UNIVERSIDAD D CIENFUEGOS
INGENIERÍA
DEPARTAMENTO DE INFORMÁTICA

EPPC
Empresa Productora de Piensos Cerdo

Universidad de Cienfuegos

"Carlos Rafael Rodríguez"

Facultad de Ingeniería

Título: Sistema Informático para la Gestión de Ventas en la Fábrica de Piensos Cienfuegos

Trabajo de diploma para optar por el título de Ingeniero Informático

Autor:

Krystian Enrique Martínez Yanes

Tutores:

Ing. Isali Azpiri Medina

Ing. Ricardo Ernesto Velázquez Herrera

Cienfuegos, Cuba

Curso 2023

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad Cienfuegos los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____ (Autor)

_____ (Autor)

_____ (Tutor)

_____ (Cotutor)

_____ (Tutor)

RESUMEN

La presente tesis aborda el desarrollo de una aplicación web para la gestión de ventas de piensos en la Fábrica de Piensos Cienfuegos. Esta investigación surge de la necesidad social de informatizar a la empresa en el proceso. El objetivo principal del estudio es diseñar y desarrollar una aplicación web innovadora que facilite y mejore el proceso de ventas, además de proporcionar herramientas estadísticas para la toma de decisiones empresariales. Los métodos principales empleados en este trabajo incluyen la realización de un exhaustivo análisis documental, una revisión bibliográfica, además de entrevistas a potenciales usuarios. Estas metodologías han permitido obtener una comprensión profunda del problema y validar la efectividad de la aplicación web propuesta. Los resultados logrados han demostrado que la implementación de esta aplicación web puede mejorar significativamente la eficiencia y precisión del proceso de ventas de la empresa, así como facilitar la gestión de entidades y representantes. Además, las herramientas estadísticas integradas en la aplicación brindan información valiosa para la toma de decisiones y la mejora continua del negocio. Se utilizaron diversas tecnologías y herramientas como Visual Studio, Python, Django, HTML, CSS, Bootstrap, SQLite, Axure, Visual Paradigm, Enterprise Architect y DbVisualizer para llevar a cabo el diseño y desarrollo del sistema informático. Esta tesis ha logrado desarrollar una solución práctica y efectiva para la Fábrica de Piensos Cienfuegos, brindando una aplicación que optimiza el proceso de ventas y proporciona herramientas estadísticas relevantes.

Palabras clave: sistema informático, aplicación web, eficiencia, tecnologías, herramientas.

ABSTRACT

This thesis addresses the development of a web application for managing the sales of animal feed at Cienfuegos Feed Factory. This research arises from the societal need to computerize the company's processes. The primary objective of the study is to design and develop an innovative web application that facilitates and enhances the sales process, in addition to providing statistical tools for business decision-making. The main methods employed in this work include an exhaustive documentary analysis, a literature review, as well as interviews with potential users. These methodologies have allowed for a deep understanding of the problem and validation of the effectiveness of the proposed web application. The achieved results have demonstrated that the implementation of this web application can significantly improve the efficiency and accuracy of the company's sales process, as well as facilitate the management of entities and sales representatives. Furthermore, the statistical tools integrated into the application provide valuable information for decision-making and continuous business improvement. A variety of technologies and tools such as Visual Studio, Python, Django, HTML, CSS, Bootstrap, SQLite, Axure, Visual Paradigm, Enterprise Architect, and DbVisualizer were used to design and develop the computer system. This thesis has succeeded in developing a practical and effective solution for Cienfuegos Feed Factory, providing an application that optimizes the sales process and offers relevant statistical tools.

Keywords: computer system, web application, efficiency, technologies, tools.

ÍNDICE

Índice de Figuras

Índice de Tablas

INTRODUCCIÓN

La informática es una disciplina que se encarga del estudio y desarrollo de sistemas informáticos, que incluyen hardware, software, redes y aplicaciones. Se trata de una herramienta fundamental en la sociedad moderna, ya que permite el procesamiento, almacenamiento y transmisión de información de manera rápida y eficiente. La informática ha revolucionado la forma en que las personas se comunican, trabajan y se relacionan entre sí, y su importancia sigue creciendo en el mundo actual.[1]

Las Tecnologías de la Información y las Comunicaciones (TIC) son un conjunto de herramientas, dispositivos y sistemas que permiten la gestión, procesamiento, almacenamiento y transmisión de información a través de medios electrónicos. Estas tecnologías son fundamentales en la sociedad actual, ya que permiten una comunicación más rápida y eficiente, así como el acceso a una gran cantidad de información en tiempo real.

La historia de las TIC se remonta a la invención del telégrafo en 1837, que permitió la transmisión de mensajes a larga distancia. Posteriormente, con la invención del teléfono en 1876 y la radio en 1895, se abrieron nuevas posibilidades de comunicación. Sin embargo, fue la invención del ordenador en los años 50 lo que realmente impulsó el desarrollo de las TIC. A partir de entonces, se han desarrollado una gran cantidad de dispositivos y sistemas electrónicos que han revolucionado la forma en que las personas se comunican y acceden a la información.

En el contexto de la ingeniería informática, las TIC son fundamentales para el diseño, desarrollo y mantenimiento de sistemas informáticos y de comunicaciones. Los ingenieros informáticos utilizan estas tecnologías para crear software, redes y sistemas de información que permiten a las empresas y organizaciones gestionar sus datos y procesos de forma eficiente.

En la actualidad, las TIC tienen una importancia fundamental en el ámbito internacional, ya que permiten la comunicación y el intercambio de información a nivel global. Las redes sociales, el correo electrónico y los servicios de mensajería instantánea han transformado la forma en que las personas se comunican entre sí, eliminando las barreras geográficas y culturales.

En Cuba, el desarrollo de las TIC ha sido un tema de gran importancia en los últimos años, ya que se ha reconocido que su uso puede contribuir significativamente al crecimiento económico y social del país. A pesar de que Cuba ha enfrentado limitaciones en cuanto a la conectividad y el acceso a internet, el gobierno ha implementado diversas políticas para fomentar el uso de las TIC en la sociedad cubana. En la actualidad, se están llevando a cabo importantes proyectos de infraestructura y servicios para mejorar la conectividad y el acceso a internet en todo el país, lo que permitirá una mayor inclusión digital y una mejor integración en la economía global. Es fundamental destacar que el uso de las TIC

en Cuba no solo tiene un impacto económico, sino también social, ya que puede contribuir a mejorar la calidad de vida de los ciudadanos y a fortalecer la participación ciudadana en la toma de decisiones.

En la actualidad, esta situación se manifiesta **en la provincia de Cienfuegos** en la falta de informatización de algunas empresas en materia de procesos de venta, así como en la dificultad para realizar pedidos. Esto se traduce en una pérdida de tiempo y recursos.

El **problema a resolver** que esta tesis busca abordar radica en la actual ineficiencia operativa en el proceso empresarial actualmente sin respaldo de la tecnología y limitaciones en la toma de decisiones estratégicas experimentadas por la Fábrica de Piensos Cienfuegos. Actualmente se gestiona las ventas de manera manual, lo que resulta en procesos propensos a errores, falta de herramientas estadísticas para análisis de ventas y reportes, así como en un acceso limitado a la información de ventas de manera oportuna y colaborativa. Esta situación obstaculiza la eficiencia operativa, la precisión en los datos registrados, y limita la capacidad de la empresa para adaptarse al crecimiento y tomar decisiones basadas en datos concretos, representando un desafío crucial que demanda una solución tecnológica eficaz.

Por lo antes planteado se puede identificar como principal interrogante en la presente investigación: **¿Cómo mejorar la gestión de ventas de la Fábrica de Piensos Cienfuegos?**

Esta investigación se enmarca en el **objeto de estudio:** el desarrollo de aplicaciones web para la mejora de procesos empresariales.

Su **campo de acción** se enfoca en el área de la venta al consumidor de piensos.

Para darle solución al problema, se plantea como **objetivo general:** desarrollar una aplicación web que permita a la Fábrica de Piensos Cienfuegos registrar, analizar y brindar información de sus productos y entidades en comercio, y otras estadísticas relevantes.

Como objetivos específicos podemos resaltar:

1. Realizar un estudio exhaustivo del estado del arte en la venta al consumidor de piensos, para identificar las mejores prácticas y herramientas tecnológicas innovadoras utilizadas por empresas nacionales e internacionales.

2. Desarrollar una aplicación web que permita a la Fábrica de Piensos Cienfuegos un proceso empresarial de manera informatizada.

3. Validar la eficacia de la aplicación web mediante pruebas de usabilidad.

Idea a defender: El desarrollo de un sistema informático en la Fábrica de Piensos Cienfuegos para gestionar las ventas mejorará el proceso operativo actual sin respaldo de tecnología.

Como **aporte práctico** se plantea: La obtención de un sistema para la informatización de la gestión del proceso de ventas en la Fábrica de Piensos Cienfuegos. Esta aplicación web permitirá realizar este proceso de gestión respaldado por las tecnologías actuales y garantizar la eficiencia en la toma de decisiones empresariales estratégicas.

El método científico de investigación es un proceso sistemático y riguroso utilizado para descubrir y validar conocimientos científicos. Se basa en la observación, la formulación de hipótesis, la experimentación, el análisis de datos y la conclusión. El objetivo principal del método científico es obtener resultados objetivos y confiables que puedan ser replicados y verificados por otros investigadores. Se puede clasificar en métodos de nivel teórico o empírico los cuales están relacionados entre sí de forma dialéctica. Métodos utilizados para el desarrollo de esta investigación:

Métodos del nivel teórico:

1. Método Analítico-Sintético: Posibilitó realizar un análisis de las distintas partes que afectan el objeto de estudio y sintetizar los elementos más significativos.

2. Método Histórico-Lógico: Para la realización de la investigación se hizo necesario estudiar la evolución del problema y la existencia de metodologías, procedimientos y sistemas informáticos similares al que se pretende elabora.

Métodos del nivel empírico:

1. Observación: Se empleó para identificar características en el proceso de gestión de tesis como la forma de realización, quiénes intervienen, qué utilizan. De igual manera se empleó para identificar las carencias más significativas en este proceso.

2. Entrevista: Se utilizó para obtener información de la empresa acerca de todos los aspectos necesario para la realización de esta investigación.

La presentación de los resultados de la investigación se divide en tres secciones en el documento de presentación:

Capítulo 1 “Fundamentación Teórico-Metodológica”:

En este capítulo se exponen los fundamentos teóricos del proceso de ventas en la Fábrica de Piensos Cienfuegos, los cuales son relevantes para la investigación del problema que se ha planteado. Se examinan los conceptos principales relacionados con el objeto de estudio, el campo de acción, los

objetivos generales y específicos, así como la metodología y la variante seleccionada, junto con una descripción de las herramientas y tecnologías que podrían ser útiles para llevar a cabo la propuesta de solución.

Capítulo 2 “Descripción y Construcción de la Solución Propuesta”:

En este capítulo se detalla la propuesta de solución y los elementos creados en el cuarto escenario de la metodología Agile Iniciad Procesos (AUP) adaptada para la Fábrica de Piensos Cienfuegos, durante las fases de análisis, diseño e implementación.

Capítulo 3 “Validación de la Solución y Estudio de Factibilidad”:

En este capítulo se examina si la solución propuesta es factible a través de la verificación de los requerimientos y el plan diseñado.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En el presente trabajo de investigación se aborda el tema de la implementación de un sistema de informatización para mejorar la gestión de procesos en una empresa. En este sentido, uno de los capítulos más importantes de la tesis es el de Fundamentos teórico-metodológicos, en el cual se profundiza en los conceptos y herramientas necesarias para llevar a cabo este proyecto. En este capítulo se analizarán los sistemas de informatización homólogos existentes en el mercado, con el fin de identificar las mejores prácticas y soluciones utilizadas por otros investigadores en el campo. Además, se abordarán los fundamentos teóricos y metodológicos necesarios para la implementación de un sistema de informatización eficiente y efectivo, así como las herramientas y técnicas que se utilizarán para su desarrollo. Todo ello con el objetivo de ofrecer una visión clara y completa del proyecto y sentar las bases para su correcta ejecución.

