Manual del programador

proyecto 2

INTEGRANTES

- **JOSUÉ FERNANDO OROZCO ROBLEDO**
  16397232
- **ANGIE MELISSA SANTIAGO RODRÍGUEZ**
  1555123
- **KAREN FLORIDALMA LAINES PABLO**
  1520722

Importaciones

Se importaron las librerías necesarias, de tiempo y de cola

```
#IMPORTACIONES
import queue
import time
```

DICCIONARIOS

DICCIONARIOS UTILIZADOS

```
#DICCIONARIOS

users_clients = {}
```

Clases

1.1 clase para la administración de inventario: se inicializaron las variables necesarias.

```
class Inventory_and_Orders:
    JosueOrozco01
    def __init__(self):
        self.principal = [[1, "Hamburguesa Volcanica", 40, "Tocino y Queso", 25, "Hamburguesas"],
            [2, "Hamburguesa Tejana", 40, "BBQ", 25, "Hamburguesas"],
            [3, "Hamburguesa Bomba de Queso", 40, "3 Quesos Fundidos", 25, "Hamburguesas"],
            [4, "Pizza de Jamon", 35, "Jamon y Queso",25, "Pizzas"],
            [5, "Pizza de Pepperoni", 35, "Pepperoni y Queso", 25, "Pizzas"],
            [6, "Pizza Hawaina", 35, "Jamon, Queso y Piña", 25, "Pizzas"],
            [7, "Tacos al Pastor", 21, "3 x 21", 30, "Tacos"],
            [8, "Tacos de Cochito Horneado", 21, "3 x 21", 30, "Tacos"],
            [9, "Tacos de Pollo", 21, "3 x 21", 30, "Tacos"],
            [10, "Pie de Queso", 25, "Torta de Queso",15, "Postres"],
            [11, "Pie de Calabaza", 25, "Pastel de Temporada",10, "Postres"],
            [12, "Pie de Manzana", 25, "Torta de Manzana",15, "Postres"],
            [13, "Gaseosas", 10, "Todos los Sabores", 45, "Bebidas"],
            [14, "Jugo Natural", 15, "Limonada y Naranjada", 30, "Bebidas"],
            [15, "Agua Pura", 5, "Agua Pura", 15, "Bebidas"]]
```

```
self.car = []
self.administrator = []
self.subtotal = 0
self.taxes = 0
self.total = 0
self.ticket = 0
self.line = queue.Queue()
self.ganancias = 0
self.iva = 0
self.dinero = 0
self.facturas = []
```

2. FUNCIONES

2.1 Función para el ingreso de productos:

1. se creó una variable account para saber cuantos elementos se iban a agregar.
2. se hizo un ciclo para agregar el número de elementos a una lista, con sus respectivos datos.
3. se usó la función append para ir agregandolos y se imprime la información del nuevo producto agregado

```python
def Ingreso_Producto_Inventario(self):
    print("-" * 50)
    account = int(input("¿Cuantos Productos Ingresara?: "))
    print("-" * 50)

    for i in range(account):
        products = []
        print("Producto No.", len(self.principal) + 1)
        name_product = input("Ingrese el Nombre del Producto: ")
        idProduct = len(self.principal) + 1
        price = int(input("Ingrese el Precio del Producto: "))
        description = input("Ingrese la Descripción del Producto: ")
        stock = input("Ingrese la Cantidad Exacta que Tiene del Producto: ")
        categoria = input("Ingrese la categoria del producto (Hamburguesas, Pizza, Tacos, Postres, Bebidas o Combos: ")

        print("-" * 50)
        products.append(idProduct)
        products.append(name_product)
        products.append(price)
        products.append(description)
        products.append(stock)
        products.append(categoria)
        self.principal.append(products)
        print("Información del Producto No.", len(self.principal), "Guardada Correctamente")
        print("-" * 50)
```

2.2 Función para ver el inventario:

1. se agregó un ciclo para poder recorrer la lista de los productos dentro del inventario y así poder imprimir cada uno de los datos de los productos.

```
def Ver_Inventario_Clientes(self, category=None):
    print("-" * 80)
    titles = ["No.", "Nombre del Producto", "Precio Q.", "Descripción"]
    print("{:<15} {:<30} {:<10} {:<15}".format(*titles))
    print("-" * 80)
    for product in self.principal:
        if category is None or product[5] == category:
            print("{:<15} {:<30} {:<10} {:<15}".format( *args: product[0], product[1], product[2], product[3]))
            print("-" * 80)
```

2.3 función para cambio de datos del inventario

1. Se usó un menú para elegir que dato cambiar de cada producto del inventario, y ver la actualización de la información cambiada.
2. Si ingresa la opción 1, 2, 3 o 4: se creó una variable que le pide y almacena el ID del producto.
3. Se crea un ciclo para comparar ese ID con cada ID de los productos en el inventario,
4. Se creó la variable position para guardar la posición según el índice de dicho producto.
5. Si se encuentra el producto, se imprimen los datos.
6. se creó una variable para comparar la confirmación del cambio y verificar que sea positiva.
7. se agrega la variable new que almacenará el nuevo cambio.
8. usando la variable position para ubicar el elemento, la variable new, y el índice del dato que se desea cambiar, se sustituye la información.

```python
def Cambio_de_Datos(self):
    while True:
        print("-" * 50)
        print("¿Que Dato Desea Cambiar?")
        print("1-. Nombre")
        print("2-. Cantidad del Producto")
        print("3-. Precio")
        print("4-. Descripción")
        print("5-. Ver Actualizaciones")
        print("6-. Regresar al Menú Anterior")
        print("7-. Salir del Programa")

        print("-" * 50)
        challenge = input("Ingrese el Número de la Opcion que Desea Actualizar: ")


        if challenge == "1":
            print("-" * 50)
            id = int(input("Ingrese Número del Producto que Desea Actualizar: "))
            print("-" * 50)
            for x in self.principal:
                if x[0] == id:
                    position = self.principal.index(x)
                    print("Los Datos del Producto son los Siguientes:")
                    print("-" * 100)
                    titles = ["No.", "Nombre del Producto", "Precio Q.", "Descripción", "Cantidad"]
                    print("{:<15} {:<30} {:<10} {:<25} {:<10}".format(*titles))
                    print("-" * 100)
                    print("{:<15} {:<30} {:<10} {:<25} {:<10}".format(*x))
                    print("-" * 100)

                    confirm = int(input("¿Desea Cambiar Nombre del Producto?\n 1. Si\n 2. No\n"))
                    print("-" * 50)
```

```python
                if confirm == 1:
                    new = input("Ingrese el Nuevo Nombre del Producto: ")
                    print("-" * 50)

                    self.principal[position][1] = new
                    print("Datos Actualizados Correctamente")
                    break
                else:
                    print("El Dato no Fue Cambiado")
                    print("-" * 50)

            else:
                print("Producto no Encontrado, Asegurese de que el Producto Exista")
                print("-" * 50)
```

