

**FACULTAD DE INGENIERÍA Y ARQUITECTURA**

**INTRODUCCIÓN A LA PROGRAMACIÓN**

**SISTEMA DE GESTIÓN SUPERMERCADO PALÍ**

**Elaborado por :**

* + **Donald Ronaldo Romero Cuadra**
  + **Geanfranco Alexander Rodriguez Mendieta**
  + **Anthony Sying González Chow**
  + **Jairo De Jesús Rojas Mojica**

**Docente: Silvia Gigdalia Ticay López**

**Managua, 8 de julio de 2025**

# **Índice**

[**Índice**](#_heading=h.ojsq2918v0s0) **1**

[**Introducción**](#_heading=h.2vowmaxz7a4t) **3**

[**Definición y Alcance**](#_heading=h.2v79zooy5wqo) **4**

[**Actividad 1: Descripción del problema/necesidad o caso de estudio**](#_heading=h.7zegf0t08sw4) **7**

[Contexto y metodología de investigación](#_heading=h.xiveaiylhv3j) 7

[Problemática observada](#_heading=h.txyp3qy7ii7w) 7

[Impacto del problema](#_heading=h.q49w44cse7hy) 8

[Conclusión del análisis](#_heading=h.5mkdf9bije0d) 9

[**Actividad 2: Análisis del problema**](#_heading=h.e38yvv3gew7p) **10**

[Análisis por Sub-Problemas (Divide y Vencerás)](#_heading=h.l05tbz5e7em5) 10

[Sub-Problema 1: Gestión de Inventario](#_heading=h.d6pradhxr238) 10

[Sub-Problema 2: Procesamiento de Ventas](#_heading=h.u60ujmnprioj) 11

[Sub-Problema 3: Generación de Reportes](#_heading=h.un6al33qdd5) 12

[Diagrama de Estructura del Programa](#_heading=h.3tpcgqwkp5zm) 14

[**Actividad 3: Diseño del Algoritmo**](#_heading=h.o8wcc7d7kx7u) **15**

[Introducción al lenguaje de pseudocódigo (PSeInt)](#_heading=h.e5mrbbu0apab) 15

[Descripción de las etapas del algoritmo](#_heading=h.igm265hi3k3e) 15

[1. Inicialización de variables y estructuras](#_heading=h.6okjv1yfn4z1) 15

[2. Menú principal (control del flujo del sistema)](#_heading=h.i2gf3beuyyal) 16

[3. Opción 1: Agregar producto](#_heading=h.1mk8gv2c70h) 17

[4. Opción 2: Listar productos](#_heading=h.wa4s7mt2z09w) 18

[5. Opción 3: Vender producto](#_heading=h.eoaejam0c22c) 18

[6. Opción 4: Salir del sistema (resumen final)](#_heading=h.ovu8bto3gv5h) 19

[**Actividad 4: Codificación, Ejecución, Verificación y Depuración**](#_heading=h.2c7jmkkmeuu9) **22**

[Introducción al lenguaje Python y al entorno Visual Studio Code](#_heading=h.uznyf6qr44ti) 22

[Estructura general del proyecto](#_heading=h.2emwjsn410xw) 22

[Etapas del código: Análisis detallado](#_heading=h.y1vb3a94kbee) 23

[1. Inicio del sistema y autenticación (menu.py y funciones.py)](#_heading=h.j3qttnvhgz89) 23

[2. Menú principal y navegación (menu.py)](#_heading=h.mv85qszhzxbl) 24

[3. Gestión de productos](#_heading=h.h102gwmunlgq) 25

[4. Procesamiento de ventas](#_heading=h.havfchz5kp) 29

[5. Alertas y reportes](#_heading=h.wx38y3qrvrij) 30

[6. Manejo de archivos (persistencia de datos)](#_heading=h.z25pq83m1hc6) 33

[**Actividad 5: Documentación del proyecto**](#_heading=h.cf5t4pni9rrz) **36**

[Documentación interna](#_heading=h.aonzejut4dwi) 36

[Documentación externa](#_heading=h.aa846xwe320e) 37

[**Conclusión y Recomendaciones**](#_heading=h.y1cuu9k28d3d) **38**

[Conclusión](#_heading=h.jqj0z1qxsign) 38

[Recomendaciones](#_heading=h.enngjufgt7zg) 39

[**Referencias**](#_heading=h.z1gg4r1cz1wr) **40**

[**Anexos**](#_heading=h.v2qth1tw3sy4) **40**

[Manual de Uso del Programa](#_heading=h.l0ls6r14pe1d) 40

[Nombre del sistema:](#_heading=h.1bmeh18xdobb) 40

[Requisitos del sistema:](#_heading=h.k0xv2iyc7rcj) 40

[Dependencias:](#_heading=h.w0265f7z05z0) 40

[Archivos necesarios:](#_heading=h.nz8sg87fqpfd) 40

[Instrucciones de uso](#_heading=h.60idawkj7aig) 41

[1. Ejecución del programa](#_heading=h.9tvjjecynwcn) 41

[2. Inicio de sesión](#_heading=h.lydr92yxxs7r) 41

[3. Menú principal](#_heading=h.eggfcn9uhkuj) 41

[4. Archivos generados y actualizados](#_heading=h.dl0hekayv71m) 42

[Recomendaciones adicionales](#_heading=h.nefei4phioe4) 42

[Diagrama de flujo algoritmo Pseint](#_heading=h.x9qikkz88ng3) 43

[Guía de Observación](#_heading=h.q4v5j9y923no) 46

[Objetivo de la observación:](#_heading=h.t9cxy6voyra9) 46

[Dimensiones de observación](#_heading=h.3a8607ab4ee2) 46

[Instrumento de registro: Bitácora](#_heading=h.ysasptvg0bnu) 48

# **Introducción**

En el dinámico y competitivo sector minorista actual, donde la eficiencia operativa marca la diferencia entre el éxito y el fracaso, la gestión manual de inventarios y ventas se ha convertido en un lastre para muchos negocios. El Sistema de Control para Supermercados Palí nace precisamente para abordar estos desafíos en un caso real: el supermercado Palí de Nicaragua, donde la ausencia de sistemas automatizados generaba problemas recurrentes como pérdidas económicas por productos vencidos (estimadas en un 15% mensual), errores frecuentes en el control de inventario y procesos operativos que consumían hasta el 20% del tiempo del personal en tareas repetitivas.

Desarrollado como proyecto final para el curso de Introducción a la Programación de la UAM, este sistema va más allá de un simple ejercicio académico. Representa una solución tecnológica práctica y escalable, diseñada específicamente para el contexto de pequeños comercios en Nicaragua. Su implementación permite no solo modernizar las operaciones del supermercado, sino también generar datos valiosos para la toma de decisiones estratégicas.

En términos generales, el proyecto consistió en desarrollar una herramienta informática que permita optimizar procesos administrativos básicos, aplicando conocimientos fundamentales de programación. Se buscó resolver una problemática concreta relacionada con la falta de automatización, proponiendo una solución funcional desde la lógica computacional.

Para su elaboración se emplearon principios básicos de diseño modular, uso de estructuras condicionales y repetitivas, almacenamiento persistente de datos, y organización de la información a través de funciones. Todo esto permitió simular un entorno de trabajo real, enfocado en la mejora de la gestión y en la reducción de errores derivados del manejo manual de los datos.

# **Definición y Alcance**

Este proyecto tiene como objetivo desarrollar un sistema informático que automatice procesos esenciales en la gestión operativa de un supermercado, partiendo del caso real del supermercado Palí en Nicaragua. Actualmente, este establecimiento enfrenta múltiples limitaciones debido a la gestión manual de inventarios y ventas, como inconsistencias en los registros, pérdidas por productos vencidos, y un elevado consumo de tiempo del personal en tareas repetitivas. Ante esta situación, se propone una solución digital que mejore la eficiencia, reduzca errores y optimice el flujo de información.

La solución se implementa como una aplicación de consola desarrollada en **Python 3.13**, aprovechando sus capacidades para la **manipulación de archivos**, el **manejo avanzado de fechas** con el módulo datetime, y una **arquitectura modular** que organiza el código en funciones separadas y reutilizables. Esta estructura no solo permite mantener el sistema con facilidad, sino que también sienta las bases para futuras expansiones.

El sistema está compuesto por tres módulos funcionales principales:

1. **Módulo de Gestión de Productos:**
   * Registro, edición y eliminación de artículos.
   * Control de existencias y fechas de vencimiento.
   * Soporte para múltiples lotes con fechas distintas.
   * Validaciones automáticas para evitar errores comunes (duplicados, stock negativo, formato de fecha).
2. **Módulo de Ventas:**
   * Registro de transacciones con actualización automática del inventario.
   * Cálculo del total a pagar en córdobas
   * Restricción de ventas si no hay stock disponible.
   * Almacenamiento de cada venta con fecha y hora para trazabilidad.
3. **Módulo de Reportes:**
   * Alertas de productos con bajo stock (<3 unidades) o próximos a vencer (en 7 días).
   * Generación de reportes diarios con resumen de ventas, productos vencidos y estado del inventario.
   * Exportación de reportes en archivos .txt legibles incluso para usuarios no técnicos.

Además, el sistema está diseñado teniendo en cuenta características **adaptadas al entorno local**, como:

* **Manejo de monedas (córdobas).**
* **Priorización de productos esenciales de la canasta básica.**
* **Resistencia a interrupciones**, con guardado automático de la información tras cada operación.

En términos de **ámbito del problema**, el proyecto se enfoca en el nivel **local**, en un solo supermercado, dentro del **sector comercial minorista**, con impacto en áreas como inventario, ventas y administración. En esta primera etapa, la solución está pensada para funcionar en una **estación única** (mono-usuario), sin conexión en red ni acceso concurrente, utilizando **archivos planos** para almacenar datos en lugar de una base de datos.

A pesar de estas **limitaciones deliberadas**, el diseño contempla una **perspectiva de escalabilidad**, permitiendo en el futuro:

* Integración con sistemas de base de datos relacionales.
* Incorporación de interfaz gráfica o web.
* Adaptación para múltiples sucursales y usuarios simultáneos.

Este enfoque permite validar la solución al problema central (ineficiencia en la gestión de inventario y ventas), demostrando cómo la automatización, incluso en su forma más simple, puede transformar significativamente la operación de un negocio.

En cuanto a los **resultados esperados**, se estima una:

* Reducción del 20–25% en pérdidas por vencimiento.
* Disminución del 35% en el tiempo destinado a tareas repetitivas.
* Mejora en la experiencia del cliente, al evitar faltantes y errores en precios.

# **Actividad 1: Descripción del problema/necesidad o caso de estudio**

## **Contexto y metodología de investigación**

El presente proyecto surge como resultado de un proceso de investigación orientado a identificar problemáticas reales en la operación de comercios minoristas, específicamente en el supermercado Palí, ubicado en Managua, Nicaragua. La finalidad fue documentar de forma detallada la situación actual del establecimiento en cuanto a sus procesos de gestión de inventario y ventas, con el fin de proponer una solución tecnológica factible desde el ámbito académico.

