

EQUIPO 6

PROYECTO

FINAL

Diego Alejandro Herrera Gutiérrez – Senior

Regina Hernández Hernández – Mid

Diego Martínez Millán – Mid

Gerardo Gaytan Alvarez – Junior

Axel Usiel Robles Sinecio – Junior

Contexto y problema

Dentro de una tienda de abarrotes se han identificado diferentes problemas que afectan de manera notable la organización del surtido y la venta dentro de una tienda de abarrotes, el cajero tiene complicaciones de salud y pierde mucho tiempo para poder buscar y cobrar correctamente el producto deseado del cliente de una manera eficaz y también pierde la capacidad de tener un control mucho más estable dentro de las ventas. Se busca resolver los problemas mencionados anteriormente y brindar un área más agradable para los que asistan a la tienda de abarrotes.

Reglas del negocio

- Cada producto debe de registrarse con un nombre y cantidad
- No se pueden registrar nombres de productos ya registrados
- La cantidad en existencia no debe de ser negativa
- La mercancía nueva que llegue se debe de registrar
- No se pueden realizar ventas si la cantidad solicitada supera la del inventario
- Cada venta debe actualizar el inventario
- El programa debe de permitir la consulta del inventario en tiempo real
- Deben de aparecer mensajes de error cuando se intente ingresar información incorrecta

Requerimientos funcionales

- **Ingresar producto**
Esto permite registrar un producto de forma específica
- **Eliminar producto**
Esta función permite eliminar cualquier producto de la base de datos
- **Lista de productos**
Esta función genera una lista con los productos ya registrados
- **Modificar cantidad**
Esta función permite modificar el precio de cualquier producto registrado
- **Consultar producto**
Esta función permite consultar la cantidad de un producto además de mostrar la cantidad de productos por categoría
- **Delimitar resurtido**
Esta función modifica a que cantidad de determinado producto se avisara en caso de necesitar más producto

Entradas y salidas

Entradas:

- **Creador de archivo**

Nombre del archivo.

¿Se desea modificarlo?

¿Se desea sobreescribirlo?

- **Menú**

(Elegir el número de la opción que se desea implementar.)

- Agregar inventario.
- Editar inventario.
- Buscar.
- Eliminar inventario.
- Mostrar inventario.
- Salir.

Seleccionar opción.

- **Agregar inventario**

(Dentro del inventario se encuentran seis categorías: fríos, snacks, enlatados, limpieza, mascotas y calientes.)

¿Cuántos productos de {categoría} desea agregar?

Nombre del producto

Cantidad del producto

Cantidad mínima del producto

Precio

- **Editar inventario**

Objeto a editar cantidad

¿Se desea editar cantidad?

Nueva cantidad

¿Se desea editar el precio?

Editar precio

- **Buscar**

Nombre del producto a buscar / escribir palabra “salir”

- **Eliminar inventario**

Ingresar nombre del producto a eliminar

Salidas:

- **Creador de archivo**

Se crea y muestra el siguiente mensaje:

“Archivo '{archivo}' creado con éxito.”

Si se desea sobreescribir:

“Archivo {archivo} sobreescrito.”

Si no se modifica el archivo:

“Archivo {archivo} sin cambios”

- **Menú**

Se procede a mostrar la función correspondiente.

- **Agregar inventario**

En caso de que no haya ningún problema:

“¡Sus productos fueron registrados exitosamente!”

Si se solicita insertar una cantidad menor a la mínima:

“AVISO: '{nombre}' tiene {piezas} unidades. Es necesario resurtir (mínimo {minimo}).”

- **Editar inventario**

Si se actualiza sin ningún problema:

“Actualizado con éxito”

Si hay un error al escribir la nueva cantidad/precio:

Cantidad: “Favor de escribir un número entero”

Precio: “Favor de escribir solamente el número”

Si hay un error al escribir el nombre:

“No hay ningún producto con ese nombre”

Si el usuario desea no cambiar la cantidad/precio:

Cantidad: "No hay ninguna cantidad por actualizar"

Precio: "No hay ningún precio por actualizar...!"

Si se escribe incorrectamente cuando se pregunta si se quiere cambiar la cantidad/precio:

"Respuesta no válida, favor de escribir 's' o 'n'"

- **Buscar**

Si se escribe "salir" se rompe el ciclo.