1.1 Referentes Teóricos

La presente tesis tiene como objetivo el desarrollo de una aplicación web para la venta de pienso por una empresa en Cienfuegos, Cuba. En este sentido, se abordan conceptos clave como es el proceso de informatización en una empresa, la experiencia del usuario, el diseño centrado en el usuario, la innovación tecnológica, tecnologías web, usabilidad, la gestión de datos, la estandarización, la eficiencia empresarial, también en que consiste un sistema de gestión de ventas y la responsabilidad social empresarial. La comprensión de estos conceptos es fundamental para el desarrollo de una aplicación eficaz que permita a la empresa mejorar su presencia en el mercado.

1. Proceso de informatización: en una empresa se refiere a la implementación de tecnologías de la información y la comunicación (TIC) en los procesos y actividades de la organización, con el objetivo de mejorar la eficiencia, productividad y competitividad. Esto implica la digitalización de documentos y datos, la automatización de procesos y la integración de sistemas informáticos para optimizar la gestión empresarial. Además, la informatización permite una mayor capacidad de análisis y toma de decisiones basadas en datos precisos y actualizados, lo que contribuye al crecimiento y éxito de la empresa en un entorno cada vez más digitalizado.

2. Experiencia de usuario (UX): La experiencia del usuario se refiere a la calidad que una persona obtiene cuando interactúa con un diseño específico. El objetivo del diseño centrado en el usuario es conseguir que el diseño sea fácil de usar. Al final, sabemos lo frustrante que es utilizar un dispositivo que no funciona de la manera que queremos o navegar por un sitio web que no es muy amigable. El diseño centrado en el usuario es un proceso iterativo que dirige sus objetivos a los usuarios y sus

necesidades. Sus creadores se apoyan en una variedad de técnicas de investigación y diseño para crear productos altamente utilizables y accesibles.

3. Diseño centrado en el usuario: El diseño centrado en el usuario es un enfoque de diseño que se centra en las necesidades y expectativas de los usuarios finales. Este enfoque es importante en el desarrollo de la aplicación web para garantizar que se adapte a las necesidades y expectativas de los consumidores.

4. Innovación tecnológica: La innovación tecnológica se refiere a la creación y desarrollo de nuevos productos, procesos o servicios con la ayuda de tecnologías avanzadas. La innovación tecnológica puede ser disruptiva o incremental. La innovación disruptiva se refiere a la creación de nuevos productos o servicios que cambian radicalmente la forma en que se hacen las cosas, mientras que la innovación incremental se refiere a mejoras graduales en los productos o servicios existentes.

5. Tecnologías web: Las tecnologías web son herramientas y recursos utilizados para crear sitios web y aplicaciones web. Las tecnologías web incluyen lenguajes de programación como HTML, CSS y JavaScript, así como marcos y bibliotecas como React, Angular y Vue.

6. Gestión de datos: La gestión de datos se refiere al proceso de recopilar, almacenar, organizar y mantener datos. La gestión efectiva de datos es importante para garantizar la precisión y la integridad de los datos.

7. Estandarización: La estandarización se refiere a la creación de normas y estándares en un área específica para garantizar la calidad y la compatibilidad. Es importante que la aplicación web que se desarrollará siga los estándares de la industria para garantizar la calidad y la compatibilidad.

8. Eficiencia empresarial: La eficiencia empresarial se refiere a la capacidad de una empresa para utilizar eficazmente sus recursos para lograr sus objetivos. Se espera que la aplicación web propuesta mejore la eficiencia empresarial.

9. Sistema de Gestión de Ventas: Es un conjunto de herramientas y procesos diseñados para optimizar la gestión de las ventas de una empresa. Este sistema incluye la informatización de los procesos de venta, la gestión de clientes, la gestión de inventario, la generación de informes y la integración con otros sistemas empresariales. El objetivo principal de un sistema de gestión de ventas interno es mejorar la eficiencia y efectividad del proceso de venta para aumentar las ventas y maximizar los beneficios de la empresa. En el caso específico de una empresa vendedora de pienso, un sistema de gestión de ventas interno puede ayudar a optimizar la gestión de los pedidos, el control del inventario, la gestión de los clientes y la facturación, entre otros aspectos clave del negocio.

10. Responsabilidad social empresarial: La responsabilidad social empresarial (RSE) se refiere a las prácticas comerciales éticas y sostenibles adoptadas por una empresa para garantizar su impacto positivo en la sociedad y el medio ambiente. Las empresas pueden adoptar prácticas RSE mediante la implementación de políticas ambientales sostenibles, prácticas laborales justas y éticas, así como contribuciones a la comunidad local.

1.2 Uso de las TIC en el desarrollo de aplicaciones web para la venta de productos

El uso de las Tecnologías de la Información y Comunicación (TIC) en el desarrollo de aplicaciones web para la venta de productos es fundamental para mejorar la eficiencia y competitividad de las empresas. En la actualidad, las empresas y consumidores buscan cada vez más la comodidad y rapidez en su comercio.

Las TIC permiten el desarrollo de plataformas web que ofrecen una experiencia de compra fácil, rápida y segura para los usuarios. Las aplicaciones web para la venta de productos pueden incluir características como la visualización de catálogos de productos, la selección de productos, la visualización de estadísticas de calidad, entre otras.

Además, el uso de las TIC permite a las empresas recopilar información valiosa sobre el comportamiento del consumidor, lo que les permite mejorar sus estrategias de marketing y ventas. Las plataformas web también pueden integrarse con herramientas de análisis de datos y estadísticas, lo que permite a las empresas tomar decisiones informadas sobre su estrategia de ventas.

En resumen, el uso de las TIC en el desarrollo de aplicaciones web para la venta de productos es esencial para mejorar la eficiencia y competitividad de las empresas en un mercado cada vez más digitalizado. Las aplicaciones web permiten una experiencia de compra fácil y segura para los usuarios, al mismo tiempo que proporcionan información valiosa para mejorar las estrategias de marketing y ventas.

1.3 Soporte para sistemas informáticos

Internacionales:

HubSpot Sales Hub: HubSpot Sales Hub es un software de ventas desarrollado por HubSpot, una empresa especializada en software de inbound marketing, ventas y servicio al cliente. Sales Hub es una herramienta de CRM para equipos de ventas que permite recopilar datos relevantes de prospectos, automatizar tareas tediosas y cerrar negocios con mayor rapidez. Con Sales Hub, podrás gestionar tus leads y actividades de ventas en un espacio de trabajo personalizado, convertir más leads en negocios y optimizar tus correos electrónicos de ventas con plantillas personalizables. También podrás hacer un

seguimiento de tus correos electrónicos y obtener información sobre cuándo los leads los abren. Además, Sales Hub ofrece funciones avanzadas como automatización de ventas, seguimiento de llamadas, asistente con IA y programador de reuniones.[2]

Infusionsoft: Es un software de ventas y marketing automatizado que combina el sistema de gestión de correo electrónico, el comercio electrónico, el marketing a través de las redes sociales y la creación de campañas a largo plazo en base a las reacciones de los posibles clientes o los que ya tienes en tu base de datos. La plataforma está diseñada explícitamente para pequeñas empresas que desean una plataforma centralizada para gestionar su CRM, marketing por correo electrónico y comercio electrónico. Infusionsoft es un software bien planteado y con buenas perspectivas de futuro.[3]

Parrot CRM: Parrot CRM es un software en la nube que optimiza el manejo de clientes, ventas, marketing y postventa. Agiliza el tiempo de respuesta de todas las conversaciones que tus asesores comerciales llevan a cabo con prospectos y clientes, para lograr mayor conversión y no perder más oportunidades. Parrot CRM es una herramienta diseñada para incrementar la productividad de los equipos de ventas, marketing y servicio al cliente. Con Parrot CRM podrás crear respuestas automáticas a las preguntas más frecuentes de tus clientes, ahorrar tiempo contestando con un solo click, asignar etiquetas, notas y actividades de seguimiento a tus prospectos para retomar la venta en el momento justo, mandar cotizaciones y recibos en PDF, gestionar tareas y actividades de seguimiento, y más. Además, Parrot CRM es adaptable para grandes empresas con equipos de ventas muy grandes, así como pymes, emprendedores y freelancers que requieren de mayor orden en sus flujos de venta y ser altamente efectivos en la conversión.[4]

Nacionales:

xEasy Business Suite: Ofrece funciones como stock, trazabilidad, envasado, materias primas y producción.[5]

GsBase: Ofrece funciones como stock, trazabilidad, envasado, materias primas y producción.[5]

SAP Business One Alimentación: Ofrece funciones como stock, trazabilidad, etiquetado, envasado y producción.[5]

Software	Tipo de aplicación	Privado / Libre	Enfoque en venta de pienso	Proporción de datos estadísticos	Reserva de ventas	Tecnologías utilizadas
HubSpot Sales Hub	Aplicación web	Privado	No especificado	Alta	Sí	Cloud, HTML, CSS, JavaScript
Infusionsoft	Aplicación web	Privado	Sí	Alta	Sí	Cloud, HTML, CSS, JavaScript
Parrot CRM	Aplicación web	Privado	No especificado	Media	Sí	Cloud, HTML, CSS, JavaScript
xEasy Business Suite	Aplicación web	Privado	Sí	Media	No especificado	Cloud, HTML, CSS, JavaScript
GsBase	Software privado	Privado	Sí	Baja	Sí	No especificado
SAP Business One Alimentación	Software privado	Privado	Sí	Alta	Sí	No especificado

Tabla 1.1 Comparativa de Softwares Internacionales y Nacionales

Al evaluar cuidadosamente las opciones disponibles se pudo concluir que estas tecnologías de venta internacionales o nacionales no se pueden aplicar en Fábrica de Piensos Cienfuegos porque no están desarrolladas con los procedimientos y políticas propias de la empresa para la venta de pienso a las entidades del país.

En consecuencia, es necesario desarrollar tecnologías de venta que estén diseñadas específicamente para la Fábrica de Piensos Cienfuegos para su gestión del comercio a entidades del país.

1.4 Metodología de desarrollo de software

La metodología de desarrollo de software es un conjunto de procesos, técnicas y herramientas que se utilizan para planificar, diseñar, construir, probar y mantener sistemas de software. Es un enfoque estructurado que permite a los equipos de desarrollo de software trabajar de manera más eficiente y efectiva para producir software de alta calidad dentro del plazo y presupuesto establecidos.

Existen diversas metodologías de desarrollo de software, cada una con sus propias ventajas y desventajas. Algunas de las metodologías más populares son:

- Modelo en cascada: esta metodología sigue un enfoque secuencial y lineal, en el que cada fase del proceso se completa antes de pasar a la siguiente. Es útil para proyectos con requisitos bien definidos y estables, pero puede ser inflexible si se requieren cambios en el proceso.

- Modelo en espiral: esta metodología se basa en un ciclo iterativo de planificación, análisis, diseño y evaluación, en el que se van refinando y mejorando los resultados en cada iteración. Es adecuada para proyectos con requisitos cambiantes o poco claros, pero puede ser costosa y compleja de implementar.

- Metodología ágil: esta metodología se centra en la entrega rápida y continua de software funcional a través de ciclos cortos de desarrollo y pruebas. Se basa en la colaboración estrecha entre los miembros del equipo y en la adaptación constante a los cambios en los requisitos del proyecto. Es adecuada para proyectos con requisitos cambiantes o poco claros y para equipos pequeños y altamente colaborativos.

Para elegir la metodología adecuada para un proyecto específico, es importante considerar factores como el tamaño del equipo, la complejidad del proyecto, los requisitos del cliente y el presupuesto disponible. Además, es importante tener en cuenta que la metodología elegida puede evolucionar y adaptarse a lo largo del proyecto a medida que se adquiere más información y se enfrentan nuevos desafíos.