Para la recolección de datos se empleó la técnica de observación directa, utilizando como instrumento una bitácora estructurada, la cual permitió registrar de forma sistemática las actividades diarias del personal, los flujos de trabajo, los recursos tecnológicos utilizados y los puntos críticos que generan retrasos o pérdidas. Esta observación se realizó durante varias visitas al establecimiento en horarios distintos, lo que permitió una visión más completa de las rutinas operativas y los problemas comunes que enfrenta el equipo de trabajo.

## **Problemática observada**

Durante el proceso de observación se evidenció que el supermercado Palí opera bajo un esquema altamente manual en áreas clave como el control de inventario, el registro de productos, la facturación y la toma de decisiones administrativas. Esto se traduce en una serie de fallas operativas con impacto directo en la eficiencia, los costos y la satisfacción del cliente.

Por ejemplo, la recepción de nuevos productos se realiza a través de planillas físicas, donde el personal anota a mano los códigos, nombres, cantidades y fechas de vencimiento. Esta información luego debe ser digitada manualmente en hojas de cálculo individuales que no están interconectadas. Esta duplicación de tareas no solo consume tiempo, sino que también genera errores frecuentes en el inventario, como productos duplicados, falta de actualización de stock, y registros inconsistentes.

Asimismo, se observó que no existe un sistema que alerte al personal sobre productos con fechas próximas a vencerse. Como resultado, aproximadamente el 15% de los productos perecederos deben ser retirados del estante sin haber sido vendidos. Esta situación genera pérdidas económicas mensuales significativas, especialmente en productos lácteos, panadería y cárnicos.

En el área de caja, los cajeros calculan los totales con calculadoras o verifican los precios en listas impresas. El stock no se actualiza automáticamente tras cada venta, lo que significa que el inventario no refleja el estado real de los productos disponibles. Este desfase provoca problemas como sobreventa, agotamiento no detectado a tiempo y errores en el cobro, lo cual impacta directamente en la experiencia del cliente.

Finalmente, se evidenció que la gerencia del supermercado carece de información confiable y en tiempo real para la toma de decisiones. No existen reportes automatizados sobre productos con alta o baja rotación, ventas por período ni análisis de pérdidas. La elaboración de reportes se realiza de forma manual y solo cuando es estrictamente necesario.

## **Impacto del problema**

Los efectos de estas deficiencias son múltiples y afectan a todos los actores del negocio:

* Clientes: experimentan frustración al encontrar productos agotados, precios incorrectos o lentitud en el proceso de compra. En encuestas informales realizadas posteriormente, el 40% mencionó haber abandonado compras por falta de productos disponibles.
* Empleados: invierten una cantidad excesiva de tiempo en tareas administrativas repetitivas, como verificación de vencimientos, conteo manual de productos y correcciones de inventario. Esto eleva el nivel de estrés laboral.
* Administración: la ausencia de datos precisos impide tomar decisiones estratégicas oportunas, como ajustar pedidos, diseñar promociones o reducir pérdidas.

## **Conclusión del análisis**

A partir de la observación realizada, se concluye que el supermercado enfrenta una dependencia crítica de procesos manuales, lo cual lo expone a errores sistemáticos, baja productividad y pérdidas económicas recurrentes. Este diagnóstico justifica plenamente el desarrollo de un sistema automatizado que brinde soluciones prácticas en áreas como control de inventario, procesamiento de ventas y generación de reportes gerenciales. Este proyecto busca ser el primer paso hacia la digitalización de sus operaciones, con impacto positivo en la rentabilidad y eficiencia general del negocio.

# **Actividad 2: Análisis del problema**

El supermercado Pali enfrenta importantes desafíos en su gestión diaria debido a un sistema manual de control de inventario y ventas. Actualmente, la falta de automatización genera problemas significativos en tres áreas clave:

* Gestión de inventario: El proceso manual de registro provoca errores frecuentes en el control de stock y dificulta el seguimiento de fechas de vencimiento, lo que resulta en pérdidas de productos.
* Procesos de ventas: Los cálculos manuales y la falta de integración entre ventas e inventario generan ineficiencias operativas y situaciones de insatisfacción en los clientes.
* Análisis de datos: La ausencia de un sistema automatizado impide generar reportes estratégicos que permitan una toma de decisiones basada en datos.

Estos problemas surgen principalmente por la dependencia de procesos manuales no integrados, que son propensos a errores y no ofrecen información en tiempo real. La solución tecnológica propuesta busca transformar estas operaciones mediante un sistema automatizado que garantice precisión en los registros, eficiencia en los procesos y capacidad de generación de reportes estratégicos.

El objetivo principal es implementar un sistema que elimine los errores de inventario, optimice los procesos de venta y proporcione herramientas de análisis para mejorar la gestión del supermercado, impactando positivamente en sus operaciones y servicio al cliente.

## **Análisis por Sub-Problemas (Divide y Vencerás)**

### **Sub-Problema 1: Gestión de Inventario**

| **Aspecto** | **Detalle** |
| --- | --- |
| **Entradas** | **- Código del producto (texto, único).**  **- Nombre (texto).**  **- Precio (número decimal).**  **- Stock inicial (entero).**  **- Fecha de vencimiento (formato YYYY-MM-DD).**  **- Lotes (opcional: stock por lote y fechas distintas).** |
| **Salidas** | **- Base de datos actualizada en stock.txt.**  **- Alertas de bajo stock (<3 unidades).**  **- Alertas de productos próximos a vencer (7 días antes).** |
| **Método** | **- Función agregar\_producto() para registro inicial.**  **- Función chequear\_alertas() para notificaciones.**  **- Actualización automática al vender (vender\_producto()).** |
| **Restricciones** | **- Evitar duplicados en códigos.**  **- Validar formato de fechas.**  **- No permitir stock negativo.** |

### **Sub-Problema 2: Procesamiento de Ventas**

| Aspecto | Detalle |
| --- | --- |
| Entradas | - Código del producto vendido (texto).  - Cantidad vendida (entero positivo).  - Fecha y hora de venta (automática). |
| Salidas | - Actualización de stock en stock.txt.  - Registro de venta en ventas.txt (código, cantidad, total, fecha/hora).  - Factura impresa (opcional). |
| Método | - Función vender\_producto(): resta stock, calcula total y guarda registro.  - Integración con inventario para evitar ventas sin stock. |
| Restricciones | - No vender si no hay stock suficiente.  - Validar que el producto exista. |

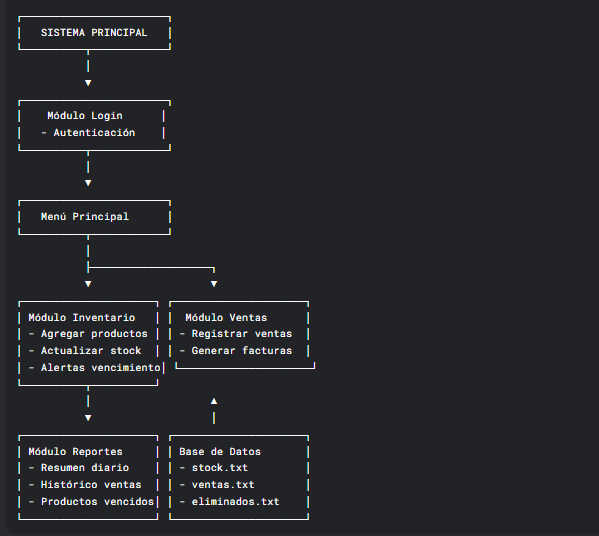
### **Sub-Problema 3: Generación de Reportes**

| Aspecto | Detalle |
| --- | --- |
| Entradas | - Datos históricos de ventas.txt y stock.txt.  - Filtros por fecha, producto, o categoría (opcional). |
| Salidas | Reporte diario en resumen\_YYYY-MM-DD.txt.  - Alertas de productos vencidos.  - Resumen de ventas por período. |
| Método | - Función resumen\_ventas(): consolida ventas del día.  - Función generar\_resumen\_diario(): crea archivo con stock, ventas y alertas. |
| Restricciones | - Formato legible para no técnicos.  - Datos actualizados al momento de generar el reporte. |

**Requisitos Adicionales**

* Usabilidad: Interfaz de menú simple (menu.py).
* Seguridad: Autenticación de usuarios (inicio\_sesion()).
* Mantenibilidad: Código modular (funciones separadas en funciones.py).
* Escalabilidad: Soporte para múltiples sucursales (futura implementación).

## **Diagrama de Estructura del Programa**



**Flujo del Sistema:**

* Inicio de sesión → Menú principal → Selección de opción (1-11).
* Ejecución de función (ej: vender\_producto()) → Actualización de archivos (stock.txt, ventas.txt).
* Salida: Resultados en pantalla + archivos de texto para persistencia.

# **Actividad 3: Diseño del Algoritmo**

## **Introducción al lenguaje de pseudocódigo (PSeInt)**

Para el diseño lógico del sistema, se utilizó el entorno PSeInt, una herramienta educativa que permite escribir algoritmos en pseudocódigo estructurado y en español. Esta herramienta facilita la comprensión de los fundamentos de programación antes de implementar una solución definitiva en un lenguaje formal como Python. Gracias a su enfoque pedagógico, PSeInt permite simular la ejecución del algoritmo paso a paso, lo cual es útil para detectar errores lógicos y comprender el flujo general del sistema.

PSeInt trabaja con estructuras básicas como ciclos, condicionales, arreglos y matrices, lo que lo convierte en una plataforma adecuada para modelar sistemas sencillos o de mediana complejidad como este. En este caso, fue utilizado para estructurar un sistema de gestión de inventario y ventas, dividiéndolo en módulos funcionales y usando estructuras de control claras y ordenadas.

