DOCUMENTO DEL ARTICULO DE INVESTIGACION DEL SISTEMA DE VENTAS DE PRODUCTOS

**Autores:**

Farfán Sánchez Abraham Andrés

Agurto Pincay José Andrés

Pincay Álvarez Pablo Salvador

Vivanco García Ángel Enrique

#### Resumen

La falta de un sistema adecuado para gestionar las operaciones comerciales en supermercados ha generado problemas en la eficiencia operativa, impactando la experiencia del cliente y la rentabilidad de la empresa. Las dificultades para manejar inventarios, controlar la facturación y coordinar los pagos de forma integrada han llevado a una carga de trabajo significativa para el personal, además de pérdidas de ventas y una disminución en la satisfacción del cliente.

Por lo tanto, se propone el desarrollo de un software integral de ventas que automatice y optimice la gestión de inventarios, ventas, y facturación en supermercados, con el objetivo de mejorar la eficiencia operativa, la experiencia del cliente, y la rentabilidad de la empresa. Este sistema permitirá un seguimiento detallado de productos, ventas, control de inventarios, y generación de reportes.

Para alcanzar estos objetivos, se utilizarán los modelos COCOMO I y COCOMO II adaptados a las necesidades del proyecto para estimar el costo, esfuerzo y tiempo de desarrollo. Además, se empleará la metodología SCRUM para permitir la colaboración continua entre usuarios finales y el equipo de desarrollo, garantizando así un producto final alineado con las expectativas del negocio. La base de datos se gestionará con SQL Server, y la codificación del proyecto se llevará a cabo en el entorno de desarrollo Microsoft Visual Studio, utilizando el lenguaje de programación C#. También se utilizarán herramientas de modelado, como diagramas de clase y casos de uso, para asegurar una arquitectura eficiente y alineada con las funcionalidades requeridas.

**Palabras clave:** Sistema de ventas, Supermercado, Metodología SCRUM, COCOMO, Automatización, Gestión de inventarios.

#### Abstract

The lack of an adequate system to manage commercial operations in supermarkets has led to inefficiencies, negatively impacting customer experience and company profitability. Challenges in managing inventories, controlling billing, and coordinating payments have led to significant workload for staff, lost sales, and decreased customer satisfaction.

Therefore, we propose the development of an integrated sales software that automates and optimizes inventory management, sales, and billing in supermarkets, aiming to enhance operational efficiency, customer experience, and profitability. This system will allow detailed tracking of products, sales, inventory control, and report generation.

To achieve these objectives, we will use the COCOMO I and COCOMO II models adapted to the project's needs to estimate the cost, effort, and time required. Additionally, the SCRUM methodology will be employed to enable ongoing collaboration between end-users and the development team, ensuring a final product aligned with business expectations. The database will be managed with SQL Server, and the project coding will be performed in Microsoft Visual Studio using C# programming language. Modeling tools, such as class diagrams and use case diagrams, will also be used to ensure an efficient architecture aligned with the required functionalities.

**Keywords:** Sales System, Supermarket, SCRUM Methodology, COCOMO, Automation, Inventory Management.

#### Introducción

En la actualidad, el uso de sistemas informáticos es esencial en las empresas, que buscan automatizar y optimizar sus servicios mediante la implementación de sistemas de gestión de ventas que aumenten su competitividad y eficiencia. En el sector de supermercados, la administración de inventarios, facturación y coordinación de ventas es fundamental para ofrecer una experiencia satisfactoria al cliente, incrementar la rentabilidad y optimizar el flujo de trabajo interno.

La gerencia de [Nombre del Supermercado] ha identificado la necesidad de un software integral de ventas que permita la optimización de estas operaciones. El presente proyecto se enfoca en desarrollar un sistema de ventas de productos que automatice y facilite la gestión de inventarios, ventas y facturación, brindando al supermercado una herramienta que optimice su operación y contribuya a la mejora de la experiencia de sus clientes.