En cuanto a las herramientas y técnicas utilizadas en la metodología de desarrollo de software, algunas de las más comunes son:

- Diagramas de flujo de datos: se utilizan para visualizar el flujo de datos a través del sistema y para identificar los procesos, entidades y relaciones involucrados.

- Diagramas de casos de uso: se utilizan para describir las interacciones entre los usuarios y el sistema y para identificar los requisitos funcionales del sistema.

- Pruebas unitarias: se utilizan para probar cada componente individual del sistema y asegurarse de que funciona correctamente.

- Integración continua: se utiliza para integrar y probar continuamente el código fuente del sistema a medida que se desarrolla.

- Control de versiones: se utiliza para gestionar y controlar los cambios en el código fuente del sistema a lo largo del tiempo.

En resumen, la metodología de desarrollo de software es un enfoque estructurado y planificado para la creación de sistemas de software. La elección de la metodología adecuada depende de varios factores, y las herramientas y técnicas utilizadas pueden variar según la metodología elegida. Lo importante es tener un enfoque riguroso y disciplinado para garantizar la calidad y eficiencia del software producido.

Metodología AUP

La metodología AUP (Proceso Unificado Ágil) es una versión simplificada del Proceso Unificado de Rational (RUP) que 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. El AUP aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (test driven development – TDD), Modelado Ágil, Gestión de Cambios Ágil, y Refactorización de Base de Datos para mejorar la productividad . La metodología AUP fue desarrollada por Scott Ambler.[6]

Una de las principales ventajas de la Metodología AUP es que es altamente adaptable a diferentes tipos de proyectos y equipos de desarrollo. A diferencia de otras metodologías ágiles, como Scrum o Kanban, AUP ofrece una estructura más completa y detallada, que incluye fases específicas para la planificación, análisis, diseño, construcción y despliegue del software.

Otra ventaja de la Metodología AUP es que se enfoca en la colaboración estrecha entre los miembros del equipo y en la adaptación constante a los cambios en los requisitos del proyecto. Esto se logra a través de reuniones diarias de seguimiento, revisiones periódicas del progreso del proyecto y un enfoque iterativo e incremental en el desarrollo del software.

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (Casos de Uso del Negocio (CUN), Descripción de Proceso de Negocio (DPN) o Mapa Conceptual (MC) y existen tres formas de encapsular los requisitos (Casos de Uso del Sistema (CUS), Historias de Usuario (HU), Descripción de Requisito por Proceso (DRP)), surgen cuatro escenarios para modelar el sistema en los proyectos, manteniendo en dos de ellos el MC, quedando de la siguiente forma:

Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.
Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.
Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.
Considerando lo previamente mencionado acerca de los escenarios en la disciplina de requisitos de software dentro de la metodología AUP, se opta por elegir el cuarto escenario, el cual es aplicable a proyectos que hayan evaluado el negocio a informatizar y cuenten con un negocio bien definido como resultado. En este escenario, el cliente estará presente en todo momento junto al equipo de desarrollo para acordar los detalles de los requisitos y así poder llevar a cabo su implementación, prueba y validación.

1.5 Arquitectura de software

La arquitectura de software es el conjunto de decisiones y principios que guían la construcción de un sistema de software. Es la estructura fundamental del sistema, que define cómo se organizan los componentes, cómo interactúan entre sí y cómo se comunican con el mundo exterior.

La arquitectura de software es importante porque tiene un impacto significativo en la calidad, el rendimiento y la mantenibilidad del sistema. Una buena arquitectura puede hacer que el sistema sea más fácil de entender, modificar y extender, mientras que una mala arquitectura puede hacer que el sistema sea difícil de mantener y evolucionar.

Hay varios principios y patrones que se utilizan en la arquitectura de software para lograr estos objetivos. Algunos de los principios más importantes son:

- Separación de preocupaciones: los diferentes componentes del sistema deben estar diseñados para realizar tareas específicas y separadas, lo que hace que el sistema sea más modular y fácil de entender.
- Abstracción: los componentes del sistema deben ser diseñados de manera abstracta, para que puedan ser reutilizados en diferentes contextos y no estén acoplados a una implementación específica.
- Encapsulamiento: los componentes del sistema deben estar encapsulados, lo que significa que su estado interno no debe ser visible desde el exterior, lo que hace que el sistema sea más seguro y fácil de mantener.
- Composición: los componentes del sistema deben ser diseñados para ser fácilmente combinables, lo que hace que el sistema sea más flexible y escalable.

Además, hay varios patrones arquitectónicos comunes que se utilizan para diseñar sistemas de software. Algunos de los patrones más comunes son:

- Arquitectura cliente-servidor: divide el sistema en dos partes: un cliente que solicita servicios y un servidor que los proporciona.

- Arquitectura basada en eventos: los componentes del sistema se comunican entre sí a través de eventos, lo que hace que el sistema sea más reactivo y escalable.
- Arquitectura basada en microservicios: divide el sistema en pequeños servicios independientes, lo que hace que el sistema sea más modular y fácil de mantener.

En resumen, la arquitectura de software es un conjunto de decisiones y principios que guían la construcción de un sistema de software. Una buena arquitectura puede hacer que el sistema sea más fácil de entender, modificar y extender, mientras que una mala arquitectura puede hacer que el sistema sea difícil de mantener y evolucionar. La arquitectura de software se basa en principios como la separación de preocupaciones, la abstracción, el encapsulamiento y la composición, y utiliza patrones arquitectónicos comunes como la arquitectura cliente-servidor, la arquitectura basada en eventos y la arquitectura basada en microservicios.

Estilos arquitectónicos

Estilo de Flujo de Datos: Es una arquitectura de computadoras que contrasta directamente con la tradicional Arquitectura de Von Neumann o de estructuras de control. A diferencia de estas arquitecturas, las arquitecturas de flujo de datos no se basan en un contador de programa, sino que la ejecución de las instrucciones solamente viene determinada por la disponibilidad de los argumentos de entrada. Aunque ningún computador de éxito comercial ha utilizado este tipo de arquitectura, esta es muy relevante en muchas arquitecturas actuales de software, incluyendo el diseño de sistemas de bases de datos o de sistemas de procesamiento paralelo.[7], [8]

Las arquitecturas de flujo de datos se centran en la transformación de los datos de entrada para obtener los datos de salida. Para lograr esto, se utilizan componentes del software como tuberías, que están interconectadas entre sí y trabajan en forma independiente.

Estilos Centrados en Datos: Es un estilo arquitectónico que se enfoca en la gestión de datos y su flujo a través de los componentes de un sistema. Los componentes se comunican entre sí mediante el intercambio de datos, y el estilo se centra en la organización y el flujo de los mismos. Este estilo se utiliza comúnmente en sistemas que manejan grandes cantidades de datos, como los sistemas de gestión de bases de datos.[9]

El estilo Centrado en Datos se caracteriza por tener una estructura en la que los componentes del sistema están diseñados para procesar datos, y no para realizar acciones o tomar decisiones. Este estilo también se enfoca en la reutilización de componentes, lo que permite una mayor flexibilidad y escalabilidad del sistema.

Estilos de Llamada y Retorno: Es un estilo arquitectónico que enfatiza la modificabilidad y la escalabilidad. Es uno de los estilos más utilizados en sistemas a gran escala. Algunos ejemplos de arquitecturas que siguen este estilo son el Modelo-Vista-Controlador (MVC), la Arquitectura Orientada a Objetos y la Arquitectura en Capas.[10]

Estilos heterogéneos: En el contexto de desarrollo de software, los estilos heterogéneos son un conjunto de patrones arquitectónicos que se utilizan para diseñar sistemas de software que constan de componentes que se ejecutan en diferentes plataformas y lenguajes de programación. Estos estilos permiten la integración de componentes heterogéneos en un sistema de software, lo que puede mejorar la flexibilidad, la escalabilidad y la interoperabilidad del sistema. Algunos ejemplos de estilos heterogéneos son SOA (Arquitectura Orientada a Servicios), EDA (Arquitectura Orientada a Eventos) y EDA (Arquitectura Orientada a Servicios y Eventos)[11]

Estilos Peer-to-Peer: Es una red de computadoras en la que todos los dispositivos conectados actúan como cliente y servidor al mismo tiempo. A diferencia de la arquitectura Cliente-Servidor, no se requiere un servidor central para administrar la red. En lugar de eso, todos los nodos de la red pueden comunicarse entre sí. La arquitectura P2P busca la descentralización y permite que cada computadora funcione como cliente y servidor simultáneamente.[12]

Existen diferentes variantes de la arquitectura P2P, pero todas comparten el objetivo de descentralizar la red y permitir que los nodos se comuniquen directamente entre sí. Algunas variantes pueden requerir un servidor central, pero en general, cada nodo tiene igualdad de privilegios y responsabilidades.

La arquitectura P2P tiene una alta complejidad para desarrollarse, ya que implica crear aplicaciones que funcionen como cliente y servidor al mismo tiempo. Sin embargo, ofrece ventajas como una mayor capacidad de escalabilidad y distribución de carga.

Es importante tener en cuenta que el estilo arquitectónico P2P no se limita al desarrollo de software, sino que también se utiliza en otros contextos, como modelos de negocio.

1.6 Tecnologías y herramientas para el desarrollo de la solución

Visual Studio es un entorno de desarrollo integrado (IDE) creado por Microsoft que permite desarrollar aplicaciones para distintas plataformas (incluyendo Windows, Android, iOS y Linux). Visual Studio ofrece herramientas para escribir, editar, depurar y compilar código en varios lenguajes como C#, C++, Visual Basic .NET, F#, Java, Python y más. Visual Studio también incluye compiladores, herramientas de finalización de código, diseñadores gráficos, control de versiones, extensiones y otras características para facilitar el proceso de desarrollo de software.[13]

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general, que se caracteriza por su sintaxis clara y simple, y su facilidad de aprendizaje. Python permite desarrollar aplicaciones para diversos ámbitos, como la ciencia, la web, el análisis de datos, la inteligencia artificial y más.[14], [15]

Django es un framework web de código abierto, escrito en Python, que sigue el patrón de diseño Modelo-Vista-Controlador (MVC). Django facilita el desarrollo rápido y limpio de aplicaciones web, proporcionando componentes reutilizables, una interfaz de administración, un sistema de plantillas, un mapeador objeto-relacional (ORM) y otras características.[16]

HTML (HyperText Markup Language) es el lenguaje de marcado estándar para crear páginas web y aplicaciones web. HTML define la estructura y el contenido de un documento web mediante etiquetas que indican cómo se debe mostrar el texto, las imágenes, los enlaces y otros elementos en un navegador.[17]

CSS (Cascading Style Sheets) es un lenguaje de diseño que se usa para describir la presentación de un documento HTML. CSS permite modificar el aspecto, el color, el tamaño, el fondo, la posición y otros atributos visuales de los elementos HTML. CSS también permite crear diseños adaptables (responsive) que se ajustan al tamaño y orientación de la pantalla del dispositivo.[13]

Bootstrap es un framework CSS y JavaScript que ofrece un conjunto de herramientas para crear interfaces web con un diseño limpio y responsive. Bootstrap incluye componentes predefinidos como botones, menús, formularios, tablas, carruseles y más, que se pueden personalizar fácilmente mediante variables y clases. Bootstrap también cuenta con una extensa documentación y una comunidad activa de desarrolladores.[18]

SQLite es un sistema de gestión de bases de datos relacionales, contenido en una pequeña biblioteca escrita en C, que se enlaza con el programa principal. SQLite es una alternativa ideal para el desarrollo de aplicaciones pequeñas que gestionen bases de datos relacionales. SQLite es un motor de base de datos SQL transaccional de código abierto, ligero, autónomo, de configuración simple y sin servidor. A diferencia de otros sistemas de gestión de bases de datos, SQLite no funciona como un servidor de bases de datos independiente, sino que se integra directamente en la aplicación que lo utiliza.[19]

