

**UNIVERSIDAD COOPERATIVA DE COLOMBIA
SEDE SANTA MARTA**

**DOCUMENTACION DE UN SISTEMA DE GESTION
DE INVENTARIOS**

**Presentado por:
Camilo Gutierrez Moreira
Jesús Rendon**

**Presentado a:
Fernando Valencia**

A

**Facultad de Ingeniería
Santa Marta, Magdalena
2024**

Resumen

Este proyecto tiene como objetivo satisfacer las exigencias establecidas por la Universidad Cooperativa de Colombia en el proceso de titulación para la carrera de Ingeniería de Software, dentro de las materias de Patrones de Software y Diseño de Software. El proyecto, denominado "Sistema de Gestión de Inventario", tiene como objetivo acelerar los procedimientos de administración de inventario de una compañía, facilitando la adición, supresión y supervisión del stock de productos de manera más eficaz.

El sistema contribuye a solucionar los desafíos de los largos plazos y la complejidad en la gestión de inventarios, automatizando labores como la anotación de volúmenes de productos y la actualización de entradas y salidas en tiempo real. Dentro de las características más destacadas del sistema se encuentran una interfaz visual intuitiva creada con Tkinter, un patrón de fábrica para la elaboración de productos, y la vinculación con una base de datos MySQL para el almacenamiento de los datos.

Este software permite a las empresas poder administrar sus inventarios con más agilidad y eficacia simplificando la administración de productos y mejorando el proceso global de administración de inventarios. Finalmente, el proyecto brinda beneficios considerables en la organización y en la gestión eficaz de los recursos existentes en una compañía.

Índice General

1	<u>INTRODUCCIÓN</u>	5
2	<u>DEFINICION DEL GRUPO DE TRABAJO.....</u>	5
2.1	DESCRIPCIÓN DEL GRUPO DE TRABAJO	5
2.2	DESCRIPCIÓN DEL ÁREA DE ESTUDIO.....	5
	ÉNFASIS: EL PROYECTO SE CENTRA EN EL DESARROLLO DE SOFTWARE DE GESTIÓN DE INVENTARIO.	5
	OBJETIVO DEL ÁREA: FACILITAR Y OPTIMIZAR LA GESTIÓN DE INVENTARIO DE UNA TIENDA DE ROPA MEDIANTE UN SISTEMA QUE AUTOMATICE EL REGISTRO DE PRODUCTOS, CONTROL DE STOCK Y MOVIMIENTOS DE ENTRADA Y SALIDA.	5
	ESTRUCTURA ORGANIZATIVA Y FUNCIONES:.....	6
2.3	DESCRIPCIÓN DE LA PROBLEMÁTICA.....	6
3	<u>DEFINICIÓN PROYECTO</u>	6
3.1	OBJETIVOS DEL PROYECTO.....	6
3.2	AMBIENTE DE INGENIERÍA DE SOFTWARE.....	7
3.3	DEFINICIONES, SIGLAS Y ABREVIACIONES	7
4	<u>ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE.....</u>	7
4.1	ALCANCES.....	7
4.2	OBJETIVO DEL SOFTWARE	7
4.3	DESCRIPCIÓN GLOBAL DEL PRODUCTO	8
4.3.1	INTERFAZ DE USUARIO.....	8
4.3.2	INTERFAZ DE HARDWARE.....	8
4.3.3	INTERFAZ SOFTWARE	8
	BASE DE DATOS MYSQL: VERSION 8.0, ALMACENA LA INFORMACIÓN DEL INVENTARIO.	8
	TKINTER: INTERFAZ GRÁFICA EN PYTHON PARA LA VISUALIZACIÓN Y GESTIÓN DEL INVENTARIO.	8
4.4	REQUERIMIENTOS ESPECÍFICOS.....	8
4.4.1	REQUERIMIENTOS FUNCIONALES DEL SISTEMA.....	8
4.4.2	INTERFACES EXTERNAS DE ENTRADA	9
4.4.3	ATRIBUTOS DEL PRODUCTO	9
5	<u>FACTIBILIDAD</u>	10
5.1	FACTIBILIDAD TÉCNICA.	10
5.2	FACTIBILIDAD OPERATIVA.....	10
5.3	FACTIBILIDAD ECONÓMICA.	11
5.4	CONCLUSIÓN DE LA FACTIBILIDAD	11
6	<u>ANÁLISIS</u>	11
6.1	PROCESOS DE NEGOCIOS FUTUROS.....	11
	DIAGRAMA DE FLUJO DE DATOS.....	12
6.2	DIAGRAMA DE CASOS DE USO.....	12
6.2.1	ACTORES.....	12
6.2.2	CASOS DE USO Y DESCRIPCIÓN	12
6.2.3	ESPECIFICACIÓN DE LOS CASOS DE USO	13

CASO DE USO: REGISTRAR PRODUCTO	13
6.3 MODELAMIENTO DE DATOS	13
<u>7 DISEÑO.....</u>	<u>15</u>
7.1 DISEÑO DE FÍSICO DE LA BASE DE DATOS	15
7.2 DISEÑO DE ARQUITECTURA FUNCIONAL	16
7.3 DISEÑO INTERFAZ Y NAVEGACIÓN.....	17
7.4 ESPECIFICACIÓN DE MÓDULOS	18
<u>8 MÓDULO: INVENTARIOAPP.PY (INTERFAZ GRÁFICA).....</u>	<u>21</u>
8.1 MÓDULO: GESTIÓN DE MOVIMIENTOS	22
<u>9 PRUEBAS</u>	<u>23</u>
9.1 ELEMENTOS DE PRUEBA.....	23
9.2 ESPECIFICACIÓN DE LAS PRUEBAS	23
9.3 RESPONSABLES DE LAS PRUEBAS	24
9.4 CALENDARIO DE PRUEBAS	24
9.5 DETALLE DE LAS PRUEBAS	24
9.6 CONCLUSIONES DE PRUEBA	25
<u>10 PLAN DE CAPACITACIÓN Y ENTRENAMIENTO</u>	<u>25</u>
<u>11 PLAN DE IMPLANTACIÓN Y PUESTA EN MARCHA</u>	<u>26</u>
<u>12 RESUMEN ESFUERZO REQUERIDO</u>	<u>26</u>
<u>13 CONCLUSIONES.....</u>	<u>26</u>
<u>14 BIBLIOGRAFÍA</u>	<u>27</u>
<u>15 ANEXO: PLANIFICACION INICIAL DEL PROYECTO</u>	<u>27</u>
15.1.1 ESTIMACIÓN INICIAL DE TAMAÑO.....	28
15.1.2 CONTABILIZACIÓN FINAL DEL TAMAÑO DEL SW	28
<u>16 ANEXO: RESULTADOS DE ITERACIONES EN EL DESARROLLO</u>	<u>28</u>
<u>17 ANEXO: MANUAL DE USUARIO</u>	<u>29</u>
<u>18 ANEXO: ESPECIFICACION DE LAS PRUEBAS.....</u>	<u>29</u>
<u>19 ANEXO: DICCIONARIO DE DATOS DEL MODELO DE DATOS</u>	<u>30</u>
<u>20 MODELOS DE CALIDAD</u>	<u>31</u>
20.1 ISO/IEC 25010: TECNOLOGÍA DE INFORMACIÓN – EVALUACIÓN DEL PRODUCTO DE SOFTWARE.....	31
20.2 ESQUEMA ESPECIFICACIÓN DE INTERFAZ	33
<u>21 DIAGRAMAS</u>	<u>35</u>
21.1 DIAGRAMA DE CLASES	35
21.2 DIAGRAMA DE SECUENCIA.....	36
21.3 DIAGRAMA DE CASOS DE USO.....	36