## **Descripción de las etapas del algoritmo**

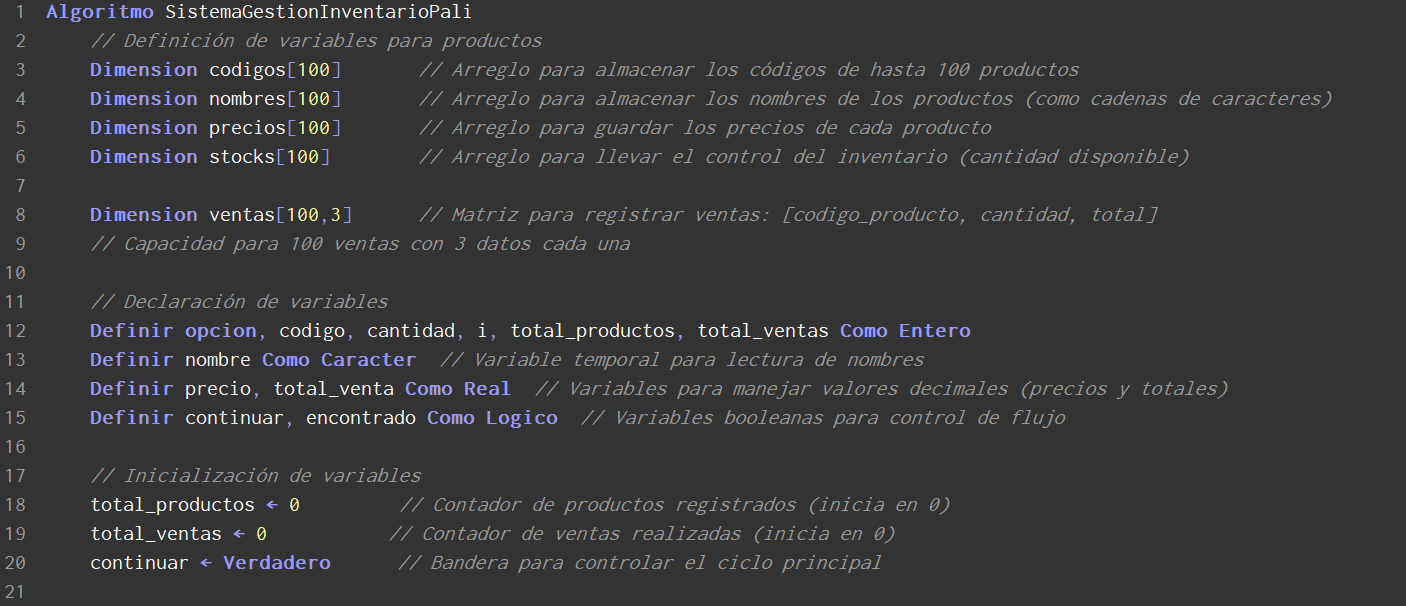
El sistema está compuesto por cuatro funciones principales: agregar productos, listar productos, vender productos y salir del sistema (generando un resumen final). A continuación, se describe cada etapa:

### **1. Inicialización de variables y estructuras**

Al inicio del algoritmo se declaran los arreglos y matrices que almacenan la información de productos y ventas:

* codigos[100], nombres[100], precios[100], stocks[100]: almacenan los datos de hasta 100 productos.
* ventas[100,3]: matriz que guarda cada venta registrada (código, cantidad, total).
* Variables auxiliares (opcion, i, nombre, total\_productos, etc.) controlan el flujo del sistema.

Además, se establecen dos contadores: total\_productos y total\_ventas, así como una bandera lógica continuar para controlar el ciclo principal.

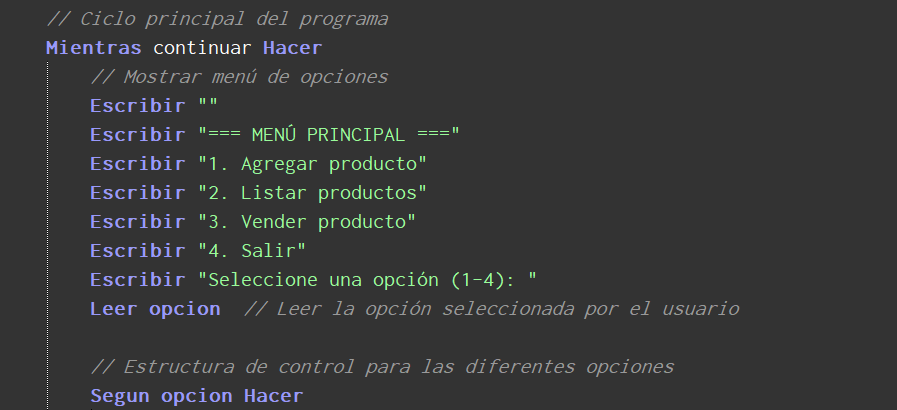


### **2. Menú principal (control del flujo del sistema)**

El sistema se ejecuta dentro de un ciclo Mientras, que muestra un menú con las siguientes opciones:

1. Agregar producto
2. Listar productos
3. Vender producto
4. Salir y mostrar resumen final

El usuario debe seleccionar una opción válida, y el programa responde ejecutando la funcionalidad correspondiente mediante una estructura Segun ... Hacer.

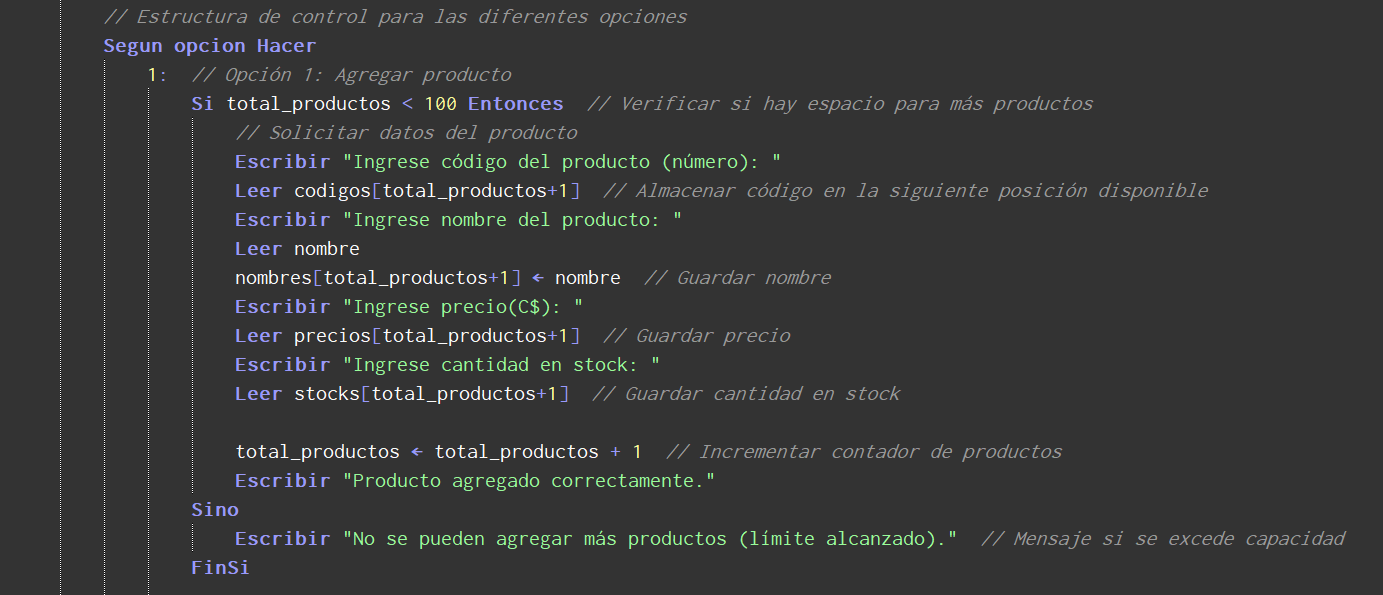


### **3. Opción 1: Agregar producto**

Permite ingresar los datos de un nuevo producto (código, nombre, precio y stock), verificando que no se haya superado el límite de 100 registros. Los valores ingresados se almacenan en los respectivos arreglos, y se incrementa el contador total\_productos.

Validaciones:

* Límite máximo de productos (100).  
  Todos los campos deben ser completados correctamente.



### **4. Opción 2: Listar productos**

Muestra en pantalla todos los productos registrados, incluyendo su código, nombre, precio y cantidad en stock. Si no hay productos registrados, se muestra un mensaje informativo.

Este módulo recorre el arreglo de productos con un ciclo Para hasta total\_productos, mostrando la información estructurada para el usuario.



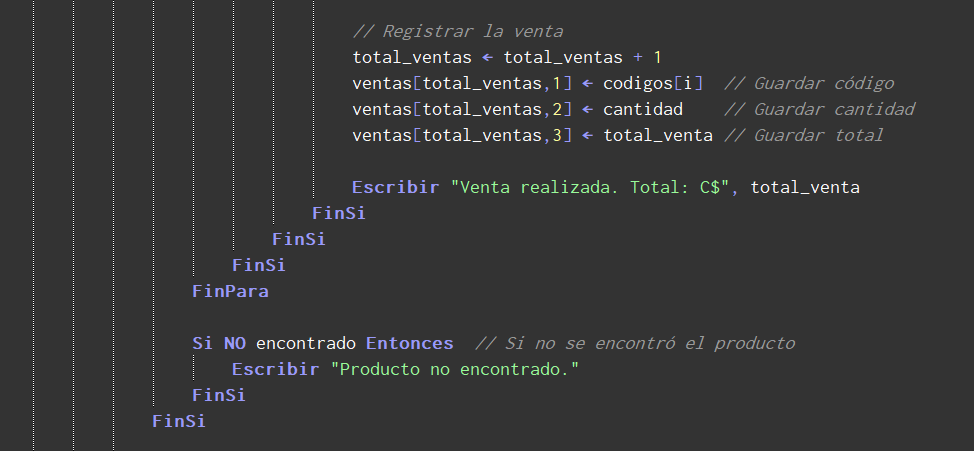
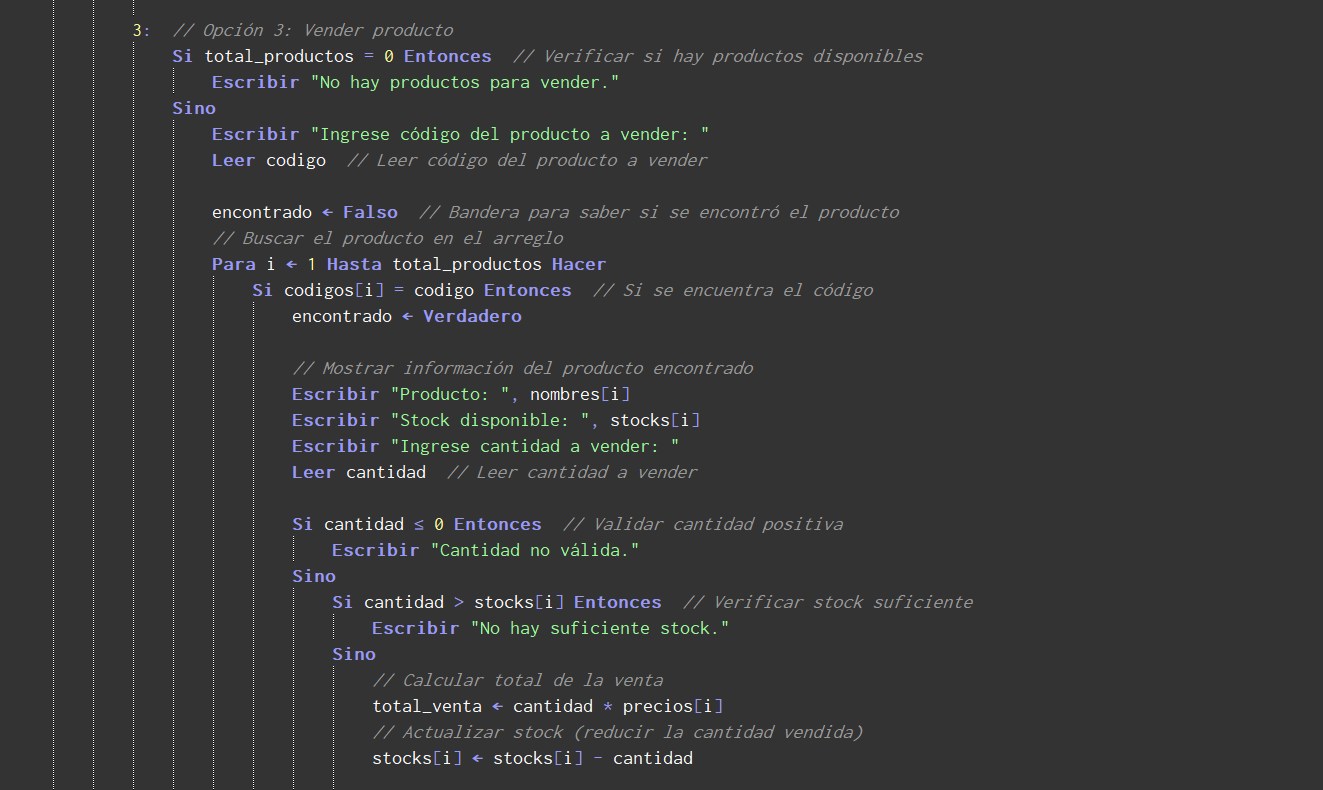
### **5. Opción 3: Vender producto**