Muestra los productos encontrados.

- **Eliminar inventario**

Si no hay objetos dentro de la biblioteca:

"No hay objetos registrados."

Si se realiza con éxito:

"Objeto eliminado."

Si el objeto no existe o se escribe de manera incorrecta:

"No se encontró el producto a eliminar."

- **Mostrar inventario**

Se muestran los productos por categoría con nombre, cantidad, cantidad mínima y precio.

Si no hay productos registrados:

"No hay productos registrados"

Si la cantidad del producto es menor al mínimo establecido:

">> ALERTA: '{producto['nombre']}' tiene {producto['cantidad']} unidades"

"(mínimo {producto['minimo']})."

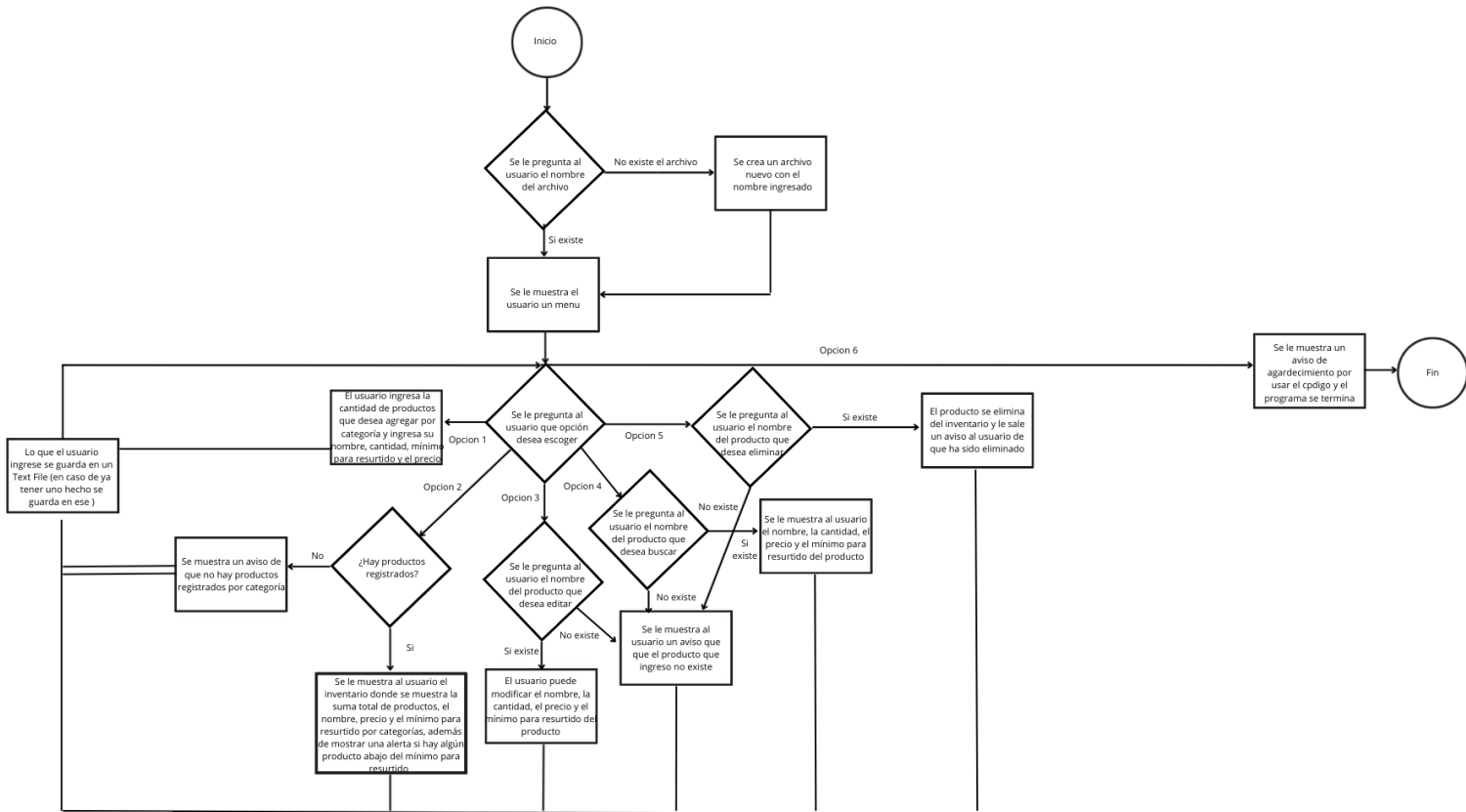
Total de piezas por categoría:

