

Universidad de San Carlos de Guatemala  
Facultad de Ingeniería  
Curso: Lab LFP  
Sección: B+  
Ing. David Morales  
Aux. Francisco Magdiel Asicono Mateo  
Segundo Semestre 2023



## **Manual Técnico Practica**

**Andres Alejandro Quezada Cabrera**

**202200174**



Guatemala, 22 de Agosto del 2023

## INTRODUCCIÓN

Bienvenido al Manual Técnico del Sistema de Gestión de Inventario. Este documento está diseñado para proporcionar una comprensión detallada de la estructura, funcionalidad y procesos involucrados en nuestro sistema de gestión de inventario. El sistema ha sido desarrollado para optimizar el proceso de seguimiento, control y gestión de inventario de productos en tu organización. A través de una interfaz intuitiva y herramientas eficientes, este sistema facilita la tarea crucial de administrar el flujo de productos y mantener registros precisos.

El sistema cuenta con varias características esenciales para un manejo efectivo del inventario. Algunas de estas características incluyen:

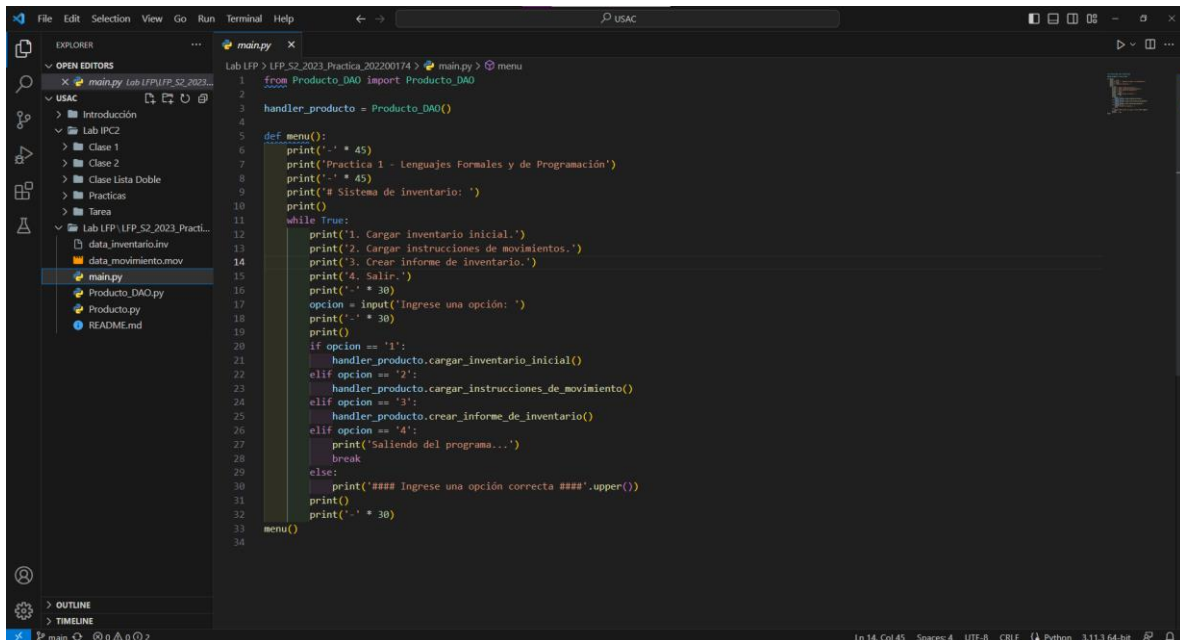
- **Menú Inicial:** Al iniciar el programa, se presenta un menú intuitivo que proporciona acceso rápido a las diversas funciones disponibles en el sistema.
- **Carga de Inventario Inicial:** Una funcionalidad que permite cargar información sobre el inventario inicial de productos. Esto se realiza a través de un archivo en formato .inv, facilitando la carga masiva de datos y evitando errores manuales.
- **Instrucciones de Movimiento:** El sistema permite cargar instrucciones de movimiento que afectan al inventario, como agregar stock o vender productos. Estas instrucciones se pueden cargar a través de un archivo en formato .mov, lo que garantiza un registro preciso y eficiente de todas las transacciones.
- **Generación de Informe de Inventario:** Se proporciona la opción de crear un informe de inventario que captura todos los detalles del inventario actual. Este informe se guarda en un archivo en formato .txt para su posterior referencia y análisis.