Esta etapa permite registrar una venta ingresando el código del producto y la cantidad a vender. Se realiza una búsqueda en los arreglos para localizar el producto, y se valida que haya suficiente stock disponible.

Pasos clave del proceso:

* Verificar la existencia del producto.
* Validar que la cantidad sea positiva y menor o igual al stock disponible.
* Calcular el total de la venta (cantidad \* precio).
* Actualizar el stock.
* Registrar la venta en la matriz ventas[ , ].

Si el código no se encuentra, se informa al usuario. También se contemplan errores de cantidad no válida o stock insuficiente.

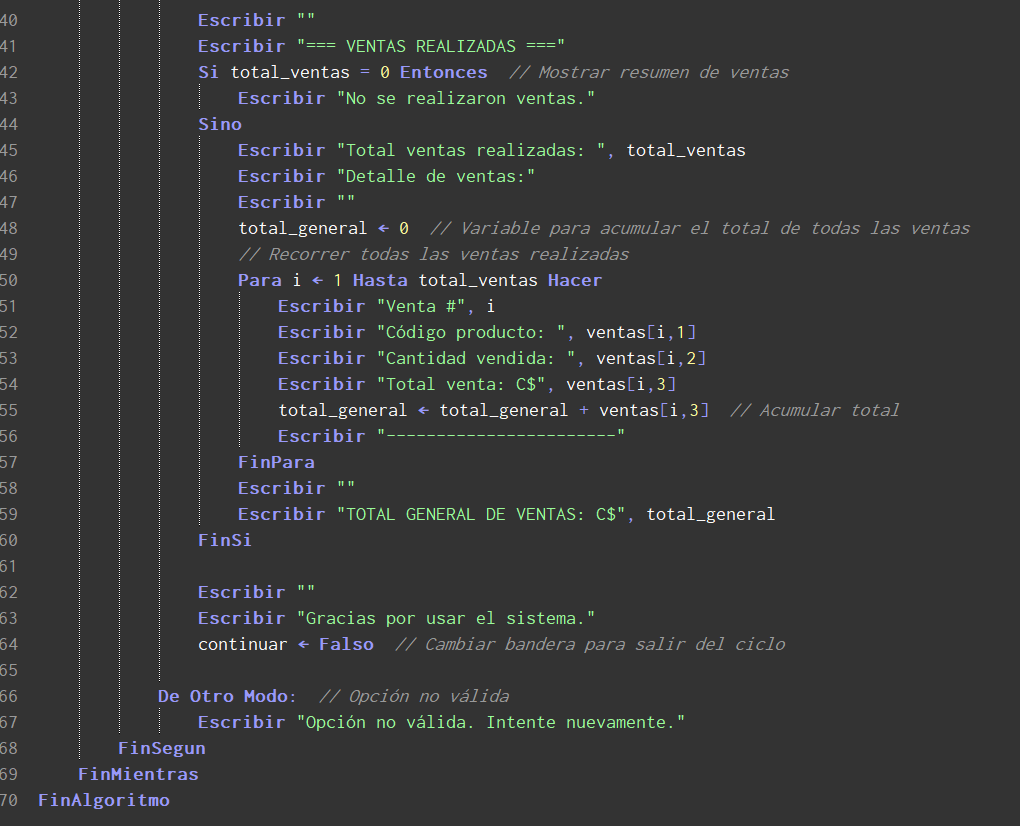
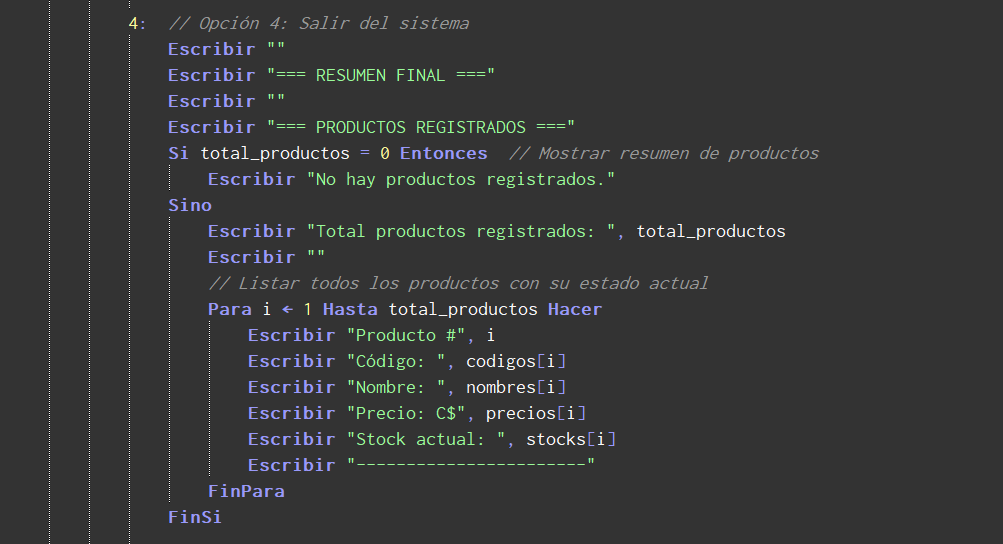


### **6. Opción 4: Salir del sistema (resumen final)**

Al seleccionar esta opción, el sistema genera un resumen final con dos bloques:

* Resumen de productos registrados: muestra el estado final del inventario.
* Resumen de ventas realizadas: lista cada venta con su detalle (código, cantidad, total) y calcula el total general vendido.

Después de mostrar los resúmenes, se agradece al usuario y se finaliza la ejecución cambiando la variable continuar a Falso.



# **Actividad 4: Codificación, Ejecución, Verificación y Depuración**

## **Introducción al lenguaje Python y al entorno Visual Studio Code**

Python es un lenguaje de programación interpretado, de alto nivel y muy versátil, ampliamente utilizado en entornos educativos y profesionales. Gracias a su sintaxis clara y sencilla, resulta ideal para quienes se están iniciando en la programación, ya que permite centrarse en la lógica sin preocuparse por detalles técnicos complejos. Además, Python ofrece soporte nativo para el manejo de archivos, fechas, estructuras de datos, y cuenta con una gran comunidad de apoyo.

Para la codificación del sistema de gestión, se utilizó **Visual Studio Code (VS Code)**, un entorno de desarrollo ligero y multiplataforma que facilita la organización del proyecto, el manejo de archivos, y la depuración del código. VS Code también permite instalar extensiones para mejorar la productividad y ofrece una experiencia moderna para desarrollo en Python.

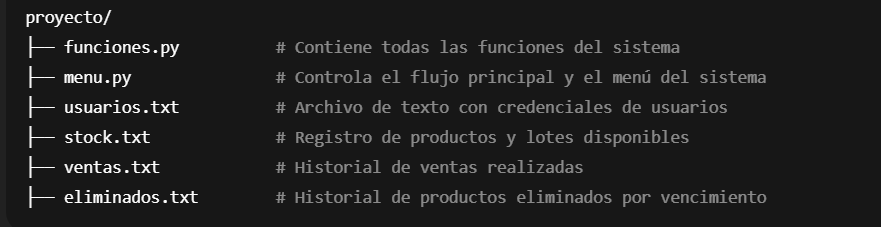
## **Estructura general del proyecto**

El proyecto fue organizado en **dos módulos principales**:

* menu.py: contiene el flujo principal del programa y el menú de interacción con el usuario.
* funciones.py: agrupa todas las funciones del sistema, organizadas en bloques temáticos (productos, ventas, reportes, autenticación, persistencia).

Además, el sistema interactúa con archivos de texto para guardar datos, como:

* stock.txt: productos registrados.
* ventas.txt: historial de ventas.
* usuarios.txt: credenciales para el inicio de sesión.
* eliminados.txt: productos descartados por vencimiento.



## **Etapas del código: Análisis detallado**

### **1. Inicio del sistema y autenticación (menu.py y funciones.py)**

El programa comienza en el archivo menu.py, específicamente en la función main(). Lo primero que hace es llamar a:

* cargar\_datos(): recupera la información desde los archivos (stock.txt, ventas.txt, etc.) y los carga en variables globales.
* inicio\_sesion(): solicita nombre de usuario y contraseña. Se permiten hasta 3 intentos antes de cerrar el programa. Esta función usa la librería pwinput para ocultar la contraseña al escribirla.

Esto garantiza que el usuario esté autenticado y que el sistema tenga toda la información previa cargada antes de operar.



### **2. Menú principal y navegación (menu.py)**

Una vez que el usuario inicia sesión, se entra en un bucle que muestra el **menú principal** con las 11 opciones del sistema. Cada opción llama a una función del módulo funciones.py. El menú está definido por la función mostrar\_menu(), y el control del flujo se hace con condicionales if...elif.

