Simulación Tema: Inteligencia Artificial 1.

Objetivo:

- Consolidar los conocimientos adquiridos en clase para la programación de una aplicación en Python.

Enunciado:

- Realizar una aplicación que permita gestionar los pedidos de una empresa almacenando la información dentro de archivos o base de datos (SQLite), para lo cual deben seguir los siguientes pasos:

La aplicación deberá: manejar clientes (se guarda su nombre, dirección, teléfono y e-mail), que pueden realizar pedidos de productos, de los cuales se anota la cantidad en stock. Un cliente puede tener una o varias cuentas para el pago de los pedidos. Cada cuenta está asociada a una tarjeta de crédito, y tiene una cierta cantidad disponible de dinero, que el cliente debe aumentar periódicamente para poder realizar nuevos pedidos.
Un cliente puede empezar a realizar un pedido sólo si tiene alguna cuenta con dinero disponible. Además, sólo es posible realizar peticiones de productos en stock.

Existe una clase responsable del cobro, orden de distribución y confirmación de los pedidos. El cobro se realiza al finalizar el pedido. Si una cuenta no tiene suficiente dinero, el pedido se rechaza. Una vez que el pedido está listo para servirse, se ordena su distribución, y una vez entregado, pasa a estar confirmado.

* Se aprobará como puntos adicionales a la practicas si se realiza una implementación visual utilizando cualquier librería GUI (Tkinter – 1 punto) o mejor aún con Flask(3 puntos).

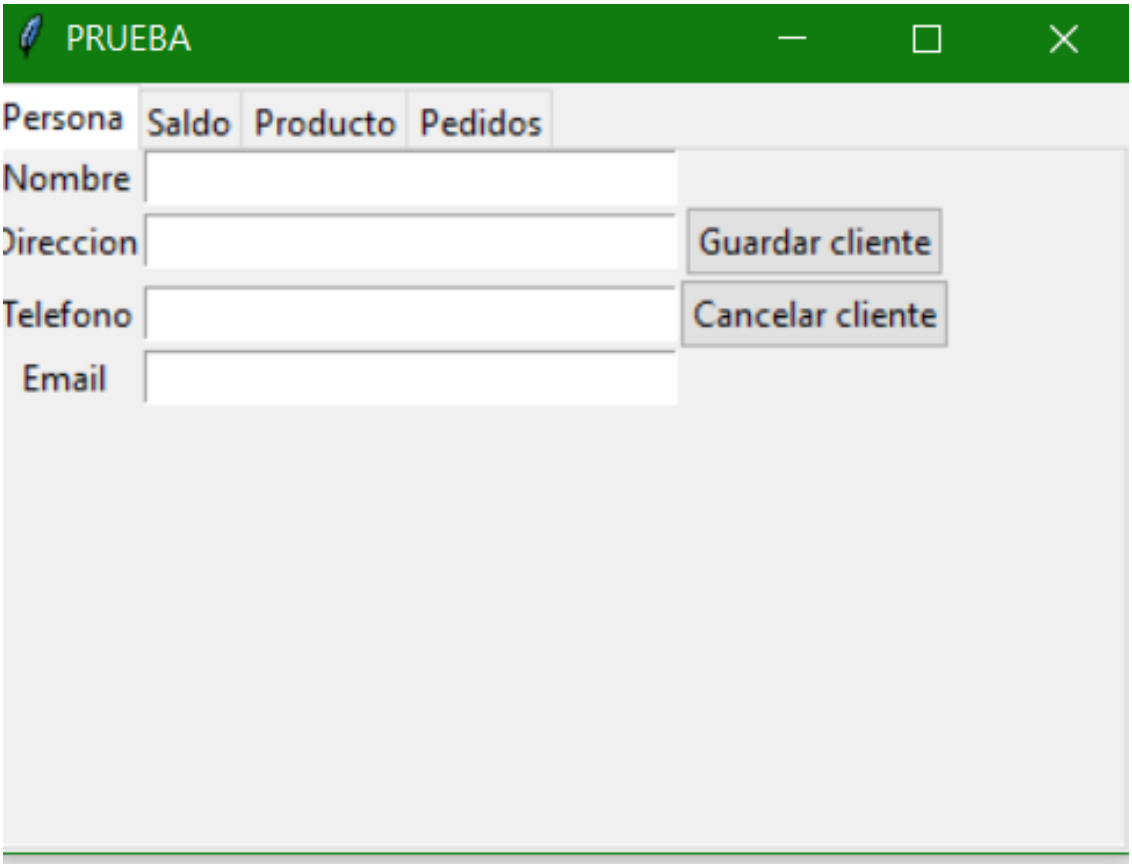
Finalmente, exportar un PDF del cuaderno de Jupyter Notebook visualizando el funcionamiento y validación del sistema.

Plazo: Se debe presentar el sistema funcionando hasta las **23:55 del 09/05/2021**, la misma que deberá ser subida al git personal y adicionalmente dentro de un cuaderno de Jupyter Notebook.