2.4 opcion para ver actualizaciones

se llama a la función
ver_inventario()

```
elif challenge == "5":
    print("Sus actualizaciones quedaron de la siguiente manera: ")
    self.Ver_Inventario()
```

2.5 regresar al menú anterior

Para regresar al menú anterior solo se rompe el ciclo.

```
elif challenge == "6":
    print("Datos Cambiados Correctamente")
    break
```

2.6 función eliminar producto

1. se creó la variable “Id” para almacenar el número de id que ingresará el usuario.
2. se creó un ciclo para recorrer la lista del inventario y comparar los id.
3. si los id coinciden, se utiliza la función remove() para eliminarlo de la lista del inventario.
4. sino coincide con ningún id, se imprime que no se encontró el producto.

```
def Eliminar_Producto(self):
    print("-" * 50)
    id = int(input("Ingrese Número del Producto que Desea Eliminar: "))
    print("-" * 50)
    for x in self.principal:
        if x[0] == id:
            self.principal.remove(x)
            print("El Producto se Elimino del Inventario con los  Siguientes Datos:")
            print("-" * 100)
            titles = ["No.", "Nombre del Producto", "Precio Q.", "Descripción", "Cantidad"]
            print("{:<15} {:<30} {:<10} {:<25} {:<10}".format(*titles))
            print("-" * 100)
            print("{:<15} {:<30} {:<10} {:<25} {:<10}".format(*x))
            print("-" * 100)
            return
        else:
            print("Producto no Encontrado, Asegurese de que el Producto Exista")
            print("-" * 50)
```

2.7 función agregar producto al carrito

```
def agregar_producto_carrito(self):
    print("-" * 50)
    id = int(input("Ingrese Número del Producto que Desea agregar al carrito: "))
    print("-" * 50)

    found = False

    for x in self.principal:
        if x[0] == id:
            found = True
            position = self.principal.index(x)
            print("Los datos son:")
            print("-" * 80)
            titles = ["No.", "Nombre del Producto", "Precio Q.", "Descripción"]
            print("{:<15} {:<30} {:<10} {:<15}".format(*titles))
            print("-" * 80)
            print("{:<15} {:<30} {:<10} {:<15}".format(*x))
            print("-" * 80)
```

1. se creó la variable “id” donde se almacenará el número de id que ingrese el usuario.
2. se declaró como predeterminada la variable “found”
3. Se creó el ciclo for para recorrer la lista de productos del inventario y comparar los id.
4. Si coinciden, found se declara como True, y se almacena la posición del producto en la variable position, y se imprimen los datos del producto.

```
            price = x[2]
            amount = int(input("Ingrese la Cantidad que Desea del Producto: "))
            if amount <= int(x[4]):
                self.principal[position][4] = str(int(x[4]) - amount)
                total_price = price * amount
                self.car.append((x[0], x[1], price, amount, total_price, x[3]))
                self.administrator.append((x[0], x[1], price, amount, total_price, x[3]))
                print("Producto agregado al carrito correctamente")
            else:
                print("No hay suficiente cantidad disponible del producto")
            break

    if not found:
        print("Producto no Encontrado, Asegúrese de que el Producto Exista")
```

5. se declarala variable price como el precio usando la posicion 2 que es el dato del precio del producto, y la variable amount, para que el usuario ingrese el número de productos que desea.
6. se coloca el bucle if, colocando como condición si amount <= a la cantidad indicada en el índice, convirtiendo el dato a números, para posteriormente multiplicar por la variable amount, y almacenar dicha cifra en la variable total_price
7. Luego se agrega a la lista “car” y a lista “administrador” con la función append, almacenando los datos específicos del producto llamandolos por su índice, y almacenando la variable price, amount y total_price.
8. pero si la condición amount>a la cantidad que hay, solamentes e imprime que no hay existencias disponibles.
9. finalmente, si el id ingresado no coincide con ninguno de la lista al ser comparada con el bucle for, se imprime, queno se encontró dicho producto.

2.8 función ver_carro

1. se inición con el bucle if, ya que si NO está vacia se declara como True, y se realizan las siguientes acciones.
2. Se usa el bucle for para recorrer cada producto en la lista “car” y poder imprimir cada dato de cada producto en dicha lista.
3. se crea la variable product, para almacenar la información de dichos productos en un orden específico.

```
def Ver_carro(self):
    if self.car:
        print("-" * 115)
        titles = ["No.", "Nombre del Producto", "Precio Q.", "Cantidad adquirida", "Total", "Descripción"]
        print("{:<15} {:<30} {:<10} {:<20} {:<10} {:<15}".format(*titles))
        print("-" * 115)
        for product in self.car:
            product_no, product_name, price, quantity_adquired, total_price, description = product
            print(f"{product_no:<15} {product_name:<30} {price:<10} {quantity_adquired:<20} {total_price:<10} {description:<15}")
            print("-" * 115)
```

2.9 función ver_ventas_del_día

1. Se inicia con el bucle if, dandole la condición que si la lista administrado no está vacía se realizan las siguientes acciones.
2. se usa el ciclo for para recorrer la lista de administrador.
3. Se declara la variable product, asignandole los datos de la venta.
4. se imprimeen pantalla cada producto con sus respectivos datos gracias al ciclo for.
5. De lo contrario, si la lista administrador está vacía, solamente se imprime que no se realizaron ventas en el día.

```
def ver_ventas_del_dia(self):
    if self.administrator:
        print("-" * 115)
        titles = ["No.", "Nombre del Producto", "Precio Q.", "Cantidad adquirida", "Total", "Descripción"]
        print("{:<15} {:<30} {:<10} {:<20} {:<10} {:<15}".format(*titles))
        print("-" * 115)
        for product in self.administrator:
            product_no, product_name, price, quantity_adquired, total_price, description = product
            print(f"{product_no:<15} {product_name:<30} {price:<10} {quantity_adquired:<20} {total_price:<10} {description:<15}")
            print("-" * 115)
    else:
        print("No se Han Hecho Ventas Durante el Día")
```