Axure es una herramienta para crear prototipos interactivos y especificaciones funcionales para sitios web y aplicaciones. Axure permite diseñar interfaces con elementos visuales como botones, menús, formularios e imágenes, y añadir interacciones como transiciones, animaciones y eventos. Axure también permite generar documentación detallada del diseño y compartir los prototipos con otros usuarios para obtener comentarios.[20]

Visual Paradigm es una herramienta CASE que proporciona un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Visual Paradigm es compatible con UML 2, SysML y la notación de modelado de procesos empresariales del grupo de gestión de objetos. Además del soporte de modelado, proporciona capacidades de ingeniería de código y generación de informes.[21]

Enterprise Architect es una plataforma visual para diseñar y construir sistemas de software, para el modelado de procesos comerciales y para propósitos de modelado más generalizados. Enterprise Architect se basa en la última especificación UML 2.5. La herramienta cubre el desarrollo de software desde el paso de los requerimientos a través de las etapas del análisis, modelos de diseño, pruebas y mantenimiento. Proporciona una trazabilidad completa desde la fase inicial del diseño a través del despliegue y mantenimiento, y también provee soporte para pruebas, mantenimiento y control de cambio.[22]

DbVisualizer es una herramienta de gestión y análisis de bases de datos para todas las principales bases de datos (por ejemplo, Oracle, SQL Server, DB2, Sybase, MySQL, SQLite) en plataformas Windows, macOS, Linux y Unix. Proporciona una interfaz simple y potente que admite varios modelos de datos. También permite la exploración y el manejo de objetos, la ejecución de consultas SQL y la visualización de información con conexiones simultáneas. Esta herramienta todo en uno es una herramienta universal de bases de datos para desarrolladores, administradores de bases de datos y analistas.[23]

1.7 Conclusiones parciales

Después de realizar una revisión exhaustiva de la literatura y el análisis documental, se puede concluir que la implementación de una aplicación web para ventas puede ser una solución efectiva para mejorar la presencia en el mercado y aumentar las ventas de la Fábrica de Piensos Cienfuegos.

Se han identificado los principales desafíos que enfrentan las empresas en el mercado de ventas directas de pienso y cómo una aplicación web puede ayudar a superar estos desafíos. Además, se ha identificado que la aplicación web debe desarrollarse siguiendo los estándares de la industria y adaptarse a las necesidades y expectativas de la empresa.

Para lograr los objetivos de la tesis, se propone un enfoque metodológico que combina métodos teóricos y empíricos, como el análisis documental, la revisión de literatura, el análisis de casos, las entrevistas y encuestas, las pruebas de usabilidad, el análisis de ventas y el análisis de procesos.

CAPÍTULO 2: PROPUESTA DE SOLUCÍON

En este capítulo se explica la solución propuesta y los elementos creados para el cuarto escenario en la metodología Agile Iniciad Procesos (AUP), adaptada para la Fábrica de Piensos Cienfuegos, durante las fases de análisis, diseño e implementación.

2.1 Análisis del proceso de informatización

El proceso de ventas en la Fábrica de Piensos Cienfuegos es un procedimiento presencial que involucra la interacción entre un representante de una entidad y el comercial encargado de reservar las ventas en la empresa. Este proceso comienza con la verificación de la información del representante en la base de datos de la empresa, lo que garantiza la autenticidad de su identidad. En caso de que el representante pertenezca a una entidad no registrada previamente en los informes de la empresa, se agregan los datos necesarios de la entidad junto con los datos del representante. Luego, se reserva el pedido de venta del tipo de pienso que desee acorde con la disponibilidad de la empresa, y se le entrega un ticket para su posterior carga. Durante el proceso de carga, se corrobora el papel de la venta reservada por parte de la empresa y se hace la carga de la mercancía, que posteriormente pasa a control de peso y cantidad. Finalmente, se liquida la venta en los papeles de la empresa y se da por realizada la venta. Todo este proceso se realiza manualmente mediante papeles para registrar todos los datos e información necesaria para realizar una venta.

La figura que se presenta a continuación ilustra lo antes descrito:

Figura 2.1 Proceso de ventas en la Fábrica de Piensos Cienfuegos

Ante este panorama, se hace evidente la necesidad de una solución más eficiente y auto matizada que proporcione a la empresa una mayor agilidad en la gestión de las ventas, tanto en términos de procesos como de registro y seguimiento de datos. La propuesta de una aplicación web, tiene el potencial de abordar estos desafíos al simplificar el proceso de ventas, automatizando la generación de documentos y brindando una interfaz intuitiva para el seguimiento de las transacciones. Esta solución no solo

mejoraría la eficiencia en la gestión de las ventas, sino que también ofrecería mayor escalabilidad, accesibilidad y facilidad de seguimiento y análisis de los datos de ventas.

Se ha diseñado una aplicación web aplicación web que presenta diversas funciones que permiten informatizar el proceso de ventas y mejorar la gestión de la empresa de manera eficiente.

En primer lugar, la aplicación web ofrecerá la capacidad de gestionar entidades, lo cual implica agregar, eliminar y actualizar información relacionada con las entidades en una base de datos centralizada. De manera similar, también se puede administrar la información de los representantes de cada entidad, permitiendo añadir, eliminar y actualizar los datos correspondientes. Además, la aplicación permite gestionar los tipos de pienso en venta, brindando la opción de agregar, eliminar y actualizar información sobre estos productos, asegurando así una gestión adecuada de los productos ofrecidos por la empresa. En términos de la reserva de ventas, la aplicación web facilita el proceso al permitir a las entidades completar todos los requisitos necesarios para realizar una reserva de manera eficiente y confiable. Esto incluye la recolección y verificación de la información relevante, así como todos los datos necesarios para completar la reserva con éxito. La aplicación también ofrece herramientas estadísticas que brindan información valiosa para mejorar las ventas y la toma de decisiones en la empresa. Estas herramientas permiten mostrar las ganancias acumuladas, el número de ventas realizadas, las entidades en negocio, el mejor cliente y el tipo de pienso más vendido. Además, se presentará esta información en gráficos representativos, que facilitarán la comprensión y el análisis de los datos para una mejor administración y seguimiento de la información.

En resumen, la aplicación web propuesta se enfoca en digitalizar y optimizar los procesos de ventas en la Fábrica de Piensos Cienfuegos, ofreciendo diversas funcionalidades para la gestión de entidades y representantes, la administración de los productos en venta, la reserva de ventas y la generación de herramientas estadísticas para mejorar las ventas y la toma de decisiones. Esta solución informatizada tiene como objetivo eliminar ambigüedades y errores asociados al registro manual en papel, permitiendo una mayor eficiencia, precisión y administración integral de la información de la empresa.

2.2 Requisitos de software

Requisitos Funcionales

Los requisitos funcionales son las capacidades que debe tener un producto para cumplir con su función. En este estudio, se obtuvieron 24 requisitos funcionales a través de entrevistas y reuniones con el cliente, todos estos se detallan en la tabla que se presenta a continuación:

Número	Nombre	Descripción	Prioridad	Complejidad
RF1	Autenticar Usuario	El usuario introduce sus datos, el sistema lo comprueba y si son válidos, el usuario queda autenticado; si los datos no son válidos el sistema muestra un mensaje de error.	Alta	Media
RF2	Mostrar Datos de la Empresa	Muestra la información en la app web relacionada con la empresa.	Media	Baja
RF3	Editar Datos de la Empresa	Permite editar la información en la app web relacionada con la empresa.	Baja	Baja
RF4	Añadir Entidad	Añade una nueva entidad (cliente).	Alta	Media
RF5	Eliminar Entidad	Elimina una entidad (cliente) existente.	Alta	Media
RF6	Actualizar Entidad	Actualiza los datos de una entidad (cliente) existente.	Alta	Media
RF7	Añadir Representante	Añade un representante.	Alta	Media
RF8	Eliminar Representante	Elimina un representante.	Alta	Media
RF9	Actualizar Representante	Actualiza los datos relacionados a un representante.	Alta	Media

RF10	Añadir Pienso	Añade a disponibilidad un tipo de pienso a la venta.	Alta	Media
RF11	Eliminar Pienso	Elimina de la disponibilidad un tipo de pienso de la venta.	Alta	Media
RF12	Reservar Venta	Gestiona una venta entre el representante de una entidad y la empresa, esta posee toda la información necesaria de acorde a las políticas de venta de la empresa.	Alta	Alta
RF13	Mostrar Reporte Semanal	Permite mostrar información sobre las ganancias acumuladas de la empresa mediante las ventas de la última semana.	Media	Media
RF14	Generar Informe Semanal	Genera un informe sobre la información sobre las ganancias acumuladas de la empresa mediante las ventas de la última semana.	Baja	Baja
RF15	Mostrar Reporte Mensual	Permite mostrar información sobre las ganancias acumuladas de la empresa mediante las ventas del último mes.	Media	Media
RF16	Generar Informe Mensual	Genera un informe sobre la información sobre las ganancias acumuladas de	Baja	Baja

		la empresa mediante las ventas del último mes.		
RF17	Mostrar Reporte Anual	Permite mostrar información sobre las ganancias acumuladas de la empresa mediante las ventas del último año.	Media	Media
RF18	Generar Informe Anual	Genera un informe sobre la información sobre las ganancias acumuladas de la empresa mediante las ventas del último año.	Baja	Baja
RF19	Mostrar Número de Ventas	Permite mostrar información del número de ventas realizadas.	Media	Media
RF20	Mostrar Entidades en Negocio	Permite mostrar información sobre la cantidad de entidades (clientes) en negocio con la empresa.	Media	Media
RF21	Mostrar Mejor Cliente	Permite mostrar información sobre la entidad (cliente) con mayores ventas, además del representante de la entidad.	Media	Media
RF22	Mostrar Pienso más Vendido	Permite mostrar información sobre la cantidad del tipo de pienso más vendido en la empresa.	Media	Media

RF23	Mostrar Ganancias Recaudadas	Permite mostrar información sobre el total de ganancias recaudadas de la empresa.	Media	Media
RF24	Cerrar Sesión	Cierra la sesión.	Baja	Baja

Tabla 2.1 Requisitos Funcionales

Requisitos No Funcionales

Los requisitos no funcionales se enfocan en las características que no están directamente relacionadas con la funcionalidad de la aplicación, sino que se centran en aspectos como la eficiencia, la velocidad y la confiabilidad. Se establecieron 8 requisitos no funcionales, todos estos se detallan en la tabla que se presenta a continuación:

Número	Descripción	Tipo de Requisito
RNF1	La interfaz es fácil de comprender, se guía por la lógica y es clara, con un diseño agradable y se mantiene uniforme en vistas similares.	Requisito de interfaz o apariencia externa
RNF2	El acceso a la aplicación web estará restringido al personal seleccionado dentro de la empresa.	Requisito de Usabilidad
RNF3	La aplicación tiene una respuesta rápida a las solicitudes de información.	Requisito de Rendimiento
RNF4	La aplicación debe funcionar correctamente en sistemas operativos como Windows, MacOS y Linux, y en navegadores populares como Google Chrome, Mozilla Firefox, Internet Explorer y Safari.	Requisito de Portabilidad
RNF5	Incluye la implementación de protocolos de seguridad como HTTPS, autenticación de	Requisito de Seguridad

	usuarios, encriptación de datos, control de acceso.	
RNF6	El desarrollo de la aplicación debe estar en correspondencia con la cultura organizacional de la empresa.	Requisito Políticos y Culturales
RNF7	La aplicación propuesta cumple con las regulaciones y normas indicadas oficialmente en la Fábrica de Pienso Cienfuegos.	Requisito Legales
RNF8	- Sistema operativo: Windows 7 o superior, Linux Ubuntu 16.04 o superior. - Navegador: Google Chrome versión 50 o superior, Mozilla Firefox versión 45 o superior, Microsoft Edge versión 25 o superior. - Procesador: Intel Pentium o superior, AMD A4 o superior. - Memoria RAM: 2 GB o superior. - Tarjeta gráfica: Intel HD Graphics o superior.	Requisito de Software y Hardware

Tabla 2.2 Requisitos No Funcionales

2.3 Historial de usuario

Las historias de usuarios son herramientas que se emplean para describir los requerimientos del software, y son similares a los casos de uso que se utilizan en el proceso unificado. Estas historias son la base para llevar a cabo las pruebas funcionales. Se han definido 21 historias de usuarios, entre las cuales se encuentra "Reservar Venta":

Número: 10	**Requisito:** Reservar Venta
Programador: Krystian Enrique Martínez Yanes	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo Estimado:** 14 horas
Riesgo en Desarrollo: Medio	**Tiempo Real:** 14 horas

Descripción: Gestiona una venta entre el representante de una entidad y la empresa, esta posee toda la información necesaria de acorde a las políticas de venta de la empresa.