**Opciones disponibles:**

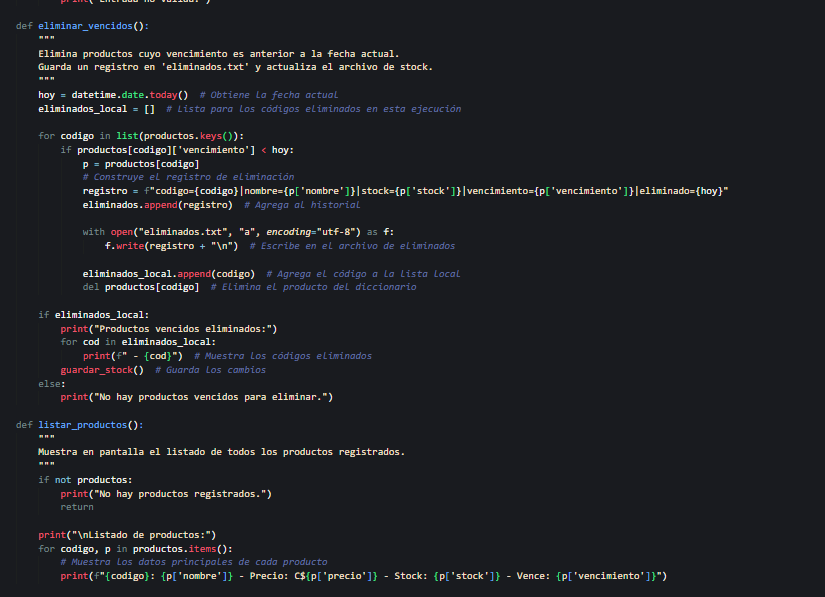
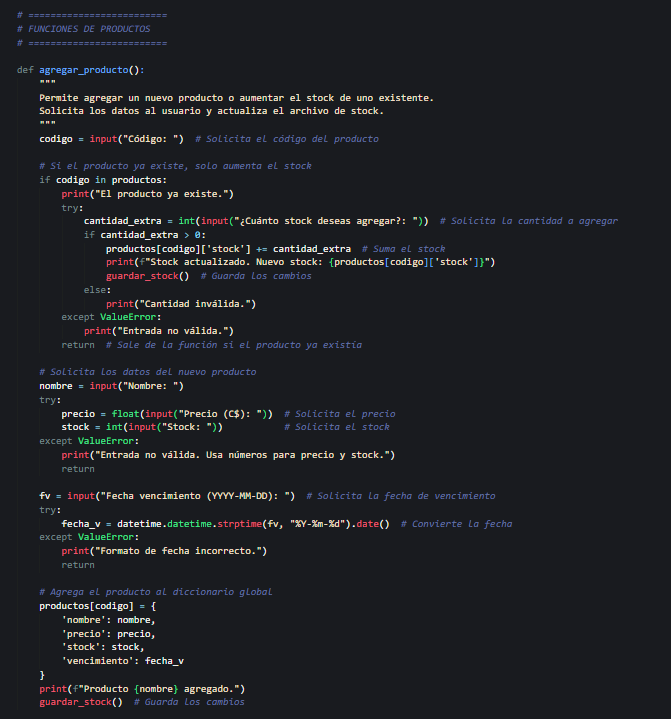
1. Agregar producto
2. Agregar producto con lotes
3. Listar productos
4. Vender producto
5. Ver alertas
6. Ver resumen de ventas
7. Aumentar stock
8. Cambiar fecha de vencimiento
9. Eliminar productos vencidos
10. Ver historial de eliminados
11. Salir del sistema y generar resumen  
    

### **3. Gestión de productos**

**Funciones relevantes:**

* agregar\_producto(): permite ingresar un producto con sus datos básicos (nombre, código, precio, stock, vencimiento). Si ya existe, solo actualiza el stock.
* agregar\_producto\_lotes(): permite agregar varios lotes con sus propias cantidades y fechas de vencimiento. Calcula el total y define la fecha más próxima como principal.
* listar\_productos(): muestra todos los productos almacenados, con sus datos actualizados.
* aumentar\_stock() y cambiar\_fecha\_vencimiento(): permiten modificar productos ya registrados.
* eliminar\_vencidos(): recorre todos los productos y elimina aquellos cuya fecha de vencimiento ya pasó. Registra la eliminación en eliminados.txt.

Estas funciones permiten mantener actualizado el inventario y trabajar tanto con productos simples como con lotes diferenciados.



### **4. Procesamiento de ventas**

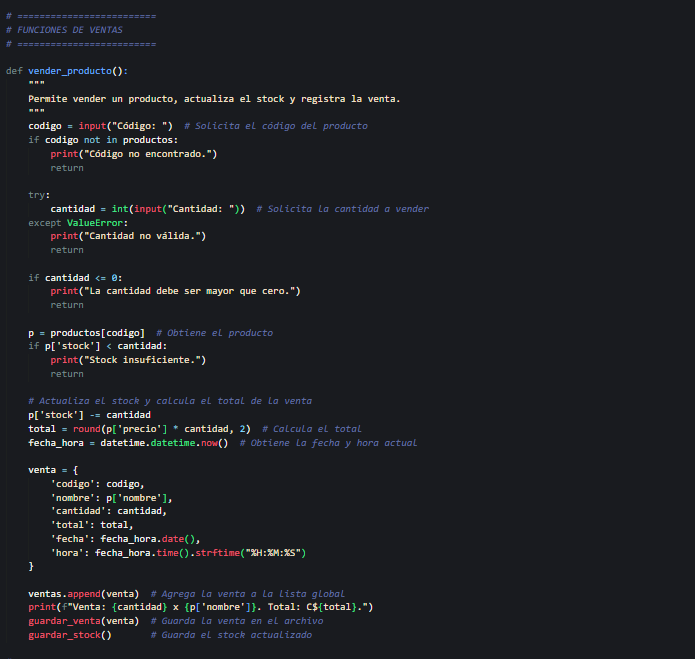
**Función clave:**

* vender\_producto(): permite registrar una venta de un producto. Verifica que exista, que tenga stock suficiente y que la cantidad sea válida. Luego actualiza el stock y guarda la venta.

**Complemento:**

* guardar\_venta(): registra la venta en ventas.txt, incluyendo código, cantidad, total, fecha y hora.

Gracias a estas funciones, el sistema asegura que no se vendan productos inexistentes o sin stock. Además, cada transacción queda registrada con precisión para generar reportes posteriores.

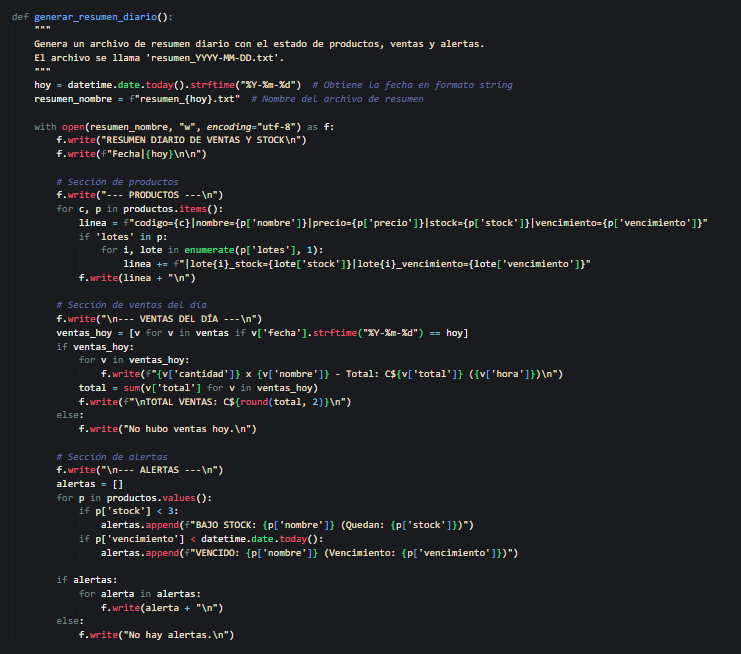


### **5. Alertas y reportes**

Estas funciones ayudan a monitorear el estado del sistema:

* chequear\_alertas(): revisa todos los productos e informa si alguno está agotado, tiene bajo stock o está vencido.
* resumen\_ventas(): calcula y muestra el total de ventas del día, con detalle por producto.
* generar\_resumen\_diario(): crea un archivo resumen\_YYYY-MM-DD.txt con:  
  + Información completa de los productos.
  + Ventas del día.
  + Alertas de stock bajo y productos vencidos.
* mostrar\_eliminados(): presenta los últimos productos eliminados, leyendo el archivo eliminados.txt.

Este conjunto de funciones proporciona una visión clara de la operación diaria del sistema y permite a los administradores tomar decisiones informadas.

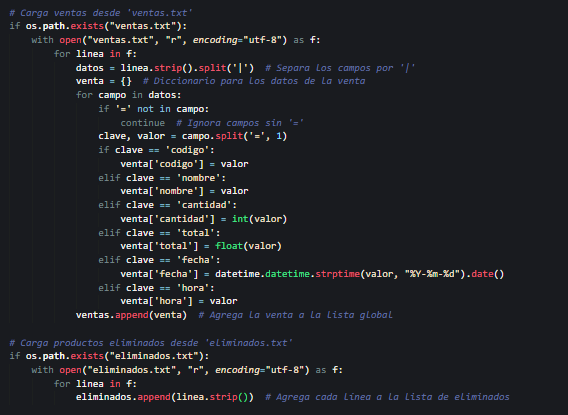
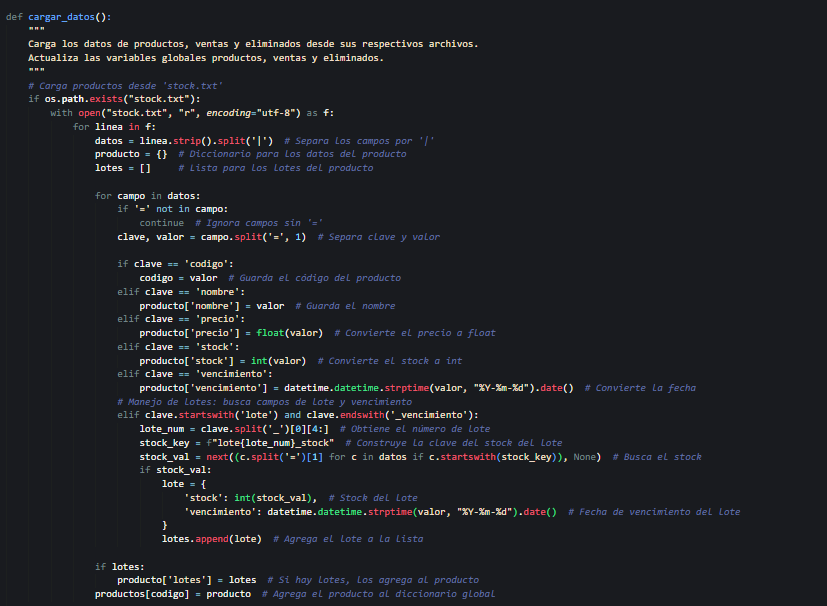
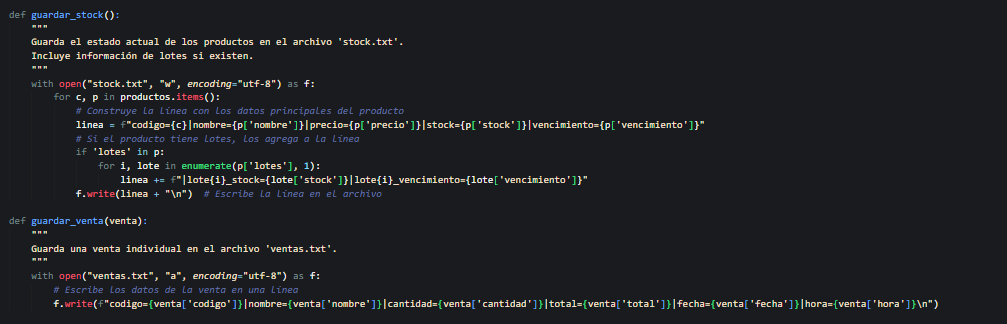


### **6. Manejo de archivos (persistencia de datos)**

La persistencia se logra mediante los archivos .txt:

* guardar\_stock(): actualiza el archivo stock.txt cada vez que se modifica el inventario.
* guardar\_venta(): guarda cada venta de forma incremental.
* cargar\_datos(): al iniciar el sistema, carga todos los productos, ventas y eliminaciones registradas.

