Instituto Tecnológico de Morelia (TECNM)

Ingeniería Bioquímica

Programación

⊙

PROYECTO FINAL

Tienda de tecnología

Isabel Pimentel Puente

Mariana Vallejo Bustillo

Diagrama de flujo del programa

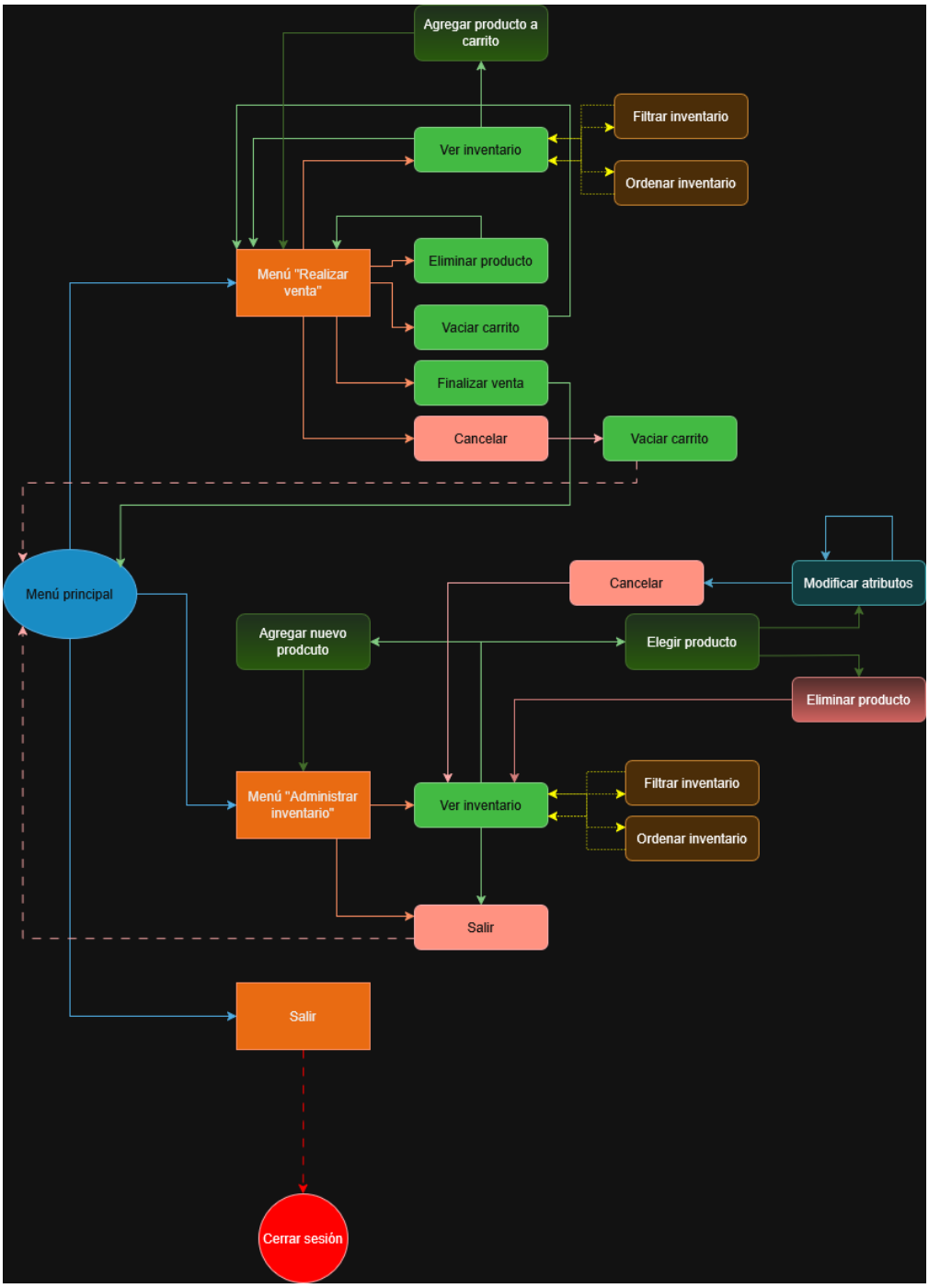

Figura 1: Diagrama de flujo del sistema

1. Funciones implementadas

1.1. Funciones generales

- **menu_x():** Muestran las opciones a elegir y permiten una navegación intuitiva por el sistema.
- **imprimir_x():** Imprimen en consola información relevante para el usuario.
- **detectar_x():** Contienen validadores de entrada del usuario.
- **total_x():** Calculan totales del carrito o del inventario.
- **obtener_mayor_id():** Obtiene el ID del último producto añadido.
- **llenar_inventario():** Llena el inventario con productos predefinidos.

