la venta	
Condiciones de Ejecución: En el sistema se tiene que haber elaborado un pedido de venta.	
Entradas/Pasos de Ejecución: 2. Seleccionar el producto deseado por el cliente .	
Resultado Esperado: El sistema debe permitir que cada vez que se seleccione un producto se valla calculando el precio total automáticamente.	
Resultado Obtenido: El precio total fue mostrado correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 47: Prueba de aceptación a la HU4

Caso de Prueba de Aceptación	
Código: HU5	**Historia de Usuario:** 1
Nombre: Verificar si se puede realizar la venta y si hay que devolver algún monto de dinero.	
Descripción: El sistema debe permitir saber si el dinero entrado es suficiente para la venta y debe calcular la cantidad a devolver.	
Condiciones de Ejecución: En el sistema se tiene que haber elaborado un pedido de venta	
Entradas/Pasos de Ejecución:	

1. Entrar el monto con que se va a pagar la venta. 2. Hacer clic en el botón Enter.
Resultado Esperado: El sistema debe calcular si el monto entrado es suficiente para realizar la venta y mostrar el cambio en caso necesario.
Resultado Obtenido: El sistema muestra el cambio correctamente.
Evaluación de la Prueba: Satisfactoria.

Tabla 48: Prueba de aceptación HU5.

Caso de Prueba de Aceptación	
Código: HU6	**Historia de Usuario:** 1
Nombre: Crear e Imprimir un recibo de pago.	
Descripción: El sistema debe permitir imprimir un recibo de pago .	
Condiciones de Ejecución: En el sistema se tiene que tener todos los datos de la venta.	
Entradas/Pasos de Ejecución: 1. Hacer clic en el botón RENDER.	
Resultado Esperado: El sistema debe imprimir un recibo de pago con los datos de la venta.	
Resultado Obtenido: El sistema imprime el recibo correctamente.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 49: Prueba de aceptación a la HU6.