Python Inicial

Jose Miguel Paz Portilla

Proyecto Final

Inventario de Productos

16 de julio de 2025

Índice

1 Introducción

Con el objetivo de realizar un inventario de productos con persistencia de información, se desarrollo 2 versiones del programa escritos en python, uno con persistencia en archivo CSV y otro manejando una base de datos con SQL.

Los requisitos con CSV son:

- Usar listas para almacenar y gestionar los datos.
- Incorporar bucles while y for según corresponda.
- Validar entradas del usuario o usuaria, asegurándote de que no se ingresen datos vacíos o incorrectos.
- Utilizar condicionales para gestionar las opciones del menú y las validaciones necesarias.
- Presentar un menú que permita elegir entre las funcionalidades disponibles: agregar productos, visualizar productos, buscar productos y eliminar productos.
- El programa debe continuar funcionando hasta que se elija una opción para salir.
- El programa persiste la información en un archivo del disco rígido el tipo CSV.

Los requisitos con SQL son:

Se debe crear una base de datos llamada **inventario.db** para almacenar los datos de los productos. La *tabla productos* debe contener las siguientes columnas:

- **id:** Identificador único del producto (clave primaria, autoincremental).
- **nombre:**Nombre del producto (texto, no nulo).
- **descripcion:** Breve descripción del producto (texto).
- **cantidad:** Cantidad disponible del producto (entero, no nulo).
- **precio:** Precio del producto (real, no nulo).
- **categoria:** Categoría a la que pertenece el producto (texto).

Las funcionalidades con SQL son:

- Registrar nuevos productos.
- Visualizar datos de los productos registrados.
- Actualizar datos de productos, mediante su ID.
- Eliminación de productos, mediante su ID.
- Búsqueda de productos, mediante su ID. De manera opcional, se puede implementar la búsqueda por los campos nombre o categoría.
- Reporte de productos que tengan una cantidad igual o inferior a un límite especificado por el usuario o usuaria.

2 Implementación

2.1 Implementación CSV

Utilizo la función main como función principal del programa, el cual muestra un menu con opciones al usuario, y según la opcion elegida realiza alguno de los requisitos solicitados.

Para ejecutar el programa desde una terminal ubicada donde se encuentran los **archivos python** y el **archivo makefile** ingresar la instrucción **make** o bien **make run**, como muesta la Figura 1.

```
josepaz@josepaz-Lenovo-IdeaPad-S145-15IGM:~/Python/ProyectoFinalPython/CodigoCSV
$ make
```

Figura 1: Ejecución con make

Al iniciar el programa, si no hay un archivo CSV se inicializa la lista de productos como vacia, caso contrario se carga la lista de productos con los productos del archivo CSV.

El programa refresca la terminal y muestra un menu con opciones, se le solicita al usuario que ingrese un número del menu entre las disponibles, como se muestra en la Figura 2.

```
--- MENU DE OPCIONES ---
1. Ingresar un nuevo producto
2. Ver productos registrados
3. Buscar producto por nombre
4. Eliminar un producto
5. Salir
Seleccionar una opción (1-5):
```

Figura 2: Mostrando el menu

Al ingresar la opción 1, se solicita el nombre, categoria y precio del **nuevo producto** para almacenarlo en la lista productos, como se muentra en la Figura 3.

```
--- NUEVO PRODUCTO ---
Ingrese el nombre del producto: zapatilla
Ingrese la categoría del producto: deportiva
Ingrese el precio del producto (sin centavos): 30000
Presione ENTER para continuar
```

Figura 3: Ingreso nuevo producto

Posteriormente al ingresar la opción 2, a partir de la lista productos se muestran los productos del inventario, como se observa en la Figura 4.

```
--- LISTA DE PRODUCTOS REGISTRADOS ---
1. NOMBRE: Garbanzo | CATEGORÍA: Legumbre | PRECIO: $2000
2. NOMBRE: Pollo | CATEGORÍA: Animal | PRECIO: $2000
3. NOMBRE: zapatilla | CATEGORÍA: deportiva | PRECIO: $30000
Presione ENTER para continuar
```

Figura 4: Mostrar productos

Para buscar un producto por su nombre se ingresa la opcion 3, el programa busca en la lista del inventario y va almacenando en una lista auxiliar todos los productos que

coinciden con el nombre del producto, y lo muestra por la terminal, como se muestra en la Figura 5.

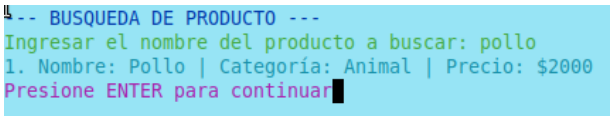


Figura 5: Buscar producto por nombre

Con la opción 4, se puede eliminar un producto con la posición en la lista mostrada por terminal, como se muestra en Figura 6.

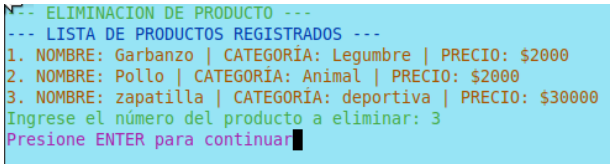


Figura 6: Eliminar producto por indice

Finalmente para finalizar el programa se ingresa la opción 5, y se muestra por terminal un mensaje de finalización, como en la Figura 7.

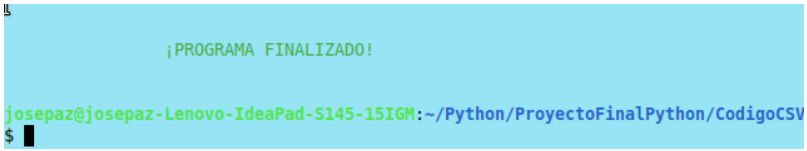


Figura 7: Finalizacion del programa

Además se muestra el archivo CSV que actua como base de datos, como se observa en la Figura 8.