Este proyecto es llevado a cabo por estudiantes de [Universidad y Carrera], quienes, bajo la guía del [Nombre del Profesor], se encuentran en la asignatura de desarrollo de software. Se espera que el sistema propuesto ofrezca una solución informática robusta y adaptable, que incremente la eficiencia y mejore la rentabilidad de la empresa.

#### Objetivos

1. Desarrollar un sistema de ventas que automatice la gestión de inventarios, facturación, y el seguimiento de ventas.
2. Optimizar la asignación de recursos internos mediante la reducción de tareas repetitivas y la mejora del flujo de información.
3. Mejorar la experiencia del cliente ofreciendo procesos de venta más ágiles y precisos.
4. Implementar herramientas de análisis y reportes que permitan a la gerencia tomar decisiones informadas sobre el negocio.

#### Metodología

El desarrollo de este sistema utilizará una combinación de técnicas de estimación de recursos mediante los modelos COCOMO I y COCOMO II. La implementación del software se realizará mediante la metodología SCRUM, permitiendo la colaboración continua entre el equipo de desarrollo y los usuarios finales para asegurar que el sistema cumpla con los requerimientos reales del negocio.

Para la arquitectura y almacenamiento de datos, se empleará SQL Server, mientras que el código fuente será desarrollado en Microsoft Visual Studio utilizando el lenguaje C#. La planificación del proyecto y la estructura modular del software se diseñarán utilizando diagramas de clase, diagramas de casos de uso y otros modelos UML que optimicen la interacción y funcionalidad del sistema.

#### Estimación COCOMO

#### COCOMO Básico – proyecto orgánico

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Submodelos básicos** | **a** | **b** | **c** | **d** |
| Orgánico | 2,4 | 1,05 | 2,5 | 0,38 |
| Semi-acoplado | 3,0 | 1,12 | 2,5 | 0,35 |
| Empotrado | 3,6 | 1,20 | 2,5 | 0,32 |

1. **SUBMODELOS DE COCOMO I**

**SIZE Estimado =** 18 KLOC

* 1. **ORGÁNICO**

**Esfuerzo** = =2.4 \* (18)1.05 = 49 PM (Personas-Mes)

**Tiempo de Desarrollo** = = 2.5 \* (49)0.38 = 10.9 M (Meses)

=> **Personas**: 49/10.9 = 4.49 personas

=> **Productividad**: 18000/49 = 367.34 LOC/PM

* 1. **SEMI-ACOPLADO**

**Esfuerzo** = =3.0 \* (18)1.12 = 76 PM (Personas-Mes)

**Tiempo de Desarrollo** = = 2.5 \* (76)0.35 = 11.3 M (Meses)

=> **Personas**: 76/11.3 = 6.72 personas

=> **Productividad**: 18000/76 = 236.84 LOC/PM

* 1. **EMPOTRADO**

**Esfuerzo** = =3.6 \* (15)1.20 = 115 PM (Personas-Mes)

**DevTime** = = 2.5 \* (115)0.32 = 11.41M (Meses)

=> **Personas**: 115/11.41 = 10.07 personas

=> **Productividad**: 18000/115 = 156.52 LOC/PM

Para el sistema de ventas, el tipo **Orgánico** es generalmente el más adecuado:

**Orgánico**: Menor complejidad, con equipo pequeño y experimentado, y requisitos relativamente estables.

1. **COCOMO Intermedio – Proyecto Orgánico**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Submodelos intermedios** | **a** | **b** | **c** | **d** |
| Orgánico | 3,2 | 1,05 | 2,5 | 0,38 |
| Semi-acoplado | 3,0 | 1,12 | 2,5 | 0,35 |
| Empotrado | 2,8 | 1,20 | 2,5 | 0,32 |