Esto garantiza que los datos no se pierdan al cerrar el sistema y que puedan ser retomados en futuras sesiones.



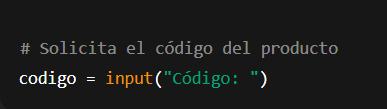
# **Actividad 5: Documentación del proyecto**

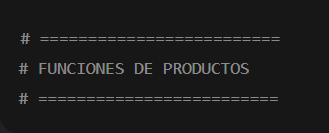
La documentación es una parte esencial del desarrollo de cualquier sistema informático, ya que facilita la comprensión, mantenimiento y evolución del código por parte de otros programadores, usuarios o evaluadores. En este proyecto, se implementaron tanto **documentación interna** como **documentación externa**, siguiendo buenas prácticas de programación y estructuración técnica.

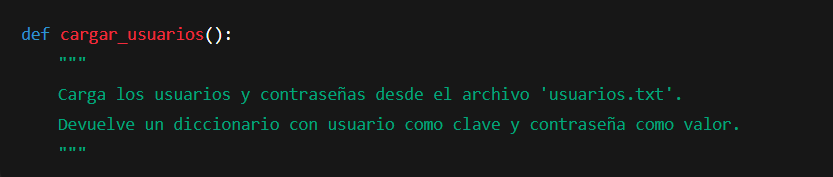
## **Documentación interna**

La **documentación interna** consiste en líneas de comentarios que se encuentran dentro del propio código fuente, explicando la función de bloques específicos, variables, estructuras o procesos. En este proyecto, se empleó de forma consistente el **método de comentarios con el símbolo #**, que es el estándar en el lenguaje de programación **Python**.

Se aplicaron dos tipos de comentarios:

**Comentarios de línea única (#)**: Se utilizan para describir instrucciones específicas, como declaraciones de variables, ciclos, condiciones o llamadas a funciones.  
  


**Bloques de comentarios descriptivos**: Se colocaron al inicio de secciones importantes para explicar la finalidad de grupos de funciones o variables. Estos comentarios ayudan a identificar la estructura lógica del archivo.  
 Ejemplo:  
  


Además, algunas funciones contienen **docstrings (comentarios multilínea con triple comillas """)**, que documentan de forma formal el propósito, entradas y salidas de la función, como es el caso de cargar\_usuarios() o vender\_producto().  


Esta documentación interna permite que otros desarrolladores puedan leer el código y comprender fácilmente qué hace cada bloque sin necesidad de revisar la implementación completa. También es útil para la depuración y futuras mejoras del sistema.

## **Documentación externa**

La **documentación externa** del proyecto se encuentra distribuida a lo largo del informe e incluye:

* **Análisis del problema** y sus subcomponentes (Actividad 1 y 2), con una descripción narrativa detallada y justificación de la solución tecnológica.
* **Diseño del algoritmo** en PSeInt (Actividad 3), incluyendo pseudocódigo estructurado y descripción de módulos.
* **Codificación modular** en Python (Actividad 4), con una explicación técnica de cada subprograma.
* **Manual de usuario** (en los anexos), donde se indican los requisitos del sistema, archivos necesarios, pasos para la ejecución y descripción de cada opción del menú.
* **Diagramas de flujo** (opcionalmente anexados), para representar visualmente el funcionamiento del sistema.

Toda esta documentación complementaria permite que el sistema pueda ser entendido, utilizado, y eventualmente mejorado, sin depender del desarrollador original.

# **Conclusión y Recomendaciones**

## **Conclusión**

El desarrollo del sistema de gestión para el supermercado Palí permitió integrar conocimientos fundamentales de lógica de programación, diseño de algoritmos, codificación en Python y documentación técnica, todo dentro de un contexto realista y aplicable. A lo largo del proyecto, se abordaron problemáticas concretas relacionadas con la gestión manual de inventario, procesamiento de ventas, control de productos vencidos y la generación de reportes estratégicos.

Gracias al uso de herramientas como **PSeInt** para el diseño lógico y **Python con Visual Studio Code** para la implementación, se logró construir una aplicación modular, funcional y con almacenamiento persistente mediante archivos de texto. El sistema cubre con eficacia las operaciones básicas de un comercio minorista, incluyendo control de stock, lotes, vencimientos, ventas y reportes diarios.

Además, el enfoque modular del sistema permite su fácil mantenimiento y adaptación a nuevas funcionalidades, lo cual representa una ventaja tanto desde el punto de vista técnico como pedagógico. Este proyecto no solo resolvió un problema específico, sino que también constituyó una valiosa experiencia de aprendizaje sobre cómo diseñar, programar, depurar y documentar una solución informática completa.

## **Recomendaciones**

A partir del desarrollo e implementación del sistema, se proponen las siguientes recomendaciones para fortalecer y escalar el proyecto:

1. **Migrar a una base de datos** Reemplazar los archivos .txt por una base de datos relacional como SQLite o MySQL, lo cual permitiría consultas más eficientes, mejor integridad de los datos y soporte para múltiples usuarios simultáneos.
2. **Agregar una interfaz gráfica (GUI)** Implementar una interfaz visual con bibliotecas como tkinter o PyQt facilitaría la interacción con el usuario, mejorando la experiencia y usabilidad del sistema.
3. **Control de acceso con roles** Definir distintos niveles de acceso (por ejemplo, administrador, cajero, supervisor) permitiría restringir o habilitar funciones según el tipo de usuario.
4. **Manejo de múltiples sucursales** Adaptar el sistema para registrar operaciones por sede o sucursal, lo cual lo haría escalable a cadenas de supermercados.
5. **Automatización de respaldos** Incluir funciones para realizar copias de seguridad automáticas de los archivos o base de datos, previniendo pérdida de información por errores o fallos del sistema.
6. **Validaciones más robustas** Incluir más validaciones de entrada, como formatos de fecha y números negativos, para asegurar que los datos almacenados sean siempre válidos.
7. **Monitoreo de rendimiento del sistema** Registrar métricas básicas como número de productos vendidos, ventas por hora o por día, y generar gráficas o informes visuales para una mejor toma de decisiones.

# **Referencias**

* Python Software Foundation. (2024). *Python 3.12.2 documentation*.<https://docs.python.org/3/>
* Microsoft. (2024). *Visual Studio Code documentation*.<https://code.visualstudio.com/docs>
* Krausz, P. (2023). *pwinput: A Python library for secure password input*. Python Package Index (PyPI).<https://pypi.org/project/pwinput/>

# **Anexos**

## **Manual de Uso del Programa**

#### **Nombre del sistema:**

**SISTEMA DE GESTIÓN SUPERMERCADO PALI**

### **Requisitos del sistema:**

* **Lenguaje:** Python 3.8 o superior
* **IDE recomendado:** Visual Studio Code
* **Sistema operativo:** Windows, Linux o macOS

### **Dependencias:**

### **Archivos necesarios:**

* menu.py (archivo principal)
* funciones.py (módulo de funciones)
* usuarios.txt (credenciales de acceso)
* stock.txt, ventas.txt, eliminados.txt (se generan o cargan automáticamente)

### **Instrucciones de uso**

#### **1. Ejecución del programa**

* Abre Visual Studio Code.
* Abre la carpeta donde se encuentran los archivos del proyecto.
* Ejecuta menu.py presionando F5 o haciendo clic en **Run Python File**.

#### **2. Inicio de sesión**

* El sistema solicitará un nombre de usuario y contraseña.
* Las credenciales deben existir previamente en el archivo usuarios.txt, por ejemplo:

admin|admin123

Las credenciales deben existir previamente en el archivo usuarios.txt.

#### **3. Menú principal**

Una vez autenticado, el usuario verá un menú con las siguientes opciones:

| **Opción** | **Descripción** |
| --- | --- |
| 1 | Agregar producto simple |
| 2 | Agregar producto con lotes |
| 3 | Listar productos registrados |
| 4 | Vender producto |
| 5 | Verificar alertas (stock bajo, vencimientos) |
| 6 | Ver resumen de ventas del día |
| 7 | Aumentar stock de un producto existente |
| 8 | Cambiar fecha de vencimiento |
| 9 | Eliminar productos vencidos automáticamente |
| 10 | Ver historial de productos eliminados |
| 11 | Salir y generar resumen diario |