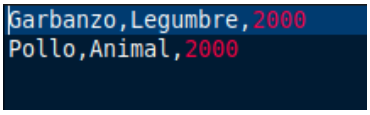


Figura 8: Contenido de archivo CSV

2.2 Implementación SQL

El programa puede implementar con la instrucción **make** o bien **make run** como se muestra en la Figura 9.

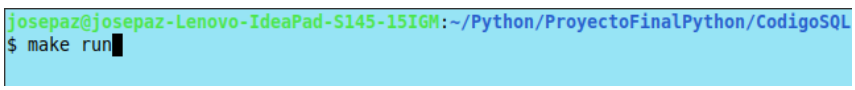


Figura 9: Ejecución con make run

La terminal se refresca y muestra un menú de opciones como se observa en la Figura 10.

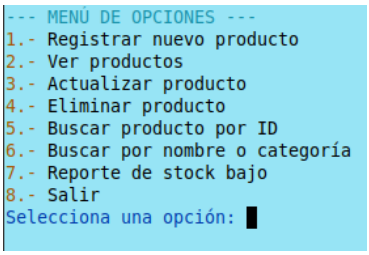


Figura 10: Menú con SQL

Al ingresar la opción 1, se solicita el nombre, descripción, cantidad, precio y categoria del **nuevo producto** para almacenarlo en la base de datos SQL, como se muentra en la Figura 11.

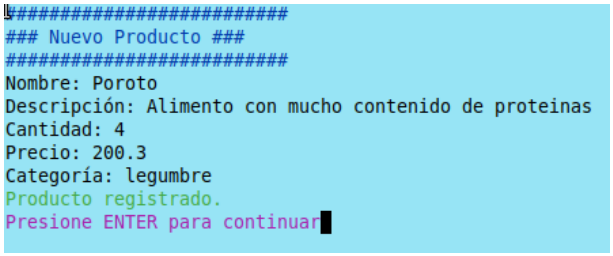


Figura 11: Nuevo Producto con SQL

Posteriormente al ingresar la opción 2, a partir de consltas a la base de datos, se muestran los productos como se observa en la Figura 12.

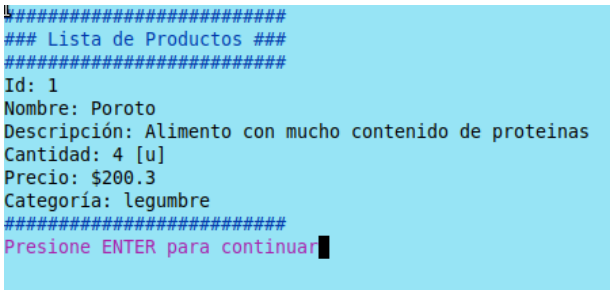


Figura 12: Lista de Productos con SQL

Con la opción 3 se puede modificar campos en el producto, si no se modifica un campo se debe ingresar **enter**, como se muesta en la Figura 13.

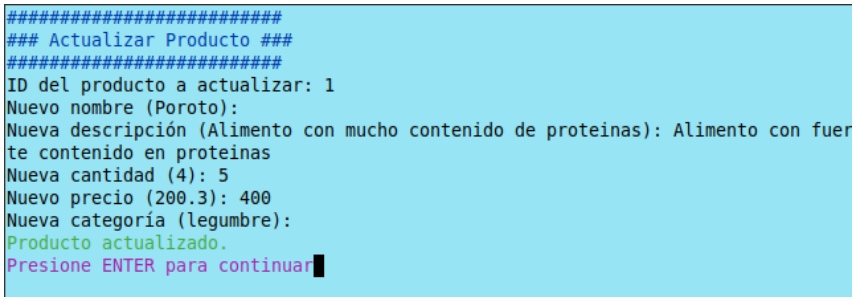


Figura 13: Actualizar Producto con SQL

Con la opción 4 se puede eliminar un producto con su id, como se muestra en la Figura 14 y Figura 15.

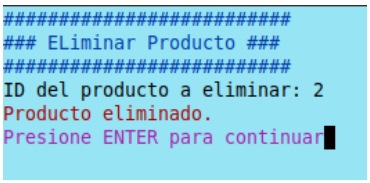


Figura 14: Eliminación de producto con id 2 en SQL

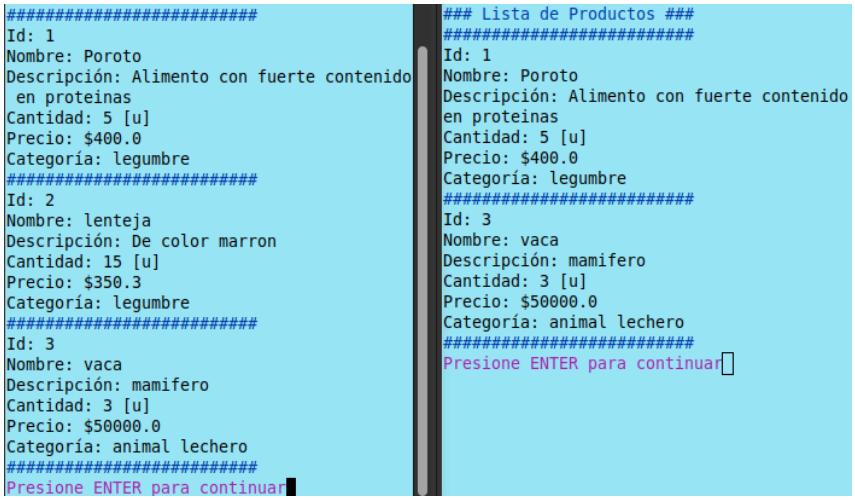


Figura 15: Antes y despues de la eliminación de producto con id 2

Con la opción 5 se puede buscar un producto por su id, como se muestra en la Figura 16.