2.10 Eliminar_prodcuto_del_carrito

1. se inición con el bucle if, ya que si está NO vacia se declara como True, y no se realizan las siguientes acciones.
2. Se declara la variable Id, donde se almacena el número de id que ingresa el usuario,
3. Se usa el bucle for para recorrer cada producto en la lista “car” y se vuelve a usar if, para comparar el id ingresado con cada uno de lista, al coincidir con uno, se usa la función remove() para eliminar el producto de la lista car.
4. si no coincide el id con ninguno de la lista, se imprime que no se encontró el producto.
5. Y si la lista car está vacía, se imprime que no hay productos en el carrito.

```python
def Eliminar_Producto_del_carrito(self):
    if self.car:
        print("-" * 50)
        id = int(input("Ingrese Número del Producto que Desea Eliminar: "))
        print("-" * 115)
        for x in self.car:
            if x[0] == id:
                self.car.remove(x)
                print("El Producto se Elimino del Inventario con los  Siguientes Datos:")
                print("-" * 115)
                titles = ["No.", "Nombre del Producto", "Precio Q.", "Cantidad adquirida", "Total", "Descripción"]
                print("{:<15} {:<30} {:<10} {:<20} {:<10} {:<15}".format(*titles))
                print("-" * 115)
                print("{:<15} {:<30} {:<10} {:<20} {:<10} {:<15}".format(*x))
                print("-" * 115)
                return
            else:
                print("Producto No Encontrado, Asegurese de que el Producto Exista")
                print("-" * 50)

    else:
        print("Asegurese de que Haya Producto en el Carro")
```

2.11 función sumar_producto_al_carrito

```python
def sumar_producto_al_carrito(self):
    if self.car:
        print("-" * 80)
        id = int(input("Ingrese el Número del Producto, al que le Desea Cambiar la Cantidad: "))
        print("-" * 80)
        for i, product in enumerate(self.car):
            if product[0] == id:
                new = int(input("Ingrese la Nueva Cantidad: "))
                product_no, product_name, price, old_total, old_quantity, description = product
                new_total = price * new
                self.car[i] = (product_no, product_name, price, new, new_total, description)
                print(f"Cantidad del producto: {product_name}, actualizada a: {new}")
                return
        print("Producto no Encontrado en el Carrito")
```

1. se inicia con el bucle if, si car no está vacía se agrega una entrada almacenada como id.
2. se usa el bucle for para recorrer la lista y comparar el id ingresado con los de la lista car.
3. si coincide con alguno de la lissta, se agrega otra entrada que se almacena como new.
4. se declara la variable product, con los datos del producto.
5. se declara la variable new_total multiplicando el precio por la variable new, y así obtener el total a pagar por la cantidad de productos.
6. Se imprime la información del producto.
7. En caso de no coincidir ningún id, se imprime producto no encontrado.

2.12 función impuesto

1. se inició con if, para que si la lista car, NO está vacía se realicen las siguientes acciones.
2. se declaró iva_rate con el valor predeterminado como 0.12
3. la variable taxes es el resultado de multiplicar el subtotal por iva_rate

4. se imprime la cantidad de impuestos,
5. Si no hay productos en la lista car, se imprime que no hay productos.

```
def Impuesto(self):
    if self.car:
        iva_rate = 0.12
        self.taxes = self.subtotal * iva_rate
        print("IMPUESTO (12%): Q.", self.taxes)
        print("-" * 115)
    else:
        print("Asegurese de que Haya Producto en el Carro")
```

2.13 función Iord

```
def Iord(self):
    if self.car:
        print("-" * 115)
        print("---PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--")
        print("-" * 115)
        print("----------------------------------------------------FACTURA----------------------------------------------------")
        print("-" * 115)
        print("No. de Orden", self.ticket)

        subtotal = 0
        taxes = 0
        total = 0

        car_copy = self.car.copy()

        factura = {"No. de Orden": self.ticket,
            "Productos": car_copy,
            "Subtotal": subtotal,
            "Impuesto (12%)": taxes,
            "Total": total}

        for product in car_copy:
            subtotal += product[4]
        taxes = subtotal * 0.12
        total = subtotal + taxes

        factura["Subtotal"] = subtotal
        factura["Impuesto (12%)"] = taxes
        factura["Total"] = total
```

```
self.facturas.append(factura)
```

1. se inició con if, si la lista car, NO está vacía, se sigue las siguientes acciones.
2. se declaran las variables subtotal, taxes, total = 0
3. se declara la variable car_copy y se hace una copia con la función copy()
4. se guarda la variable factura con los datos antes especificados.
5. se realiza un blucle for para recorrer los productos en la copia del carrito.
6. a la variable sub_total se le va sumando el precio del producto ya multiplicado por la cantidad ingresada.
7. Se calcula el impuesto
8. Se le suma todo eso al subtotal.
9. con la función append() se agregar a la lista factura.

2.14 función ver_factura

1. Si la lista facturas NO está vacía, se realizan las siguientes acciones.
2. se inicializa con el bucle for y enumarate para devolver i y factura en la lista de facturas
3. se agrega el ciclo for para recorrer los productos en la lista factura.
4. se almacenan losdatos en la variable product
5. para imprimen los datos de cada elemento de la lista.
6. si la lista está vacía, se imprime queno hay facturas.

```
def Ver_facturas(self):
    if self.facturas:
        print("")
        print("")
        print("-----------------------------------------------REALIZAR PEDIDOS-----------------------------------------------")
        print("")
        print("")
        print("")
        for i, factura in enumerate(self.facturas):
            print("-" * 115)
            print("---PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--")
            print("-" * 115)
            print("-------------------------------------------------------FACTURA-------------------------------------------------------")
            print("-" * 115)
            print("No. de Orden", factura["No. de Orden"])
            print("-" * 115)
            titles = ["No.", "Nombre del Producto", "Precio Q.", "Cantidad adquirida", "Total", "Descripción"]
            print("{:<15} {:<30} {:<10} {:<20} {:<10} {:<15}".format(*titles))
            print("-" * 115)
            for product in factura["Productos"]:
                product_no, product_name, price, quantity_adquired, total_price, description = product
                print(
                    f"{product_no:<15} {product_name:<30} {price:<10} {quantity_adquired:<20} {total_price:<10} {description:<15}")
                print("-" * 115)
            print(f"SUBTOTAL: Q. {factura['Subtotal']}")
            print("-" * 115)
            print(f"IMPUESTO (12%): Q. {factura['Impuesto (12%)']}")
            print("-" * 115)
            print(f"TOTAL Q. {factura['Total']}")
            print("-" * 115)
```

```
                print("-" * 115)
                print("")
                print("")
                print("")

    else:
        print("No hay Facturas disponibles")
```

2.15 función sub_total_carrito

```
def Sub_total_carrito(self):
    if self.car:
        self.subtotal = sum(product[4] for product in self.car)
        print("SUBTOTAL: Q.", self.subtotal)
        print("-" * 115)
```

1. Con el bucle if, designamos que si la lista car, NO está vacía se realizan las siguientes acciones.
2. se declara subtota, y con con el ciclo for se suman los precios en los productos en car.
3. se imprime el subtotal.

