MÓDULO 4 ACTIVIDAD GRUPAL AE6_GRUPAL

Integrantes:
Emily Quintana
Jorge Cardenas
Alvaro Ortega

Participación de Emily:

Crear archivos de código:
leer.py

Definí la función leer_inventario(): dentro de un bloque de excepción try/except donde me abría el archivo txt de inventario y lo leía y lo devuelve como una línea de string (porque tiene palabras y números).

y la función leer_inventario_completo(): donde imprime cada línea del inventario (dentro de un for), le puse .strip para imprimir pero que antes le quite espacios en blancos.

```
def leer_inventario():
    try:
        with open("inventario.txt", "r", encoding="utf-8") as archivo:
            return archivo.readlines()
    except FileNotFoundError:
        print("El archivo 'inventario.txt' no existe.")
        return []


def leer_inventario_completo():
    print("\nINVENTARIO COMPLETO")
    lineas = leer_inventario()
    if lineas:
        for linea in lineas:
            print(linea.strip())
    else:
        print("El inventario está vacío o el archivo no existe.")
```

visualizar.py
primero que nada importar leer, puesto que lo que iba a visualizar “ el producto” debía buscar si estaba o no dentro del inventario.

defini la función visualizar_producto(): donde se le pide al usuario escribir el producto que desea verificar/visualizar y dentro de un if ingrese la condición de si ese nombre está en el inventario (recorriendo con for) lo imprima con toda su información sino pues que indique que ese producto no esta en inventario.

```
from leer import leer_inventario

def visualizar_producto():
    print("\nVISUALIZAR PRODUCTO")
    nombre_busqueda = input("Ingrese el nombre del producto que desea ver: ")
    encontrado = False

    lineas = leer_inventario()
    if lineas:
        for linea in lineas:
```

```
            if nombre_busqueda.lower() in linea.lower():
                print("Detalles del producto:")
                print(linea.strip())
                encontrado = True
                break
        if not encontrado:
            print(f"Producto '{nombre_busqueda}' no encontrado en el inventario.")
    else:
        print("El inventario está vacío o el archivo no existe.")
```

Main.py trabajo en conjunto con mi compañero Jorge.

Primero, importar cada archivo creado más la librería OS;
Luego, se procedió a crear un menú en el cual el usuario pueda hacer input de lo que quiere realizar, esto dentro de un While/True o bloque infinito hasta que el usuario desee salir de él. Es importante señalar que cada alternativa hace un llamado a una función:

```
import os
import registrar #Jorge
import leer        #Emily
import visualizar  #Emily
import modificar #Jorge
import backup    #Jorge
import consultar #Álvaro
import eliminar  #Álvaro
```

M4_AE6_Grupal > main.py

```
1   import os
2   import registrar #jorge
3   import leer        #Emily
4   import visualizar  #emily
5   import modificar #jorge
6   import backup    #jorge
7   #import consultar #Álvaro
8   #import eliminar  #Álvaro
9
    Windsurf: Refactor | Explain | Generate Docstring | ×
10  def menu_principal():
11      while True:
12          print("\nSISTEMA DE GESTIÓN DE INVENTARIO Moda Xpress")
13          print("1. Registrar nuevo producto")
14          print("2. Modificar producto")
15          print("3. Visualizar producto")
16          print("4. Consultar inventario completo")
17          print("5. Crear respaldo")
18          print("6. Consultar producto")
19          print("7. Eliminar producto")
20          print("8. Salir")
21
22          opcion = input("Ingrese una opción: ")
23
24          if opcion == '1':
25              registrar.registrar_producto()
26          elif opcion == '2':
27              modificar.modificar_producto()
28          elif opcion == '3':
29              visualizar.visualizar_producto()
30          elif opcion == '4':
31              leer.leer_inventario_completo()
32          elif opcion == '5':
33              backup.backup()
34          elif opcion == '6':
35              consultar.consultar_producto()
36          elif opcion == '7':
```