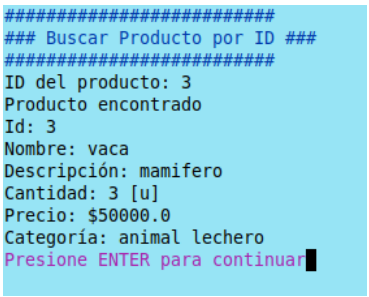


Figura 16: Busqueda de producto por ID

Con la opción 6 se puede buscar un producto por su nombre o categoría como se muestra en la Figura 17.

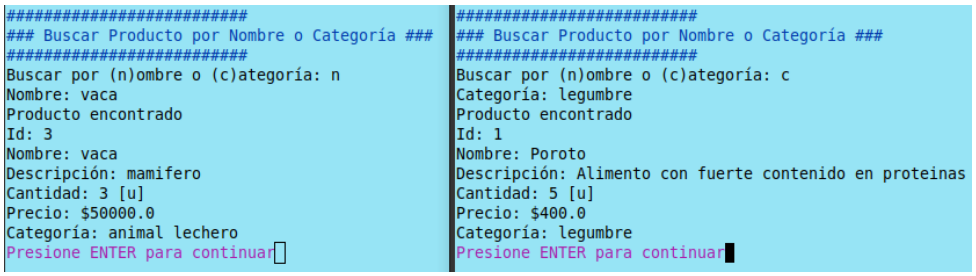


Figura 17: Busqueda de producto por nombre o categoría

Con la opción 7 se puede buscar todos los productos tal que su cantidad sea inferior al indicado por teclado, como se muestra en la Figura 18.

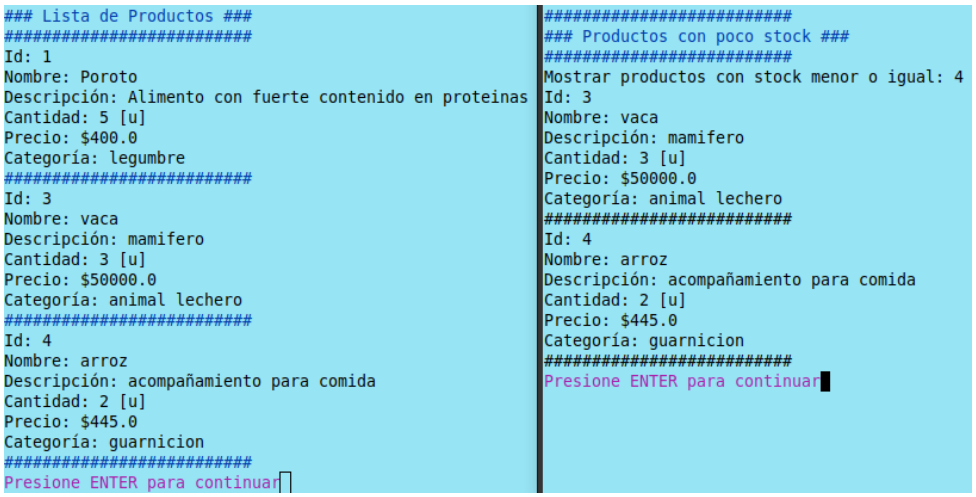


Figura 18: Busqueda de productos con poco stock

Con la opción 8 se puede finalizar el programa como se muestra en la Figura 19.

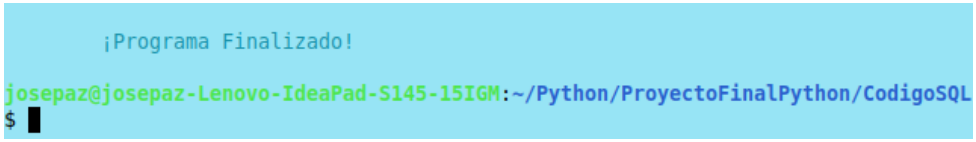


Figura 19: Finalizar el programa

Como se muetra en la Figura 20, el contenido del archivo sql no es facil de entender directamente.

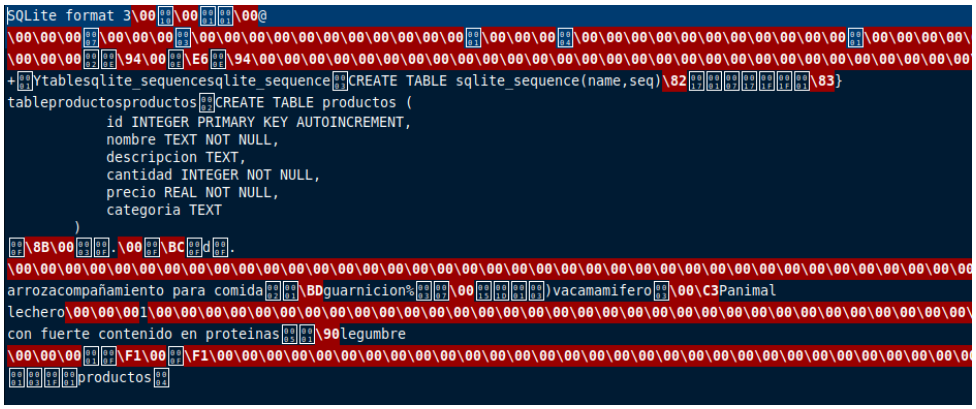


Figura 20: Contenido de archivo SQL

3 Pruebas

3.1 Pruebas CSV

3.1.1 Opción fuera del rango

Como se observa en la Figura 21, al ingresar una opción fuera de las opciones del menu el programa evita que se ejecuten.