Campos:

Tipo de pienso a vender: Especifica el tipo de pienso a vender

Cantidad de toneladas: Especifica la cantidad de toneladas del tipo de pienso a vender

Empresa Vendedora: Especifica la empresa a la que se le va a vender

Comprador: Especifica el representante de la empresa a la que se le va a vender

Botones:

Guardar: Guarda la información y reserva la venta

Observaciones: N/A.

Prototipo elemental de interfaz gráfica de usuario:

Tabla 2.3 Historial de Usuario "Reservar Venta"

2.4 Descripción de la arquitectura

El diseño del software se basa en una arquitectura que establece las ideas principales y proporciona una descripción detallada de cómo construir el sistema.[24]

Patrón de arquitectura Modelo Vista Template

El patrón de arquitectura Modelo Vista Template (MTV) es una variante del patrón Modelo Vista Controlador (MVC)[25]. En este patrón, los componentes se dividen en tres categorías:

Modelo (Model): Es el backend que contiene toda la lógica de datos.

Plantilla (Template): Es el frontend o interfaz gráfica de usuario (GUI).

Vista (View): Es el cerebro de la aplicación que controla cómo se muestran los datos.

Este patrón permite a varios desarrolladores trabajar simultáneamente en la aplicación y facilita la administración y modificación del código frontend y backend en componentes separados. Esto es especialmente útil cuando varios desarrolladores necesitan actualizar, modificar o depurar una aplicación completada simultáneamente.

El patrón MTV es muy útil para el desarrollo de aplicaciones web modernas porque permite que la aplicación sea escalable, mantenible y fácil de expandir.

La figura que se presenta a continuación ilustra la manera en que los tres elementos de la arquitectura definida interactúan entre sí:

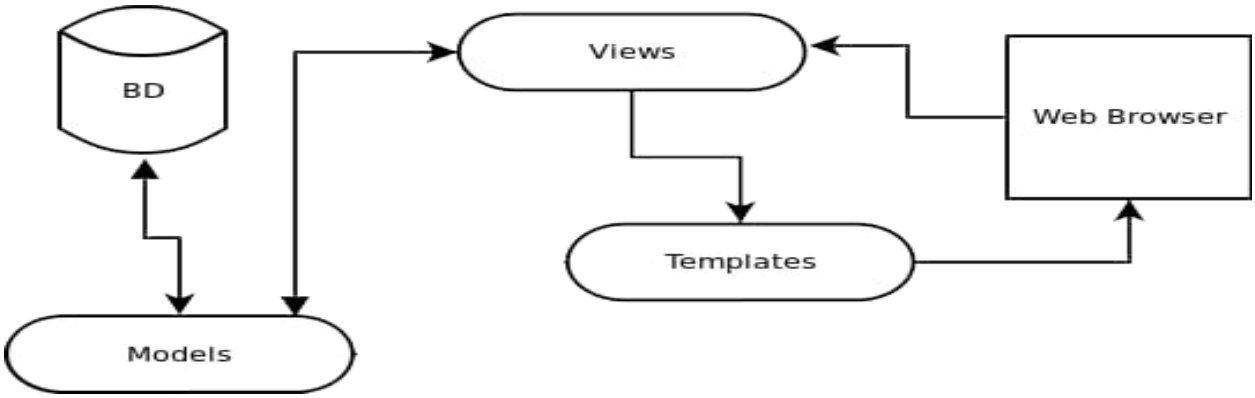

Figura 2.2 Patrón de Arquitectura Modelo Vista Template

La figura que se presenta a continuación ilustra la aplicación en paquetes de la arquitectura a la propuesta de solución:

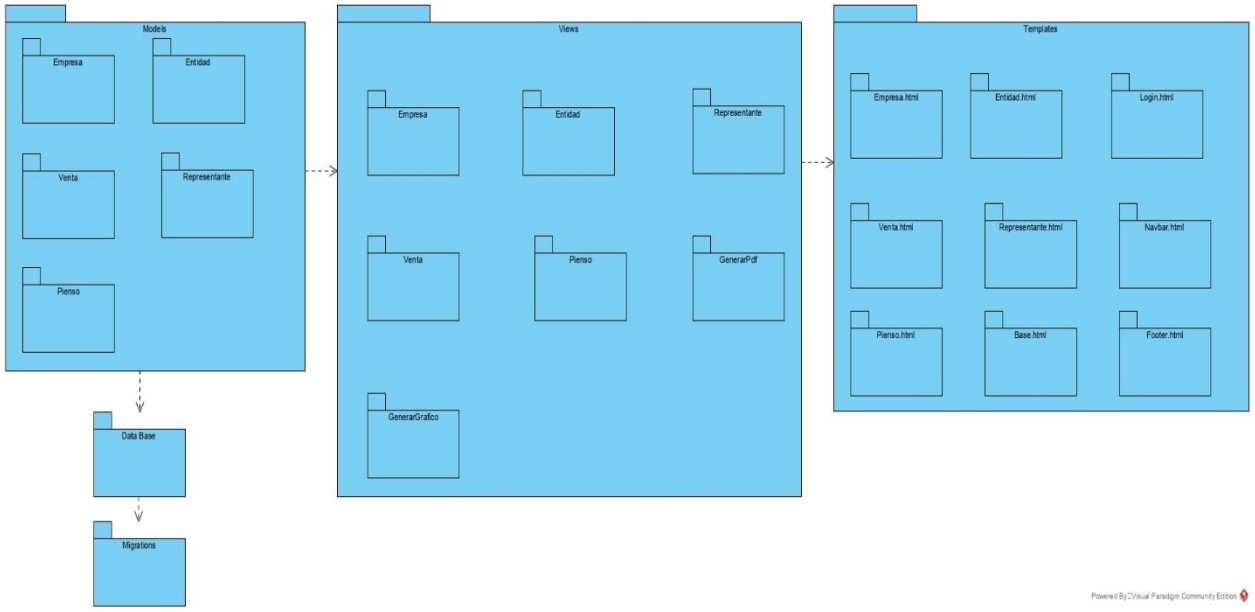

Figura 2.3 Arquitectura a la Propuesta de Solución

La figura que se presenta a continuación ilustra la aplicación en paquetes de la arquitectura relacional a la propuesta de solución:

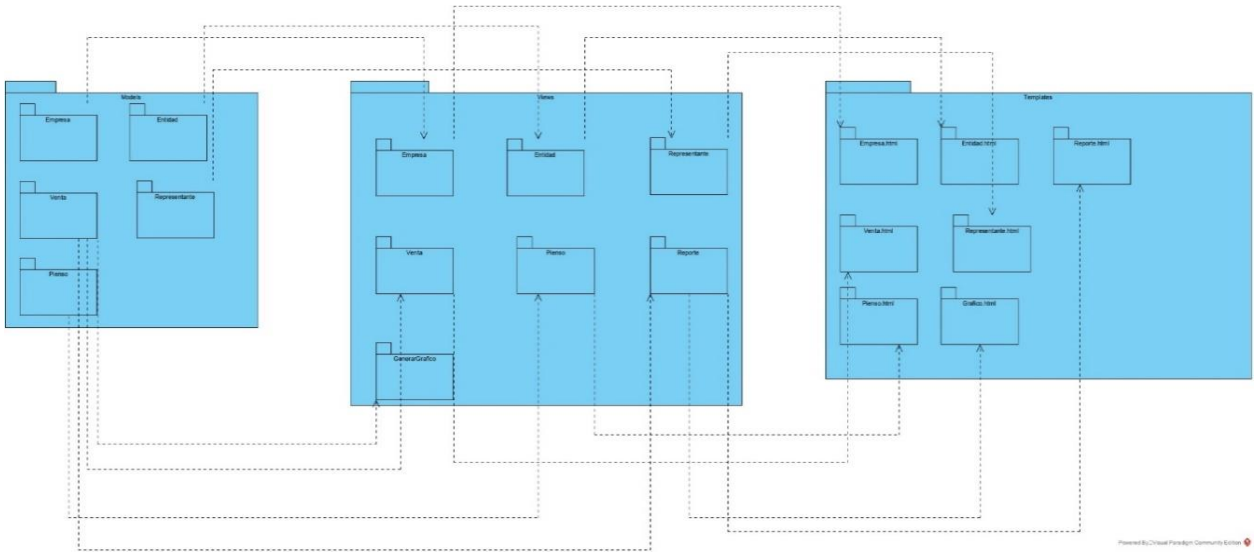

Figura 2.4 Arquitectura relacional a la Propuesta de Solución

2.5 Patrones de diseño utilizados

GASP es un acrónimo de General Responsibility Assignment Software Patterns. Estos patrones son una serie de pautas generales para asignar responsabilidades en el diseño orientado a objetos. Proporcionan una guía para asignar responsabilidades a las clases y objetos en un sistema de software.[26]

Experto en información: Este patrón asigna la responsabilidad a la clase que tiene la información necesaria para cumplir con una solicitud. Al asignar esta responsabilidad a una clase específica, se mantiene el encapsulamiento y se evita la dispersión de información. Este patrón se refleja en mi tesis en la forma en que el sistema asigna responsabilidades, asegurándose de que las clases u objetos que poseen la información necesaria para llevar a cabo ciertas tareas (como la generación de informes estadísticos) sean las encargadas de realizar esas tareas. Esto ayuda a promover un diseño más coherente y eficiente.

Alta cohesión y Bajo acoplamiento: La Alta cohesión agrupa las responsabilidades relacionadas en una sola clase. Al tener una alta cohesión, las clases se vuelven más fáciles de entender y mantener. Mientras que el Bajo acoplamiento reduce las dependencias entre clases y objetos. Al tener un bajo acoplamiento, los cambios en una clase tienen menos impacto en otras clases. En el contexto de mi tesis se busca lograr alta cohesión al agrupar funcionalidades relacionadas de manera lógica y consistente dentro del sistema de gestión de ventas. Al mismo tiempo, se busca lograr bajo acoplamiento, asegurándose de que los distintos módulos o componentes del sistema estén lo menos acoplados posible, lo que promueve la flexibilidad y la facilidad de mantenimiento a largo plazo.

Creador: Este patrón asigna la responsabilidad de crear instancias de objetos a una clase específica. Al asignar esta responsabilidad a una clase, se mantiene un bajo acoplamiento y se facilita la reutilización de código. Este patrón se refleja en mi tesis en la forma en que el sistema maneja la creación y la gestión de objetos relacionados con las ventas. Por ejemplo, el patrón Creador se manifiesta en la forma en que se implementan las clases de objetos Entidades, Representantes, ventas, etc., asegurando que la creación de estos objetos esté desacoplada de su utilización dentro del sistema.

Controlador: Este patrón asigna la responsabilidad de manejar eventos y coordinar acciones a una clase específica. Al asignar esta responsabilidad a una clase, se separa la lógica de negocios de la capa de presentación, lo que facilita la reutilización de código y proporciona un mayor control. Este patrón se refleja en el contexto de mi tesis en la forma en que se separan claramente las responsabilidades entre la presentación de datos al usuario (a través de la interfaz de usuario) y la lógica de gestión de las ventas y generación de informes (a través de componentes de control o capas de lógica de negocio).