Participación de Jorge:

Creación del Repositorio;
Código en módulos: backup(); modificar(); registrar();

backup()

Este módulo importa las librerias ‘os’ y ‘datetime’, para, luego, definir una función backup(), la cual almacena la hora actual en variable x (linea 5), y la formatea como string con formato ‘Año-mes-día-HoraMinutoSegundo’.

posteriormente, se crea el archivo a respaldar como “inventario_bak{x}.txt”
Finalmente, se informa al Usuario respecto del archivo creado, junto con su tamaño.

modificar()

La función ‘modificar_producto()’ permite actualizar un producto en el archivo "inventario.txt". Primero, lee el archivo y muestra todos los productos numerados para que el usuario seleccione cuál modificar. Luego, se valida que el número ingresado esté dentro del rango existente. Si es válido, solicita los nuevos datos (asegurándose de que no estén vacíos), reemplazando la línea correspondiente en la lista y sobrescribiendo el archivo completo con los datos actualizados. Finalmente, se imprime en pantalla el inventario modificado. Cabe señalar que el código incluye manejo de excepciones para errores comunes como archivo no encontrado, entrada inválida o índices incorrectos.

Muestras del código:

1. Validación de entrada y rango:
‘if num_producto < 1 or num_producto > len(lineas):’

Verifica que el número de producto seleccionado esté dentro de los límites de la lista, evitando accesos inválidos.

2. Prevención de datos vacíos:
‘while len(nuevo_dato) == 0 : …’
Garantiza que el usuario no ingrese cadenas vacías, asegurando que el producto tenga datos válidos antes de actualizar.

3. Sobrescritura segura del archivo:
‘with open('inventario.txt', 'w') as archivo: archivo.writelines(lineas)’
En esta sección se utiliza el modo de escritura (`'w'`) para reemplazar todo el contenido del archivo con la lista actualizada, manteniendo la integridad de los datos restantes.

registrar()

La función `registrar_producto()` permite añadir nuevos productos al archivo de inventario. Primero , se solicita al usuario cuatro datos: nombre/color, precio, cantidad y talla. Luego escribe estos datos como una nueva línea en el archivo `inventario.txt` usando el modo de anexión (`'a'`, de *append*) para preservar el contenido existente. Finalmente, reabre el archivo en modo lectura para recuperar y mostrar el último registro añadido como confirmación. Es importante señalar que el código no incluye manejo de errores, lo que podría causar fallos en casos como permisos de archivo o entradas inesperadas.

Partes importantes citadas:

1. **Apertura segura en modo anexión**:

`with open('inventario.txt', 'a') as file:`
Con esto, se garantiza que el archivo se cierre automáticamente tras escribir y permite añadir datos sin borrar el contenido existente.

2. **Formato estructurado de datos**:
`file.write(f'{producto}, {precio}, {cantidad}, {talla}\n')`
Combina las entradas del usuario en una línea organizada con comas, manteniendo consistencia en el almacenamiento.

3. **Verificación de la inserción**:
`ultima_linea = file.readlines()[-1]`
Lee el último registro del archivo para confirmar visualmente que los datos se guardaron correctamente, proporcionando retroalimentación inmediata.

Participación de Álvaro

Función `leer_inventario_completo()`

Objetivo: Mostrar todos los productos del inventario almacenados en el archivo `inventario.txt`.