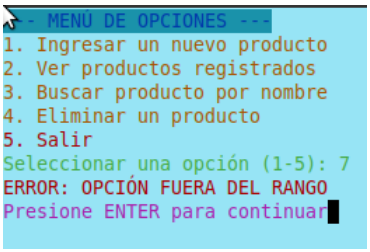


Figura 21: Error por elegir opciones fuera del rango

3.1.2 Campos del producto

Como se observa en la Figura 22, al ingresar campos vacios del nombre o categoria del producto o bien si se ingresa valores no enteros al precio del producto, el sistema no permite continuar con la ejecución del programa.

```
--- NUEVO PRODUCTO ---
Ingrese el nombre del producto:
ERROR: NOMBRE DE PRODUCTO ESTA VACIA
Ingrese el nombre del producto:
ERROR: NOMBRE DE PRODUCTO ESTA VACIA
Ingrese el nombre del producto: zanahoria
Ingrese la categoría del producto:
ERROR: NOMBRE DE LA CATEGORIA ESTA VACIA
Ingrese la categoría del producto: verdura
Ingrese el precio del producto (sin centavos): 0.9
ERROR: EL PRECIO NO ES UN VALOR ENTERO
Ingrese el precio del producto (sin centavos):
ERROR: EL PRECIO NO ES UN VALOR ENTERO
Ingrese el precio del producto (sin centavos): 1500
Presione ENTER para continuar
```

Figura 22: Pruebas al ingresar los campos del producto

3.1.3 Busqueda de producto por nombre

Al ingresar varios productos, se guardan en la lista de productos en forma ordenada alfabeticamente por nombre. Cuando se busca un nombre de un producto se genera una sublista con las coincidencias y se muestra por la terminal en caso en encontrar como se muestra en la Figura 23, sino muestra un mensaje que el producto no esta en el inventario.

```
--- LISTA DE PRODUCTOS REGISTRADOS ---
1. NOMBRE: Garbanzo | CATEGORÍA: Legumbre | PRECIO: $2000
2. NOMBRE: Pollo | CATEGORÍA: Animal | PRECIO: $2000
3. NOMBRE: pOlLo | CATEGORÍA: ave | PRECIO: $9000
4. NOMBRE: zanahoria | CATEGORÍA: verdura | PRECIO: $1500
Presione ENTER para continuar
```

(a) Productos en el inventario

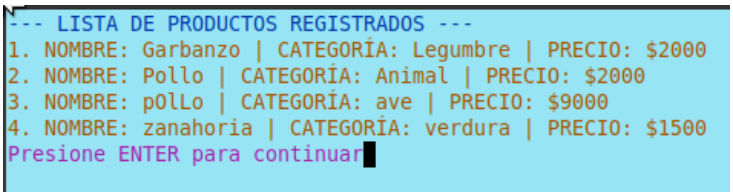
```
--- BUSQUEDA DE PRODUCTO ---
Ingresar el nombre del producto a buscar: pollo
1. Nombre: Pollo | Categoría: Animal | Precio: $2000
2. Nombre: pOlLo | Categoría: ave | Precio: $9000
Presione ENTER para continuar
```

(b) Busqueda de un producto por nombre encontrado

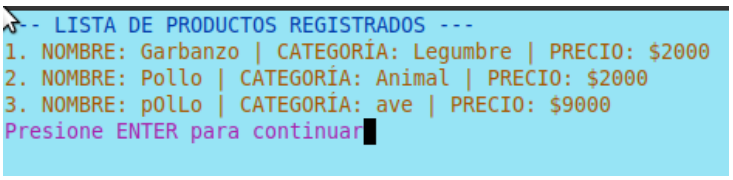
Figura 23: Busqueda de producto por nombre

3.1.4 Eliminación de producto por posición

Como se muestra en la Figura 24, en la terminal se observan los productos que hay en el inventario, junto a un numero, para eliminar por ejemplo naranja, se ingresa su posición en la lista mostrada que es 3, luego se vuelve a mostrar los productos y se observa que fue eliminado del inventario.



(a) Posición del producto a eliminar



(b) Inventario despues de eliminar el producto

Figura 24: Eliminación de producto por posición en el menú

3.2 Pruebas SQL

3.2.1 Opción fuera del rango

Al ingresar en el menú de opciones un número fuera del 1 al 8, el programa muestra un mensaje de error, como se observa en la Figura 25.

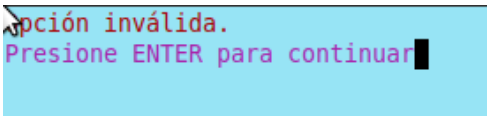


Figura 25: Error por elegir opciones fuera del rango

3.2.2 Campos del producto

El programa valida que al menos no esten vacios el nombre, cantidad y precio de un nuevo producto, como se muestra en la Figura 26.