2.16 función total_ventas_del_día

```
def total_ventas_dia(self):
    if self.administrator:
        self.ganancias = sum(product[4] for product in self.administrator)
        iva_rate = 0.13
        self.iva = self.ganancias * iva_rate
        print("SUBTOTAL: Q.", self.ganancias)
        print("-" * 115)
        print("IMPUESTO (12%): Q.", self.iva)
        self.dinero = self.ganancias + self.iva
        print("-" * 115)
        print("TOTAL Q.", self.dinero)
        print("-" * 115)
```

1. Con el bucle if si la lista adiministrator no está vacía, se realizan las siguientes acciones
2. se declara que las ganacias es igual a la suma del precio de los productos, y con el ciclo for se recorren los elementos en administrator.
3. Las ganancias se multiplican con iva_rate declarado como 0.13
4. Se imprime el subtotal, el IVA, y se suman para indicar las ventas del día.

```python
def calcular_total(self):
    if self.car:
        self.total = self.subtotal + self.taxes
        print("TOTAL Q.", self.total)
        print("-" * 115)
        print("---PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--PEDIDO--")
        print("-" * 115)
```

2.17 función calcular_total

1. Con el bucle if, si car NO está vacía, se realiza lo sigueinte.
2. a la variable total se le asigna el varlo del subtotal suamdno losimpuestos
3. se imprime la variable total en pantalla.

```python
def Pago(self):
    while True:
        print("\n------------------Metodo de Pago------------------")
        print("\n¿Como Desea Pagar?")
        print("\n1-. Efectivo")
        print("2-. Tarjeta\n")
        print("-" * 50)
        opcion_pago = input("Ingrese el Número de la Opción que Desee: ")
        print("-" * 50)

        if opcion_pago == "1":
            efectivo = float(input("\nIngrese la Cantidad con la que Pagara (Q.): "))
            if efectivo >= self.total:
                cambio = efectivo - self.total
                print("\n-------------------Pago Exitoso--------------------")
                print(f"Su cambio es Q. {cambio:.2f}")
                break
            else:
                print("\n----------------------ERROR!----------------------")
                print("------La Cantidad Ingresada es Insuficiente-------")

        elif opcion_pago == "2":
            print("--------- Acerque la Tarjeta al Lector... --------")
            time_eight = 5
            for tiempo_restante in range(time_eight, -1, -1):
                print("\rTiempo Restante: {} segundos".format(tiempo_restante), end='')
                time.sleep(1)
            print("\n-----------------Tarjeta Aprobada-----------------")
            print("---------------Gracias por su Compra--------------")
            break

            else:
                print("\n----------------------ERROR!----------------------")
                print("-----------------Intente Nuevamente---------------\n")
```

2.18 función pagar

1. Se usa el bucle while para imprimr el menú.
2. se agrega una entrada declarada como opcion_pago.
3. si opcion_pago =1 se realiza lo siguiente.
4. se agrega una entrada declarada como efectivo haciendola decimal con la función float
5. Con el bucle if, se coloca la condición que si es mayor o igual al total debe haber cambio.
6. se declara la variable cambio, restando el efectivo menos el total
7. se imprime en pantalla el cambio.
8. se detiene el bucle.
9. En caso de no cumplir la condición se imprime error.
10. Si opcion_pago=2, se realiza lo siguiente.
11. Se define una variable time_eight y se le asigna el valor inicial de 5. Esto indica que el temporizador comenzará desde 5 segundos.
12. Se utiliza un bucle for para iterar sobre una secuencia generada por range. La secuencia comienza en time_eight (5 en este caso), va hasta -1 (sin incluirlo), con un paso de -1. Esto significa que el bucle contará hacia atrás desde time_eight hasta 0.
13. En cada iteración del bucle, se imprime el mensaje "Tiempo Restante: x segundos", donde x es el valor actual de tiempo_restante. La opción end='' evita que se agregue automáticamente un carácter de nueva línea al final de la impresión, lo que contribuye a la sobreescritura.
14. Después de imprimir el mensaje, el programa se detiene durante 1 segundo usando la función sleep del módulo time. Esto crea un efecto de temporización.
15. se detiene el bucle.
16. sino se cumple la condición imprime error.

```python
def atender_clientes(self):
    while not self.line.empty():
        cliente = self.line.get()
        print(f"Atendiendo al Cliente No. {cliente}")


        time_twenty = 20
        for tiempo_restante in range(time_twenty, -1, -1):
            print("\rTiempo Restante: {} segundos".format(tiempo_restante), end='')
            time.sleep(1)

        print("\nEntregando Pedido al Cliente No. {}".format(cliente))

        time_eight = 8
        for tiempo_restante in range(time_eight, -1, -1):
            print("\rTiempo Restante: {} segundos".format(tiempo_restante), end='')
            time.sleep(1)

        print("\nPedido Entregado Exitosamente")
        self.car.clear()
```

2.19 función atender_clientes

1. Este bucle while se ejecuta mientras la cola (self.line) no esté vacía. En cada iteración, un cliente se extrae de la cola utilizando self.line.get().
2. se utiliza un bucle for para contar hacia atrás desde time_twenty hasta 0, representando el tiempo restante para la entrega del pedido. Se utiliza time.sleep(1) para pausar la ejecución del programa durante 1 segundo en cada iteración, creando un efecto de cuenta regresiva.
3. Se imprime un mensaje indicando que el pedido se está entregando al cliente actual.
4. el segundo bucle funcionan de manera similar al primer bucle, pero representan un segundo periodo de espera después de entregar el pedido.
5. Finalmente, se imprime un mensaje indicando que el pedido ha sido entregado exitosamente, y se limpia la cola (self.car.clear()), presumiblemente para indicar que el cliente ha sido atendido y no está en la cola.

2.20 función take_ticket

```
def take_ticket(self):
    if self.car:
        self.line.put(self.ticket)
        print("")
        print("")
        print("")
        print("-" * 50)
        print(f"Tu número de ticket es: {self.ticket}")
```

1. Esta función toma un ticket para un cliente si hay al menos un cliente en la cola (self.car). Si hay clientes en la cola, se coloca un nuevo ticket (self.ticket) en la cola (self.line) y se imprime un mensaje que indica el número de ticket. Además, la función incrementa el número del ticket para el siguiente cliente.

