Manual Técnico

Bienvenido al manual técnico del Sistema de gestión de Inventario. Acá se presentarán los distintos módulos utilizados para llevar a cabo cada una de las acciones necesarias para el funcionamiento de dicho proyecto.

Menú principal:

Se creó un módulo menú principal con el fin de presentar un mensaje de bienvenida y que importará del módulo opciones, una función denominada escoger_opciones para dar continuidad a la lógica del proyecto.

menu principal.py

```
import opciones #importando el modulo necesario
print("-." * 25)
print("Practica 1 - Lenguajes formales y de Programacion")
print(" "*13, "B I E N V E N I D O", " "*10)
print("-." * 25)
print("")
opciones.escoger_opciones() #se hace llamada a la funcion necesaria
```

Esta sería la presentación de la primer salida en la consola.

```
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.
Practica 1 - Lenguajes formales y de Programacion
             B I E N V E N I D O
-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.

     ¿Qué desea hacer?

1. Cargar Inventario Inicial
2. Cargar Instrucciones de Movimientos
3. Crear Informe de Inventario
4. Salir
Ingrese una opcion:
```

Opciones:

En este modulo se creó un ciclo para el cual se presenten las diversas opciones que el usuario puede tomar. Se inicia importando todos los demás módulos para mantener un código ordenado.

```
#importaciones necesarias
import cargarInventario
import cargarMov
import informe
```

En este caso presentara 4 opciones encerradas en un bucle while, las cuales serían: Cargar inventario Inicial, Cargan Instrucciones de movimientos, Crear informe de Inventario y Salir. Eso fue almacenado en una tupla con el fin de poder limitar las opciones y validarlas, ya que, si el usuario ingresa una opción que no se encuentra en la tupla, mostrará un mensaje de error.

```
def escoger_opciones():
    #estas son las unicas opciones validas, de no ser asi mostraraun error
    op = ("1","2","3","4")
    option = input("Ingrese una opcion:  ")
    if not option in op:
        print('Esa opcion no es valida')
    elif option == "1":
        print("Elegiste cargar inventario...")
        cargarInventario.leerDoc()
        #del modulo cargarInvetario se extraerá la funcion en cargada de mostrar
    elif option == "2":
        print("Elegiste movimientos")
        cargarMov.movDoc() #con este modulo se haran los movimientos en inventario
    elif option == "3":
        print("Elegiste generar informe")
        print("...Informe en proceso...")
        informe.generar_reporte()
        #se creara un archivo de texto con ayuda de este modulo
    elif option == "4":
        print("Regresa pronto")
        break
```

```
      ¿Qué desea hacer?

 1. Cargar Inventario Inicial
 2. Cargar Instrucciones de Movimientos
 3. Crear Informe de Inventario
 4. Salir
 Ingrese una opcion:  7
 Esa opcion no es valida
      ¿Qué desea hacer?

 1. Cargar Inventario Inicial
 2. Cargar Instrucciones de Movimientos
 3. Crear Informe de Inventario
 4. Salir
 Ingrese una opcion:  4
 Regresa pronto
```

Como no presenta una opción válida, el bucle se encargará de volver a preguntar hasta encontrar una respuesta valida y así ejecutar la acción que el usuario requiera.

CargarInventario:

Para este módulo se creo un diccionario principal denominado *Ubicaciones* esto con el fin de almacenar los datos necesarios para un inventario.

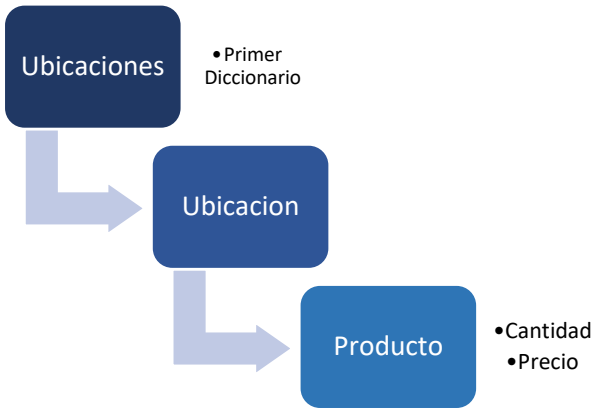
Se creó una variable “*archivo*” con el propósito de solicitar al usuario el nombre del documento que desea abrir y se encerró en try en un bucle while para poder validar que dicho archivo exista o esté bien escrito.

```
ubicaciones = {}
def leerDoc(): #se buscará el documento y se leerá
    while True:
        archivo = input("Ingrese el nombre del archivo: ") #se almacena su nombre
        try:
            with open(archivo + ".inv", encoding="utf-8") as file:
                #con esto se leera el archivo solicitado
                for linea in file:
                    valores = linea.strip().split(";") #se le quitarán las separaciones
                    if len(valores) == 4:  # Asegurarse de que hay exactamente 4 valores en la línea
                        producto, cantidad, precio, ubicacion = valores
                        inst, producto_Valido = producto.split(" ")
```