MVT: Este patrón es muy útil porque separa claramente las distintas capas de la aplicación, lo que facilita su mantenimiento y escalabilidad. En el modelo vista template, el modelo representa la base de datos y la lógica de negocio de la aplicación, la vista es la interfaz de usuario y el template es la estructura que define cómo se mostrará la información en la vista. Esta separación permite que los desarrolladores trabajen en cada capa de forma independiente, lo que facilita la colaboración y reduce el riesgo de errores. Además, permite que se realicen cambios en una capa sin afectar a las demás, lo que hace que las actualizaciones sean más sencillas y menos costosas. Otra ventaja del MVT es que facilita la reutilización de código. Al separar las distintas capas, se pueden utilizar los mismos modelos y templates en distintas vistas, lo que reduce el tiempo y el esfuerzo necesarios para desarrollar nuevas funcionalidades.

GoF (Gang of Four) son un conjunto de patrones de diseño de software creados por cuatro autores: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides. Estos patrones se describen en el libro "Design Patterns: Elements of Reusable Object-Oriented Software" y se han convertido en una referencia importante para los desarrolladores de software. Los patrones GoF son soluciones probadas y comprobadas para problemas comunes en el diseño de software. Estos patrones se dividen en tres categorías: patrones creacionales, patrones estructurales y patrones de comportamiento. Los patrones creacionales se utilizan para crear objetos y clases de forma eficiente y flexible. Los patrones estructurales se utilizan para organizar las relaciones entre objetos y clases, y los patrones de comportamiento se utilizan para definir la interacción entre objetos y clases.[27], [28]

Estrategia: Con el uso del patrón Strategy, puedes encapsular cada estrategia en una clase separada y cambiarlas dinámicamente según los requisitos del negocio. El patrón Estrategia se refleja en mi tesis en la forma en que el sistema gestiona diferentes estrategias o algoritmos para el procesamiento de ventas, generación de informes u otras tareas relacionadas.

Observador: Este patrón sería útil para mantener actualizados a los diferentes componentes de la aplicación web sobre los cambios en las ventas, entidades o cualquier otro evento relevante. Al utilizar el patrón Observer, puedes establecer una relación de suscriptor-observador para notificar y actualizar de manera eficiente los diferentes elementos de la aplicación cuando ocurran cambios. Este patrón se utiliza para implementar la actualización en tiempo real de los diferentes componentes del sistema cuando se produzcan cambios en los datos de ventas.

Método de Fábrica: Este patrón puede ser útil al crear instancias de diferentes tipos de productos dentro de tu aplicación, lo que te brindaría flexibilidad y facilidad de mantenimiento. El patrón Método de Fábrica se refleja en el contexto de mi tesis en la forma en que el sistema gestiona la creación de diferentes tipos de objetos relacionados con las ventas, como por ejemplo, la creación reportes, o elementos de la interfaz de usuario. Esto podría proporcionar una mayor flexibilidad en la creación de objetos sin comprometer la coherencia del sistema.

Patrón Método de Fábrica Abstracta: Este patrón proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar sus clases concretas. En mi tesis, se aplica este patrón para proporcionar una manera de crear no solo objetos individuales, como en el patrón Método de Fábrica, sino también para crear familias enteras de objetos relacionados, como por ejemplo, diferentes tipos de productos, ventas o informes, que pueden estar interrelacionados en el contexto de la gestión de ventas.

2.6 Diagrama de Clases

El diagrama de clases es una representación visual estática de la estructura y las relaciones entre las clases en un sistema de software. Es una parte integral del modelado orientado a objetos y es ampliamente utilizado en proyectos de ingeniería informática y desarrollo de software.

La figura que se presenta a continuación ilustra el diseño del modelo de datos:

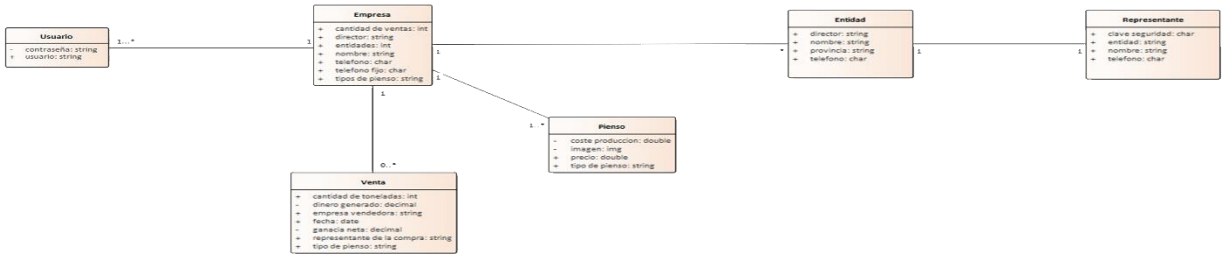

Figura 2.5 Diagrama de Clases

2.7 Diagrama del Modelo de la Base de Datos

El diagrama de modelo de la base de datos es una representación visual que muestra la estructura y las relaciones entre los datos de una base de datos. Este tipo de diagrama proporciona una vista

esquemática de las tablas, campos y relaciones dentro de la base de datos, lo que facilita la comprensión de cómo se organizan y almacenan los datos. Además, el diagrama de modelo de la base de datos ayuda a visualizar la arquitectura general de la base de datos, incluyendo las tablas y sus atributos, las relaciones entre las tablas, las claves primarias y foráneas, así como las restricciones de integridad. En resumen, el diagrama de modelo de la base de datos es una herramienta crucial para comprender la estructura y el diseño de una base de datos, lo que a su vez facilita su implementación, mantenimiento y optimización.

La figura que se presenta a continuación ilustra el modelo de la base de datos:

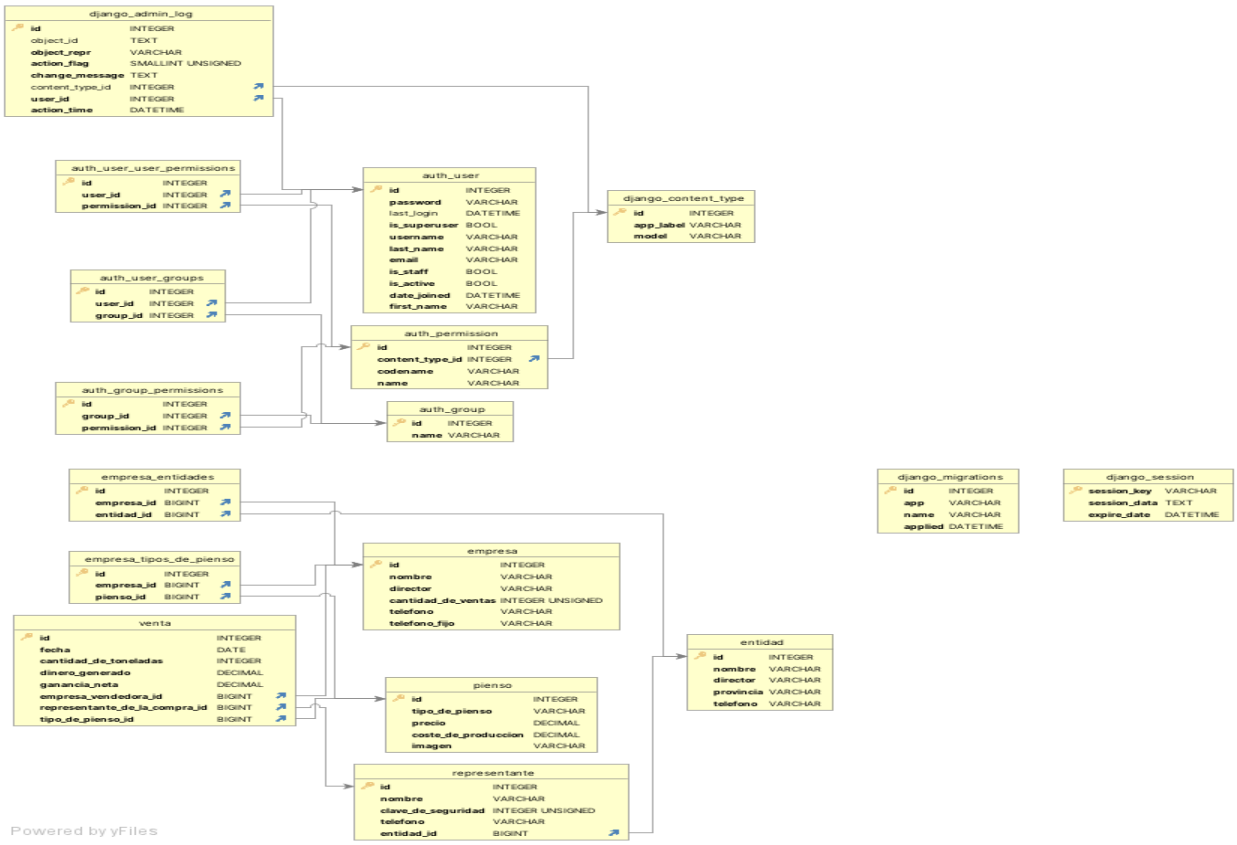

Figura 2.6 Diagrama del Modelo de la Base de Datos

2.8 Estándar de codificación empleado

Para lograr una uniformidad en la programación del sistema, se ha optado por implementar el uso del estándar de codificación CamelCase, se usa a menudo como una convención de nomenclatura en la programación, especialmente en lenguajes como Java, JavaScript, C#, PHP, Python y otros. Dependiendo del lenguaje y del tipo de elemento que se nombra (variable, función, clase, etc.), se puede usar el lowerCamelCase (primera letra en minúscula) o el UpperCamelCase (primera letra en mayúscula), el cual permitirá una lectura, comprensión y mantenimiento del código más sencillos[29].
A continuación, se detallan de manera general las convenciones de nomenclatura utilizadas:

Convenciones de nomenclatura:

General:

En todo momento se utilizarán nombres que sean claros, concretos y libres de ambigüedades.

El nombre de todas las variables y métodos comenzarán con letra minúscula y si este está compuesto por varias palabras, todas las palabras internas que lo componen comienzan con mayúscula.

Identación:

El contenido siempre se denomina con tb para las tablas y nm para los nomencladores, nunca utilizando espacios en blanco.

Clases:

El nombre de las clases comenzará con mayúsculas, todas las palabras internas que lo componen comienzan con mayúscula.

Intentar mantener los nombres de las clases descriptivos y simples. Usar palabras completas, evitar acrónimos y abreviaturas, a no ser que la abreviatura sea mucho más conocida que el nombre completo, como URL o HTML.

Nombre de variables:

No se utilizarán nombres de variables que puedan ser ambiguos.

Se procurará evitar dar nombres sin sentido a variables temporales.

2.9 Conclusiones parciales

La solución propuesta en este capítulo a través de la aplicación web desarrollada ofrece una serie de mejoras significativas para la Fábrica de Piensos Cienfuegos. La implementación de esta solución permitirá agilizar el proceso de ventas, simplificar la gestión de entidades y representantes, gestionar las ventas y proporcionar información estadística relevante para la toma de decisiones. Esta solución garantiza un enfoque centrado en el usuario, siguiendo los estándares de la industria y mejorando la eficiencia y precisión de la empresa en la gestión de la información. La aplicación web establece una base sólida para mejorar la satisfacción del cliente. Las características incorporadas en la solución, como la reserva de ventas, la administración de entidades y la visualización de estadísticas, brindarán una gestión más eficiente y una comprensión más profunda del rendimiento del negocio.

CAPÍTULO 3: VALIDACIÓN

En este capítulo se examina la posibilidad de implementar la solución propuesta mediante la comprobación de los requisitos y el diseño propuesto.

3.1 Técnicas de validación de requisitos

Se realizó una validación de los requisitos del software con el fin de asegurarse de que éstos definan el sistema que el cliente desea. Se utilizaron las siguientes técnicas para llevar a cabo la validación de los requisitos del software:

Revisiones de los requisitos: El equipo de desarrollo y el cliente revisaron cuidadosamente cada uno de los requisitos. Durante las revisiones internas, se identificaron algunas no conformidades técnicas y ortográficas, las cuales se corrigieron oportunamente. Con el cliente, se llevaron a cabo dos revisiones. En la primera, el cliente quedó satisfecho con el trabajo realizado por el equipo de desarrollo, pero se hicieron algunos ajustes a algunos requisitos funcionales para mejorarlo aún más. En la segunda reunión, se agregaron detalles a los requisitos funcionales y finalmente se aprobaron todos los requisitos.