2.21 función suma_ticket

```
def suma_ticket(self):
    self.ticket += 1
```

Esta función incrementa el número del ticket (self.ticket) en uno.

3. funciones para validaciones

3.1 función validar_nombres

Esta función toma un argumento nombres y verifica si contiene solo caracteres alfabéticos utilizando el método .isalpha(). Si no cumple con esta condición, se lanza una excepción ValueError con un mensaje de error indicando que los nombres no deben contener caracteres especiales.

```
def Validar_Nombres(nombres):
    if not nombres.isalpha():
        raise ValueError ("\n----------------------ERROR!----------------------"
                          "\nLos Nombres NO Deben Contener Caracteres Especiales")
```

3.2 función validar_apellidos

esta función verifica si la cadena apellidos contiene solo caracteres alfabéticos. Si no cumple con esta condición, se lanza una excepción ValueError con un mensaje de error específico.

```
def Validar_Apellidos(apellidos):
    if not apellidos.isalpha():
        raise ValueError ("\n----------------------ERROR!----------------------"
                          "\nLos Appellidos NO Deben Contener Caracteres Especiales")
```

3.3 función validar_fecha_nacimiento

Esta función toma una cadena fecha que supuestamente representa una fecha en formato de día/mes/año. La función divide la cadena en partes usando '/' como delimitador y verifica si hay exactamente tres partes y si todas las partes son dígitos. Si no cumple con estas condiciones, se lanza una excepción ValueError con un mensaje de error indicando que el formato de la fecha de nacimiento es incorrecto.

```
def Validar_Correo_Electronico(correo):
    if correo.count('@') != 1:
        raise ValueError ("\n----------------------ERROR!----------------------"
                          "\n-----Direccion de Correo Electronica NO Valida----")
```

3.4 función validar_correo_electronico

Esta función toma una cadena correo y verifica que contenga exactamente un símbolo '@'. Si no cumple con esta condición, se lanza una excepción ValueError con un mensaje de error indicando que la dirección de correo electrónico no es válida.

```
def Validar_Numero_Telefonico(telefono):
    if not telefono.isdigit() or len(telefono) != 8:
        raise ValueError ("\n----------------------ERROR!----------------------"
                          "\n---Formato Incorrecto para el Número Telefonico---")
```

```python
def Registro_Clientes():
    while True:
        print("\n-------------BIENVENIDOS A FASTFOOD-------------")
        print("\nIngrese sus Datos a Continuacion:")
        try:
            names = input("Ingrese sus Nombres: ")
            Validar_Nombres(names)

            last_name = input("Ingrese sus Apellidos: ")
            Validar_Apellidos(last_name)

            birth_date = input("Ingrese su Fecha de Nacimiento (DD/MM/AA): ")
            Validar_Fecha_Nacimiento(birth_date)

            email = input("Ingrese su Correo Electronico: ")
            Validar_Correo_Electronico(email)

            phone = input("Ingrese su Número Telefonico: ")
            Validar_Numero_Telefonico(phone)

            username = input("\nIngrese un Nombre de Usuario Unico: ")
            if username in users_clients:
                print("\nERROR!\nEl Nombre de Usuario ya Existe, Elija otro")
                continue

            password = input("Elige una Contraseña: ")
            confirmation = input("Confirma tu Contraseña: ")


            if password != confirmation:
                print("\nERROR!\nLas Contraseñas no Coinciden. Intente de Nuevo")
                continue

            users_clients[username] = {
                "nombres": names,
                "apellidos": last_name,
                "fecha_nacimiento": birth_date,
                "email": email,
                "no_telefonico": phone,
                "contraseña": password
            }
            print("\n----------Registro Exitoso como Cliente!----------")
            break
        except ValueError as ve:
            print(f"Error: {ve}")
```

4.1 función Registro_clientes

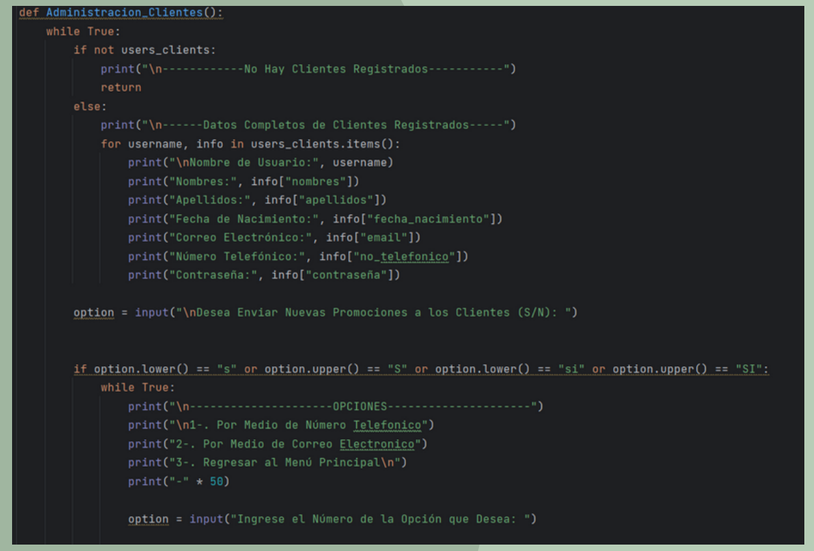
1. Inicia un bucle infinito, lo que significa que el código dentro del bucle se ejecutará repetidamente hasta que se rompa explícitamente con break.Dentro del bucle, se imprime un mensaje de bienvenida y se solicita al usuario que ingrese sus datos personales, como nombres, apellidos, fecha de nacimiento, correo electrónico, número telefónico, nombre de usuario y contraseña.
2. Se utiliza un bloque try...except para manejar posibles errores que puedan ocurrir durante la entrada de datos. Si se lanza una excepción ValueError (por ejemplo, debido a datos no válidos ingresados), se imprime un mensaje de error específico.
3. Después de ingresar cada tipo de dato, se llama a las funciones de validación correspondientes, como Validar_Nombres, Validar_Apellidos, etc., para asegurarse de que los datos ingresados cumplan con ciertos criterios.
4. Se verifica si el nombre de usuario ya existe en una estructura de datos llamada users_clients. Si existe, se imprime un mensaje de error y se solicita al usuario que elija otro nombre de usuario.
5. Se solicita al usuario que ingrese y confirme una contraseña. Se verifica si las contraseñas coinciden. Si no coinciden, se imprime un mensaje de error y se solicita al usuario que lo intente de nuevo.
6. Si todas las validaciones son exitosas y no se han producido errores, se crea un diccionario que representa la información del cliente y se agrega al diccionario users_clients. Luego se imprime un mensaje de registro exitoso.
7. El bucle se rompe con break, y la función sale de la repetición.