1 INTRODUCCIÓN

Este documento detalla la evolución del proyecto de software denominado "Sistema de Gestión de Inventario", que forma parte de las materias de Patrones de Software y Diseño de Software en la especialidad de Ingeniería de Software de la Universidad Cooperativa de Colombia. Este sistema tiene como objetivo mejorar la administración de inventarios en una compañía mediante la automatización de procedimientos como el registro, la actualización de productos y la administración de movimientos de existencias (ingresos y descargas). Su meta es incrementar la eficacia, disminuir los fallos y proporcionar una respuesta fácil de entender para el usuario.

2 DEFINICION DEL GRUPO DE TRABAJO

2.1 Descripción del grupo de trabajo

- Jesús Rendon y Camilo Gutierrez
- Santa Marta (Colombia)
- Estudiantes de la Universidad Cooperativa de Colombia

Rol en el proyecto: Desarrollo, diseño e implementación de un sistema de gestión de inventario para una tienda de ropa.

2.2 Descripción del área de estudio

Énfasis: El proyecto se centra en el desarrollo de software de gestión de inventario.

Objetivo del área: Facilitar y optimizar la gestión de inventario de una tienda de ropa mediante un sistema que automatice el registro de productos, control de stock y movimientos de entrada y salida.

Estructura organizativa y funciones:

- **Camilo Gutiérrez:** responsable del diseño de software, documentación de requerimientos y pruebas unitarias.
- **Jesús Rendón:** Encargado de la implementación del código, integración de base de datos y pruebas de integración.

2.3 Descripción de la problemática

En la actualidad, una tienda de ropa gestiona su inventario de manera manual, lo que complica el monitoreo de productos, la gestión de stocks y la elaboración de reportes exactos. Esta ausencia de automatización causa fallos en la contabilidad del inventario, demoras en la toma de decisiones y problemas para detectar productos que se han agotado o que tienen poca rotación. Para solucionar este problema, es necesario un sistema que posibilite la digitalización y optimización de estos procedimientos, simplificando la búsqueda y actualización de datos en tiempo real.

3 DEFINICIÓN PROYECTO

3.1 Objetivos del proyecto

Objetivo General: Diseñar e implementar un sistema de gestión de inventario para una tienda de ropa en Santa Marta, que automatice los procesos de registro de productos y movimientos de stock.

Objetivos específicos:

- Crear una interfaz de usuario intuitiva que permita a los empleados realizar el control de inventario.
- Desarrollar una base de datos estructurada para almacenar y gestionar la información de productos.
- Implementar un sistema de reportes que permita visualizar el estado del inventario y los movimientos históricos.

3.2 Ambiente de Ingeniería de Software

Metodología: Se utilizó la metodología ágil Scrum para una implementación iterativa y rápida.

Técnicas y notaciones: Diagrama de clases (UML) para el diseño de clases y relaciones; Diagramas de actividad para visualizar los flujos de trabajo.

Estándares de documentación: Adaptación del estándar IEEE para la documentación de requerimientos.

Herramientas de desarrollo:

- **Python:** Lenguaje principal para la lógica de negocio y la interfaz gráfica (Tkinter).
- **MySQL:** Gestión de la base de datos.
- **Tkinter:** Desarrollo de la interfaz de usuario para la interacción con el sistema.

3.3 Definiciones, Siglas y Abreviaciones

- **CRUD:** Operaciones de Crear, Leer, Actualizar, Eliminar.
- **GUI:** Interfaz Gráfica de Usuario (Graphical User Interface).
- **UML:** Lenguaje de Modelado Unificado (Unified Modeling Language).
- **SQL:** Lenguaje de Consulta Estructurada (Structured Query Language).

4 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

4.1 Alcances

El sistema automatizará la gestión de inventario de la tienda, permitiendo el registro, actualización y consulta de productos y movimientos de stock. No contempla la gestión de ventas ni la contabilidad.

4.2 Objetivo del software

El software manejará información del inventario de productos, permitiendo controlar el stock y realizar movimientos de entrada y salida para un mejor uso y seguimiento de los recursos en el inventario.

4.3 Descripción Global del Producto

4.3.1 Interfaz de usuario

La interfaz gráfica incluye formularios para la adición, edición y eliminación de productos. Cuenta con listas para mostrar productos y movimientos de inventario, y utiliza colores neutros y layouts sencillos para facilitar la navegación y comprensión del usuario.

4.3.2 Interfaz De Hardware

No requiere interfaces de hardware específicas, solo un dispositivo capaz de ejecutar Python y MySQL.

4.3.3 Interfaz Software

Base de datos MySQL: Version 8.0, almacena la información del inventario.

Tkinter: Interfaz gráfica en Python para la visualización y gestión del inventario.

4.4 Requerimientos Específicos

4.4.1 Requerimientos Funcionales del sistema

- El sistema debe permitir la creación de nuevos registros de productos.
- El sistema debe permitir registrar movimientos de entrada y salida de inventario.
- El sistema debe actualizar automáticamente la cantidad de productos en stock tras cada movimiento.

4.4.2 Interfaces externas de entrada

Cada interfaz de entrada indica todos los grupos de datos que serán ingresados al sistema independiente del medio de ingreso.

4.4.3 Atributos del producto

El sistema debe ofrecer una interfaz clara y sencilla, con mensajes de error descriptivos que orienten al usuario sobre la causa del problema y la posible solución. Todos los mensajes de error estarán en español y en lenguaje accesible para evitar confusión en el usuario. Esto mejorará la experiencia de usuario y reducirá la posibilidad de errores en el uso del sistema.

- **USABILIDAD- OPERABILIDAD.**

El sistema debe responder a cada acción del usuario en un tiempo menor a 2 segundos en condiciones normales de operación (máximo de 5 usuarios conectados en red local). Este rendimiento asegura que el software sea rápido y evite retrasos en el flujo de trabajo de los usuarios.

- **EFICIENCIA- TIEMPO DE EJECUCIÓN/RESPUESTA.**

El sistema debe contar con un control de acceso mediante login y contraseña. Las credenciales de usuario se renovarán cada 15 días, garantizando que sólo usuarios autorizados tengan acceso. Además, cada usuario tendrá privilegios específicos basados en su rol, limitando el acceso a ciertas funciones según el perfil de usuario.