## INSATACIÓN DEL PROGRAMA Y USO DEL PROGRAMA

Ir al siguiente enlace para obtener el programa

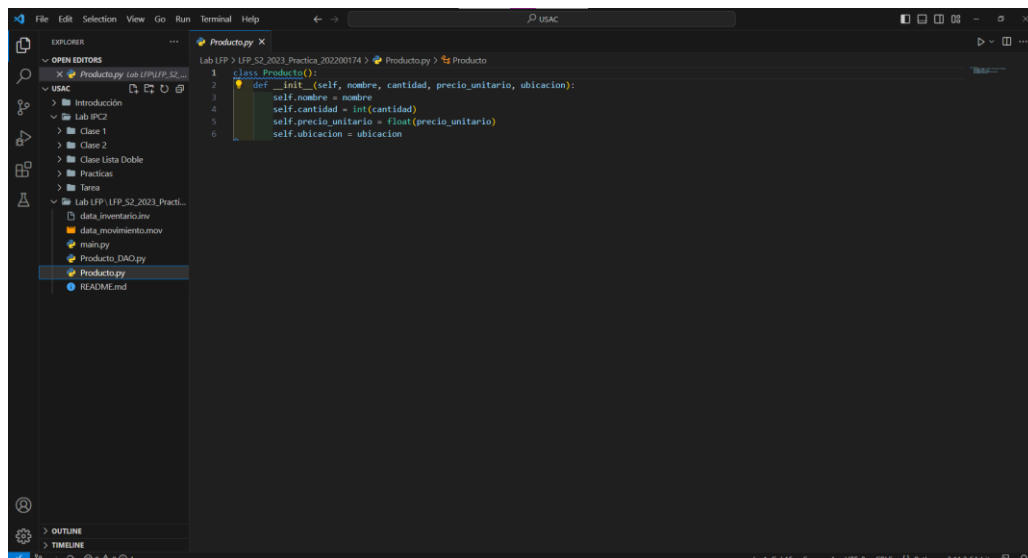
[https://github.com/MrQS94/LFP\\_S2\\_2023\\_Practica\\_202200174](https://github.com/MrQS94/LFP_S2_2023_Practica_202200174)

En el archivo main.py, podremos encontrar que existe un menú, en el cual nos servirá para acceder a las funcionalidades del programa.