4.2 iniciar_sesion_cliente

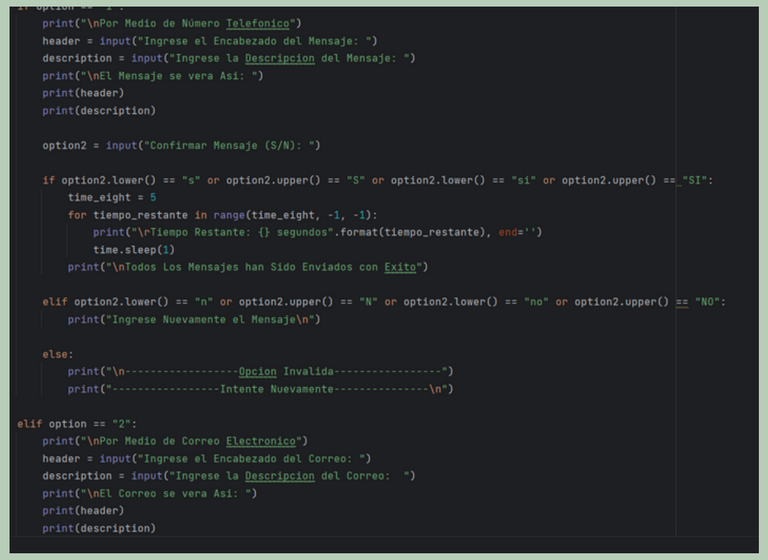
```python
def Iniciar_Sesion_Cliente():
    username = input("Ingrese su Nombre de Usuario: ")
    password = input("Ingrese su Contraseña: ")
    print("-" * 50)

    if username in users_clients:
        usuario = users_clients[username]
        if usuario["contraseña"] == password:
            print("\nIniciando Sesión como Cliente:", usuario["nombres"], usuario["apellidos"])
            Menu_Clientes()
        else:
            print("\n----------------------ERROR!----------------------")
            print("----Nombre de Usuario o Contraseña Incorrectos----")
    else:
        print("\n----------------------ERROR!----------------------")
        print("----------Nombre de Usuario no Encontrado---------")
```

1. Solicita al usuario que ingrese su nombre de usuario y almacena la entrada en la variable username.
2. Solicita al usuario que ingrese su contraseña y almacena la entrada en la variable password.
3. if username in users_clients, Verifica si el nombre de usuario ingresado está presente en la estructura de datos users_clients. Si es así, procede a la siguiente verificación.
4. usuario = users_clients[username], Obtiene la información del usuario correspondiente al nombre de usuario ingresado desde la estructura de datos users_clients y la almacena en la variable usuario.
5. con el bucle if compara la contraseña ingresada con la contraseña almacenada en la información del usuario. Si coinciden, se imprime un mensaje de inicio de sesión exitoso y se llama a la función Menu_Clientes. Si no coinciden, se imprime un mensaje de error indicando que el nombre de usuario o la contraseña son incorrectos.
6. Si el nombre de usuario no se encuentra en users_clients, se imprime un mensaje de error indicando que el nombre de usuario no se encontró.

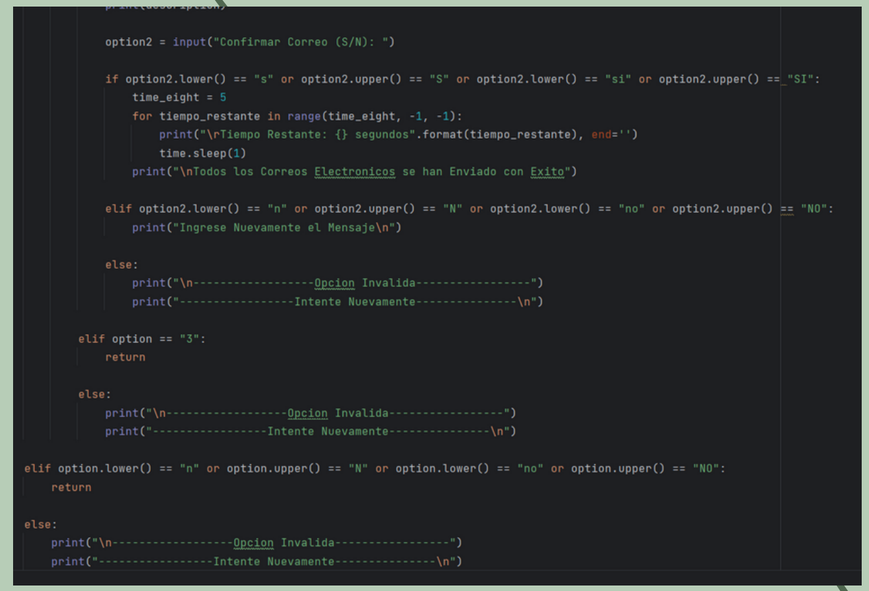
4.3 función administracion_clientes



```
def Administracion_Clientes():
    while True:
        if not users_clients:
            print("\n-----------No Hay Clientes Registrados----------")
            return
        else:
            print("\n------Datos Completos de Clientes Registrados-----")
            for username, info in users_clients.items():
                print("\nNombre de Usuario:", username)
                print("Nombres:", info["nombres"])
                print("Apellidos:", info["apellidos"])
                print("Fecha de Nacimiento:", info["fecha_nacimiento"])
                print("Correo Electrónico:", info["email"])
                print("Número Telefónico:", info["no_telefonico"])
                print("Contraseña:", info["contraseña"])

        option = input("\nDesea Enviar Nuevas Promociones a los Clientes (S/N): ")


        if option.lower() == "s" or option.upper() == "S" or option.lower() == "si" or option.upper() == "SI":
            while True:
                print("\n--------------------OPCIONES--------------------")
                print("\n1-. Por Medio de Número Telefonico")
                print("2-. Por Medio de Correo Electronico")
                print("3-. Regresar al Menú Principal\n")
                print("-" * 50)

                option = input("Ingrese el Número de la Opción que Desea: ")
```



```
    print("\nPor Medio de Número Telefonico")
    header = input("Ingrese el Encabezado del Mensaje: ")
    description = input("Ingrese la Descripcion del Mensaje: ")
    print("\nEl Mensaje se vera Asi: ")
    print(header)
    print(description)

    option2 = input("Confirmar Mensaje (S/N): ")

    if option2.lower() == "s" or option2.upper() == "S" or option2.lower() == "si" or option2.upper() == "SI":
        time_eight = 5
        for tiempo_restante in range(time_eight, -1, -1):
            print("\rTiempo Restante: {} segundos".format(tiempo_restante), end='')
            time.sleep(1)
        print("\nTodos Los Mensajes han Sido Enviados con Exito")

    elif option2.lower() == "n" or option2.upper() == "N" or option2.lower() == "no" or option2.upper() == "NO":
        print("Ingrese Nuevamente el Mensaje\n")

    else:
        print("\n-----------------Opcion Invalida----------------")
        print("-----------------Intente Nuevamente--------------\n")

elif option == "2":
    print("\nPor Medio de Correo Electronico")
    header = input("Ingrese el Encabezado del Correo: ")
    description = input("Ingrese la Descripcion del Correo:  ")
    print("\nEl Correo se vera Asi: ")
    print(header)
    print(description)
```



```
        option2 = input("Confirmar Correo (S/N): ")

        if option2.lower() == "s" or option2.upper() == "S" or option2.lower() == "si" or option2.upper() == "SI":
            time_eight = 5
            for tiempo_restante in range(time_eight, -1, -1):
                print("\rTiempo Restante: {} segundos".format(tiempo_restante), end='')
                time.sleep(1)
            print("\nTodos los Correos Electronicos se han Enviado con Exito")

        elif option2.lower() == "n" or option2.upper() == "N" or option2.lower() == "no" or option2.upper() == "NO":
            print("Ingrese Nuevamente el Mensaje\n")

        else:
            print("\n-----------------Opcion Invalida----------------")
            print("-----------------Intente Nuevamente--------------\n")

    elif option == "3":
        return

    else:
        print("\n-----------------Opcion Invalida----------------")
        print("-----------------Intente Nuevamente--------------\n")

elif option.lower() == "n" or option.upper() == "N" or option.lower() == "no" or option.upper() == "NO":
    return

else:
    print("\n-----------------Opcion Invalida----------------")
    print("-----------------Intente Nuevamente--------------\n")
```