Diseño de prototipos web: Se presentaron al cliente prototipos del sistema que permitieron una vista previa del mismo en funcionamiento. A través de la interacción con los prototipos, se evaluó si cumplían con las necesidades del cliente y fueron aprobados por él.

Generación de casos de prueba: Para validar los requisitos funcionales del sistema, se crearon casos de pruebas específicas para cada historia de usuario. Además, se presentaron al cliente prototipos del sistema para que pudiera interactuar con ellos y evaluar si cumplían con sus necesidades, lo cual fue aprobado por el cliente.

3.2 Pruebas de Software

Los niveles de prueba son conjuntos de actividades que se planifican y ejecutan en conjunto durante el proceso de pruebas con el fin de comprobar y validar los diferentes componentes de un producto. Estos niveles permiten la selección de diversas técnicas y tipos de pruebas para llevar a cabo en cada etapa.[30]

Prueba de unidad: La prueba de unidad se enfoca en evaluar las unidades de software que conforman el producto. Por lo general, se aplica a componentes como el código fuente, archivos binarios y de datos, entre otros, con el fin de asegurarse de que los flujos de control, funcionales y de datos del

elemento de software sean cubiertos y funcionen adecuadamente. Su propósito es verificar que estas unidades operen de acuerdo a lo esperado.

Prueba de integración: La prueba de integración tiene como objetivo verificar que los componentes de software y hardware del producto funcionen correctamente al interactuar entre sí para llevar a cabo una transacción de negocio o caso de uso. Esta evaluación identifica posibles errores en las especificaciones de la interfaz de cada paquete de software que se está integrando con los demás, lo que permite detectar fallos y asegurar el correcto funcionamiento del sistema en su conjunto.

Prueba de sistema: La prueba de sistema se lleva a cabo para comprobar si un producto de software cumple con los requisitos previamente establecidos. Normalmente, esta evaluación se realiza al finalizar el desarrollo del producto. Sin embargo, un enfoque iterativo permite realizar pruebas más tempranas, evaluando subconjuntos de requisitos funcionales bien definidos. De esta manera, se pueden detectar posibles fallos en el sistema de forma temprana y asegurar que el producto final cumpla con los requerimientos establecidos.

Prueba de aceptación: La prueba de aceptación de usuario es la última evaluación que se realiza antes de implementar el software en los entornos de operación. Su finalidad es comprobar que el software está preparado para su uso, que cumple con los criterios de aceptación establecidos y que satisface las necesidades y expectativas de los clientes para los que fue desarrollado. De esta forma, se asegura que el software está listo para ser utilizado y que cumple con los requerimientos del usuario final.

Existen diversas estrategias de prueba de software que han sido propuestas en diferentes publicaciones. Todas estas estrategias ofrecen una plantilla que sirve como guía para los profesionales y un conjunto de hitos para los jefes de proyecto, con el objetivo de llevar a cabo las pruebas de manera correcta. Estas pruebas deben incluir tanto pruebas de bajo nivel, que verifiquen la correcta implementación de pequeñas partes del código, como pruebas de alto nivel, que validen que las funcionalidades del sistema se correspondan con lo especificado por el cliente. Para el presente estudio, se sugiere seguir la siguiente estrategia de pruebas:

Tipo de Prueba	Método de Prueba	Técnica de Prueba
Unitarias	Caja Blanca	Camino Básico
Funcionalidad	Caja Negra	Partición de Equivalencia

Tabla 3.1 Estrategia de Prueba Aplicada a la Solución

Pruebas de Caja Blanca

Las pruebas de caja blanca son una técnica de monitorización o prueba de software en la que se analiza el diseño, código y estructura interna de un sistema, con el objetivo de mejorar propiedades como la seguridad y el uso eficiente del sistema. Estas pruebas se caracterizan principalmente porque son los propios sistemas y aplicaciones quienes exponen sus métricas para que el usuario pueda leerlas, analizarlas y tomar decisiones y acciones en función de la obtención de un resultado u otro. Las pruebas de caja blanca se basan en los detalles referentes al código fuente, es decir, que se enfocan en su análisis y pueden llevarse a cabo a nivel de integración o unidad del sistema para el desarrollo de software.[31]

Se llevaron a cabo pruebas para asegurar que las funciones internas del sistema se ejecuten correctamente, centrándose en las principales funcionalidades. Se utilizó la técnica de prueba del camino básico, que implica garantizar que cada sentencia del programa sea probada al menos una vez. Para lograr esto, se midió la complejidad del procedimiento o algoritmo y se obtuvo un conjunto básico de caminos de ejecución que se utilizaron para generar casos de prueba.[32]

Se describirá a continuación el procedimiento de pruebas llevado a cabo en la funcionalidad empresa(request), debido a su importancia crítica en el sistema. En primer lugar, se examinó el código y se hizo una lista de las instrucciones:

```
def empresa(request):
    try:
        empresa = Empresa.objects.get(pk=1)  #1
        ventas = Venta.objects.all()  #2
        ganancia = 0  #3
        ganancia_neta = 0  #4
        for venta in ventas:  #5
            ganancia += venta.dinero_generado  #6
            ganancia_neta += venta.ganancia_neta  #7
    except Empresa.DoesNotExist:  #8
        empresa = None  #9
        ganancia = 0  #10
        ganancia_neta = 0  #11
    return {'empresa': empresa, 'ganancia': ganancia, 'ganacia_neta':
ganancia_neta}  #12
```

Figura 3.1 Procedimiento de pruebas a la funcionalidad empresa(request)

Para generar los casos de prueba basados en la técnica elegida, es necesario crear un diagrama de flujo que represente el código de la función, tal como se ilustra en la figura:

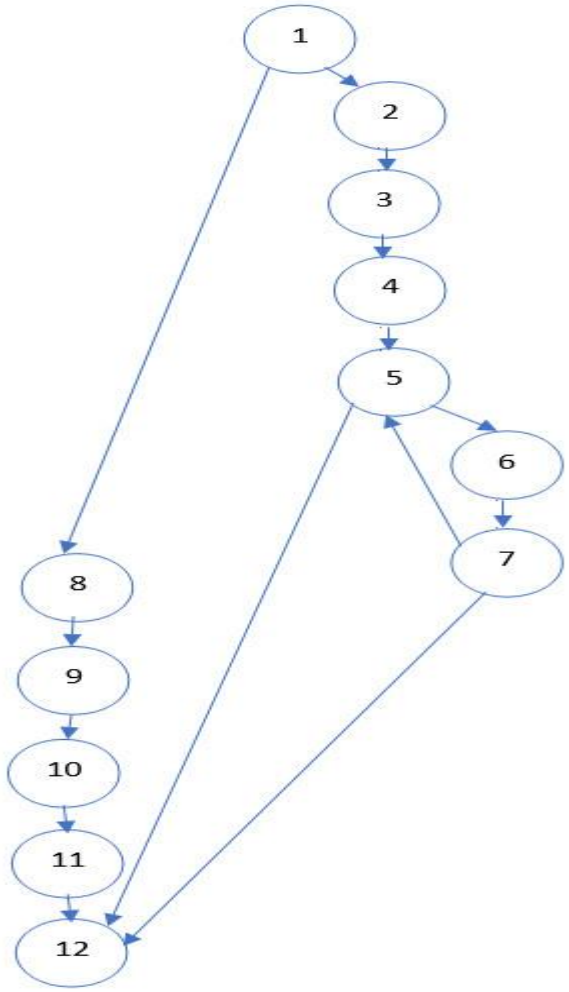

Figura 3.2 Diagrama de Flujo a la Función empresa(request)

Después de esto, se debe calcular la complejidad ciclomática V (G) del grafo generado, que representa el número de caminos independientes presentes en el código. Esto implica cualquier camino dentro del código que introduzca al menos un nuevo conjunto de sentencias de proceso o una nueva condición. Hay tres métodos para calcular la complejidad ciclomática:

1. V (G) = a - n + 2, siendo a el número de arcos o aristas del grafo y n el número de nodos.
2. V (G) = r, siendo r el número de regiones cerradas del grafo.
3. V (G) = c + 1, siendo c el número de nodos de condición.

Al hacer los cálculos necesarios, se llega al mismo resultado utilizando cualquiera de las opciones disponibles:

1. V(G)= a – n + 2
   V(G)= 14-12+2
   V(G)=4

2. V(G)= r
   V(G)= 4

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

Entonces, la agrupación de caminos básicos estaría compuesta por:

Camino básico 1: 1-8-9-10-11-12

Camino básico 2: 1-2-3-4-5-6-7-12

Camino básico 3: 1-2-3-4-5-12

Camino básico 4: 1-2-3-4-5-6-7-5-12

Pruebas de Caja Negra

Las pruebas de caja negra son una técnica de análisis de la funcionalidad de un sistema que no tiene en cuenta la estructura interna del código. Estas pruebas se enfocan en la entrada y salida de un software, tomando como base sus especificaciones y requisitos.[33]

En otras palabras, las pruebas de caja negra buscan verificar las funcionalidades del software o aplicación analizada sin tomar como referente la estructura del código interno, las rutas de tipo internas ni la información referente a la implementación. Las pruebas de caja negra tienen una gran importancia en la verificación del estado del sistema. Algunas de sus características principales son el uso de herramientas y plataformas externas que le permitan realizar una verificación del funcionamiento para comprobar que todo marche de manera correcta. Además, ofrecen un nivel de abstracción del código y se dedican a realizar las comprobaciones del comportamiento del sistema establecido, lo que contribuye a tener una mejor comunicación entre los módulos o subsistemas del software.[33]

Existen diversos métodos para llevar a cabo las pruebas de caja negra, siendo algunos de ellos los siguientes:

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

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

Entre los métodos utilizados para realizar pruebas de caja negra, destaca la técnica de Partición de Equivalencia debido a su eficacia en la detección de valores válidos e inválidos de las entradas del software. Esta técnica permite identificar rápidamente errores genéricos que, de otra manera, requerirían la ejecución de numerosos casos para ser detectados. La partición equivalente se enfoca en la definición de casos de prueba que permitan descubrir clases de errores.