"Total de piezas en categoría '{categoria}': {suma_total}"

- **Salir**

Se termina el programa

Diagrama de Flujo



Ejecución en Python

```
C: > Users > 11127 > OneDrive > Documentos > PROYECTOFPD > EDITAR_BORRADOR.py > ...
66 def actualizar(inventario):
99     print("Precio actualizado con éxito.")
100     except ValueError:
101         print("Favor de escribir solamente el número")
102         break
103     elif precio == "n":
104         print("No hay ningún precio por actualizar...!")
105         break
106     else:
107         print("Respuesta no válida, favor de escribir 's' o 'n'.")
108
109     # Alerta de que se está quedando sin stock
110     if producto["cantidad"] < producto["minimo"]:
111         print(f"    >> ALERTA: '{producto['nombre']}' tiene {producto['cantidad']} unidades "
112               f"    (mínimo {producto['minimo']}).")
113
114     return # salir después de actualizar un producto
115
116
117 print("\nNo hay ningún producto con ese nombre.")
118
119 inventario = agregar_inventario()
120 #ostrar_inventario(inventario)
121 actualizar(inventario)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
> ¿Desea actualizar su cantidad? (s/n): s
Ingresar nueva cantidad: 5

Actualizado con éxito.

¿Desea actualizar el precio? (s/n): s
Ingresar nuevo precio: 60
Precio actualizado con éxito.
>> ALERTA: 'wasd' tiene 5 unidades (mínimo 32).
PS C:\Users\11127\OneDrive\Documentos\PROYECTOFPD>
```

```
C: > Users > 11127 > OneDrive > Documentos > PROYECTOFPD > EDITAR_BORRADOR.py > actualizar
66 def actualizar(inventario):
97     producto["precio"] = nuevo_precio
98     print("Precio actualizado con éxito.")
99
100     except ValueError:
101         print("Favor de escribir solamente el número")
102         break
103     elif precio == "n":
104         print("No hay ningún precio por actualizar...!")
105         break
106     else:
107         print("Respuesta no válida, favor de escribir 's' o 'n'.")
108
109
110     if producto["cantidad"] < producto["minimo"]:
111         print(f"    >> ALERTA: '{producto['nombre']}' tiene {producto['cantidad']} unidades "
112               f"    (mínimo {producto['minimo']}).")
113
114     return #salir después de actualizar
115
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

TERMINAL

```
> ¿Cuántos productos de "snacks" quiere registrar?: 0
¿Cuántos productos de "enlatados" quiere registrar?: 0
¿Cuántos productos de "limpieza" quiere registrar?: 0
¿Cuántos productos de "mascotas" quiere registrar?: 0
¿Cuántos productos de "calientes" quiere registrar?: 0
¿A qué objeto desea actualizar su cantidad?: Grassel

No hay ningún producto con ese nombre.
PS C:\Users\11127\OneDrive\Documentos\PROYECTOFPD> python EDITAR_BORRADOR.py
```



```
49         if producto["nombre"].lower() == product_actualizar:
50             try:
51                 numero = int(input("Ingresar nueva cantidad: "))
52                 producto["cantidad"] = numero
53                 print("Actualizado con éxito")
54                 return
55             except ValueError:
56                 print("Favor de escribir un número entero")
57                 return
58         else:
59             print("\nNo hay ningún producto con ese nombre")
60
61
62
63 inventario = agregar_inventario()
64 suma(inventario)
65 actualizar(inventario)
66 suma(inventario)
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