Una vez abierto el documento con el nombre y extensión valida, se procederá a recorrer el documento a través de un ciclo for para hacerlo línea por línea. Se creó una variable *valores* que almacenara una lista del contenido en donde separará por partes los lugares en donde encuentre “ ; ” para así darle un nombre a cada parte, que serían producto, cantidad, precio, ubicación. A la otra parte se le denominó *inst* en donde se encontrará la instrucción con el fin de separar el nombre del producto ya que es importante que se encuentre así. Se realiza una validación para comprobar que el formato en el documento sea el adecuado y contenga solo los datos útiles.

```
                    if ubicacion in ubicaciones:#diccionario anidadoso
                        ubicaciones[ubicacion][producto_Valido] = {
                            #si algo ya existe en la ubicacion
                            "cantidad": float(cantidad), #solo actualizara
                            "precio": float(precio)
                        }
                    else:
                        ubicaciones[ubicacion] = { #pero sino, debera crear un diccionario
                            #en donde guardarlo
                            producto_Valido: {"cantidad": float(cantidad), "precio": float(precio)}
                        }
                else:
                    print("Formato de línea incorrecto:", linea)
                #si el formato esta mal en el archivo lo demostrara
                break
        print("...Carga Exitosa...")
        print("--------------------")
    except FileNotFoundError:
        print("El documento no existe o se escribio de manera incorrecta")
    break
```

Se procederá con dos caminos, si la *ubicación* existe en el diccionario *ubicaciones*, entonces se accederá al *producto* para poder actualizar los datos sobre su *precio* y *cantidad* y sino existe entonces creará este diccionario en el cual la llave será otro diccionario con cantidad y precio.



Se concretará con el cierre del try manejando el error de un nombre incorrecto y un mensaje a consola indicando que el proceso se concretó con éxito.

CargarMov:

Se inicia importando el diccionario ubicaciones para poder realizar cambios en él, se abre el documento con la extensión requerida y se realizan las mismas validaciones a su nombre así como a su formato. En este caso se dividirá en dos partes las cuales serán instrucción y valores.

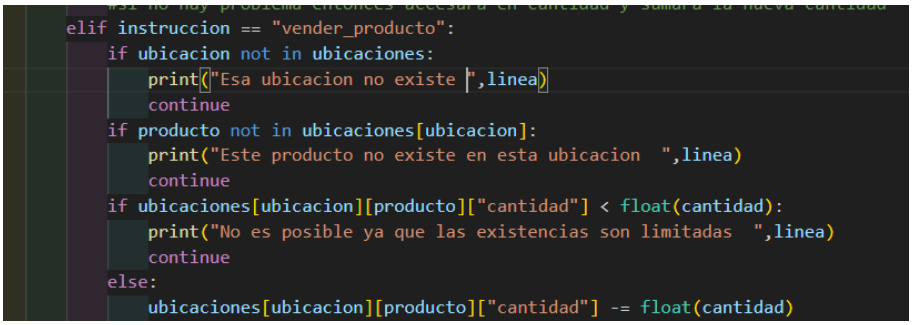
```
from cargarInventario import ubicaciones
def movDoc():
    while True:
        archivo = input("Ingrese el nombre del archivo: ")
        try:
            with open(archivo + ".mov", encoding="utf-8") as file:
                #con esto se leera el archivo solicitado
                for linea in file:
                    instruccion, resto = linea.strip().split(" ")  # Separar la instrucción del resto
                    valores = resto.split(";")  # Separar los valores usando ;
                    if len(valores) == 3:  # Asegurarse de que hay exactamente 4 valores en la línea
                        producto, cantidad, ubicacion = valores
```

En este caso es muy necesario dividir nuevamente las partes de cada línea del documento solicitado. Se dividirá la instrucción para validar qué se desea hacer, si agregar o vender lo cual contendrá una acción en base a condicionales.

```
c():
                #con la primera parte se podra valuar que es lo que se desea hacer
                    if instruccion == "agregar_stock":
            #los mensajes de error si en caso no es posible concretar la opcion
                        if ubicacion  not in ubicaciones :
                            print("No es posible concretar la accion ",linea )
                            continue
                        if producto not in ubicaciones[ubicacion]:
                            print("La ubicacion no coincide" ,linea)
                            continue
                        ubicaciones[ubicacion][producto]["cantidad"] += float(cantidad)
                        valor1= ubicaciones[ubicacion][producto]["cantidad"]
                        valor2 = cantidad
                        print(str(valor1)+" eso se sumrara" + str(valor2), linea)
                        #si no hay problema entonces accesara en cantidad y sumara la nueva cantidad
                    elif instruccion == "vender_producto":
```

Si la instrucción es agregar, entonces se validará que la ubicación exista, así mismo que el producto exista en la ubicación otorgada, y si todo marcha bien entonces se accederá al diccionario ubicaciones del modulo anterior y se accederá hasta la cantidad dentro del producto para poder

sumarla y así actualizar la cantidad. En el caso de que la instrucción sea vender se realizaran las mismas validaciones, pero también se comprobará que la cantidad en inventario no sea menor a loque se quiere vender, de lo contrario mostrará un error.