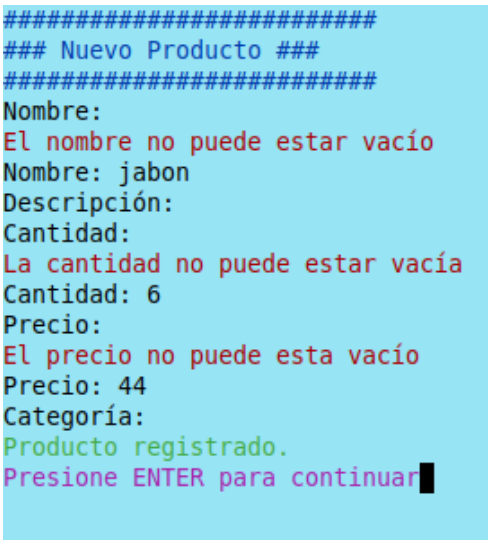


Figura 26: Errores al intentar ingresar campos vacios

4 Código

4.1 Código - CSV = Comma Separated Values(Valores Separados por Coma)

4.1.1 makefile

```
#Para ejecutar el archivo utilizando este makefile escribir en terminal:
# make
# o sino:
# make run
run: main.py metodos_productos.py menu.py opciones.py
	python3 main.py
```

4.1.2 main.py

```
# Pre_Entrega
# Alumno: Paz Portilla, Jose Miguel
# Comision: 25010

#De metodos_productos.py importa los metodos:
#ingresar_producto, mostrar_productos, buscar_producto, eliminar_producto.
from metodos_productos import (
	ingresar_producto,
	mostrar_productos,
	buscar_producto,
	eliminar_producto,
	cargar_productos,
	guardar_productos
)

#De menu.py importa los metodos:
#limpiar_pantalla, mostrar_menu, ejecutar_opcion, ingresar_opcion.
from menu import (
	limpiar_pantalla,
	mostrar_menu,
	ejecutar_opcion,
	ingresar_opcion
)

"""
Es el programa principal, repetitivamente limpia la terminal(hasta ingresar
↪  terminar program),
muestra un menu de opciones, toma la opcion y realiza alguna accion sobre
↪  productos a partir de ella.
Parámetros:-
Retorna:-
"""
def main () :

	productos = cargar_productos()
	volver_al_menu = True
```

```
        while volver_al_menu == True :
                limpiar_pantalla ()
                mostrar_menu ()
                opcion = ingresar_opcion ()
                volver_al_menu = ejecutar_opcion ( opcion , productos )
        guardar_productos(productos)
        limpiar_pantalla ()
        print("\n\n\t\t¡PROGRAMA FINALIZADO!\n\n")


if __name__ == "__main__":
    main()
```

4.1.3 menu.py

```
#Importa el módulo estándar os, esto proporciona
#funciones para interactuar con el sistema operativo.
import os
import opciones

from colorama import Fore, Style, Back, init
init(autoreset=True)

from metodos_productos import (
        ingresar_producto,
        mostrar_productos,
        buscar_producto,
        eliminar_producto
)

"""
Refresca o limpia la terminal
Parámetros:-
Retorna:-
"""
def limpiar_pantalla():
        # os.system: Ejecuta un comando del sistema operativo
        # como si se escribiera en la terminal.
        # Si el nombre del sistema operativo es windows utiliza cls,
        # sino utiliza clear para refrescar la pantalla
    os.system('cls' if os.name == 'nt' else 'clear')

"""
Muestra las opciones del menu por la terminal
Parámetros:-
Retorna:-
"""
def mostrar_menu():
    print(Fore.BLUE + Back.CYAN + "--- MENÚ DE OPCIONES ---" + Style.RESET_ALL)
```

```
    print(Fore.YELLOW + "1. Ingresar un nuevo producto")
    print(Fore.YELLOW + "2. Ver productos registrados")
    print(Fore.YELLOW + "3. Buscar producto por nombre")
    print(Fore.YELLOW + "4. Eliminar un producto")
    print(Fore.RED + "5. Salir")

"""
Muestra las opciones del menu por la terminal
Parámetros:-
Retorna:la opcion que se ingreso por teclado,
eliminando los espacios en blanco al inicio y al final
"""
def ingresar_opcion ():
        opcion = input ( Fore.GREEN + "Seleccionar una opción (1-5): " ).strip()
        return opcion

"""
Realiza la opcion sobre la lista de productos
Parámetros:- opcion a realizar y la lista de productos
Retorna:Bool para indicar si debe seguir pidiendo opciones o no
"""
def ejecutar_opcion ( opcion , productos ) :
        match opcion:
                case opciones.INGRESAR_NUEVO_PRODUCTO:
                        limpiar_pantalla()
                        ingresar_producto(productos)
                case opciones.MOSTRAR_PRODUCTOS_POR_TERMINAL:
                        limpiar_pantalla()
                        mostrar_productos(productos)
                case opciones.BUSCAR_PRODUCTO_POR_NOMBRE:
                        limpiar_pantalla()
                        buscar_producto(productos)
                case opciones.ELIMINAR_PRODUCTO_POR_INDICE:
                        limpiar_pantalla()
                        eliminar_producto(productos)
                case opciones.FINALIZAR_PROGRAMA:
                        return False  # salir del programa
                case _:
                        print(Fore.RED + "ERROR: OPCIÓN FUERA DEL RANGO")

        input(Fore.MAGENTA + "Presione ENTER para continuar")
        return True  # continuar el bucle
```

4.1.4 opciones.py

```
#Constantes que representan las opciones del menu
INGRESAR_NUEVO_PRODUCTO = '1'
MOSTRAR_PRODUCTOS_POR_TERMINAL = '2'
BUSCAR_PRODUCTO_POR_NOMBRE = '3'
ELIMINAR_PRODUCTO_POR_INDICE = '4'
```

```
FINALIZAR_PROGRAMA = '5'
```