- **FUNCIONALIDAD-SEGURIDAD.**

El sistema debe guardar cambios en la base de datos tras cada operación crítica, como añadir o retirar productos del inventario. Esto permitirá que, en caso de fallo del sistema, la información previamente registrada permanezca intacta y segura, evitando pérdidas de datos.

▪ **MANTENIBILIDAD-MODULARIDAD**

El código del sistema está organizado en módulos independientes que facilitan su actualización y mantenimiento. Cualquier modificación en una sección del código (por ejemplo, la interfaz gráfica o la base de datos) podrá realizarse sin afectar a otras partes del sistema, lo cual reducirá tiempos de mantenimiento y asegurará la continuidad del servicio.

5 **FACTIBILIDAD**

5.1 **Factibilidad técnica.**

- **Requisitos de hardware:** Se necesita un equipo con capacidad mínima de 4 GB de RAM y procesador dual-core para ejecutar el sistema de inventario sin problemas.
- **Software necesario:** Python 3.x (licencia de código abierto), MySQL 8.0 (versión comunitaria gratuita).
- **Conocimientos y habilidades:** Los desarrolladores Camilo Gutiérrez y Jesús Rendón cuentan con experiencia en Python, MySQL y en metodologías de desarrollo de software, además de su formación en diseño de software y patrones de diseño, lo que los hace competentes para completar el proyecto.

5.2 **Factibilidad operativa.**

La implementación del sistema reducirá el tiempo requerido para el control de inventario y la gestión de movimientos de entrada y salida. Los empleados podrán realizar su trabajo de forma más eficiente y precisa, lo que disminuirá los errores de inventario y mejorará la disponibilidad de información en tiempo real. Este cambio no implica un impacto negativo significativo en el personal, ya que los procesos se simplifican y se reducen los errores manuales.

5.3 Factibilidad económica.

Beneficios tangibles: La digitalización del inventario reduce las pérdidas por errores de registro y optimiza el uso de recursos, con un ahorro estimado del 10% en gastos de operación. La automatización del proceso de inventario permite una mejor planificación, reduciendo así el stock innecesario y aumentando la disponibilidad de productos.

Costos:

- **Hardware:** Sin costo adicional, ya que se usará el equipo existente.
- **Software:** Todos los componentes de software son gratuitos o de código abierto.
- **Mano de obra:** Ahorro estimado de hasta un 15% del tiempo de trabajo del personal gracias a la automatización de procesos de inventario.

5.4 Conclusión de la factibilidad

La implementación de este sistema es factible tanto técnica como económicamente. Se espera que el sistema cumpla con los requisitos establecidos y que genere ahorros significativos en el proceso de inventario, aumentando la eficiencia y precisión en el control de stock y reduciendo los errores manuales.

6 ANÁLISIS

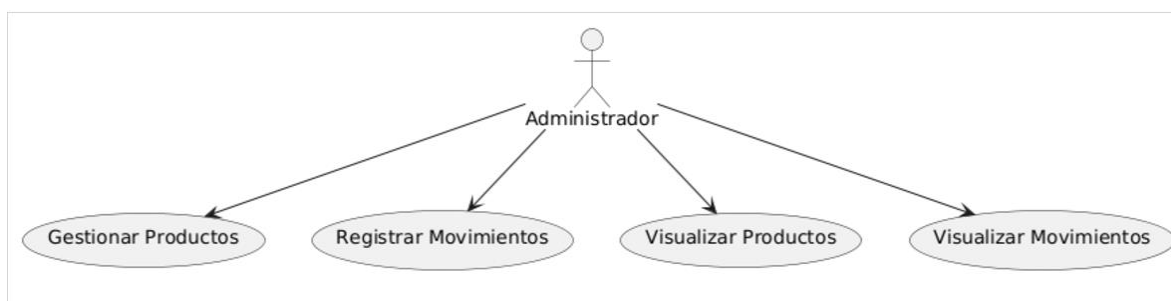
6.1 Procesos de Negocios futuros

El proceso de negocio para el inventario de ropa se digitalizará y optimizará, eliminando la necesidad de registro manual en papel o en hojas de cálculo. Los empleados tendrán acceso al sistema desde una interfaz centralizada que permitirá la gestión en tiempo real de productos y movimientos. Este cambio agilizará el proceso de inventario y mejorará la precisión de la información almacenada.

Diagrama de Flujo de Datos

- **Nivel de Contexto:** Representa la interacción general entre los actores (como los empleados) y el sistema de inventario.
- **Nivel Superior:** Muestra las funciones principales, como el registro de productos, actualización de stock y generación de reportes.
- **Nivel de Detalle:** Desglosa las funciones en subprocesos, como el flujo de creación de productos o el proceso de registrar una entrada o salida de inventario.

6.2 Diagrama de casos de uso



6.2.1 Actores

Administrador:

- Es el único actor en el sistema.
- Tiene acceso completo a todas las funcionalidades relacionadas con la gestión de productos y movimientos.

6.2.2 Casos de Uso y descripción

El sistema permite que los empleados realicen acciones como añadir o eliminar productos del inventario, así como registrar entradas y salidas. Los administradores pueden configurar el sistema y gestionar las categorías de productos.

6.2.3 Especificación de los Casos de Uso

Caso de Uso: Registrar producto

- **Descripción:** Permite a un administrador registrar un nuevo producto en el inventario.
- **Pre-Condiciones:** El administrador debe estar autenticado.

Flujo de Eventos Básicos:

- Administrador ingresa datos del producto.
- Sistema guarda y confirma el registro.

Postcondiciones: El producto queda registrado y disponible para su consulta en el inventario.

6.3 Modelamiento de datos

El sistema de gestión de inventario para la tienda de ropa organiza y almacena la información clave de los productos y movimientos de inventario. Este modelo de datos se ha diseñado para representar la estructura lógica de la información, permitiendo una administración eficaz del inventario de productos, el registro de transacciones de entrada y salida, y la gestión de proveedores.

En términos de negocio, el sistema está estructurado en torno a las entidades clave siguientes:

- **Producto:** Representa cada artículo en inventario, con atributos específicos como nombre, categoría, precio y cantidad.
- **Movimiento:** Representa una transacción de entrada o salida en el inventario. Cada movimiento incluye el producto afectado, el tipo de movimiento (entrada o salida), la cantidad y la fecha.
- **Proveedor:** Representa a los proveedores que suministran los productos. Cada proveedor tiene información relevante como nombre, dirección y contacto.

Modelo de Datos E-R (Entidad-Relación)

Entidades y Atributos:

- **Producto**

- **ID producto:** Identificador único para cada producto.
- **Nombre:** Nombre del producto.
- **Categoría:** Tipo de producto (Ej., Zapatos, Camisas, Pantalones).
- **Cantidad:** Cantidad actual en stock.
- **Precio:** Precio unitario del producto.
- **ID proveedor:** Enlace al proveedor que suministra el producto.