1. while True:: Este bucle se ejecuta indefinidamente hasta que se alcanza una condición de salida (return). Se encarga de gestionar la administración de clientes y el envío de promociones.
2. if not users_clients:: Verifica si no hay clientes registrados. Si es así, imprime un mensaje indicando que no hay clientes y sale del bucle usando return.
3. Si hay clientes, muestra información detallada de cada cliente utilizando un bucle for y la función items() para recorrer el diccionario users_clients.
4. La variable option recoge la respuesta del usuario sobre si desea enviar promociones a los clientes.
5. Si el usuario decide enviar promociones, se inicia un bucle while True para presentar opciones sobre cómo enviarlas.
6. Se solicita al usuario que elija entre enviar promociones por número telefónico, por correo electrónico o regresar al mSe utiliza un bucle while True para presentar el submenú de opciones y manejar la entrada del usuario.
7. Para cada opción (1, 2, 3), se solicita al usuario que ingrese el encabezado y la descripción del mensaje o correo electrónico.
8. Se muestra el mensaje o correo electrónico y se pide confirmación. Dependiendo de la confirmación, se inicia un temporizador y se imprime un mensaje de éxito o se solicita al usuario que ingrese nuevamente el mensaje.enú principal.
9. Se utiliza un bucle else para manejar el caso en que el usuario ingrese una opción no válida en el submenú. Imprime un mensaje indicando que la opción es inválida y le pide al usuario que intente nuevamente.

5.1 función menu_clientes

```python
def Menu_Clientes():
    while True:
        print("-----------------------------------------------------")
        print("--------------BIENVENIDOS A FASTFOOD--------------")
        print("-----------------------------------------------------")
        print("\n1-. Ver Menu")
        print("2-. Agregar Producto al Carrito")
        print("3-. Ver Carrito")
        print("4-. Editar Carrito")
        print("5.- Finalizar Comprar")
        print("6.- Regresar al Menu Principal")
        print("7.- Salir del Programa\n")
        print("-" * 50)
        option = input("Ingrese el Número de la Opción que Desee: ")
        print("-" * 50)

        if option == "1":
            print("-----------------------------------------------------")
            print("\n------------------MENU FASTFOOD------------------")
            print("-----------------------------------------------------")
            print("\n1-. Hamburguesas ")
            print("2-. Pizzas")
            print("3-. Tacos")
            print("4-. Postres")
            print("5-. Bebidas")
            print("6-. Combos\n")
            print("-" * 50)
            option2 = input("Ingrese el Número de la Opción que Desee: ")
            print("-" * 50)
```

```python
            if option2 == "1":
                Administration.Ver_Inventario_Clientes(category="Hamburguesas")
            elif option2 == "2":
                Administration.Ver_Inventario_Clientes(category="Pizzas")
            elif option2 == "3":
                Administration.Ver_Inventario_Clientes(category="Tacos")
            elif option2 == "4":
                Administration.Ver_Inventario_Clientes(category="Postres")
            elif option2 == "5":
                Administration.Ver_Inventario_Clientes(category="Bebidas")
            elif option2 == "6":
                Administration.Ver_Inventario_Clientes(category="Combos")
            else:
                print("\n-----------------Opcion Invalida-----------------")
                print("-----------------Intente Nuevamente--------------\n")

        elif option == "2":
            Administration.agregar_producto_carrito()

        elif option == "3":
            Administration.Ver_carro()

        elif option == "4":
                print("-----------------------------------------------------")
                print("\n-------------BIENVENIDOS A FASTFOOD-------------")
                print("-----------------------------------------------------")
                print("\n1-. Eliminar Producto")
                print("2-. Cambiar Cantidad del Producto\n")
                print("-" * 50)
                option3 = input("Ingrese el Número de la Opción que Desee: ")
```

```python
                if option3 == "1":
                    Administration.Eliminar_Producto_del_carrito()

                elif option3 == "2":
                    Administration.sumar_producto_al_carrito()

                else:
                    print("\n-----------------Opcion Invalida-----------------")
                    print("-----------------Intente Nuevamente--------------\n")

        elif option == "5":
            Administration.suma_ticket()
            Administration.Iord()
            Administration.Ver_carro()
            Administration.Sub_total_carrito()
            Administration.Impuesto()
            Administration.calcular_total()
            Administration.Pago()
            Administration.take_ticket()
            print("-" * 50)
            Administration.atender_clientes()
            print("\n-------------Gracias Por Tu Compra-------------")
            print("----------Esperamos que Vuelvas Pronto!----------\n")
            break

        elif option == "6":
            print("\n----------------- Regresando... ----------------")
            break

        elif option == "7":
            print("\n----------Esperamos que Vuelvas Pronto!----------")
```

```python
        else:
            print("\n-----------------Opcion Invalida-----------------")
            print("-----------------Intente Nuevamente--------------\n")
```