4.1.5 metodos_productos.py

```
# Se definen las funciones que sirven para manipular los productos

import os
import csv # CSV = Comma-Separated Values: utiliza los metodos read() y write()
↪  para manejar la base_datos.csv con separacion de comas, writerows convierte
↪  la lista productos como sublistas producto con comas y los almacena por
↪  filas.
from colorama import Fore, Style, Back, init
init(autoreset=True)
NOMBRE = 0 # producto[0]: NOMBRE
CATEGORIA = 1 #  producto[1]: CATEGORIA
PRECIO = 2 # producto[2]: PRECIO
"""
Genera un producto a partir del nombre, categoria y precio ingresados por
↪  teclado
Ingresa el producto a la lista de productos
Parámetros:
productos(list): lista de productos
Retorna:-
"""
def ingresar_producto ( productos ):
        print (Fore.BLUE + "--- NUEVO PRODUCTO ---")
        while True:
                nombre = input(Fore.YELLOW + "Ingrese el nombre del producto:
                ↪  ").strip()
                if nombre != "":
                        break
                print(Fore.RED + "ERROR: NOMBRE DE PRODUCTO ESTA VACIA")

        while True:
                categoria = input(Fore.YELLOW + "Ingrese la categoría del
                ↪  producto: ").strip()
                if categoria != "":
                        break
                print(Fore.RED + "ERROR: NOMBRE DE LA CATEGORIA ESTA VACIA")

        while True:
                #Bloque para atrape el error de conversion de cadena a entero
                try:
                        precio = int(input(Fore.YELLOW + "Ingrese el precio del
                        ↪  producto (sin centavos): "))
                        if precio < 0:
                                print(Fore.RED + "ERROR: EL PRECIO INGRESADO ES
                                ↪  NEGATIVO")
                                continue
                        break
                except ValueError:
```

```
                        print(Fore.RED + "ERROR: EL PRECIO NO ES UN VALOR
                        ↪  ENTERO")

        producto = [nombre, categoria, precio]

        # Insertar producto en la lista de productos ordenado por nombre
        producto_fue_insertado = False
        for indice in range(len(productos)):#i va de 0 hasta len(productos)-1
                # compara ignorando mayúsculas/minúsculas
                if nombre.lower() < productos[indice][NOMBRE].lower():
                        productos.insert (indice, producto)
                        producto_fue_insertado = True
                        break
        # Si no fue ingresado producto en la lista de productos,
        # debido a que al compararlo no hubo otro nombre menor alfabeticamente,
        # Simplemente de agrega el producto al final de la lista productos
        if not producto_fue_insertado:
                productos.append(producto)

"""
Muestra por terminal la lista de productos
con el formato: nombre|categoria|precio
Parámetros:
productos(list): lista de productos
Retorna:-
"""
def mostrar_productos ( productos ) :
    if len (productos) == 0:
        print (Fore.RED + "No hay productos registrados.")
        return

    print (Fore.BLUE +  "--- LISTA DE PRODUCTOS REGISTRADOS ---")
    # Empieza a enumerar desde 1 el i, y va deserializando producto de la
    ↪  lista_productos
    for indice, producto in enumerate( productos , start = 1 ):
        nombre, categoria, precio = producto
        print(Fore.YELLOW +  f"{indice}. NOMBRE: {nombre} | CATEGORÍA:
        ↪  {categoria} | PRECIO: ${precio}")

"""
Busca en la lista de productos los productos que coinciden con el nombre de
↪  productos
ingresado por teclado. Luegos los muestra por terminal en caso de encontrarlo
Parámetros:
productos(list): lista de productos
Retorna:-
"""
def buscar_producto ( productos ):
        print (Fore.BLUE +  "--- BUSQUEDA DE PRODUCTO ---")
        # Borra espacios de los bordes y lo paso a minusculas
```

```
        nombre = input(Fore.GREEN + "Ingresar el nombre del producto a buscar:
        ↪  ").strip().lower()
        if nombre == "":
                print(Fore.RED + "ERROR: NOMBRE DE PRODUCTO A BUSCAR ESTA
                ↪  VACIA")
                return

        encontrados = []
        for producto in productos:
                if nombre in producto[NOMBRE].lower():
                        encontrados.append(producto)

        if len(encontrados) != 0:
                for indice, encontrado in enumerate ( encontrados , start = 1 )
                ↪  :
                        nombre, categoria, precio = encontrado
                        print(Fore.CYAN + f"{indice}. Nombre: {nombre} |
                        ↪  Categoría: {categoria} | Precio: ${precio}")
        else:
                print(Fore.RED + "NOMBRE DE PRODUCTO NO ESTA EN EL INVENTARIO")

"""
Elimina un producto de la lista de productos a partir de la posicion mostrada
↪  por terminal
Parámetros:
productos(list): lista de productos
Retorna:-
"""
def eliminar_producto ( productos ):
        print ("--- ELIMINACION DE PRODUCTO ---")
        if len(productos) == 0:
                print (Fore.RED + "INVENTARIO ESTÁ VACIO")
                return

        mostrar_productos(productos)
        while True:
                try:
                        indice = int(input(Fore.GREEN + "Ingrese el número del
                        ↪  producto a eliminar: "))
                        if 1 <= indice and indice <= len(productos):
                                #Se resta 1 porque la posicion en la lista
                                ↪  empieza en 0
                                eliminado = productos.pop ( indice - 1 )
                                break
                        else:
                                print(Fore.RED + "ERROR: OPCION DE LA LISTA DE
                                ↪  PRODUCTOS FUERA DE RANGO")
                except ValueError:
                        print(Fore.RED + "ERROR: INGRESO DE UN NUMERO NO
                        ↪  ENTERO")
```

```
"""
Carga la lista de productos con los datos del archivo csv, que tiene la
↪  informacion cargada separando sus atributos por comas
Parámetros:
archivo de base de datos, por defecto se guarda un archivo llamado
↪  base_datos.csv
Retorna:
Lista de productos
"""
def cargar_productos (archivo='base_datos.csv'):
        productos = []
        if os.path.exists (archivo):
                with open (archivo, 'r', newline = '') as base_datos:
                        productos_csv = csv.reader(base_datos)
                        for producto in productos_csv:
                                nombre, categoria, precio = producto[NOMBRE],
                                ↪  producto[CATEGORIA], int(producto[PRECIO])
                                productos.append ([nombre, categoria, precio])
        return productos

"""
Guarda la lista de productos en el archivo csv
Parámetros:
archivo de base de datos, por defecto se guarda un archivo llamado
↪  base_datos.csv
Retorna:-
"""
def guardar_productos (productos, archivo='base_datos.csv'):
    with open(archivo, 'w', newline='') as base_datos:
        productos_csv = csv.writer (base_datos)
        productos_csv.writerows (productos)
```