* 1. **Tabla Factores de Costo**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Atributos | Valor | | | | | |
| Muy bajo | Bajo | Normal | Alto | Muy Alto | Extra alto |
| Atributos de Software | | | | | | |
| Fiabilidad | 0,75 | 0,88 | 1 | 1,15 | 1,4 |  |
| Tamaño de Base de datos |  | 0,94 | 1 | 1,08 | 1,16 |  |
| Complejidad | 0,7 | 0,85 | 1 | 1,15 | 1,3 | 1,65 |
| Atributos de hardware | | | | | | |
| Restricciones de tiempo de ejecución |  |  | 1 | 1,11 | 1,3 | 1,66 |
| Restricciones de memoria virtual |  |  | 1 | 1,06 | 1,21 | 1,56 |
| Volatilidad de la máquina virtual |  | 0,87 | 1 | 1,15 | 1,3 |  |
| Tiempo de respuesta |  | 0,87 | 1 | 1,15 | 1,3 |  |
| Atributo de personal | | | | | | |
| Capacidad de análisis | 1,46 | 1,19 | 1 | 0,86 | 0,71 |  |
| Experiencia en la aplicación | 1,29 | 1,13 | 1 | 0,91 | 0,82 |  |
| Calidad de los programadores | 1,42 | 1,17 | 1 | 0,86 | 0,7 |  |
| Experiencia en la máquina virtual | 1,21 | 1,1 | 1 | 0,9 |  |  |
| Experiencia en lenguaje | 1,14 | 1,07 | 1 | 0,95 |  |  |
| Atributos del proyecto | | | | | | |
| Técnicas actualizadas de programación | 1,24 | 1,1 | 1 | 0,91 | 0,82 |  |
| Utilización de herramientas de software | 1,24 | 1,1 | 1 | 0,91 | 0,83 |  |
| Restricciones de tiempo de desarrollo | 1,23 | 1,08 | 1 | 1,04 | 1,1 |  |

* 1. **Cálculo del Size**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Factores de Peso** | | | | |
| Factores Funcionales de Peso | Parámetros de Medida (1) | | | Contador (2) | Total Multiplicación (1) \*(2) |
| Simple | Media | Compleja |
| N.º Entrada de usuario | 7 | 10 | 15 | 20 | 200 |
| N.º Salida usuario | 5 | 7 | 10 | 10 | 70 |
| N.º Consulta usuario | 3 | 4 | 6 | 8 | 32 |
| N.º Archivos Lógicos Internos (tablas) | 4 | 5 | 7 | 4 | 20 |
| N.º Interfaces externas | 3 | 4 | 6 | 6 | 24 |
|  |  |  |  | | 346 |

* 1. **Calculo formulaLOC**

**Tabla de conversión de: Correlación código fuente a PF**

|  |  |
| --- | --- |
| Lenguaje | Correlación Código Fuente por PF (aprox) |
| Assembler | 320 |
| C | 128 |
| ALGOL | 105 |
| FORTRAN | 105 |
| PASCAL | 91 |
| RPG | 80 |
| PL/1 | 80 |
| Modula-2 | 80 |
| Prolog | 64 |
| LISP | 64 |
| BASIC | 64 |
| 4GL para BD | 40 |
| APL | 32 |
| Smaltallk | 29 |
| Query | 13 |
| Spreadsheet | 6 |
| Sql | 13 |
| VB | 24 |
| Java | 46 |
| Html | 14 |
| Delphi | 118 |
| C++ | 53 |
| COBOL | 107 |
| C# | 58 |