1. while True:: Este bucle se ejecuta indefinidamente hasta que se alcance una condición de salida (break).
2. Presenta un menú con diversas opciones para que el usuario interactúe.
3. Imprime un menú principal con varias opciones como ver el menú, agregar productos al carrito, ver el carrito, editar el carrito, finalizar la compra, regresar al menú principal y salir del programa.
4. Utiliza una estructura if-elif-else para manejar la opción ingresada por el usuario.
5. Para la opción "1", muestra un submenú para elegir categorías de productos (Hamburguesas, Pizzas, Tacos, Postres, Bebidas, Combos).
6. Para la opción "2", llama a la función Administration.agregar_producto_carrito().
7. Para la opción "3", llama a la función Administration.Ver_carro().
8. Para la opción "4", presenta un submenú para editar el carrito, donde el usuario puede eliminar productos o cambiar la cantidad.
9. Para la opción "5", realiza una serie de acciones relacionadas con el proceso de compra (suma de ticket, mostrar carrito, calcular subtotal, impuestos, total, pago, tomar ticket, atender clientes) y finaliza la compra.
10. Para la opción "6", imprime un mensaje indicando que se está regresando y sale del bucle principal con break.
11. Para la opción "7", imprime un mensaje de despedida y sale del programa con exit().
12. Si el usuario ingresa una opción no válida, se imprime un mensaje indicando que la opción es inválida y le pide al usuario que intente nuevamente.

6. LLAMAR A LAS FUNCIONES

```
predetermined = "FASTFOOD023"

while True:
    print("-" * 50)
    print("--------------BIENVENIDOS A FASTFOOD--------------")
    print("-" * 50)
    print("1-. Hacer Pedido")
    print("2-. Salir del Programa")
    print("-" * 50)
    position = input("Ingrese el Número de la Opción que Desee: ")
    print("-" * 50)

    if position == "0":
        #La contraseña administrativa es FASTFOOD023
        password = input("Ingrese Contraseña Administrativa: ")

        if password == predetermined:
            full_name = input("Ingrese su Nombre Completo: ")
            while True:
                print("-" * 50)
                print("--------------------BIENVENID@--------------------")
                print("--------------------------------------------------")
                print("\nIngresando Como:", full_name)
                print("\n1-. Ingresar Producto al Inventario")
                print("2-. Ver el Inventario")
                print("3-. Cambiar algún Dato de un Producto")
                print("4-. Eliminar Producto del Inventario")
```

```
                print("5-. Realizar Pedidos")
                print("6-. Administracion de Clientes")
                print("7-. Ventas del Día")
                print("8-. Regresar al Menú Principal")
                print("9-. Salir del Programa\n")
                print("-" * 50)
                option = input("Ingrese el Número de la Opción que Desee: ")
                print("-" * 50)
                if option == "1":
                    Administration.Ingreso_Producto_Inventario()

                elif option == "2":
                    Administration.Ver_Inventario()

                elif option == "3":
                    Administration.Cambio_de_Datos()

                elif option == "4":
                    Administration.Eliminar_Producto()

                elif option == "5":
                    Administration.Ver_facturas()

                elif option == "6":
                    Administracion_Clientes()

                elif option == "7":
                    Administration.ver_ventas_del_dia()
                    Administration.total_ventas_dia()
```

```
                elif option == "8":
                    print("\n----------------- Regresando... -----------------")
                    break

                elif option == "9":
                    print("\n----------Esperamos que Vuelvas Pronto!----------")
                    exit()

                else:
                    print("\n-----------------Opcion Invalida-----------------")
                    print("-----------------Intente Nuevamente--------------\n")

        else:
            print("\n----------------------ERROR!----------------------")
            print("---------------Contraseña Incorrecta--------------")

if position == "1":
    while True:
        print("--------------------------------------------------")
        print("\n--------------------BIENVENID@--------------------")
        print("--------------------------------------------------")
        print("\n¿Que Desea Hacer?")
        print("1-. Registarme")
        print("2-. Iniciar Sesion")
        print("3-. Modo Invitado")
        print("4-. Regresar al Menú Principal")
        print("5-. Salir del Programa\n")
        print("-" * 50)
        option = input("Ingrese el Número de la Opción que Desee: ")
        print("-" * 50)
```

```
        if option == "1":
            Registro_Clientes()

        elif option == "2":
            Iniciar_Sesion_Cliente()

        elif option == "3":
            Menu_Clientes()

        elif option == "4":
            print("\n----------------- Regresando... -----------------")
            break

        elif option == "5":
            print("\n----------Esperamos que Vuelvas Pronto!----------")
            exit()

        else:
            print("\n-----------------Opcion Invalida-----------------")
            print("-----------------Intente Nuevamente--------------\n")

elif position == "2":
    print("\n----------Esperamos que Vuelvas Pronto!----------")
    exit()

else:
    print("\n-----------------Opcion Invalida-----------------")
    print("-----------------Intente Nuevamente--------------\n")
```

1. Se establece una contraseña administrativa predeterminada (predetermined = "FASTFOOD023").
2. Un bucle while True maneja el ingreso de la contraseña administrativa.
3. Si la contraseña ingresada coincide con la predeterminada, se permite el acceso al menú de administración.
4. El administrador puede realizar diversas acciones como ingresar productos al inventario, ver el inventario, cambiar datos de un producto, eliminar un producto, realizar pedidos, administrar clientes, ver las ventas del día, regresar al menú principal o salir del programa.
5. Si se elige la opción "1" en el menú principal, se presenta un submenú para que los clientes realicen acciones como registrarse, iniciar sesión, utilizar el modo invitado, regresar al menú principal o salir del programa.
6. Si se elige la opción "2" en el menú principal, el programa se cierra.
7. En ambos menús, se incluye una sección para manejar opciones incorrectas. Si el usuario ingresa una opción no válida, se imprime un mensaje indicando que la opción es inválida y se le pide al usuario que intente nuevamente.