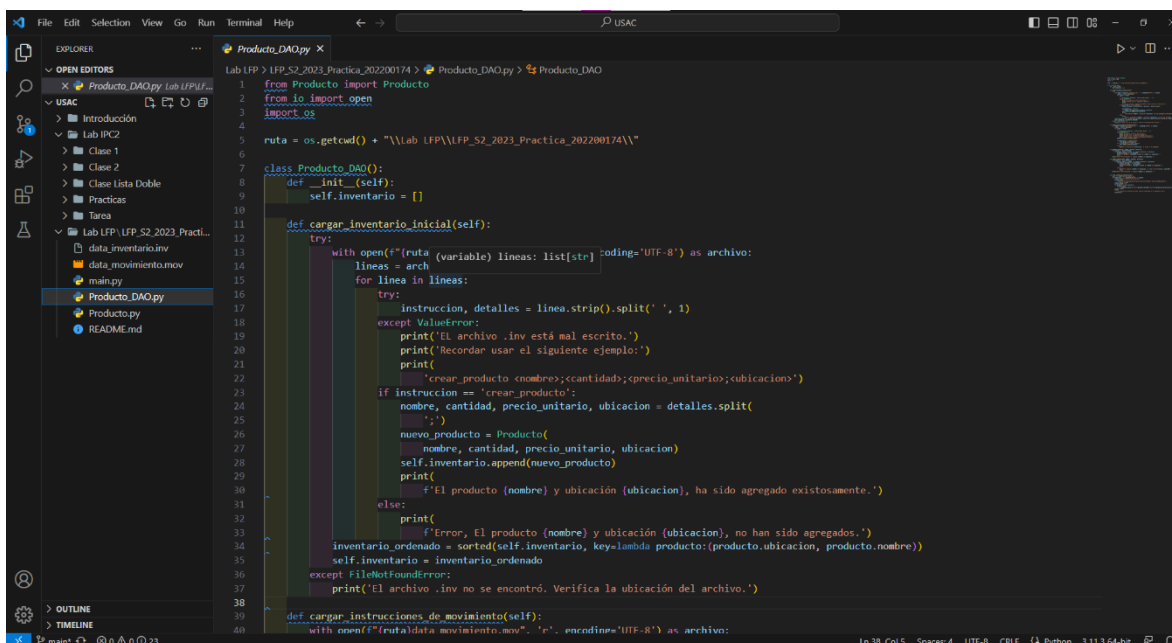
```
1 from Producto.DAO import Producto.DAO
2
3 handler_producto = Producto.DAO()
4
5 def menu():
6     print('\n * 45')
7     print('Practica 1 - Lenguajes Formales y de Programación')
8     print('\n * 45')
9     print('# Sistema de inventario: ')
10    print()
11    while True:
12        print('1. Cargar inventario inicial.')
13        print('2. Cargar instrucciones de movimientos.')
14        print('3. Crear informe de inventario.')
15        print('4. Salir.')
16        print('\n * 30')
17        opcion = input('Ingrese una opción: ')
18        print('\n * 30')
19        print()
20        if opcion == '1':
21            handler_producto.cargar_inventario_inicial()
22        elif opcion == '2':
23            handler_producto.cargar_instrucciones_de_movimiento()
24        elif opcion == '3':
25            handler_producto.crear_informe_de_inventario()
26        elif opcion == '4':
27            print('Saliendo del programa...')
28            break
29        else:
30            print('### Ingrese una opción correcta ###'.upper())
31        print()
32    print('\n * 30')
33 menu()
34
```

En el archivo Producto.py vamos a encontrar el modelo que vamos a utilizar para guardar los productos, en este caso, sería el nombre, cantidad, precio\_unitario, ubicación, este último será uno de los más importantes del programa.



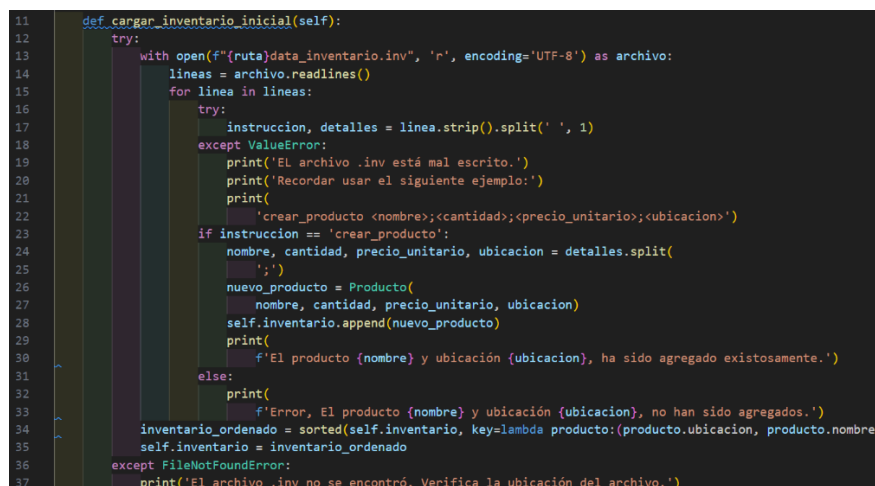
```
1 class Producto():
2     def __init__(self, nombre, cantidad, precio_unitario, ubicacion):
3         self.nombre = nombre
4         self.cantidad = int(cantidad)
5         self.precio_unitario = float(precio_unitario)
6         self.ubicacion = ubicacion
```