* 1. **Calculo de la Variable FAE (multiplicador)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Atributos | Valor | | | | | |
| Muy bajo | Bajo | Normal | Alto | Muy Alto | Extra alto |
| Atributos de Software | | | | | | |
| Fiabilidad | 0,75 | 0,88 | 1 | 1,15 | 1,4 |  |
| Tamaño de Base de datos |  | 0,94 | 1 | 1,08 | 1,16 |  |
| Complejidad | 0,7 | 0,85 | 1 | 1,15 | 1,3 | 1,65 |
| Atributos de hardware | | | | | | |
| Restricciones de tiempo de ejecución |  |  | 1 | 1,11 | 1,3 | 1,66 |
| Restricciones de memoria virtual |  |  | 1 | 1,06 | 1,21 | 1,56 |
| Volatilidad de la máquina virtual |  | 0,87 | 1 | 1,15 | 1,3 |  |
| Tiempo de respuesta |  | 0,87 | 1 | 1,15 | 1,3 |  |
| Atributo de personal | | | | | | |
| Capacidad de análisis | 1,46 | 1,19 | 1 | 0,86 | 0,71 |  |
| Experiencia en la aplicación | 1,29 | 1,13 | 1 | 0,91 | 0,82 |  |
| Calidad de los programadores | 1,42 | 1,17 | 1 | 0,86 | 0,7 |  |
| Experiencia en la máquina virtual | 1,21 | 1,1 | 1 | 0,9 |  |  |
| Experiencia en lenguaje | 1,14 | 1,07 | 1 | 0,95 |  |  |
| Atributos del proyecto | | | | | | |
| Técnicas actualizadas de programación | 1,24 | 1,1 | 1 | 0,91 | 0,82 |  |
| Utilización de herramientas de software | 1,24 | 1,1 | 1 | 0,91 | 0,83 |  |
| Restricciones de tiempo de desarrollo | 1,23 | 1,08 | 1 | 1,04 | 1,1 |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Submodelos intermedios | a | b | c | d |
| Orgánico | 3,2 | 1,05 | 2,5 | 0,38 |
| Semi-acoplado | 3,0 | 1,12 | 2,5 | 0,35 |
| Empotrado | 2,8 | 1,20 | 2,5 | 0,32 |

**ENTONCES SE TIENE:**

#### COCOMO II

* 1. **Datos iniciales**

**A** = 2.5

**SIZE** = 18 KLOC (18,000 líneas de código)

**Escalamiento B**:

* Factores de escala dados:
  + PREC = nominal (3.72)
  + FLEX = very high (1.01)
  + RESL = high (2.83)
  + TEAM = extra high (0.00)
  + PMAT = nominal (4.68)

**Multiplicadores de Esfuerzo (M)**:

* PERS = very high (0.63)
* PREX = high (0.87)

**Multiplicadores M**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Cost Drivers** | **Extra Low** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** |
| **RCPX** | 0.73 | 0.81 | 0.98 | 1.0 | 1.30 | 1.74 | 2.38 |
| **RUSE** | - | - | 0.95 | 1.0 | 1.07 | 1.15 | 1.24 |
| **PDIF** | - | - | 0.87 | 1.0 | 1.29 | 1.81 | 2.61 |
| **PERS** | 2.12 | 1.62 | 1.26 | 1.0 | 0.83 | 0.63 | 0.50 |
| **PREX** | 1.59 | 1.33 | 1.12 | 1.0 | 0.87 | 0.71 | 0.62 |
| **FCIL** | 1.43 | 1.30 | 1.10 | 1.0 | 0.97 | 0.73 | 0.62 |
| **SCED** | - | 1.43 | 1.14 | 1.0 | 1.0 | 1.0 | - |

**Person Months**

**Person Months**

1. **MODELO POST ARQUITECTURA**

**Cost Drivers:**

RELY = Low (0.75)

TIME = Very high (1.31)

ACAP = Very low (1.50)

PCAP = Very low (1.37)

TOOL = Low (1.12)

SCED = Very low (1.29)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Cost Drivers** | **Very Low** | **Low** | **Nominal** | **High** | **Very High** | **Extra High** |
| RELY | 0.75 | 0.88 | 1.00 | 2.48 | 1.24 | 0.00 |
| DATA |  | 0.93 | 1.00 | 2.03 | 2.03 | 0.00 |
| CPLX | 0.75 | 0.88 | 1.00 | 2.83 | 1.41 | 0.00 |
| RUSE |  | 0.91 | 1.00 | 2.19 | 1.10 | 0.00 |
| DOCU | 0.89 | 0.95 | 1.00 | 3.12 | 1.56 | 0.00 |
| TIME |  |  | 1.00 | 1.11 | 1.31 | 1.67 |
| STOR |  |  | 1.00 | 1.06 | 1.21 | 1.57 |
| PVOL |  | 0.87 | 1.00 | 1.15 | 1.30 |  |
| ACAP | 1.50 | 1.22 | 1.00 | 0.83 | 0.67 |  |
| PCAP | 1.37 | 1.16 | 1.00 | 0.87 | 0.74 |  |
| PCON | 1.24 | 1.10 | 1.00 | 0.92 | 0.84 |  |
| AEXP | 1.22 | 1.10 | 1.00 | 0.89 | 0.81 |  |
| PEXP | 1.25 | 1.12 | 1.00 | 0.88 | 0.81 |  |
| LTEX | 1.22 | 1.10 | 1.00 | 0.91 | 0.84 |  |
| TOOL | 1.24 | 1.12 | 1.00 | 0.86 | 0.72 |  |
| SITE | 1.25 | 1.10 | 1.00 | 0.82 | 0.84 | 0.78 |
| SCED | 1.29 | 1.10 | 1.00 | 1.00 | 1.00 |  |