4.2 Código - SQL = Structured Query Language(Lenguaje de Consulta Estructurado)

4.2.1 makefile

```
#Para ejecutar el archivo utilizando este makefile escribir en terminal:
# make
# o sino:
# make run
run: main.py productos.py menu.py
        python3 main.py
```

4.2.2 main.py

```
from productos import crear_base
from menu import mostrar_menu, ejecutar_opcion, limpiar_pantalla
from colorama import Fore, Style, init
```

```
def main():
    crear_base()
    while True:
        limpiar_pantalla ()
        mostrar_menu ()
        opcion = input (Fore.BLUE +"Selecciona una opción: ")
        limpiar_pantalla ()
        ejecutar_opcion (opcion)
        input (Fore.MAGENTA + "Presione ENTER para continuar")

if __name__ == "__main__":
    main()
```

4.2.3 menu.py

```
from colorama import Fore, Style, init
import productos
import os

init (autoreset=True)


"""
Refresca o limpia la terminal
Parámetros:-
Retorna:-
"""
def limpiar_pantalla ():
        # os.system: Ejecuta un comando del sistema operativo
        # como si se escribiera en la terminal.
        # Si el nombre del sistema operativo es windows utiliza cls,
        # sino utiliza clear para refrescar la pantalla
    os.system('cls' if os.name == 'nt' else 'clear')

def mostrar_menu ():
    print(Fore.CYAN + "--- MENÚ DE OPCIONES ---")
    print(Fore.YELLOW + "1.-" + Fore.RESET + " Registrar nuevo producto")
    print(Fore.YELLOW + "2.-" + Fore.RESET + " Ver productos")
    print(Fore.YELLOW + "3.-" + Fore.RESET + " Actualizar producto")
    print(Fore.YELLOW + "4.-" + Fore.RESET + " Eliminar producto")
    print(Fore.YELLOW + "5.-" + Fore.RESET + " Buscar producto por ID")
    print(Fore.YELLOW + "6.-" + Fore.RESET + " Buscar por nombre o categoría")
    print(Fore.YELLOW + "7.-" + Fore.RESET + " Reporte de stock bajo")
    print(Fore.YELLOW + "8.-" + Fore.RESET + " Salir")

def ejecutar_opcion(opcion):
    match opcion:
        case "1":
            print(Fore.BLUE +"############################")
            print (Fore.BLUE + "### Nuevo Producto ###")
            print(Fore.BLUE + "############################")
```

```
        nombre = input("Nombre: ")
        while nombre == "":
                print (Fore.RED +"El nombre no puede estar vacío")
                nombre = input("Nombre: ")
        descripcion = input("Descripción: ")
        cantidad = input("Cantidad: ")
        while cantidad == "":
                print (Fore.RED +"La cantidad no puede estar vacía")
                cantidad = input("Cantidad: ")
        try:
                cantidad = int(cantidad)
        except ValueError:
                print (Fore.RED +"La cantidad debe ser un valor entero")
        if cantidad <= 0:
                print(Fore.RED +"La cantidad debe ser mayor a cero")
                return
        precio = input("Precio: ")
        while precio == "":
                print (Fore.RED +"El precio no puede esta vacío")
                precio = input("Precio: ")
        try:
                precio = float (precio)
        except ValueError:
                print (Fore.RED +"El precio debe ser un valor real")
        if precio <= 0.0:
                print (Fore.RED +"El precio debe ser mayor a cero")
                return
        categoria = input("Categoría: ")
        productos.agregar (nombre, descripcion, cantidad, precio, categoria)
        print(Fore.GREEN + "Producto registrado.")

    case "2":
        print(Fore.BLUE + "############################")
        print (Fore.BLUE +"### Lista de Productos ###")
        print(Fore.BLUE + "############################")
        for producto in productos.listar():
            print(f"Id: {producto[0]}")
            print(f"Nombre: {producto[1]}")
            print(f"Descripción: {producto[2]}")
            print(f"Cantidad: {producto[3]} [u]")
            print(f"Precio: ${producto[4]}")
            print(f"Categoría: {producto[5]}")
            print(Fore.BLUE +"############################")

    case "3":
        print(Fore.BLUE + "############################")
        print (Fore.BLUE + "### Actualizar Producto ###")
        print(Fore.BLUE + "############################")
        identificador = int(input("ID del producto a actualizar: "))
        producto = productos.buscar_por_id (identificador)
        if producto:
```

```
            nombre = input(f"Nuevo nombre ({producto[1]}): ") or
            ↪  producto[1]
            descripcion = input(f"Nueva descripción ({producto[2]}): ") or
            ↪  producto[2]
            cantidad = int(input(f"Nueva cantidad ({producto[3]}): ") or
            ↪  producto[3])
            precio = float(input(f"Nuevo precio ({producto[4]}): ") or
            ↪  producto[4])
            categoria = input(f"Nueva categoría ({producto[5]}): ") or
            ↪  producto[5]
            productos.actualizar(identificador, nombre, descripcion,
            ↪  cantidad, precio, categoria)
            print(Fore.GREEN + "Producto actualizado.")
        else:
            print(Fore.RED + "Producto no encontrado.")

    case "4":
        print(Fore.BLUE + "############################")
        print (Fore.BLUE + "### ELiminar Producto ###")
        print(Fore.BLUE + "############################")
        identificador = int(input("ID del producto a eliminar: "))
        productos.eliminar (identificador)
        print(Fore.RED + "Producto eliminado.")

    case "5":
        print(Fore.BLUE + "############################")
        print (Fore.BLUE + "### Buscar Producto por ID ###")
        print(Fore.BLUE + "############################")
        pid = int(input("ID del producto: "))
        producto = productos.buscar_por_id(pid)
        if producto:
                print ("Producto encontrado")
                print(f"Id: {producto[0]}")
                print(f"Nombre: {producto[1]}")
                print(f"Descripción: {producto[2]}")
                print(f"Cantidad: {producto[3]} [u]")
                print(f"Precio: ${producto[4]}")
                print(f"Categoría: {producto[5]}")
        else:
                print (Fore.RED + "Producto no encontrado.")

    case "6":
        print(Fore.BLUE + "############################")
        print (Fore.BLUE + "### Buscar Producto por Nombre o Categoría ###")
        print(Fore.BLUE + "############################")
        tipo = input("Buscar por (n)ombre o (c)ategoría: ").lower()
        if tipo == "n":
            nombre = input("Nombre: ")
            encontrados = productos.buscar_por_nombre(nombre)
        elif tipo == "c":
            categoria = input("Categoría: ")
```

```
                encontrados = productos.buscar_por_categoria(categoria)
            else:
                print(Fore.RED + "Opción inválida.")
                return

            if encontrados:
                    for producto in encontrados:
                            print ("Producto encontrado")
                            print(f"Id: {producto[0]}")
                            print(f"Nombre: {producto[1]}")
                            print(f"Descripción: {producto[2]}")
                            print(f"Cantidad: {producto[3]} [u]")
                            print(f"Precio: ${producto[4]}")
                            print(f"Categoría: {producto[5]}")
            else:
                    print (Fore.RED + "Producto no encontrado.")

        case "7":
            print(Fore.BLUE + "############################")
            print (Fore.BLUE + "### Productos con poco stock ###")
            print(Fore.BLUE + "############################")
            cantidad_maxima = int(input("Mostrar productos con stock menor o
            ↪  igual: "))
            for producto in productos.stock_bajo (cantidad_maxima):
                print(f"Id: {producto[0]}")
                print(f"Nombre: {producto[1]}")
                print(f"Descripción: {producto[2]}")
                print(f"Cantidad: {producto[3]} [u]")
                print(f"Precio: ${producto[4]}")
                print(f"Categoría: {producto[5]}")
                print("############################")

        case "8":
            print(Fore.CYAN + "\n\t¡Programa Finalizado!\n")
            exit()

        case _:
            print(Fore.RED + "Opción inválida.")
```