En el archivo `Producto_DAO.py` será nuestro “cerebro” el cuál será utilizado para recolectar, actualizar e imprimir la información.



```
1 from Producto import Producto
2 from io import open
3 import os
4
5 ruta = os.getcwd() + "\\Lab LFP\\LFP_S2_2023_Practica_202200174\\"
6
7 class Producto_DAO():
8     def __init__(self):
9         self.inventario = []
10
11     def cargar_inventario_inicial(self):
12         try:
13             with open(f"{ruta}data_inventario.inv", 'r', encoding='UTF-8') as archivo:
14                 lineas = archivo.readlines()
15                 for linea in lineas:
16                     try:
17                         instruccion, detalles = linea.strip().split(' ', 1)
18                     except ValueError:
19                         print('El archivo .inv está mal escrito.')
20                         print('Recordar usar el siguiente ejemplo:')
21                         print('crear_producto <nombre>;<cantidad>;<precio_unitario>;<ubicacion>')
22                     if instruccion == 'crear_producto':
23                         nombre, cantidad, precio_unitario, ubicacion = detalles.split(
24                             ';')
25                         nuevo_producto = Producto(
26                             nombre, cantidad, precio_unitario, ubicacion)
27                         self.inventario.append(nuevo_producto)
28                         print(f'El producto {nombre} y ubicación {ubicacion}, ha sido agregado existosamente.')
29                     else:
30                         print(f'Error, El producto {nombre} y ubicación {ubicacion}, no han sido agregados.')
31                 inventario_ordenado = sorted(self.inventario, key=lambda producto: (producto.ubicacion, producto.nombre))
32                 self.inventario = inventario_ordenado
33             except FileNotFoundError:
34                 print('El archivo .inv no se encontró. Verifica la ubicación del archivo.')
35
36     def cargar_instrucciones_de_movimiento(self):
37         with open(f"{ruta}data_movimiento.mov", 'r', encoding='UTF-8') as archivo:
```

En el archivo `Producto_DAO.py`, tenemos la función “`cargar_inventario_inicial`”, el cual se encarga por medio de `with open(ruta)` y `readLines()` de conocer donde se encuentra el archivo y leer lo que tiene el archivo, en este caso tenemos un `.inv`. En el momento que ya sido leído, la variable `instrucción` se encarga de conocer lo que vamos a realizar, en este caso será `crear_producto`, y la función `detalles` se encarga de conocer, el nombre, cantidad, `precio_unitario` y `ubicación`, muy importante porque con ello podremos saber cuales productos vamos a añadir. Luego de eso, lo insertamos dentro de un array llamado `inventario`, que anteriormente creamos y ahí es donde se guarda toda nuestra información.



```
11 def cargar_inventario_inicial(self):
12     try:
13         with open(f"{ruta}data_inventario.inv", 'r', encoding='UTF-8') as archivo:
14             lineas = archivo.readlines()
15             for linea in lineas:
16                 try:
17                     instruccion, detalles = linea.strip().split(' ', 1)
18                 except ValueError:
19                     print('El archivo .inv está mal escrito.')
20                     print('Recordar usar el siguiente ejemplo:')
21                     print('crear_producto <nombre>;<cantidad>;<precio_unitario>;<ubicacion>')
22                 if instruccion == 'crear_producto':
23                     nombre, cantidad, precio_unitario, ubicacion = detalles.split(
24                         ';')
25                     nuevo_producto = Producto(
26                         nombre, cantidad, precio_unitario, ubicacion)
27                     self.inventario.append(nuevo_producto)
28                     print(f'El producto {nombre} y ubicación {ubicacion}, ha sido agregado existosamente.')
29                 else:
30                     print(f'Error, El producto {nombre} y ubicación {ubicacion}, no han sido agregados.')
31             inventario_ordenado = sorted(self.inventario, key=lambda producto: (producto.ubicacion, producto.nombre))
32             self.inventario = inventario_ordenado
33         except FileNotFoundError:
34             print('El archivo .inv no se encontró. Verifica la ubicación del archivo.')
```