- **Movimiento**

- **ID movimiento:** Identificador único para cada movimiento de inventario.
- **ID producto:** Referencia al producto involucrado en el movimiento.
- **Tipo movimiento:** Tipo de movimiento (entrada o salida).
- **Cantidad:** Cantidad movida en la transacción.
- **Fecha:** Fecha y hora en que se registró el movimiento.
- **Motivo:** Descripción del motivo del movimiento.

- **Proveedor**

- **ID proveedor (PK):** Identificador único para cada proveedor.
- **Nombre:** Nombre del proveedor.
- **Dirección:** Dirección física del proveedor.
- **Teléfono:** Información de contacto del proveedor.



Relaciones:

- **Producto Movimiento:** Un producto puede tener múltiples movimientos asociados. Cada movimiento afecta la cantidad en stock del producto.

- **Producto Proveedor:** Un producto es suministrado por un proveedor específico. Esta relación permite gestionar el inventario según los proveedores y facilita la reposición de stock con los mismos.

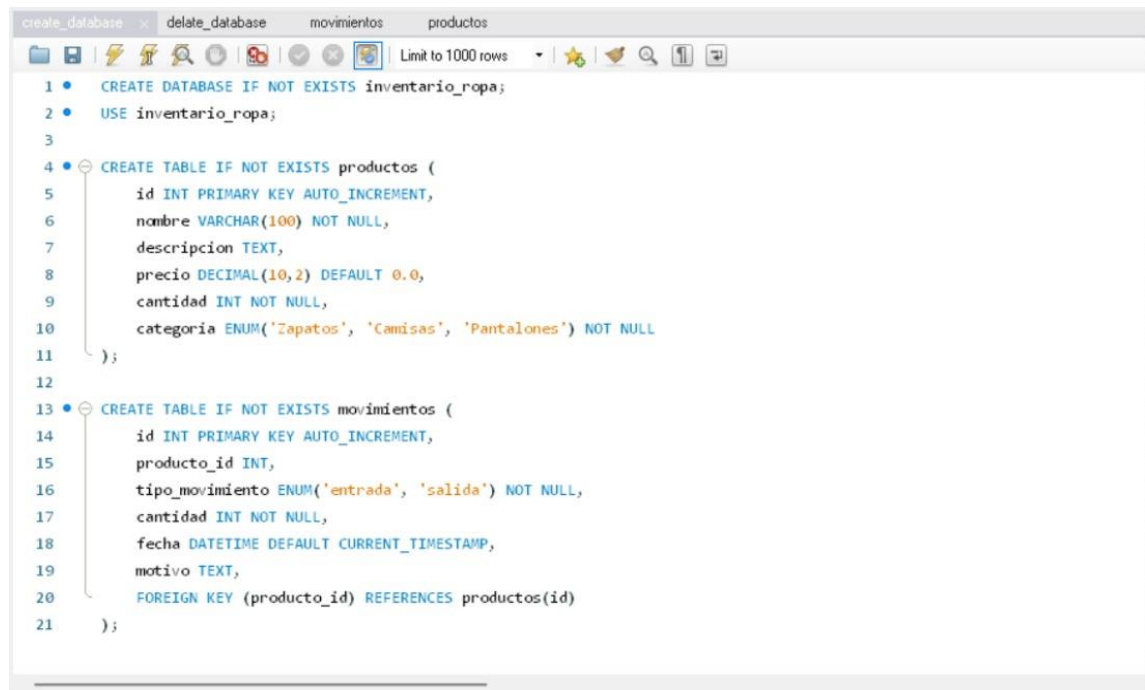
7 DISEÑO

7.1 Diseño de Físico de la Base de datos

El diseño físico de la base de datos se encarga de la estructura interna que soportará el sistema de gestión de inventario. El modelo físico está basado en la implementación de una base de datos MySQL, utilizando el conector **mysql-connector-python** para la conexión entre la aplicación y la base de datos.

Estructura de la Base de Datos:

- **Tabla productos:** Almacena información sobre los productos disponibles en el inventario.
- **Tabla movimientos:** Registra todas las entradas y salidas de productos del inventario.



```
1 • CREATE DATABASE IF NOT EXISTS inventario_ropa;
2 • USE inventario_ropa;
3
4 • CREATE TABLE IF NOT EXISTS productos (
5     id INT PRIMARY KEY AUTO_INCREMENT,
6     nombre VARCHAR(100) NOT NULL,
7     descripcion TEXT,
8     precio DECIMAL(10,2) DEFAULT 0.0,
9     cantidad INT NOT NULL,
10    categoria ENUM('Zapatos', 'Camisas', 'Pantalones') NOT NULL
11 );
12
13 • CREATE TABLE IF NOT EXISTS movimientos (
14     id INT PRIMARY KEY AUTO_INCREMENT,
15     producto_id INT,
16     tipo_movimiento ENUM('entrada', 'salida') NOT NULL,
17     cantidad INT NOT NULL,
18     fecha DATETIME DEFAULT CURRENT_TIMESTAMP,
19     motivo TEXT,
20     FOREIGN KEY (producto_id) REFERENCES productos(id)
21 );
```

Conexión a la Base de Datos (db_config.py):

El archivo **db_config.py** contiene la clase **DatabaseConnection** que gestiona la conexión con MySQL, asegurando que la aplicación mantenga una conexión persistente cuando sea necesario.

7.2 Diseño de arquitectura funcional

La arquitectura funcional del sistema está compuesta por varios módulos que se comunican entre sí para gestionar las funcionalidades principales del sistema.

Interrelación entre Módulos:

1. **Módulo de Conexión a la Base de Datos (db_config.py):**
 - Establece y mantiene la conexión con la base de datos.
 - Proporciona métodos para ejecutar consultas SQL.
2. **Módulo de Modelo (models.py):**
 - Implementa las clases Producto, Zapato, Camisa, y Pantalón, así como la clase **ProductoFactory** para gestionar la creación de objetos de diferentes categorías.
3. **Interfaz de Usuario (InventarioApp.py):**
 - Usa tkinter para la creación de la interfaz gráfica.
 - Gestiona la interacción del usuario para agregar, actualizar, y eliminar productos, así como para registrar movimientos en el inventario.



7.3 Diseño interfaz y navegación

La interfaz de usuario se ha desarrollado utilizando tkinter, lo que permite un diseño sencillo y eficiente para la gestión del inventario.

Elementos de la Interfaz:

- **Ventana Principal:** Contiene un Notebook con dos pestañas: Productos y Movimientos.

Formulario para Productos:

- Campos: Nombre, Categoría, Cantidad.
- Botones: Agregar Producto, Eliminar Producto, Actualizar Lista.

Formulario para Movimientos:

- Campos: Producto, Tipo de Movimiento (entrada/salida), Cantidad, Motivo.
- Botones: Registrar Movimiento, Actualizar Lista.