4.2.4 productos.py

```
import sqlite3

def crear_base ():
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS productos (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            nombre TEXT NOT NULL,
            descripcion TEXT,
```

```
            cantidad INTEGER NOT NULL,
            precio REAL NOT NULL,
            categoria TEXT
        )
    """)
    conexion.commit ()
    conexion.close ()

def agregar (nombre, descripcion, cantidad, precio, categoria):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("INSERT INTO productos (nombre, descripcion, cantidad,
    ↪  precio, categoria) VALUES (?, ?, ?, ?, ?)",
              (nombre, descripcion, cantidad, precio, categoria))
    conexion.commit ()
    conexion.close ()

def listar ():
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("SELECT * FROM productos")
    productos = cursor.fetchall ()
    conexion.close ()
    return productos

def buscar_por_id (identificador):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute("SELECT * FROM productos WHERE id=?", (identificador,))
    producto = cursor.fetchone ()
    conexion.close ()
    return producto

def actualizar (identificador, nombre, descripcion, cantidad, precio,
↪  categoria):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("""UPDATE productos SET nombre=?, descripcion=?, cantidad=?,
    ↪  precio=?, categoria=? WHERE id=?""",
              (nombre, descripcion, cantidad, precio, categoria, identificador))
    conexion.commit ()
    conexion.close ()

def eliminar (identificador):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("DELETE FROM productos WHERE id=?", (identificador,))
    conexion.commit ()
    conexion.close ()

def buscar_por_nombre (nombre):
```

```
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("SELECT * FROM productos WHERE nombre LIKE ?", ('%' + nombre
    ↪  + '%',))
    productos = cursor.fetchall ()
    conexion.close ()
    return productos

def buscar_por_categoria (categoria):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("SELECT * FROM productos WHERE categoria LIKE ?", ('%' +
    ↪  categoria + '%',))
    productos = cursor.fetchall ()
    conexion.close ()
    return productos

def stock_bajo (cantidad_maxima):
    conexion = sqlite3.connect ("inventario.db")
    cursor = conexion.cursor ()
    cursor.execute ("SELECT * FROM productos WHERE cantidad <= ?",
    ↪  (cantidad_maxima,))
    productos = cursor.fetchall ()
    conexion.close ()
    return productos
```

5 Conclusión

En este informe realizado al combinar LaTeXy Python se presenta una simplificación del programa inventario, donde se pudo aplicar lo aprendido en clases, como listas, bases de datos SQL, condicionales, bucles, ademas se uso modulacion y funciones, además de el uso de la función main y un makefile para ejecutar de un forma más práctica el código.

El programa con archivo CSV tiene la desventaja que ante un corte de energía eléctrica la información de la lista se pierde, mientras que un programa con SQL no necesita una lista, sino que trabaja directamente con una conexión a una base de datos y por ende no hay perdida de información en caso de un corte de energía eléctrica.