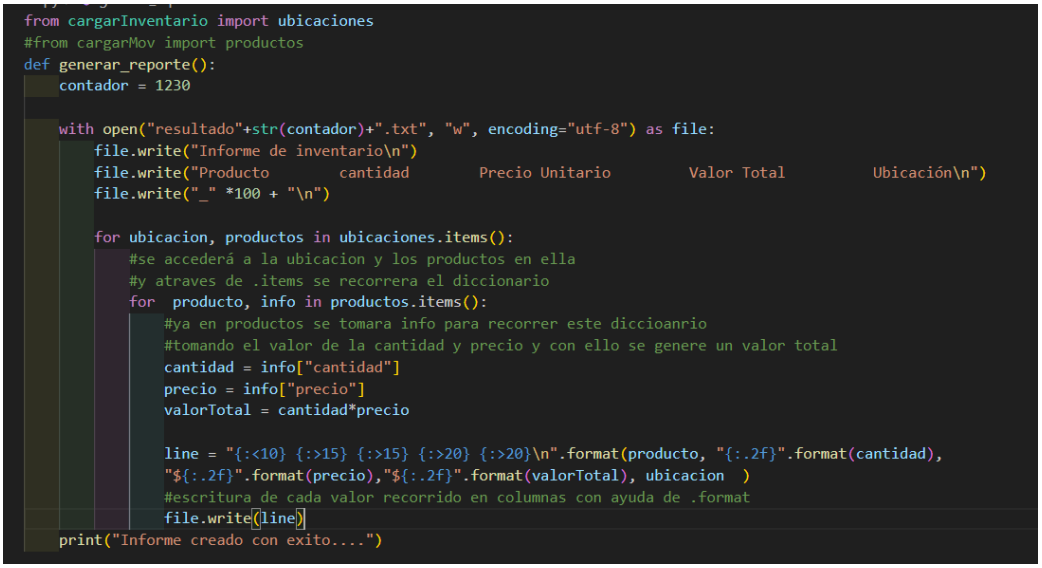
```
    elif instruccion == "vender_producto":
        if ubicacion not in ubicaciones:
            print("Esa ubicacion no existe ",linea)
            continue
        if producto not in ubicaciones[ubicacion]:
            print("Este producto no existe en esta ubicacion  ",linea)
            continue
        if ubicaciones[ubicacion][producto]["cantidad"] < float(cantidad):
            print("No es posible ya que las existencias son limitadas  ",linea)
            continue
        else:
            ubicaciones[ubicacion][producto]["cantidad"] -= float(cantidad)
```

Y se concretará cerrando el bucle e imprimiendo un mensaje de que el proceso ha sido concretado exitosamente.

Informe:

Se inició importando el inventario y se crea el documento con la extensión ".txt". Se añaden los títulos necesario utilizando *file.write*. Para los valores es necesario recorrer los diccionarios con *.item* ya que se accedera al valor por su llave teniendo en cuenta que producto es una llave en el diccionario ubicación. Así mismo se recorrerá producto, para tomar los valores del precio y la cantidad y con el producto de ambos se creará el dato: *ValorTotal*.

Se creara una lista llamada *line* en donde se almacenara cada dato y con ayuda de *.format* se escribirá como columna con distancias antes establecidas y así se escribirá en el documento por medio de *.write.*



```
from cargarInventario import ubicaciones
#from cargarMov import productos
def generar_reporte():
    contador = 1230

    with open("resultado"+str(contador)+".txt", "w", encoding="utf-8") as file:
        file.write("Informe de inventario\n")
        file.write("Producto        cantidad        Precio Unitario        Valor Total          Ubicación\n")
        file.write("_" *100 + "\n")

        for ubicacion, productos in ubicaciones.items():
            #se accederá a la ubicacion y los productos en ella
            #y atraves de .items se recorrera el diccionario
            for  producto, info in productos.items():
                #ya en productos se tomara info para recorrer este diccioanrio
                #tomando el valor de la cantidad y precio y con ello se genere un valor total
                cantidad = info["cantidad"]
                precio = info["precio"]
                valorTotal = cantidad*precio

                line = "{:<10} {:>15} {:>15} {:>20} {:>20}\n".format(producto, "{:.2f}".format(cantidad),
                "${:.2f}".format(precio),"${:.2f}".format(valorTotal), ubicacion  )
                #escritura de cada valor recorrido en columnas con ayuda de .format
                file.write(line)
    print("Informe creado con exito....")
```

Sé concluirá con un mensaje expresando que el informe se ha creado con éxito.