#### Metodología de Desarrollo

#### Herramientas para el desarrollo del sistema

#### Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) creado por Microsoft. Proporciona un conjunto completo de herramientas y servicios para el desarrollo de software, incluyendo la escritura de código, la depuración, el diseño de interfaces de usuario y la administración de proyectos.

Visual Studio admite múltiples lenguajes de programación, como C#, C++, Visual Basic, F# y otros. Al utilizar Visual Studio en el proyecto, hemos obtenido varios beneficios y ventajas que han contribuido al éxito y eficiencia del desarrollo:

Amplia compatibilidad de lenguajes: Visual Studio ofrece soporte para una amplia gama de lenguajes de programación, lo cual me permitió seleccionar el lenguaje más adecuado para el proyecto.

Entorno de desarrollo integrado completo: Visual Studio proporciona un conjunto completo de herramientas y características que agilizaron mi flujo de trabajo. El entorno integrado incluye un editor de código altamente funcional, herramientas de depuración avanzadas y capacidades de prueba integradas. Esto nos permitió realizar todas las tareas de desarrollo desde un único lugar, lo que mejoró la productividad y la eficiencia general.

Depuración y análisis avanzados: La funcionalidad de depuración y análisis de Visual Studio resultó invaluable durante el desarrollo del proyecto. Nos permitió identificar y solucionar rápidamente errores y problemas en el código, lo que aceleró el proceso de depuración y garantizó un software más estable y libre de errores.

#### C#

Como lenguaje de programación, escogimos C#, pues la sintaxis es clara, legible, y nos facilita la comprensión y escritura del código. Además, cuenta con un amplio conjunto de bibliotecas y frameworks, un entorno de desarrollo integrado poderoso y una comunidad activa de desarrolladores que brindan soporte y recursos adicionales. Al utilizar una estructura de código bien definida y fácilmente comprensible, nos reduce la posibilidad de cometer errores.

Además de su sintaxis, C# nos ofrece otras características:

Amplio conjunto de bibliotecas y frameworks: C# cuenta con un vasto conjunto de bibliotecas y frameworks en la plataforma .NET, que proporcionan una funcionalidad predefinida y reutilizable para una amplia gama de tareas comunes en el desarrollo de software. Estas bibliotecas y frameworks permiten ahorrar tiempo y esfuerzo al aprovechar soluciones ya existentes en lugar de tener que desarrollar todo desde cero.

Entorno de desarrollo integrado (IDE): Visual Studio es el entorno de desarrollo integrado (IDE) principal para el desarrollo en C#. Proporciona herramientas poderosas que facilitan la escritura, depuración y prueba del código. La interfaz intuitiva de Visual Studio, junto con las numerosas características de productividad, como la finalización automática de código, la refactorización y la depuración visual, contribuyen a una experiencia de programación más eficiente y cómoda.

#### Metodología Scrum

**Visión del proyecto**

- **Objetivo**: Desarrollar un sistema de ventas que permita una gestión ágil y eficaz de inventarios, facturación, control de ingresos y análisis de ventas, mejorando así la experiencia del cliente y optimizando las operaciones internas.