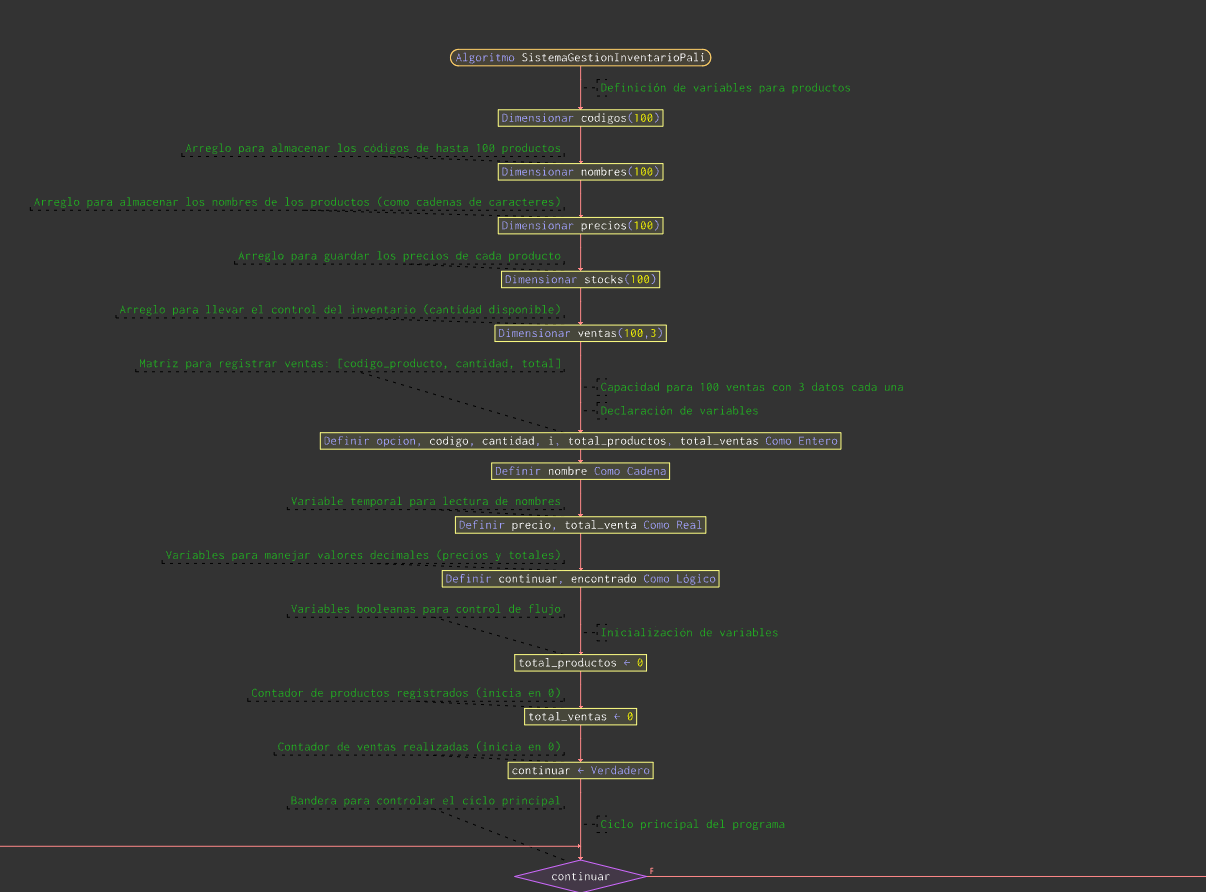
#### **4. Archivos generados y actualizados**

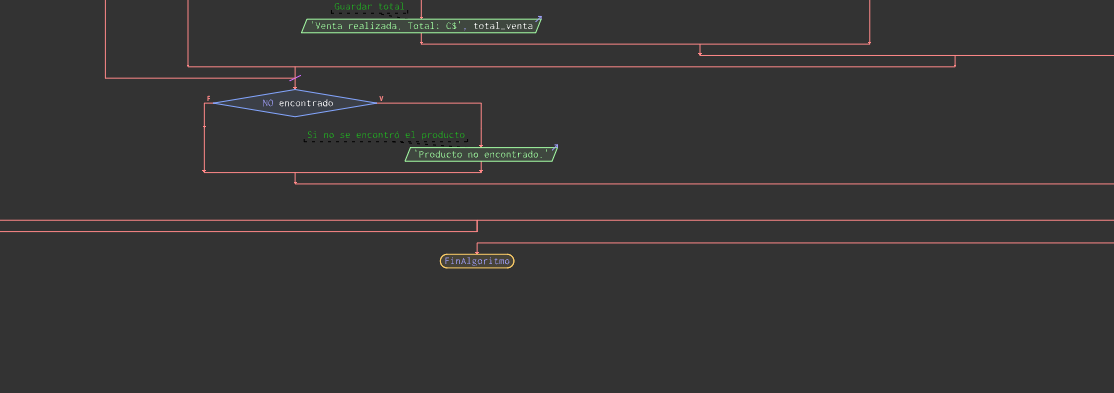
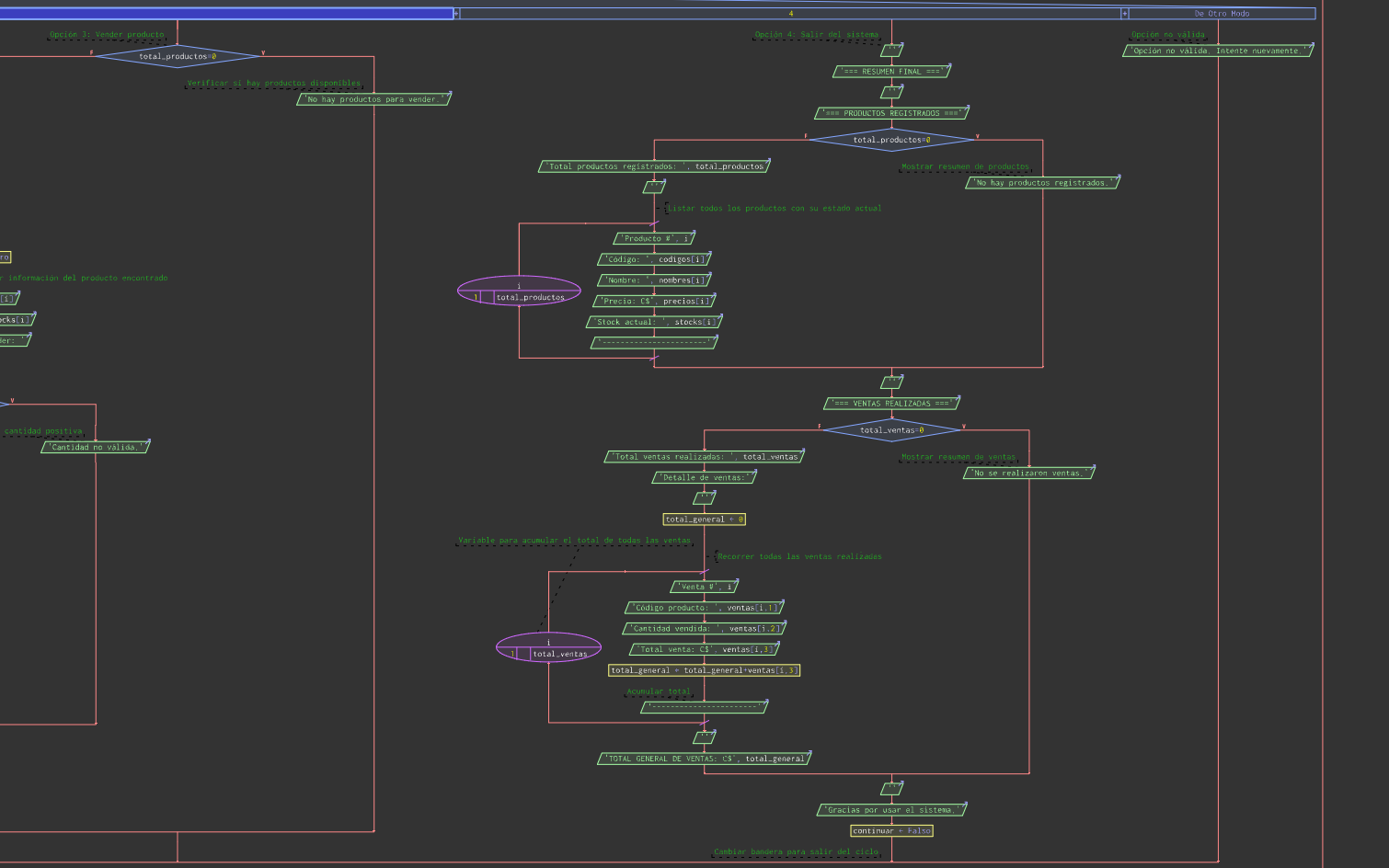
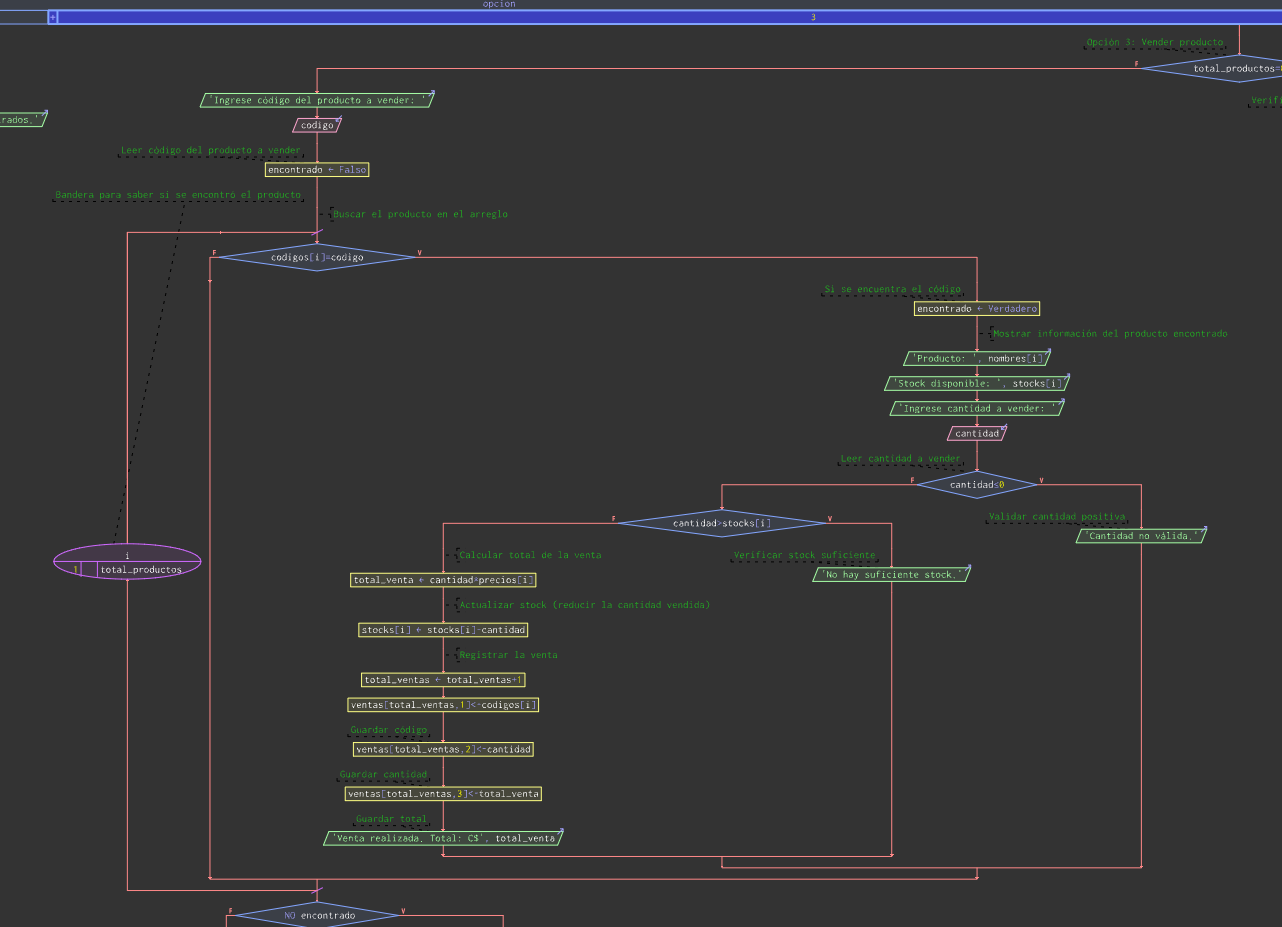
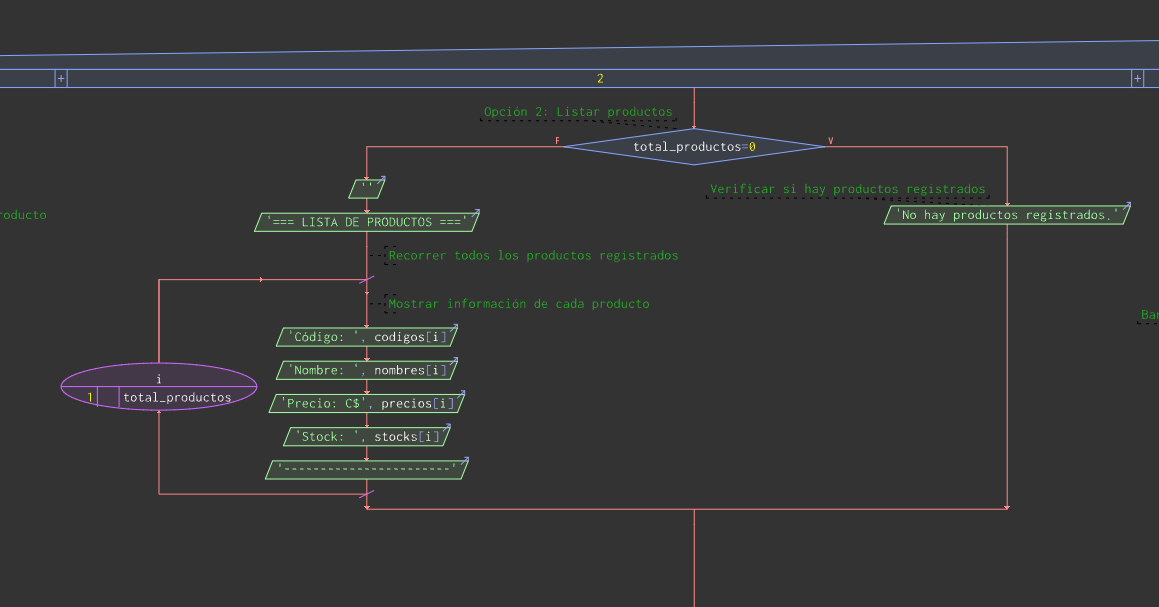
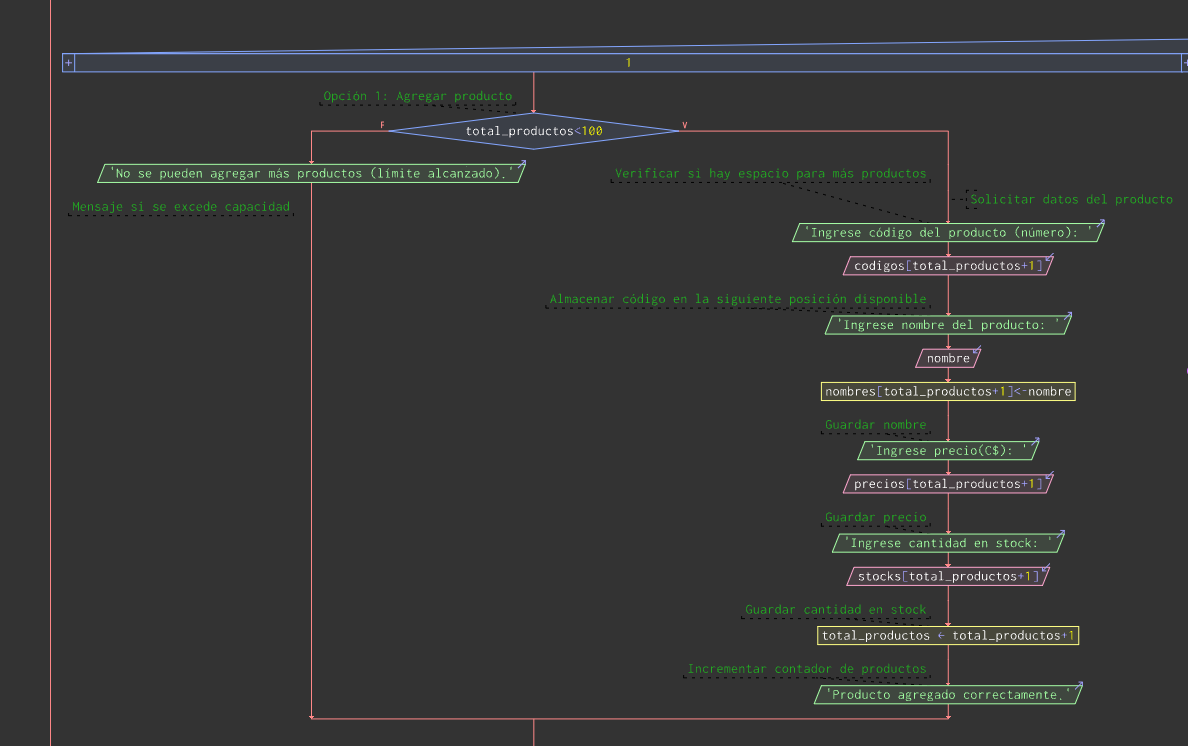
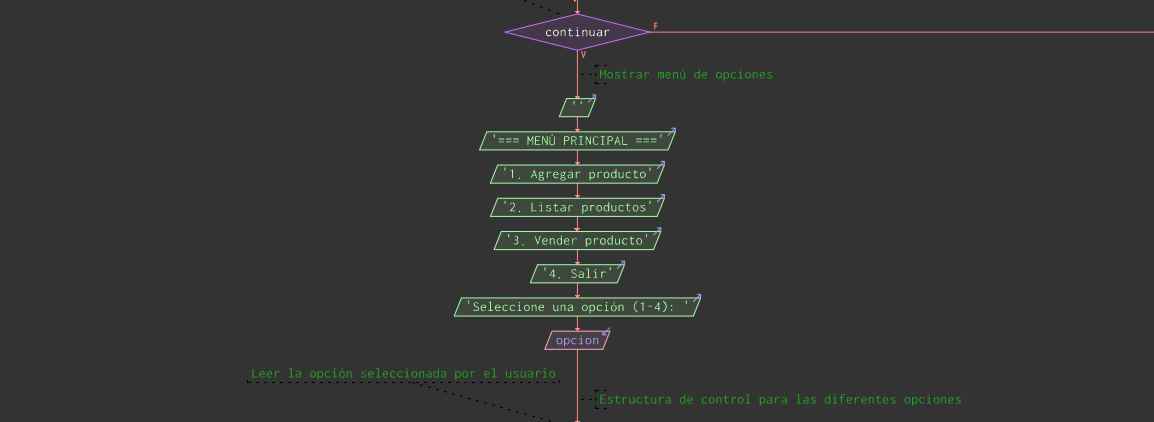
* stock.txt: almacena el inventario de productos y lotes.
* ventas.txt: guarda cada venta realizada con fecha y hora.
* eliminados.txt: registra los productos eliminados por vencimiento.
* resumen\_YYYY-MM-DD.txt: genera un resumen automático al salir del programa.

