**Documentación proyecto POO**

**Gestor de tienda de componentes electrónicos**

**Estudiante:**

**Andres Camilo Muñoz Mejía T00064244**

**Profesor:**

**Yuranis Henriquez Nuñez**

**29/05/2025**

**Objetivo del proyecto:**

Desarrollar una aplicación para la gestión de una tienda de componentes electrónicos utilizando principios de Programación Orientada a Objetos (POO).

**Descripción general:**

Este proyecto tiene como objetivo desarrollar una aplicación que permita gestionar el inventario de una tienda de componentes electrónicos. Se aplicarán los principios de Programación Orientada a Objetos (POO) para facilitar la gestión de productos, control de stock y registro de ventas.

**Análisis de Requisitos**

Para el análisis de requisitos se pensó en cuales serían los requisitos principales que debería tener el sistema

**Requerimientos Funcionales**

Estos requerimientos definen las funciones y características que el sistema debe cumplir:

1. **Gestión de Inventario**
   * Permitir el registro de nuevos productos con sus atributos (ID, nombre, categoría, marca, precio, stock, descripción, fecha de ingreso).
   * Posibilitar la eliminación de productos del inventario.
   * Actualizar automáticamente el stock tras cada venta.
   * Mostrar la lista de productos disponibles con opciones de búsqueda y filtrado.
2. **Gestión de Ventas**
   * Registrar las ventas realizadas, asociando productos y cantidades vendidas.
   * Calcular el total de la venta automáticamente.
   * Generar recibos detallados con los productos comprados, cantidades, precios y total de la compra.
3. **Control de Acceso**
   * Implementar un sistema de autenticación para el gerente mediante una contraseña.
   * Permitir la recuperación o cambio de contraseña en caso de olvido.
4. **Generación de Reportes**
   * Permitir al gerente generar informes de ventas en un período determinado.
   * Mostrar ganancias totales y productos más vendidos.

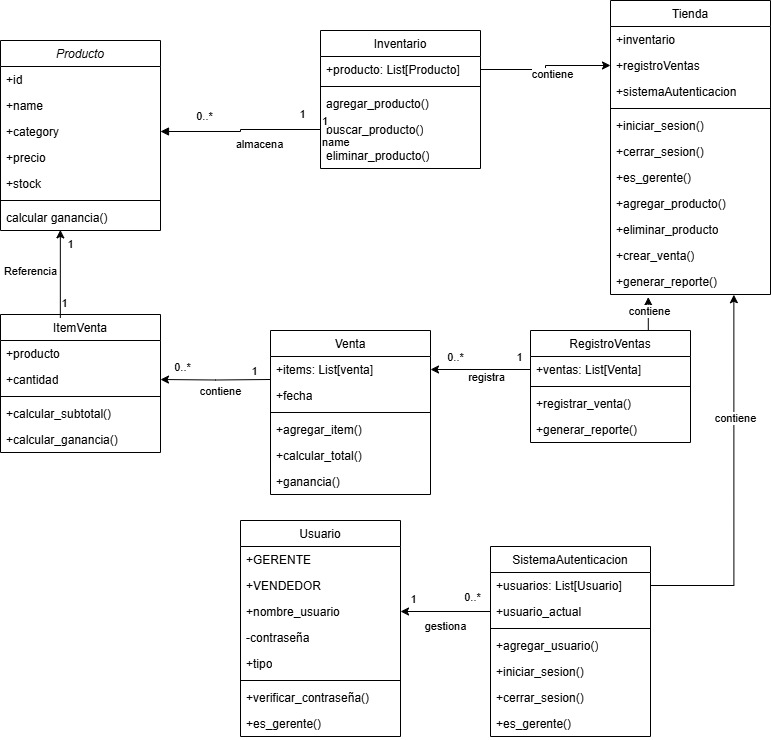
**Requerimientos No Funcionales**

Estos describen características de calidad y restricciones del sistema:

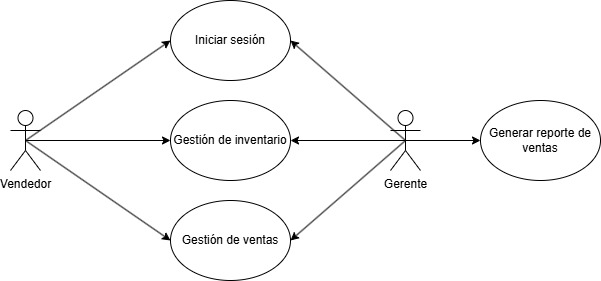
1. **Usabilidad**
   * La interfaz debe ser intuitiva y fácil de usar, con menús organizados y opciones accesibles.
2. **Seguridad**
   * Implementar medidas para proteger la información del inventario y ventas.
   * Garantizar que solo el gerente tenga acceso a funciones administrativas.
3. **Eficiencia y Rendimiento**
   * La aplicación debe responder de manera rápida a las operaciones de inventario y ventas.
   * El sistema debe manejar grandes volúmenes de productos sin degradar el rendimiento.
4. **Escalabilidad**
   * La aplicación debe poder expandirse para soportar más productos y funcionalidades en el futuro.
5. **Compatibilidad**
   * Debe poder ejecutarse en diferentes dispositivos o plataformas según la implementación (PC, móvil o web).
6. **Mantenimiento**
   * El código debe estar bien documentado y estructurado para facilitar futuras mejoras o correcciones.

**Diseño del sistema**

**Diagrama de clases**

****

**Diagrama de casos de uso**

****

**Implementación**

El código se trabajó en tres archivos para poder trabajar de forma más controlada, de modo que cada archivo trabaja un aspecto concreto:

**producto.py:** se encarga del apartado de las clases y metodos de productos, clientes y ventas

**gestora\_ventas.py:** se encarga de las operaciones que se hacen con los productos, clientes y ventas, y se encarga de gestionar los archivos Excel donde se almacenan los mismos.

**main.py:** se encarga de implementar la interfaz gráfica.

Las clases y métodos principales que se utilizaron fueron las siguientes:

**Clase**Producto

Representa un producto en el sistema.

* **Atributos**:
  + id: Identificador único del producto.
  + nombre: Nombre del producto.
  + categoria: Categoría del producto.
  + precio: Precio del producto.
  + stock: Cantidad disponible en inventario.
* **Métodos**:
  + actualizar\_stock(cantidad): Modifica el stock del producto, asegurándose de que no sea negativo.
  + \_\_str\_\_(): Devuelve una representación en cadena del producto.

**Clase**Cliente

Representa un cliente en el sistema.

* **Atributos**:
  + id: Identificador único del cliente.
  + nombre: Nombre del cliente.
  + email: Correo electrónico del cliente.
* **Métodos**:
  + \_\_str\_\_(): Devuelve una representación en cadena del cliente.

**Clase**Venta

Representa una venta realizada en el sistema.

* **Atributos**:
  + id: Identificador único de la venta.
  + cliente: Cliente asociado a la venta.
  + productos: Lista de tuplas con productos y cantidades vendidas.
  + fecha: Fecha de la venta (por defecto, la fecha actual).
* **Métodos**:
  + calcular\_total(): Calcula el total de la venta sumando el precio de los productos por su cantidad.
  + \_\_str\_\_(): Devuelve una representación en cadena de la venta, incluyendo los productos y el total.

**Clase**GestoraVentas

Gestiona las operaciones de ventas, productos y clientes.