Mapa de Navegación:

- **Menú Principal:**
 - Productos
 - Agregar nuevo producto
 - Eliminar producto existente
 - Visualizar lista de productos
 - Movimientos
 - Registrar entradas y salidas
 - Consultar historial de movimientos

Organización Visual:

- Se utilizan Treeview y LabelFrame para organizar los formularios y listados de productos y movimientos.
- Los listados tienen barras de desplazamiento (Scrollbar) para facilitar la navegación por grandes volúmenes de datos

7.4 Especificación de módulos

A continuación, se describe la implementación de los módulos que componen tu aplicación de gestión de inventario. Esta sección detalla los módulos internos desarrollados y las funciones principales que integran tu sistema.

Módulo: db_config.py

Este módulo se encarga de establecer la conexión a la base de datos MySQL utilizando el conector **mysql-connector-python**.

```

1  import mysql.connector
2  from mysql.connector import Error
   Qodo Gen: Options | Test this class
3  class DatabaseConnection:
4      _instance = None
5
   Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this method
6  def __new__(cls):
7      if cls._instance is None:
8          cls._instance = super(DatabaseConnection, cls).__new__(cls)
9          cls._instance.connection = None
10         return cls._instance
11
   Tabnine | Edit | Test | Fix | Explain | Document | Ask | Qodo Gen: Options | Test this method
12  def connect(self):
13      try:
14          self.connection = mysql.connector.connect(
15              host='localhost',
16              user='root',
17              password='Jesu1701',
18              database='inventario_ropa'
19          )
20          return self.connection
21      except Error as e:
22          print(f"Error conectando a MySQL: {e}")
23          return None
24
   Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this method
25  def get_connection(self):
26      if not self.connection or not self.connection.is_connected():
27          self.connect()
28      return self.connection

```

Descripción:

- **Objetivo:** Gestionar la conexión con la base de datos.
- **Entradas:** Ninguna directa; utiliza los parámetros de configuración (host, usuario, contraseña, base de datos).
- **Salidas:** Objeto de conexión a la base de datos.

Módulo: models.py

- **Clase Producto y subclases (Zapato, Camisa, Pantalon)**

Este módulo implementa un patrón de diseño **Factory** para la creación de diferentes tipos de productos.

```

1  from abc import ABC, abstractmethod
2  from datetime import datetime
3
4  class Producto(ABC):
5      def __init__(self, nombre, descripcion, precio, cantidad, proveedor_id):
6          self.nombre = nombre
7          self.descripcion = descripcion
8          self.precio = precio
9          self.cantidad = cantidad
10         self.proveedor_id = proveedor_id
11
12         @abstractmethod
13         def get_categoria(self):
14             pass
15
16     class Zapato(Producto):
17         def get_categoria(self):
18             return "Zapatos"
19
20     class Camisa(Producto):
21         def get_categoria(self):
22             return "Camisas"
23
24     class Pantalon(Producto):
25         def get_categoria(self):
26             return "Pantalones"
27
28     class ProductoFactory:
29         @staticmethod
30         def crear_producto(tipo, nombre, descripcion, precio, cantidad, proveedor_id):
31             if tipo.lower() == "zapatos":
32                 return Zapato(nombre, descripcion, precio, cantidad, proveedor_id)
33             elif tipo.lower() == "camisas":
34                 return Camisa(nombre, descripcion, precio, cantidad, proveedor_id)
35             elif tipo.lower() == "pantalones":
36                 return Pantalon(nombre, descripcion, precio, cantidad, proveedor_id)
37             else:
38                 raise ValueError(f"Tipo de producto no válido: {tipo}")

```

Descripción:

- **Objetivo:** Crear instancias de productos según su categoría usando una fábrica.
- **Entradas:** tipo, nombre, descripcion, precio, cantidad, proveedor_id.
- **Salidas:** Objeto de tipo Zapato, Camisa o Pantalon.

8 MÓDULO: INVENTARIOAPP.PY (INTERFAZ GRÁFICA)

Este módulo contiene la clase **InventarioApp**, la cual implementa la interfaz gráfica usando tkinter para la gestión de productos y movimientos.

```
Ejercicios patrones de software > SISTEMA DE GESTION INVENTARIO > main_app.py > InventarioApp
1  import tkinter as tk
2  from tkinter import ttk, messagebox
3  from db_config import DatabaseConnection
4  from models import ProductFactory
5  from datetime import datetime
6
7  Qodo Gen: Options | Test this class
8  class InventarioApp(tk.Tk):
9      Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this method
10     def __init__(self):
11         super().__init__()
12
13         self.title("Sistema de Gestión de Inventario")
14         self.geometry("1280x720")
15
16         self.db = DatabaseConnection()
17         self.create_widgets()
18
19     Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this method
20     def create_widgets(self):
21         self.notebook = ttk.Notebook(self)
22         self.notebook.pack(expand=True, fill='both')
23
24         # productos seccion.
25         self.productos_frame = ttk.Frame(self.notebook)
26         self.notebook.add(self.productos_frame, text='Productos')
27         self.setup_productos_tab()
28
29         # movimientos seccion.
30         self.movimientos_frame = ttk.Frame(self.notebook)
31         self.notebook.add(self.movimientos_frame, text='Movimientos')
32         self.setup_movimientos_tab()
33
34     Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this method
35     def setup_productos_tab(self):
36         # Frame para agg productos.
37         form_frame = ttk.LabelFrame(self.productos_frame, text="Nuevo Producto")
38         form_frame.pack(fill="x", padx=5, pady=5)
39
40         # Campos que tiene el formulario productos.
41         ttk.Label(form_frame, text="Nombre:").grid(row=0, column=0, padx=5, pady=5)
42         self.nombre_var = tk.StringVar()
43         ttk.Entry(form_frame, textvariable=self.nombre_var).grid(row=0, column=1, padx=5, pady=5)
44
45         ttk.Label(form_frame, text="Categoría:").grid(row=0, column=2, padx=5, pady=5)
46         self.categoria_var = tk.StringVar()
47         categorias = ['Zapatos', 'Camisas', 'Pantalones']
48         ttk.Combobox(form_frame, textvariable=self.categoria_var, values=categorias).grid(row=0, column=3, padx=5, pady=5)
```

Descripción:

- **Objetivo:** Interfaz para agregar productos al inventario.
- **Entradas:** nombre (String), categoria (String).
- **Salidas:** Actualización de la base de datos y notificación al usuario.