**Planificación del proyecto**

**- Backlog del Producto**: Crear una lista detallada con los requerimientos y funcionalidades que incluirán módulos como gestión de inventarios, facturación, control de usuarios y generación de reportes.

**- Stakeholders**: Identificar a los interesados (administradores, personal de ventas y gerentes) para alinear los requerimientos con las necesidades de cada rol.

- **Equipo Scrum**: Formar el equipo con un Product Owner, que represente los intereses de los usuarios, un Scrum Master que guíe al equipo, y desarrolladores con habilidades técnicas necesarias.

**Sprint 0 (Preparación)**

**- Objetivo**: Configurar el entorno de desarrollo, definir la arquitectura del sistema y detallar historias de usuario iniciales, como el flujo de ventas y la administración de inventario.

**- Planificación y Estimación**: Crear un plan de trabajo inicial y una estimación del esfuerzo y tiempo necesario para el Sprint 0.

- **Definición de Sprints**: Determinar el número y duración de los sprints en base a las historias de usuario y establecer un cronograma.

**Sprint 1**

**- Reuniones Diarias**: Realizar daily scrums para que el equipo informe sobre avances y obstáculos.

- **Desarrollo Prioritario**: Implementar y probar historias de usuario claves, como la funcionalidad de ventas o facturación.

**- Revisión del Sprint**: Al terminar el sprint, mostrar el progreso al Product Owner y a otros stakeholders, recolectando feedback.

**- Retrospectiva**: Evaluar las prácticas del sprint y acordar mejoras para los siguientes.

**Sprints subsiguientes**

- **Desarrollo Incremental:** En cada sprint, implementar nuevas historias de usuario y mejorar las existentes. Los sprints pueden enfocarse en módulos específicos como el de reportes de ventas o control de inventario.

- **Backlog Refinement:** Ajustar las prioridades en el backlog de acuerdo con el feedback y las necesidades emergentes.

- **Comunicación Constante:** Realizar revisiones periódicas y mantener una comunicación fluida con los stakeholders para asegurar que el desarrollo esté alineado con los objetivos del negocio.

**Entrega del producto**

- **Revisión Final**: Al finalizar los sprints planificados, realizar una revisión del sistema completo, asegurando que se cumplen todos los requisitos.

- **Pruebas y Calidad**: Realizar pruebas exhaustivas para verificar la integridad del sistema.

- **Entrega al Cliente**: Hacer la entrega formal del sistema al cliente o usuario final.

**Mantenimiento y mejora continua**

- **Seguimiento y Soporte**: Proveer soporte técnico, resolviendo problemas y ajustando el sistema según sea necesario.

- **Retroalimentación y Aprendizaje**: Recoger opiniones de los usuarios para mejorar futuras iteraciones y usar las lecciones aprendidas para otros desarrollos futuros.

**Distribución de roles entre los integrantes**

|  |  |  |
| --- | --- | --- |
| **Nombres** | **Contacto** | **Rol** |
| Farfán Sánchez Abraham Andrés | [abraham.farfansan@ug.edu.ec](mailto:abr%61h%61m.%66%61%72fa%6es%61%6e@u%67%2ee%64%75.ec) | Líder de diseño arquitectónico |
| Agurto Pincay José Andrés | [jose.agurtopin@ug.edu.ec](mailto:j%6fs%65.a%67%75rto%70%69n@%75%67.e%64u.%65%63) | Líder de gestión y mejora |
| Pincay Álvarez Pablo Salvador | [pablo.pincayalv@ug.edu.ec](mailto:pab%6c%6f%2e%70%69%6e%63%61%79%61%6cv@%75%67%2e%65%64u%2e%65c) | Líder de diseño |
| Vivanco García Ángel Enrique | [angel.vivancogar@ug.edu.ec](mailto:%61%6eg%65%6c.%76%69%76%61%6eco%67%61%72@%75%67%2e%65d%75.%65%63) | Líder de requerimientos |

#### Resultado

Para el proyecto del sistema de ventas para el supermercado, el resultado esperado sería un sistema de gestión integral que optimice las operaciones de ventas, mejore la experiencia del cliente, y aumente la productividad del personal. El sistema permitiría:

**Objetivos y Funcionalidades del Sistema:**

* **Optimización de las ventas**: El sistema facilita la realización de ventas, la emisión de facturas, y el procesamiento rápido de pagos, lo que agiliza la atención al cliente.
* **Control de inventario**: Registra los productos disponibles, las entradas y salidas de inventario, y emite alertas para la reposición de productos.
* **Gestión de clientes y empleados**: Guarda información relevante de clientes y empleados, permitiendo una atención personalizada y una supervisión del desempeño del personal.
* **Generación de reportes**: Proporciona reportes de ventas, inventarios, y desempeño, lo que facilita la toma de decisiones basadas en datos.
* **Interfaz amigable y accesible**: Ofrece una interfaz fácil de usar que permite a los empleados del supermercado operar el sistema de forma intuitiva y eficiente.

**Arquitectura:**

El sistema implementa una variante de la arquitectura **Modelo-Vista-Controlador (MVC)**. En este caso, se usará la arquitectura **Datos-Entidad-Negocio-Presentación (DNP)**, donde:

* **Datos**: Gestiona el acceso a la base de datos donde contiene toda la lógica relacionada con la manipulación directa de datos, como consultas, inserciones, actualizaciones y eliminaciones en el sistema de ventas.
* **Entidad**: Incluye todas las clases que representan las entidades extraídas de los requerimientos funcionales. Además, estas clases actúan como modelos de datos que son independientes de la lógica de negocio o de presentación.
* **Negocio**: Coordina las interacciones entre el modelo y la vista, gestionando las solicitudes de los usuarios y asegurando que la interfaz responda a sus necesidades. Además, contiene la lógica de negocio del sistema, como las validaciones y las reglas específicas para buscar, registrar, actualizar y eliminar datos.
* **Presentación**: Presenta la información de manera clara y accesible a los usuarios del sistema, facilitando la administración del negocio.

**Calidad del Sistema:**

**Extensibilidad:**

* La arquitectura DNP, similar a MVC, separa responsabilidades, lo que facilita la incorporación de nuevas funcionalidades.

**Confiabilidad:**

* Diseñar con separación de capas asegura que los procesos críticos, como ventas y compras, se ejecuten sin interrupciones, incluso si hay cambios en otras áreas del sistema.

**Portabilidad:**

* Aunque es una aplicación de escritorio, la arquitectura permite adaptarla a diferentes entornos tecnológicos con ajustes mínimos.

**Requisitos de Hardware y Software:**

El sistema especifica los requisitos mínimos necesarios para su correcto funcionamiento, asegurando compatibilidad con las plataformas y el hardware del supermercado.

#### Conclusión

Se ha desarrollado una solución de software eficiente y robusta basada en el patrón de arquitectura **Datos-Entidad-Negocio-Presentación (DNP)** para la gestión de ventas de un supermercado. El objetivo principal del proyecto es automatizar los procesos de ventas y control de inventario, optimizando la operatividad del supermercado y mejorando tanto la experiencia del cliente como la productividad del personal.

La solución implementada permite registrar ventas, administrar inventarios, y gestionar datos de clientes y empleados de forma precisa, eliminando errores comunes en procesos manuales y agilizando la atención en el punto de venta. Adicionalmente, el sistema ofrece funcionalidades de generación de reportes, lo que facilita la toma de decisiones informadas por parte de los gerentes y permite un control exhaustivo de la actividad de ventas, el inventario, y el rendimiento del personal.

Se ha diseñado pensando en los diversos usuarios que interactuarán con el sistema, tales como cajeros, supervisores, gerentes y proveedores, asegurando que cada uno pueda acceder de manera intuitiva a las funciones que necesita para desempeñar sus tareas. La implementación del sistema garantiza una interfaz amigable y un entorno confiable, minimizando las interrupciones y contribuyendo significativamente a la satisfacción del cliente, la eficiencia operativa y la rentabilidad del supermercado.