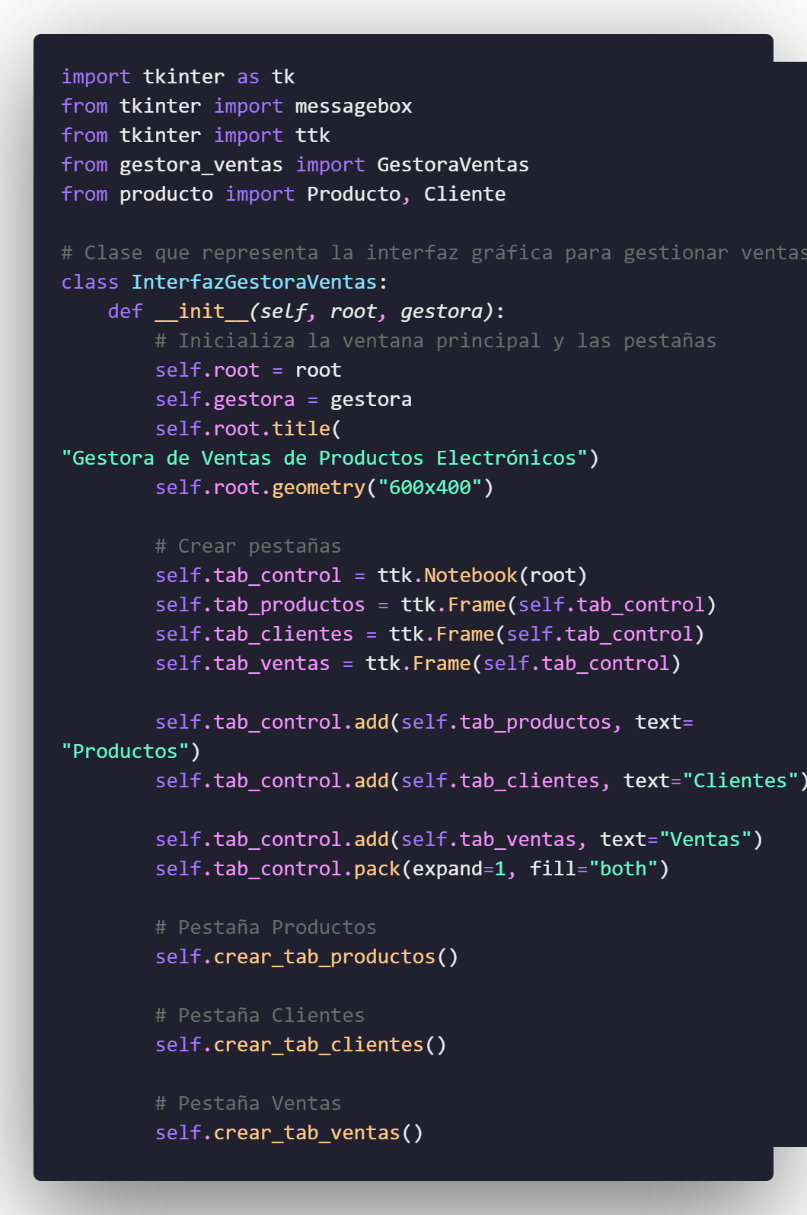
* **Atributos**:
  + productos\_file: Archivo Excel para almacenar productos.
  + clientes\_file: Archivo Excel para almacenar clientes.
  + ventas\_file: Archivo Excel para almacenar ventas.
  + productos: Lista de objetos Producto.
  + clientes: Lista de objetos Cliente.
  + ventas: Lista de objetos Venta.
* **Métodos**:
  + cargar\_datos(): Carga los datos desde los archivos Excel.
  + guardar\_datos(): Guarda los datos en los archivos Excel.
  + agregar\_producto(producto): Agrega un nuevo producto al sistema.
  + agregar\_cliente(cliente): Agrega un nuevo cliente al sistema.
  + realizar\_venta(cliente\_id, productos\_vendidos): Realiza una nueva venta, actualizando el stock de los productos.
  + crear\_producto(id, nombre, categoria, precio, stock): Crea una instancia de Producto.
  + crear\_cliente(id, nombre, email): Crea una instancia de Cliente.
  + crear\_venta(id, cliente, productos, fecha): Crea una instancia de Venta.