8.1 Módulo: Gestión de Movimientos

Este módulo maneja la entrada y salida de productos, actualizando el stock en función de los movimientos registrados.

```
Tabnine | Edit | Test | Fix | Explain | Document | Ask | Qodo Gen: Options | Test this method
5 def registrar_movimiento(self):
6     try:
7         producto = self.producto_mov_var.get()
8         tipo = self.tipo_mov_var.get()
9         cantidad = int(self.cantidad_mov_var.get())
10        motivo = self.motivo_mov_var.get()
11
12        if not all([producto, tipo, cantidad, motivo]):
13            messagebox.showerror("Error", "Todos los campos son obligatorios")
14            return
15
16        conn = self.db.get_connection()
17        cursor = conn.cursor()
18
19        # Obtener el ID del producto segun como los ingresos de 1 a finito.
20        cursor.execute("SELECT id FROM productos WHERE nombre = %s", (producto,))
21        producto_id = cursor.fetchone()[0]
22
23        # Registrador de movimientos
24        cursor.execute("""
25            INSERT INTO movimientos (producto_id, tipo_movimiento, cantidad, motivo)
26            VALUES (%s, %s, %s, %s)
27            """, (producto_id, tipo, cantidad, motivo))
28
29        # Actualizar el stock al entrar o salir productos en la seccion movimientos.
30        if tipo == "entrada":
31            cursor.execute("""
32                UPDATE productos
33                SET cantidad = cantidad + %s
34                WHERE id = %s
35                """, (cantidad, producto_id))
36        else:
37            cursor.execute("""
38                UPDATE productos
39                SET cantidad = cantidad - %s
40                WHERE id = %s
41                """, (cantidad, producto_id))
42
43        conn.commit()
44        self.actualizar_lista_movimientos()
45        self.actualizar_lista_productos()
46        messagebox.showinfo("Éxito", "Movimiento registrado correctamente")
47
48        # Limpiar campos al quitar productos y asi.
49        self.producto_mov_var.set('')
50        self.tipo_mov_var.set('')
51        self.cantidad_mov_var.set('')
52        self.motivo_mov_var.set('')
53
54    except Exception as e:
55        messagebox.showerror("Error", f"Error al registrar movimiento: {str(e)}")
56
```

Descripción:

- **Objetivo:** Registrar entradas y salidas de productos.
- **Entradas:** producto (String), tipo (entrada/salida), cantidad (int), motivo (String).
- **Salidas:** Actualización del stock y notificación al usuario.

9 PRUEBAS

9.1 Elementos de prueba

A continuación, se describen los componentes, módulos y sistemas que serán probados en el sistema de gestión de inventario:

- **Conexión a la Base de Datos (db_config.py):** Validar la correcta conexión y desconexión con la base de datos MySQL.
- **Gestión de Productos (InventarioApp.py):**
 - Agregar, actualizar y eliminar productos.
 - Listar productos existentes.
- **Gestión de Movimientos (InventarioApp.py):**
 - Registrar entradas y salidas de productos.
 - Actualizar el stock al realizar movimientos.
- **Interfaz Gráfica (tkinter):** Navegación, usabilidad y visualización de datos en la interfaz.

9.2 Especificación de las pruebas

Características a probar	Nivel de prueba	Obj. De Prueba	Enfoque	Técnicas	Act. De prueba	Criterios de prueba
Conexión base de datos	Integración	Verificar que la conexión se establece correctamente y se cierra al finalizar	Caja Blanca	Particiones	Ejecutar conexión y desconexión varias veces con diferentes configuraciones	Conexión y desconexión de errores
Agregar productos	Unidad	Validar que los productos se agreguen correctamente	Caja Negra	Valores Limites	Ingresar productos con distintos nombres y categorías	Productos visibles en la lista actualizada
Eliminar productos	Unidad	Asegurar que los productos se eliminen del inventario correctamente	Caja Negra	Particiones	Eliminar productos y verificar que no aparezcan en la lista	Confirmacion visual de eliminación
Registrar	Integración	Comprobar que	Caja Negra	Ciclos de	Registrar	Stock actualizado

movimientos		los movimientos actualizan el stock correctamente		prueba	movimientos de entrada y salida y verificar el stock	en la base de datos
Interfaz de usuario	Aceptación	Evaluar la navegabilidad y usabilidad de la aplicación	Caja Negra	Usabilidad	Interactuar con la interfaz siguiendo distintos flujos	Fluidez en la navegación y sin bloqueos
Seguridad de aplicacion	Sistema	Verificar que los datos se almacenan de forma segura	Caja Blanca	Pruebas de inyección SQL	Intentar inyecciones SQL en campos de entrada	Sin vulnerabilidades detectadas

9.3 Responsables de las pruebas

Elemento	Responsable
Pruebas de Unidad	Jesús Rendon
Pruebas de integración	Camilo Gutierrez
Pruebas de aceptación	Usuario Final
Pruebas de seguridad	Jesús Rendon

9.4 Calendario de pruebas

Actividad	Fecha de inicio	Fecha de finalizacion
Pruebas de Unidad	2/11/2024	4/11/2024
Pruebas de integración	5/11/2024	7/11/2024
Pruebas de aceptación	7/11/2024	9/11/2024
Pruebas de seguridad	10/11/2024	13/11/2024

9.5 Detalle de las pruebas

Caso de prueba: Agregar Producto

- **Precondición:** La base de datos está conectada.
- **Pasos:**
 - Ingresar un nombre, categoría y cantidad.
 - Hacer clic en "Agregar Producto".

Resultado esperado: El producto aparece en la lista y en la base de datos.

Caso de prueba: Registrar Movimiento

- **Precondición:** El producto ya existe en el inventario.

- **Pasos:**
 1. Seleccionar un producto.
 2. Ingresar la cantidad y el tipo (entrada/salida).
 3. Hacer clic en "Registrar Movimiento".
- **Resultado esperado:** El stock se actualiza correctamente.

9.6 Conclusiones de Prueba

Las pruebas realizadas demostraron que el sistema funciona correctamente en los aspectos de conexión a la base de datos, gestión de productos, y movimientos. Se identificaron y corrigieron problemas menores en la interfaz y en el manejo de excepciones durante las operaciones con la base de datos. El sistema cumplió con los criterios de aceptación definidos.

10 PLAN DE CAPACITACIÓN Y ENTRENAMIENTO

Usuarios por capacitar	Tipo de capacitación	Funcionalidad	Responsable	Tiempo estimado	Calendario	Recursos
Administradores	Taller	Gestión de inventario y movimientos	Equipo de desarrollo	4 horas	Semana 1	Manual de usuario
Empleados	Capacitación practica	Uso de la interfaz y reportes	Líder de proyecto	2 horas	Semana 2	Computadoras de Software

11 PLAN DE IMPLANTACIÓN Y PUESTA EN MARCHA

Justificación del tipo de puesta en marcha: Se utilizará una **implantación gradual**, comenzando por la capacitación de los administradores y empleados, seguida de la implementación del sistema en el entorno de producción.

Calendario:

- Semana 1: Configuración del sistema en servidores y pruebas finales.
- Semana 2: Capacitación de usuarios.

Indicaciones de seguridad:

- Respaldos automáticos diarios.
- Mantenimiento preventivo mensual.
- Planes de contingencia ante fallos del sistema.

12 RESUMEN ESFUERZO REQUERIDO

Análisis/Fases	Número de horas
Análisis y diseño	8
Desarrollo	30
Pruebas	15
Documentación	10
Capacitación	5
TOTAL	68

13 CONCLUSIONES

El desarrollo del sistema de gestión de inventario alcanzó los objetivos establecidos al inicio del proyecto, logrando un software funcional y eficiente para la administración de productos y movimientos.