En el archivo `Producto_DAO.py`, tenemos la función “`cargar_instrucciones_de_movimiento`”, el cuál es el encargado de agregar o vender stock de un producto, realiza una vez más la lectura del archivo, y en este caso tenemos 2 funciones llamadas “`agregar_stock`” y “`vender_producto`”, las cuáles se encargan de recorrer un for por cada producto que existe e ir comparando si existe en esa ubicación y si el nombre es el correcto, con ello imprime cuanto se añadió en su stock, lo mismo pasa en “`vender_producto`” solo que en este existe un if adicional que se encarga de que el stock que se piensa vender sea menor al existente, y dentro de “`cargar_instrucciones_de_movimiento`”, escribimos un if anidado que se encarga de ejecutar y comparar cuales instrucciones contiene el archivo `.mov`.

```

39 def cargar_instrucciones_de_movimiento(self):
40     with open(f"{ruta}data_movimiento.mov", 'r', encoding='UTF-8') as archivo:
41         lineas = archivo.readlines()
42         for linea in lineas:
43             try:
44                 instruccion, detalles = linea.strip().split(' ', 1)
45             except ValueError:
46                 print('El archivo .mov está mal escrito.')
47                 print('Recordar usar el siguiente ejemplo:')
48                 print('agregar_stock <nombre>;<cantidad>;<ubicacion>')
49                 print('vender_producto <nombre>;<cantidad>;<ubicacion>')
50                 partes = detalles.strip().split(';')
51
52                 if instruccion == 'agregar_stock':
53                     self.agregar_stock(*partes)
54                 elif instruccion == 'vender_producto':
55                     self.vender_producto(*partes)
56                 else:
57                     print(f'La instrucción {instruccion}, no existe en el programa')
58
59 def agregar_stock(self, nombre, cantidad, ubicacion):
60     for producto in self.inventario:
61         if producto.nombre == nombre and producto.ubicacion == ubicacion:
62             producto.cantidad += int(cantidad)
63             print(f"Se agregaron {cantidad} unidades de {nombre} en {ubicacion}.")
64             return
65     print(f"Error: No se encontró el producto {nombre} en {ubicacion}.")
66
67 def vender_producto(self, nombre, cantidad, ubicacion):
68     for producto in self.inventario:
69         if producto.nombre == nombre and producto.ubicacion == ubicacion:
70             if producto.cantidad >= int(cantidad):
71                 producto.cantidad -= int(cantidad)
72                 print(f"Se vendieron {cantidad} unidades de {nombre} en {ubicacion}.")
73                 return
74             else:
75                 print(f"El producto {nombre} en {ubicacion}, no tiene el stock deseado ({cantidad}).")
76                 return
77     print(f"Error: No se encontró el producto {nombre} en {ubicacion}.")

```

Por último, tenemos la función “`crear_informe_de_inventario`”, esto se encarga reconocer que exista el archivo “`resultado_123123.txt`”, sino crea el archivo, recorre todo el inventario buscando cada nombre, cantidad, precio\_unitario, total y ubicación del producto.

```

40 def crear_informe_de_inventario(self):
41     ruta = f"{ruta}resultado_123123.txt"
42     with open(ruta, 'a', encoding='UTF-8') as archivo:
43         archivo.write("Informe de Inventario:\n")
44         archivo.write(
45             "Producto\t\tCantidad\t\tPrecio Unitario\t\tValor Total\t\tUbicación\n")
46         archivo.write("\n")
47         for producto in self.inventario:
48             archivo.write(
49                 f"{producto.nombre}\t\t{producto.cantidad}\t\t{producto.precio_unitario}\t\t{int(producto.cantidad) * float(producto.precio_unitario)}\t\t{producto.ubicacion}\n")
50         archivo.close()
51     print(
52         f"El informe del inventario ha sido creado o actualizado en la siguiente ruta:")
53     print(ruta)

```