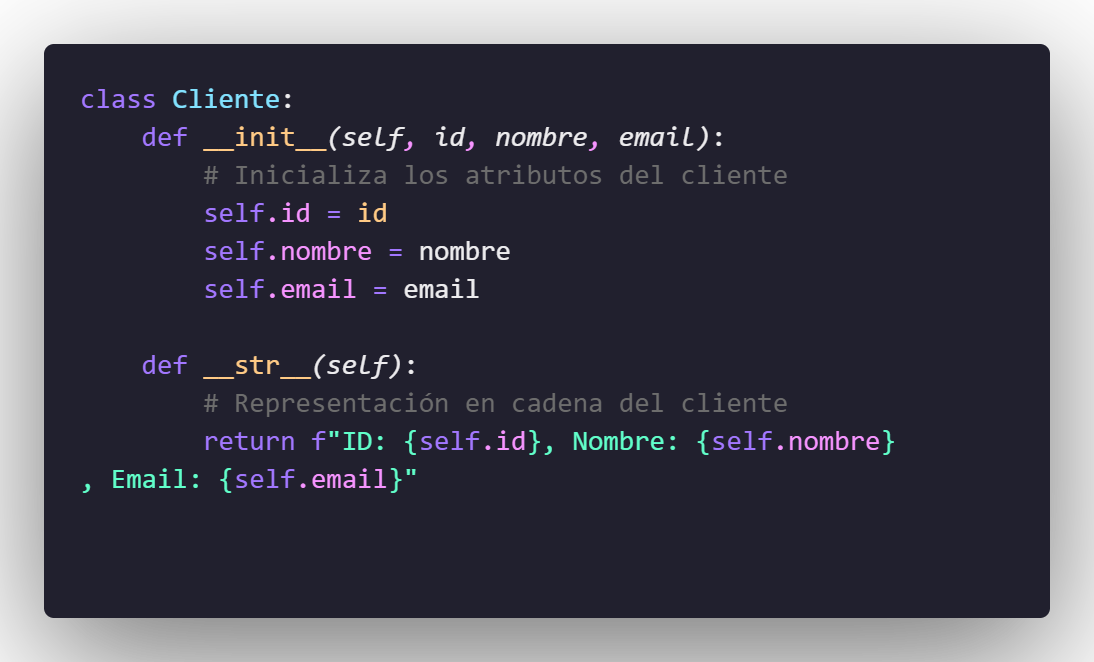
**Clase**InterfazGestoraVentas

Representa la interfaz gráfica para gestionar ventas.

* **Atributos**:
  + root: Ventana principal de la aplicación.
  + gestora: Instancia de GestoraVentas.
* **Métodos**:
  + crear\_tab\_productos(): Configura la pestaña de gestión de productos.
  + crear\_tab\_clientes(): Configura la pestaña de gestión de clientes.
  + crear\_tab\_ventas(): Configura la pestaña de gestión de ventas.
  + actualizar\_lista\_productos(): Actualiza la lista de productos en la interfaz.
  + actualizar\_lista\_clientes(): Actualiza la lista de clientes en la interfaz.
  + actualizar\_lista\_ventas(): Actualiza la lista de ventas en la interfaz.
  + agregar\_producto(): Agrega un nuevo producto al sistema.
  + agregar\_cliente(): Agrega un nuevo cliente al sistema.
  + realizar\_venta(): Realiza una nueva venta en el sistema.

**Fotos de algunas secciones del código**





**Resultados**

Los resultados no fueron los esperados, si se llegó a un producto funcional, pero no se alcanzaron todos los objetivos, por ejemplo, no fue posible implementar la función de autenticación del usuario ni la generación de reporte, no obstante, si fue posible implementar las funciones básicas, como la gestión del inventario, clientes y ventas.

Los principales problemas encontrados al momento del desarrollo del proyecto fueron el tiempo, el cual debido a factores externos a la materia no fue el requerido para la implementación de todas las features, además de la misma complejidad del proyecto, que si bien fue sencillo hubo aspectos que se dificultaron más de lo esperado, como por ejemplo el desarrollo de la interfaz

**Aspectos técnicos**

**Tecnologías utilizadas:**

* Python: como único lenguaje para el desarrollo del proyecto
* Draw.io: para la creación de diagramas
* Visual Studio Code: como editor de código
* GitHub: para el alojamiento del proyecto
* Git: para registrar los cambios en el tiempo
* Pandas: librería utilizada para la gestión de los archivos Excel
* Tkinter: librería utilizada para el desarrollo de la interfaz
* Datatime: librería utilizada para obtener la fecha y hora
* GitHub copilot: como IA principal para ayudar en la solución de problemas
* ClaudeIA: como IA para consultas rápidas

**Bibliografía:**

* [**https://docs.python.org/3/**](https://docs.python.org/3/)
* [**https://docs.python.org/es/3.13/library/tkinter.html**](https://docs.python.org/es/3.13/library/tkinter.html)
* [**https://pandas.pydata.org/docs/**](https://pandas.pydata.org/docs/)
* [**https://github.com/AndresM2003/Proyecto\_POO\_Andres\_T00064244**](https://github.com/AndresM2003/Proyecto_POO_Andres_T00064244)
* [**https://refactoring.guru/es/design-patterns**](https://refactoring.guru/es/design-patterns)
* [**https://www.youtube.com/watch?v=TQgB9JFbui0**](https://www.youtube.com/watch?v=TQgB9JFbui0)