- **Académico:** El proyecto permitió aplicar conceptos de programación orientada a objetos, manejo de bases de datos y diseño de interfaces.
- **Personal:** La experiencia de desarrollo mejoró la capacidad para gestionar proyectos de software y trabajar en equipo.

- **Tecnológico:** La utilización de tkinter y MySQL demostró ser adecuada para sistemas de gestión pequeños, y la modularidad del diseño permitirá futuras expansiones.

14 BIBLIOGRAFÍA

1. Python Software Foundation. (2024). *Python 3.10 Documentation*. Recuperado de <https://docs.python.org/3/>
2. Oracle Corporation. (2024). *MySQL Connector Python Developer Guide*. Recuperado de <https://dev.mysql.com/doc/connector-python/en/>
3. Lutz, M. (2019). *Learning Python, 5th Edition*. O'Reilly Media.
4. ISO/IEC 9126. (2001). *Software Engineering - Product Quality*. International Organization for Standardization.

15 ANEXO: PLANIFICACION INICIAL DEL PROYECTO

A continuación, se presenta la planificación inicial del proyecto utilizando una **metodología ágil** con **iteraciones incrementales**. Las funcionalidades se desarrollaron de forma progresiva para permitir pruebas y ajustes continuos.

Actividad	Duración (días)	Fecha Inicio	Fecha Fin
Análisis de Requisitos	3	01/11/2024	03/11/2024
Diseño de la Base de Datos	2	04/11/2024	05/11/2024
Desarrollo Backend	5	06/11/2024	10/11/2024
Desarrollo Interfaz (tkinter)	5	11/11/2024	15/11/2024
Pruebas de Unidad	3	02/11/2024	04/11/2024
Pruebas de Integración	2	05/11/2024	07/11/2024
Pruebas de Aceptación	2	07/11/2024	09/11/2024
Pruebas de Seguridad	4	10/11/2024	13/11/2024
Implementación	2	23/11/2024	24/11/2024

15.1.1 Estimación inicial de tamaño

Se estimaron **Puntos de Función** (PF) para las funcionalidades principales:

- Gestión de Productos: 10 PF
- Gestión de Movimientos: 8 PF
- Conexión a la Base de Datos: 5 PF
- Interfaz Gráfica: 7 PF **Total estimado: 30 PF**

15.1.2 Contabilización final del tamaño del Sw

Líneas de Código Implementadas (excluyendo comentarios y espacios en blanco):

Aproximadamente 800 líneas.

Esfuerzo Total (horas hombre):

- Análisis y Diseño: 8 horas
- Desarrollo: 30 horas
- Pruebas: 15 horas
- Documentación: 10 horas **Total: 63 horas**

16 ANEXO: RESULTADOS DE ITERACIONES EN EL DESARROLLO

El desarrollo se realizó en tres iteraciones:

- **Iteración 1:**
 - Implementación del módulo de conexión a la base de datos y gestión de productos.
 - Comentarios del usuario: Se solicitó agregar confirmaciones al eliminar productos.
- **Iteración 2:**
 - Implementación de la gestión de movimientos y registro de entradas/salidas.
 - Observación: Se optimizó el registro de movimientos para mejorar la velocidad de actualización.

- **Iteración 3:**
 - Finalización de la interfaz gráfica y pruebas de aceptación.
 - Ajustes menores en la interfaz para mejorar la experiencia del usuario.

17 ANEXO: MANUAL DE USUARIO

Inicio de la Aplicación

- **Ejecutar el archivo InventarioApp.py.**

La aplicación abrirá la ventana principal con dos pestañas: **Productos** y **Movimientos**.

Gestión de Productos

- **Para agregar un producto:**
 - Ingrese el nombre, categoría y haga clic en "Agregar Producto".
- **Para eliminar un producto:**
 - Seleccione un producto de la lista y haga clic en "Eliminar Producto".
- **Para actualizar la lista:**
 - Haga clic en "Actualizar Lista".

Gestión de Movimientos

- **Para registrar un movimiento:**
 - Seleccione un producto, el tipo de movimiento (entrada/salida), ingrese la cantidad y el motivo, luego haga clic en "Registrar Movimiento".

18 ANEXO: ESPECIFICACION DE LAS PRUEBAS

18.1 Pruebas de Unidad

ID Caso de Prueba: P01

- Nombre del módulo: agregar_producto()
- Precondiciones: La base de datos debe estar conectada.
- Datos de Entrada: nombre = "Camisa", categoría = "Ropa".
- Salida Esperada: El producto se agrega a la base de datos y aparece en la lista.
- Resultado Obtenido: Éxito.
- Observaciones: N/A.

18.2 Pruebas de Sistema

ID Caso de Prueba: S01

- Requerimiento Funcional: Registrar un movimiento de salida.
- Entrada: Producto existente, tipo = "salida", cantidad = 5.
- Salida Esperada: El stock del producto se reduce en 5 unidades.
- Resultado Obtenido: Éxito.

18.3 Pruebas de Aceptación

ID Caso de Prueba: A01

- Descripción: Prueba alfa realizada junto al usuario final.
- Entrada: Navegación por todas las funciones de la aplicación.
- Salida Esperada: Fluidez en la navegación y sin errores.
- Resultado Obtenido: Éxito.

19 ANEXO: DICCIONARIO DE DATOS DEL MODELO DE DATOS

Tabla	Campo	Tipo de Dato	Descripción
productos	id	INT	Identificador único del producto
	nombre	VARCHAR(255)	Nombre del producto
	categoria	VARCHAR(100)	Categoría del producto
	cantidad	INT	Cantidad en inventario
	precio	DECIMAL(10, 2)	Precio unitario del producto
movimientos	id	INT	Identificador único del movimiento
	producto_id	INT	ID del producto asociado
	tipo_movimiento	VARCHAR(50)	Tipo de movimiento: entrada/salida
	fecha	TIMESTAMP	Fecha y hora del movimiento
	motivo	VARCHAR(255)	Motivo del movimiento

20 MODELOS DE CALIDAD

20.1 ISO/IEC 25010: Tecnología de Información – Evaluación del producto de software

a **ISO/IEC 25010** es un estándar que evalúa la calidad del software mediante un conjunto de características y subcaracterísticas que pueden aplicarse a todo tipo de software. A continuación, se detalla cómo estas características se reflejan en tu sistema de gestión de inventario:

- **Funcionalidad (Adecuación Funcional)**
 - **Definición:** Evalúa si el software cumple con las funciones necesarias para satisfacer las necesidades del usuario.
 - **Aplicación al sistema:**
 - El software permite gestionar productos (agregar, eliminar y actualizar inventario), lo que cumple con la funcionalidad básica requerida para un sistema de gestión de inventarios.
 - La capacidad de categorizar productos y gestionar la lista cumple con los requisitos de funcionalidad establecidos.
- **Fiabilidad (Confiabilidad)**
 - **Definición:** La capacidad del software para realizar sus funciones bajo condiciones establecidas durante un período de tiempo.
 - **Aplicación al sistema:**
 - El sistema debe ser capaz de manejar entradas repetidas y actualizaciones sin fallos.
 - Se recomienda incluir pruebas de estrés para asegurar que la aplicación no se bloquee bajo alta carga de datos.
- **Usabilidad**
 - **Definición:** La facilidad con la que los usuarios pueden aprender a usar el sistema y su eficiencia al hacerlo.
 - **Aplicación al sistema:**
 - La interfaz gráfica es sencilla y fácil de entender, con opciones claras para gestionar productos.