1.2. Funciones específicas

1.2.1. Inventario

- **ver_productos_inventario():**
  Muestra todos los productos disponibles en el inventario en una tabla con diseño, incluyendo opciones de búsqueda y filtrado. También integra la generación del reporte de inventario, mostrando:
  - Total de productos existentes.
  - Valor total del inventario.
  - Productos con stock menor a 6 resaltados en amarillo.
  - Productos sin stock resaltados en rojo.
- **agregar_producto():** Permite agregar un producto requiriendo: nombre, proveedor, categoría, precio unitario y stock inicial.
- **modificar_atributo():** Permite modificar atributos de un producto (excepto el ID) y también eliminarlo si así se desea.

1.2.2. Ventas

- **agregar_producto_carrito():** Agrega un producto existente al carrito mediante su ID, siempre que tenga stock disponible. Cada inserción descuenta una unidad del inventario.
- **eliminar_producto_carrito():** Elimina una unidad de un producto del carrito y regresa esa unidad al inventario.
- **ver_carrito():** Muestra el contenido del carrito, el total acumulado y la cantidad total de artículos.
- **vaciar_carrito():** Vacía el carrito y regresa todo el stock al inventario, previa confirmación del usuario.
- **finalizar_compra():** Calcula subtotales por producto, el total final del carrito y, tras confirmación, genera un ticket de venta en formato JSON.
- **generar_ticket_venta():** Crea un archivo JSON con nombre `Ticket_fecha_hora.json` dentro de la carpeta `Tickets_venta`, que se genera automáticamente si no existe.
- **buscar_productos():** Implementa un filtro de búsqueda por coincidencia parcial (nombre, proveedor, categoría). Permite aplicar múltiples filtros consecutivos y ordenarlos.
- **ordenar_inventario():** Permite ordenar por ID, nombre, proveedor, categoría, precio y stock (este último solo ascendente). El filtro de búsqueda se puede aplicar sobre el ordenamiento.

2. Estructura de datos

2.1. Inventario

El inventario está implementado como un diccionario llave–valor:

```
Inventario = { ID : Producto }
```

Cada producto es también un diccionario:

```
Producto = {ÏD": ID, "nombre": nombre, çategoría": categoría,
"precio": precio_unitario, "stock": stock_inventario, "proveedor": proveedor}
```

Esto garantiza unicidad por ID, acceso directo a los atributos y fácil modificación.

2.2. Carritos

Existen dos estructuras:

- **Carrito principal:** Lista de productos agregados.

```
Carrito = [Producto_1, Producto_2, ...]
```

- **Carrito auxiliar:** Diccionario que relaciona cada ID con la cantidad agregada.

```
Carrito_auxiliar = { ID : cantidad }
```

3. Justificación del diseño

La estructura del programa es **estructural** para mantener claridad y simplicidad.

El inventario y los productos se manejan con diccionarios, usando las herramientas nativas de Python, lo que evita dependencias externas. Los diccionarios permiten acceso directo por llave, garantizan unicidad y evitan incoherencias.

El carrito principal se maneja como lista, pues únicamente se requiere almacenar elementos de forma ordenada. El carrito auxiliar simplifica el conteo de productos repetidos sin recorrer la lista completa cada vez.

La vista en consola facilita la implementación de menús y validaciones, y permite un diseño visual claro mediante tablas y colores ANSI.

El sistema de búsqueda y ordenamiento permite aplicar múltiples filtros consecutivos. El ordenamiento convierte temporalmente el inventario en una lista para simplificar el uso de `sort()`.

Los tickets se guardan en JSON para mantener una estructura legible, portable y fácil de procesar posteriormente.

La paleta de colores busca mejorar la experiencia de usuario:

- **Blanco:** Información normal.
- **Verde:** Operaciones exitosas.
- **Amarillo:** Advertencias sobre posibles incoherencias.
- **Rojo:** Errores que interrumpen el flujo.
- **Azul:** Confirmaciones.
- **Cian:** Información relevante del menú actual.
- **Morado:** Resalte de secciones activas.

4. Limitaciones conocidas

- Los cambios al inventario existen solo durante la ejecución del programa; no hay persistencia global.
- Los valores de producto no deben sobrepasar el ancho diseñado en las tablas (p. ej. nombres menores a 30 caracteres).
- El diseño está optimizado para UTF-8; en una codificación distinta, el diseño visual podría verse afectado.