Se llevaron a cabo tres iteraciones de pruebas de caja negra para verificar que las funcionalidades del sistema se ejecutan correctamente y satisfacen las necesidades del cliente. Los resultados obtenidos se presentan a continuación:

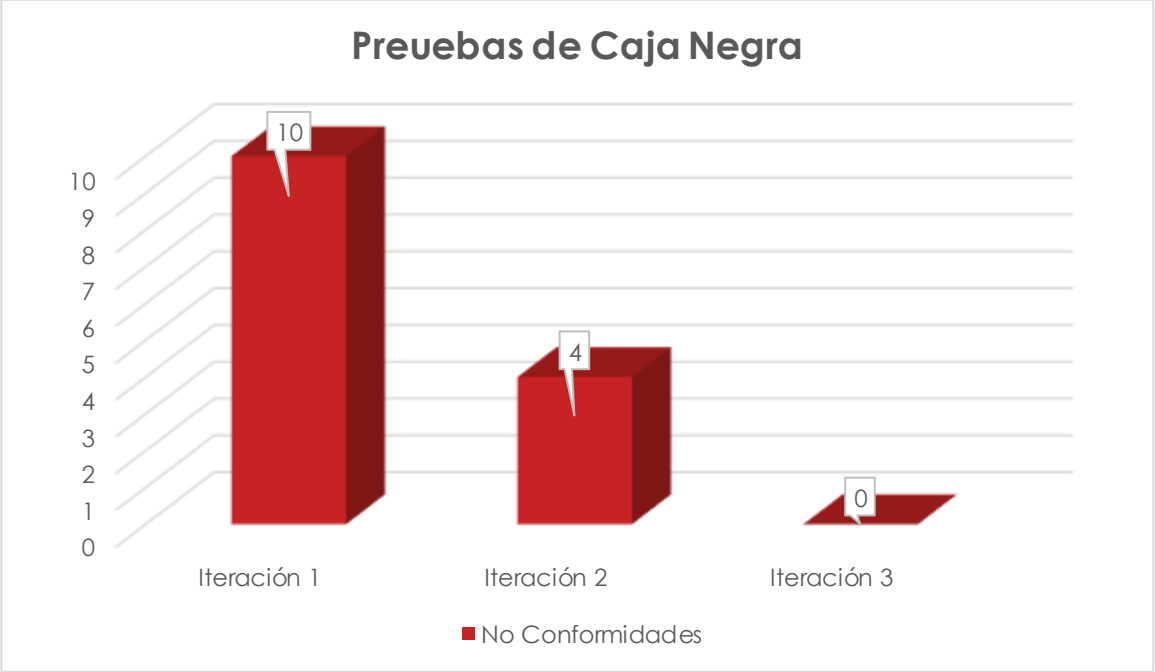

Gráfico 3.1 Pruebas de Caja Negra

En la primera iteración, se identificaron un total de 10 problemas de no conformidad (NC), de los cuales 3 fueron errores de validación, 3 fueron problemas funcionales, 2 fueron problemas de correspondencia entre la aplicación y el documento, 1 fue un problema de errores ortográficos y 1 fue un problema de funcionalidad. En la segunda iteración, se encontraron 4 problemas de no conformidad, donde aún existían errores de validación y se detectaron problemas adicionales de escritura. Finalmente, en la tercera iteración, se obtuvieron resultados satisfactorios ya que no se detectaron problemas de no conformidad.

3.3 Pruebas de Aceptación

Las pruebas de aceptación son un conjunto de pruebas que se realizan para verificar si el software desarrollado cumple con los requisitos del cliente y si está listo para su uso en el entorno de producción. Estas pruebas se realizan después de las pruebas unitarias, de integración y del sistema. El objetivo principal de las pruebas de aceptación es asegurarse de que el software cumple con los requisitos del cliente y que está listo para su uso en el entorno de producción. Las pruebas de aceptación se pueden realizar manualmente o mediante herramientas automatizadas. Las pruebas manuales son realizadas por el equipo de pruebas, mientras que las herramientas automatizadas son utilizadas por los desarrolladores y los miembros del equipo de pruebas. Las pruebas de aceptación son importantes porque ayudan a garantizar que el software cumpla con los requisitos del cliente y esté listo para su uso en el entorno de producción.[34]

Con el fin de verificar si el cliente estaba satisfecho con la solución desarrollada, se llevaron a cabo dos iteraciones, y se obtuvieron los siguientes resultados:

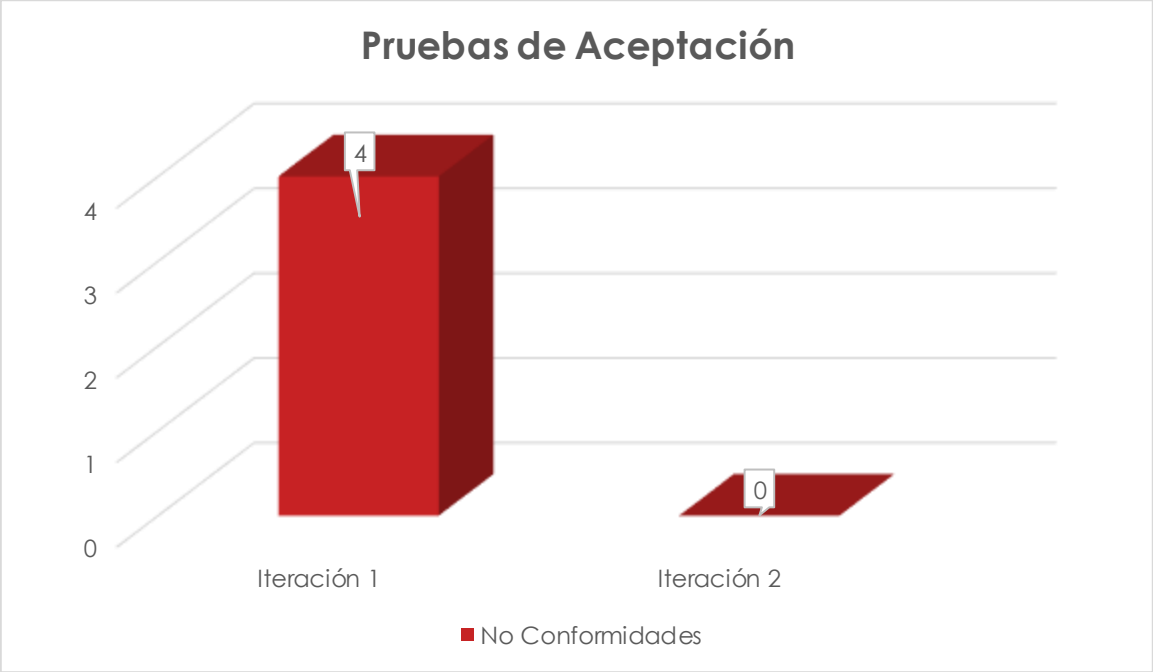

Gráfico 3.2 Pruebas de Aceptación

En una primera iteración se detectaron un total de 4 no conformidades (NC) de errores ortográficos. En la segunda iteración se obtuvieron resultados satisfactorios al no detectarse no conformidades. Finalmente, el cliente emitió un acta de aceptación.

3.4 Conclusiones parciales

El desarrollo de la solución propuesta en este capítulo ha permitido avanzar en la resolución de los desafíos planteados, obteniendo resultados que respaldan el valor y la viabilidad de la aplicación web desarrollada. La aplicación web desarrollada ha demostrado ser una solución efectiva y práctica para mejorar el proceso de gestión de ventas, la gestión de entidades y representantes y la generación de herramientas estadísticas relevantes. Al implementar esta solución, se ha optimizado la eficiencia operativa de la empresa, facilitado la toma de decisiones estratégicas y mejorado la satisfacción del cliente. Los resultados obtenidos han validado la propuesta de solución, demostrando que la aplicación web ofrece mejoras significativas en la gestión de las ventas, la generación de información estadística relevante y la calidad del servicio ofrecido.

CONCLUSIONES GENERALES

En el desarrollo de la presente tesis, se ha abordado de manera exitosa el objetivo principal de desarrollar una aplicación web para la gestión de ventas por la Fábrica de Piensos Cienfuegos. A lo largo de los capítulos, se ha llevado a cabo una rigurosa investigación que ha validado la relevancia y efectividad de la solución propuesta.

La necesidad social de mejorar la presencia en el mercado y mejorar la gestión de ventas de la empresa ha sido abordada mediante el diseño e implementación de una aplicación web innovadora. Esta solución ha demostrado su capacidad para agilizar el proceso de ventas, mejorar la eficiencia operativa y proporcionar herramientas estadísticas para la toma de decisiones empresariales.

La tesis ha cumplido con los objetivos planteados gracias a la aplicación de metodologías apropiadas, como el análisis documental, la revisión bibliográfica y el uso de entrevistas. Estas metodologías han permitido la obtención de una comprensión profunda de la problemática a resolver y han validado la efectividad de la aplicación web desarrollada.

RECOMENDACIONES

En futuros desarrollos, se recomienda continuar mejorando y actualizando la aplicación web para adaptarse a las cambiantes necesidades del mercado y mantener la competitividad de la empresa.

REFERENCIAS

[1] R. S. Pressman y B. R. Maxim, *Software Engineering: A Practitioner's Approach*, 8th ed. McGraw-Hill Education. 2014.

[2] «Software de Sales Hub Enterprise | HubSpot.» https://www.hubspot.es/products/sales/enterprise.

[3] «¿Qué es Infusionsoft? - La Consultoría Digital.» https://laconsultoriadigital.com/que-es-infusionsoft/

[4] «El Mejor CRM para Vendedores y Asesores | Parrot CRM.» https://theparrotcrm.com/

[5] «Los 6 mejores Software de Alimentación [2023]». https://www.softwaredoit.es/software-industrial/software-alimentacion-alimentaria.html.

[6] «AUP – Metodología.» https://metodologia.es/aup/

[7] «¿Qué es la arquitectura de flujo de datos? - Spiegato.» https://tareasuniversitarias.com/estilos-arquitectonicos.html.

[8] «Estilos arquitectónicos – Tareas Universitarias.» https://tareasuniversitarias.com/estilos-arquitectonicos.html.

[9] «Calaméo - ESTILOS Y PATRONES.» https://www.calameo.com/books/0051661110a1c15881ba8.

[10] «Estilo de Llamada y Retorno - Estilo Arquitectónico - 1Library.Co.» https://bing.com/search?q=Estilos+de+Llamada+y+Retorno+arquitectonico.

[11] «EstilosHeterogéneos on emaze.» https://www.emaze.com/@ATZTQZRQ/EstilosHeterog%C3%A9neos.

[12] «Arquitectura Peer To Peer (P2P) - Reactive Programming.» https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/p2p.

[13] «Microsoft Learn». https://bing.com/search?q=qu%C3%A9+es+python

[14] «¿Qué es Python?» https://bing.com/search?q=qu%C3%A9+es+python

[15] «Python.org». https://aws.amazon.com/es/what-is/django/

[16] «¿Qué es Django? - Explicación del software Django - AWS». https://aws.amazon.com/es/what-is/django/

[17] «Conceptos básicos de HTML - Aprender desarrollo web | MDN». https://bing.com/search?q=qu%C3%A9+es+python

[18] «Bootstrap: ¿qué es, para qué sirve y cómo instalarlo? - Rock Content». https://rockcontent.com/es/blog/bootstrap/

[19] «SQLite: ventajas y desventajas | Blog | HostingPlus.cl», *SQLite: ventajas y desventajas | Blog | HostingPlus.cl*. https://www.hubspot.es/products/sales/enterprise.

[20] «Axure RP 10 - Prototypes, Specifications, and Diagrams in One Tool». https://developer.mozilla.org/es/docs/Learn/CSS/First_steps/What_is_CSS

[21] «¿QUE ES Y COMO USAR VISUAL PARADIGM? - Yo Androide». https://yoandroide.xyz/que-es-y-como-usar-visual-paradigm/

[22] «EA en 30 minutos - proagile 20: Enterprise Architect en Español». https://enterprisearchitect.es/

[23] «SQL Client and Database Management Software - DbVisualizer.» https://www.dbvis.com/

[24] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, 2008.

[25] «El patrón modelo-vista-controlador: Arquitectura y frameworks explicados». https://www.freecodecamp.org/espanol/news/el-modelo-de-arquitectura-view-controller-pattern/

[26] E. Gamma, R. Helm, R. Johnson, y J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

[27] «Patrones de diseño GoF». https://www.laraveltip.com/aplica-mejores-practicas-siguiendo-los-patrones-graps/

[28] «Patrones Gof - EcuRed». https://www.ecured.cu/Patrones_Gof.

[29] «Nomenclaturas de programación: camelCase, PascalCase, snake_case.» https://platzi.com/clases/2218-pensamiento-logico-2020/35651-nomenclaturas-de-programacion-camelcase-pascalcase/

[30] A. Hunt y D. Thomas, *The Pragmatic Programmer: Your Journey to Mastery*. Addison-Wesley Professional, 1999.

[31] «¿Qué son las pruebas de caja blanca? - KeepCoding Bootcamps.» https://keepcoding.io/blog/que-son-pruebas-de-caja-blanca/

[32] T. H. Cormen, C. E. Leiserson, R. L. Rivest, y C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge. MIT Press, 2009.

[33] «¿Qué son las pruebas de caja negra? | KeepCoding Bootcamps.»

[34] «Pruebas de aceptación: el qué y el por qué - Nimblework.» https://www.nimblework.com/es/agile/pruebas-de-aceptacion/