Pasos:

1. Abrir el archivo: Utilicé `with open("inventario.txt", "r", encoding="utf-8") as archivo:` para abrir el archivo en modo lectura. El uso de `with` asegura que el archivo se cierre automáticamente después de ejecutar el bloque de código, liberando así recursos.
2. Leer el contenido: Con `archivo.readlines()`, leí todo el contenido del archivo y lo almacené en la variable `lineas`. Este método devuelve una lista donde cada elemento es una línea del archivo.
3. Verificar si el inventario está vacío: Si `lineas` está vacía, imprimo "El inventario está vacío." y salgo de la función.
4. Imprimir el inventario: Si hay productos, imprimo "INVENTARIO COMPLETO:" seguido de cada línea (producto) del inventario. Utilicé `linea.strip()` para eliminar caracteres de nueva línea al final de cada línea.
5. Manejo de excepciones: Implementé bloques `except` para manejar errores como `FileNotFoundError` (si el archivo no existe) y otros errores inesperados, imprimiendo un mensaje correspondiente en cada caso.

Función `eliminar_producto()`

Objetivo: Eliminar un producto específico del inventario basado en el número proporcionado por el usuario.

Pasos:

1. Leer el inventario: Abrí el archivo en modo lectura, leí su contenido y lo almacené en la lista `lineas` sin incluir caracteres de nueva línea final (`strip()`).

2. Listar productos: Imprimí "Lista de productos:" seguido del número y detalles de cada producto en el inventario.
3. Obtener la selección del usuario: Solicité al usuario que ingresara el número del producto que desea eliminar con `input()`.
4. Validar la selección: Verifiqué si el número ingresado era válido (entre 1 y el total de productos). Si no lo era, imprimí un mensaje de error y salí de la función.
5. Eliminar el producto: Creé una nueva lista `new_lines` que incluía todas las líneas del inventario excepto la del producto a eliminar. Utilicé una lista de comprehension para filtrar la línea correspondiente al producto seleccionado.
6. Guardar el inventario actualizado: Abrí el archivo en modo escritura (`w`) y escribí de nuevo el contenido actualizado (sin la línea del producto eliminado). Esto sobrescribe el archivo original con el contenido filtrado.
7. Confirmar la eliminación: Imprimí un mensaje de confirmación indicando que el producto había sido eliminado con éxito.
8. Manejo de excepciones: Implementé bloques `except` para manejar errores como `FileNotFoundError` (si el archivo no existe), `ValueError` (si el usuario no ingresa un número entero) y otros errores inesperados, imprimiendo un mensaje correspondiente en cada caso.

Participación en GitHub/Repositorio


GRAPH
Automático
Ajustes al main e inventario Ortega Hamel
main
inventario.txt M
inventario_bak_2025-08-11-200331.txt M
main.py M
Merge branch 'main' into Alvaro Ortega Hamel
.gitignore A
README.md M
backup.py A
inventario.txt A
inventario_bak_2025-08-11-200331.txt A
leer.py A
main.py A
modificar.py A
registrar.py A
visualizar.py A
Agregando el archivo inventario.txt de nuevo Emy Quintana
inventario.txt A
Actualizando el sistema de inventario y arreglando archivos Emy Quintana
backup.py M
inventario.txt D
leer.py M
main.py M
principal_modificado.py D
visualizar.py M
Para facilitar integración, se ha eliminado 'hora:minuto:segundo' del archiv...
backup.py M
inventario_bak_2025-08-11-200331.txt A
inventario_bak_2025080913:52:45.txt D
Merge branch 'Jorge' JorgeCardenasY
.gitignore A
README.md M
backup.py A
git.pdf A
inventario.txt A
inventario_bak_2025080913:52:45.txt A
main.py A
modificar.py A
registrar.py A
Sección Jorge al 09-08-25 JorgeCardenasY
.gitignore A
README.md M
backup.py A
git.pdf A
inventario.txt A
inventario_bak_2025080913:52:45.txt A
main.py A
modificar.py A
registrar.py A
subiendo mi parte Emy Quintana
leer.py A
principal_modificado.py A
visualizar.py A
Segunda subida de archivos Alvaro Ortega Hamel
consultar.py A
eliminar.py A
Subiendo funciones Alvaro Ortega Hamel
git.pdf A
Creacion_del_Repositorio JorgeCardenasY
README.md A