### **Recomendaciones adicionales**

* Evitar cerrar el programa abruptamente para no perder datos no guardados.
* Hacer copia de seguridad de los archivos .txt regularmente.
* Revisar los mensajes en pantalla antes de cada confirmación o salida.

## **Diagrama de flujo algoritmo Pseint**





## **Guía de Observación**

### **Objetivo de la observación:**

Identificar problemáticas operativas relacionadas con la gestión manual de inventarios, procesamiento de ventas y administración de información, con el fin de fundamentar el desarrollo de una solución tecnológica adecuada.

### **Dimensiones de observación**

| Categoría | Aspectos a observar | Evidencia registrada (ejemplos reales) |
| --- | --- | --- |
| Gestión de inventario | - ¿Cómo se registran los productos?  - ¿Se usan medios físicos o digitales?  - ¿Se controlan las fechas de vencimiento? | Se registran en hojas de Excel sin validación automática. Productos vencidos no se identifican a tiempo. |
| Procesamiento de ventas | - ¿Cómo se calculan los precios?  - ¿El stock se actualiza en tiempo real?  - ¿Hay demoras o errores frecuentes? | El cajero busca precios en una lista impresa. El stock no se actualiza tras la venta. Se observan errores. |
| Alertas y control | - ¿Existen alertas por productos próximos a vencer o con bajo stock?  - ¿Cómo se gestionan los vencidos? | No existen alertas. El personal revisa manualmente las fechas de vencimiento una vez por semana. |
| Toma de decisiones | - ¿La gerencia tiene acceso a reportes actualizados?  - ¿Se generan análisis de ventas o pérdidas? | No se generan reportes periódicos. La información se recopila solo cuando hay problemas o auditorías. |
| Impacto en el personal | - ¿El personal realiza tareas repetitivas?  - ¿Se generan errores por fatiga o carga laboral?  - ¿Se duplican tareas? | Los cajeros anotan ventas y luego ingresan datos en Excel. Se duplican tareas. Errores por falta de tiempo. |
| Impacto en el cliente | - ¿Se generan molestias por productos agotados?  - ¿Existen errores en cobros o precios?  - ¿Cómo reaccionan? | Clientes se retiran por falta de productos. Se quejan de precios mal aplicados. Hay quejas en el mostrador. |

### 

### **Instrumento de registro: Bitácora**

Cada observador llevará una bitácora con anotaciones cronológicas de situaciones observadas, registrando:

* Fecha y hora exacta del evento.
* Lugar o área específica (cajas, bodega, pasillos, etc.).
* Descripción breve del hecho.
* Actores involucrados (ej.: cajero, supervisor, cliente).
* Observaciones adicionales (ej.: reacción de los clientes, solución aplicada, consecuencias observadas).

**Ejemplo de entrada en bitácora:**

**Fecha/Hora:** 28/06/2025 – 3:34 p.m.  
 **Área:** Caja N°2  
 **Situación observada:** Cajera no encuentra el precio de un producto. Busca lista impresa, cliente espera 2 minutos. Producto no se vende.  
 **Observación adicional:** Cliente expresa molestia y deja parte de su compra.