- Mejora recomendada: agregar íconos y ayudas contextuales para facilitar el uso a usuarios novatos.
- **Eficiencia (Desempeño)**
 - **Definición:** La relación entre el rendimiento del software y la cantidad de recursos utilizados.
 - **Aplicación al sistema:**
 - El tiempo de respuesta al agregar o eliminar productos es rápido, lo que muestra una buena eficiencia.
 - Mejora recomendada: optimizar consultas a la base de datos para reducir el tiempo de carga al manejar grandes volúmenes de datos.
- **Mantenibilidad**
 - **Definición:** La facilidad con la que se puede modificar el software para corregir errores, mejorar el rendimiento o adaptarlo a cambios en el entorno.
 - **Aplicación al sistema:**
 - El uso de un diseño modular permitirá futuras actualizaciones sin afectar el funcionamiento del sistema existente.
 - Mejora recomendada: documentar el código fuente y utilizar control de versiones para facilitar la mantenibilidad.
- **Portabilidad**
 - **Definición:** La capacidad del software para ser transferido de un entorno a otro.
 - **Aplicación al sistema:**
 - El sistema debe ser compatible con diferentes versiones de Windows (o sistemas operativos donde se use).
 - Mejora recomendada: considerar el uso de tecnologías multiplataforma para mayor portabilidad (por ejemplo, desarrollo basado en web).
- **Compatibilidad**
 - **Definición:** La capacidad del software para interactuar con otros sistemas.
 - **Aplicación al sistema:**

- Actualmente no se observa integración con otros sistemas externos, pero sería útil considerar la posibilidad de conectarse a sistemas de facturación o gestión de ventas.

- **Seguridad**

- **Definición:** La capacidad del software para proteger la información y los datos.
- **Aplicación al sistema:**
 - Actualmente, la seguridad no es evidente en la interfaz observada. Se recomienda implementar autenticación de usuarios y roles para gestionar el acceso al sistema.

20.2 Esquema especificación de Interfaz

ID	Nombre	Categoría	Cantidad	Precio
1	Nike	Zapatos	10	0.00
2	Manga larga	Camisas	10	0.00
3	Clasicos	Pantalones	10	0.00

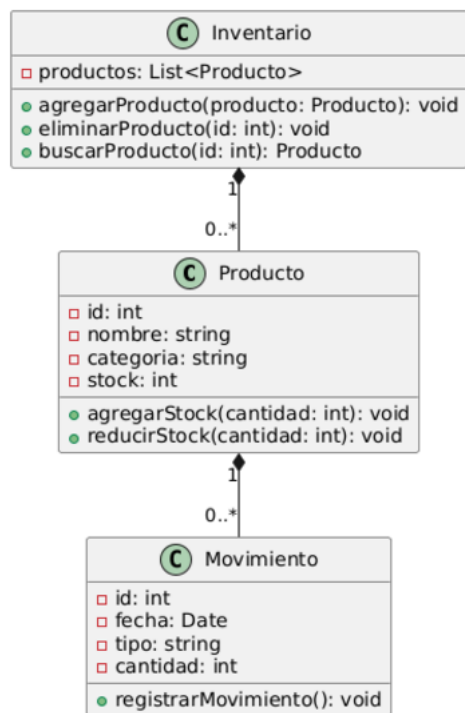
- **Área 1: Menú Principal**

- Ubicado en la parte superior, el menú incluye dos opciones:
 - **Productos:** Permite gestionar el inventario de productos.
 - **Movimientos:** Posiblemente para gestionar las entradas y salidas de inventario.

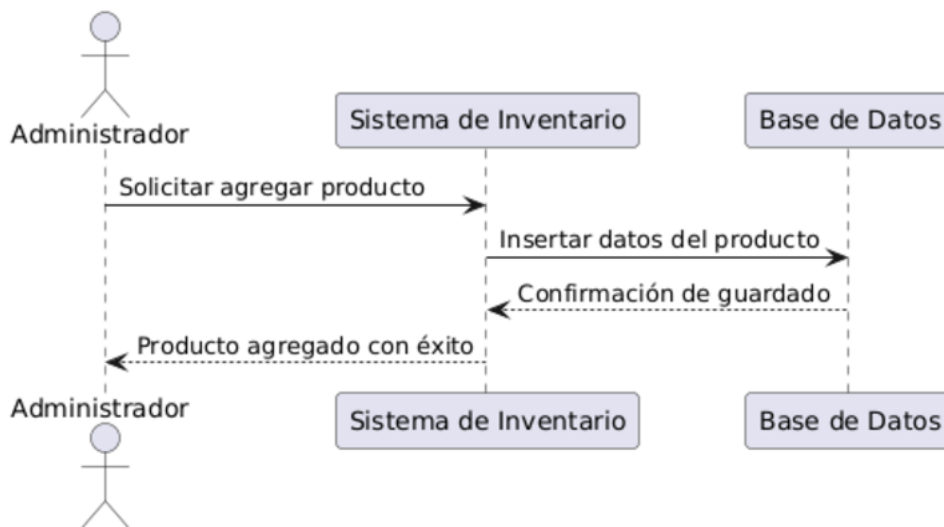
- **Área 2: Barra de Herramientas**
- Incluye botones funcionales para la gestión de productos:
 - **Agregar Producto**
 - **Eliminar Producto**
 - **Actualizar Lista**
- **Área 3: Imagen Corporativa**
- No se observan elementos gráficos corporativos en la interfaz actual, pero este espacio podría incluir un logo en versiones futuras.
- **Área 4: Título de Ventana**
- El título "Sistema de Gestión de Inventario" indica claramente el propósito del software, mostrando el contexto actual.
- **Área 5: Despliegue e Ingreso de Datos**
- **Campos de entrada:**
 - **Nombre del Producto:** Campo para ingresar el nombre.
 - **Categoría:** Menú desplegable para seleccionar la categoría del producto.
- **Botones asociados:**
 - **Agregar Producto** para añadir un nuevo ítem al inventario.
 - **Eliminar Producto** para remover un producto seleccionado.
 - **Actualizar Lista** para refrescar la tabla de inventario.

21 DIAGRAMAS

21.1 Diagrama de Clases



21.2 Diagrama de Secuencia



21.3 Diagrama de Casos de Uso