No hay productos registrados

¿A qué objeto desea actualizar su cantidad?:coco
Ingresar nueva cantidad: 10
Actualizado con éxito
Categoria: frios
Suma total: 10
- coco: 10
Categoria: snacks
Suma total: 0
No hay productos registrados

```
Inventario.py x
Mi_proyecto > src > Inventario.py > ...
185 def menu():
220     else:
221         print("No se encontró el producto")
222     elif opcion == "5":
223         inventario = cargar_inventario(archivo_inventario)
224         if inventario:
225             eliminar_objeto(inventario)
226             guardar_inventario(inventario, archivo_inventario)
227     elif opcion == "6":
228         if not inventario:
229             inventario = inventario_vacio()
230             guardar_inventario(inventario, archivo_inventario)
231             print("¡Gracias por usar nuestro programa!")
232             break
233         else:
234             guardar_inventario(inventario, archivo_inventario)
235             print("¡Gracias por usar nuestro programa!")
236             break
237     else:
238         print("Opción no válida. Intente de nuevo.")
239 menu()
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

/src/Primera_parte.py"
Cual es el nombre del archivo?
papas
Archivo papas.txt ya existe
Desea modifica el contenido del archivo? (s/n)
s

===== MENÚ DE INVENTARIO =====
1. Agregar productos
2. Mostrar inventario
3. Actualizar cantidad de un producto
4. Buscar producto
5. Eliminar producto
6. Salir

Aprendizajes

En este trabajo logramos aplicar algunos temas que aprendimos en clase, Los cuales son:

–**For**: Recorrer las categorías en las funciones, Recorrer los productos en las categorías, Y repetir un número de veces según la cantidad de productos a registrar.

Ejemplos:

```
for categoria in categorias:
```

```
for producto in productos:
```

```
for i in range(cantidad):
```

–**While**: Se usa para repetir un código mientras se cumpla una condición.

Ejemplo:

```
while True: # Menú principal siempre activo hasta que el usuario decida salir
    print("\n==== MENÚ DE INVENTARIO =====")
    print("1. Agregar productos")
    print("2. Mostrar inventario")
    print("3. Actualizar cantidad de un producto")
    print("4. Buscar producto")
    print("5. Eliminar producto")
    print("6. Salir")
```

–**If (elif,else)**: Se usa para tomar una decisión siempre y cuando se cumplan las condiciones que se solicita:

Ejemplos:

```
if len(productos) == 0:
    print("No hay productos registrados")

if producto["cantidad"] < producto["minimo"]:

if opcion == "1":
    ...
elif opcion == "2":
    .....
else:
    print("Opción no válida")
```

–**Def:** Sirven para hacer funciones para utilizarlo en el código y no tener que andar copiando y pegando y hacer el código más largo.

Ejemplo:

```
def eliminar_objeto(inventario):
```

```
    if not inventario:
```

```
        print("\n> No hay productos registrados.")
```

```
    retur
```

–**Except/Try:** Sirve para manejar errores y así que no se detenga el programa hasta que se le diga.

Ejemplo:

```
if cantidad > 0:
```

```
    for i in range(cantidad):
```

```
        nombre = input("\nNombre del producto: ")
```

```
        try:
```

```
            piezas = int(input("Cantidad en stock: "))
```

```
        except ValueError:
```

```
            piezas = 0
```

–**Diccionario:** Almacena datos con palabras clave.

Ejemplo:

```
{  
    "nombre": "Refresco",  
    "cantidad": 10,  
    "minimo": 5,  
    "precio": 15.5  
}
```

–**Archivo de Texto:** Es un archivo de texto que se crea o se accede para almacenar datos. Lo usamos para guardar el inventario permanentemente.

Ejemplo:

```
def creador_de_archivo(nombre_archivo = ""):
```

```
    if nombre_archivo == "":
```

```
        nombre_archivo = input("Con que nombre desea llamar el archivo?\n").strip()
```

```
        archivo = (f"{nombre_archivo}.txt")
```

```
    try:
```

```
        with open(archivo, "x", encoding="utf-8") as file:
```

```
            pass
```

```
        print(f"Archivo '{archivo}' creado con éxito.")
```

Conclusión

Con este proyecto aprendimos a crear un gestor de inventario en Python (Algo rudimentario pero funcional) con los conceptos antes mencionados. Además, reforzamos el entendimiento de los programas y cómo aprovechar las listas, bibliotecas y archivos de texto para almacenar y organizar datos, y representarlo de una manera entendible.

No solo nos exige usar la lógica de programación, sino que también a fortalecer la colaboración en equipo ya que cada uno se tuvo que encargar en idear una solución para cada parte del código para que al final unir nuestras ideas y que resultara en este programa (El cual salió bastante bien a nuestro parecer). En conclusión, Gracias al trabajo en equipo y a las clases de nuestro Gran Profesor que fuimos capaces de programar este código y entendemos mejor los fundamentos de la programación.