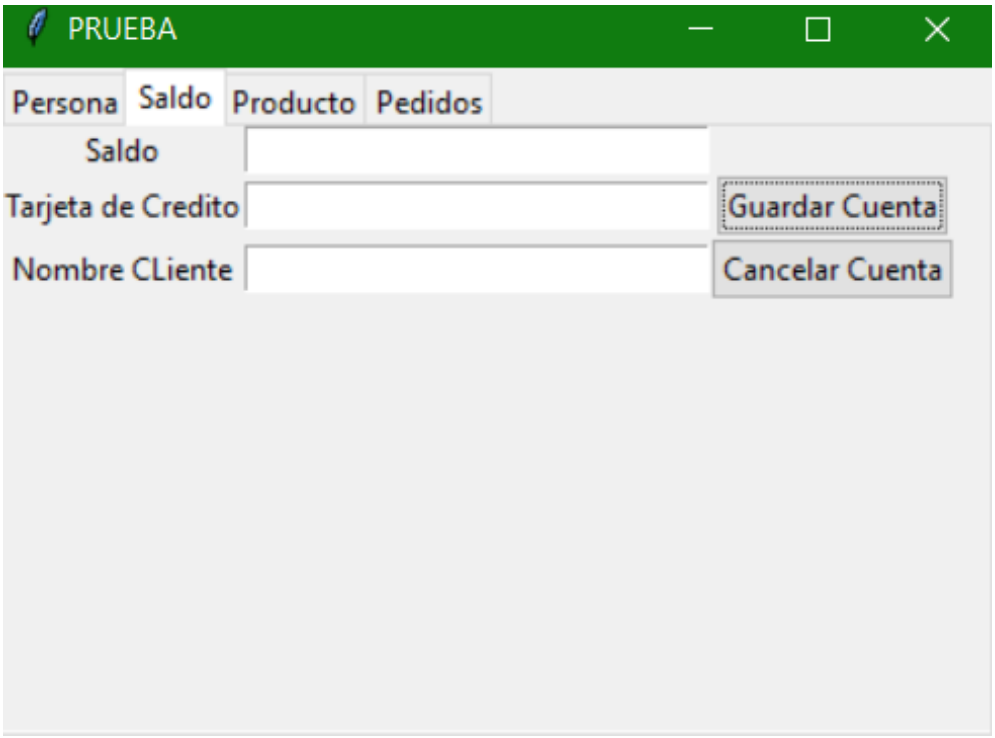
Simulación Tema: Inteligencia Artificial 1.

Nombre: Esteban Sibri

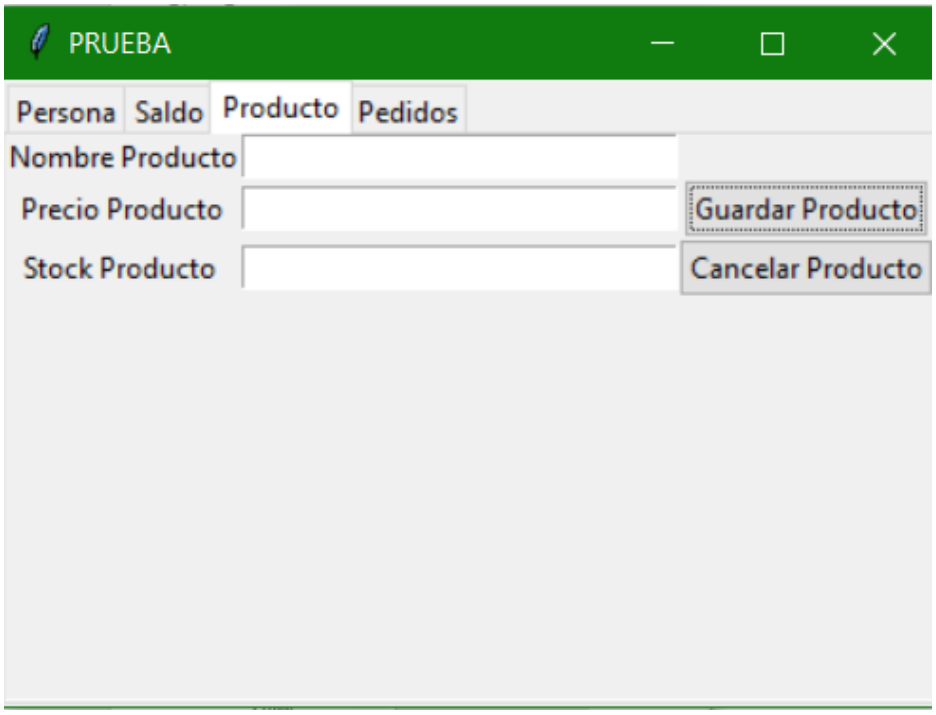
Página Persona:



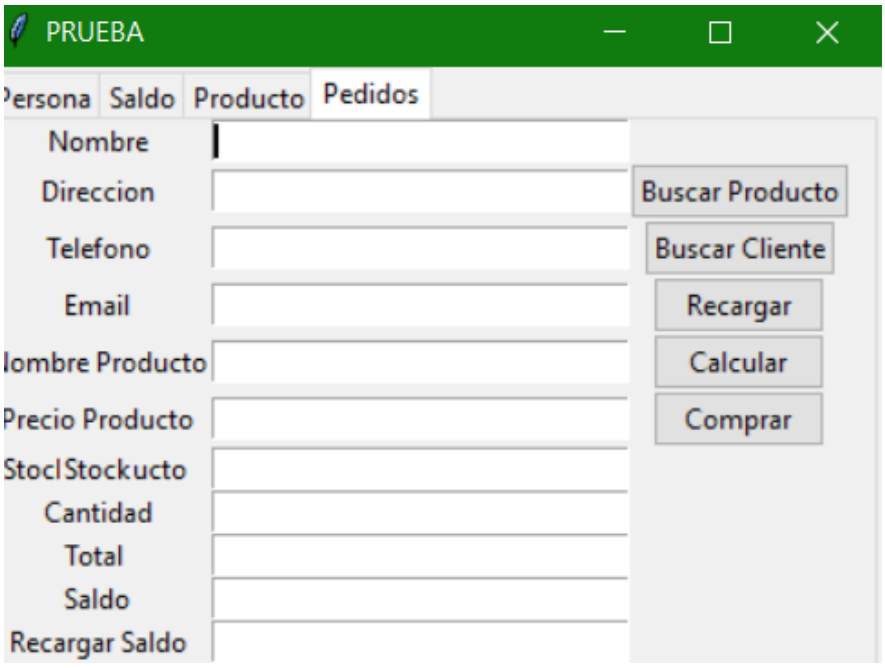
Página Saldo:



Página Producto:

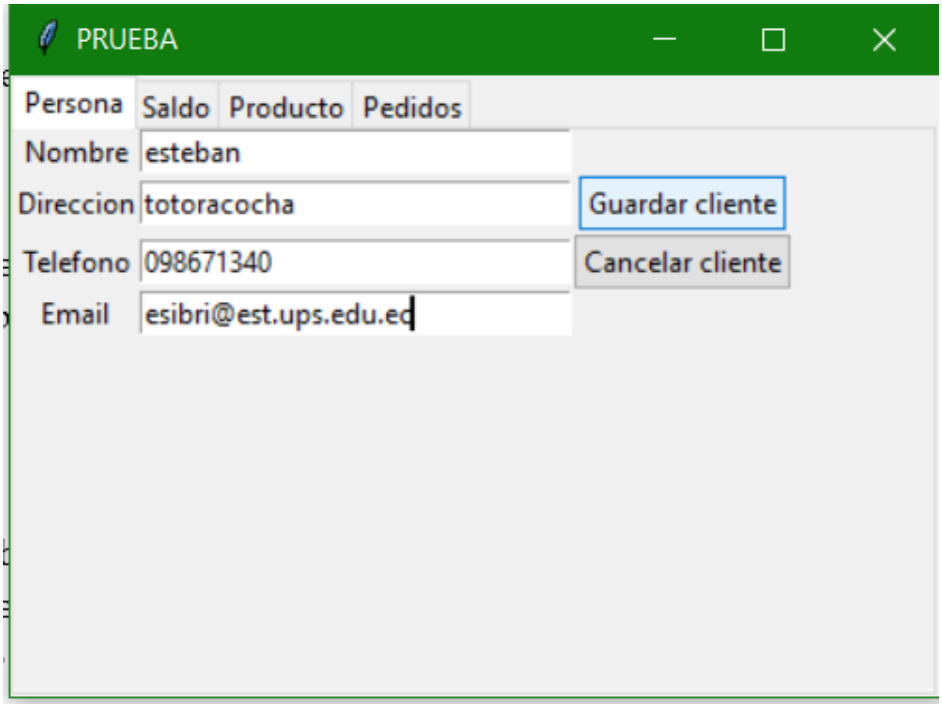


Página Pedidos:



Funcionamiento:

Registrar Cliente:

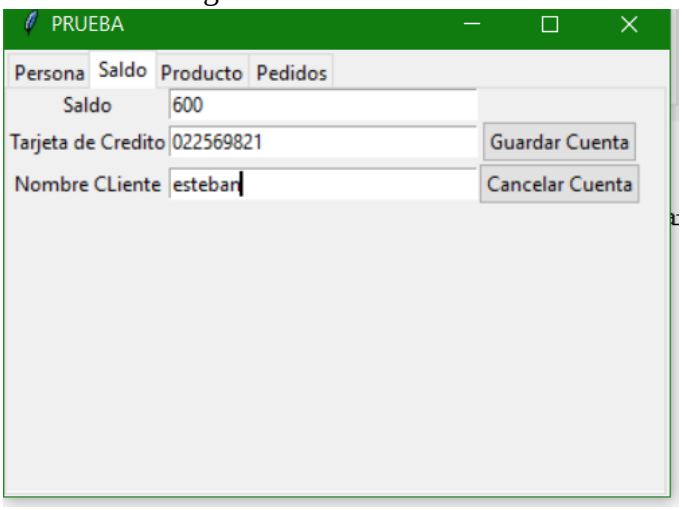


Al dar click en el botón guardar cliente, los datos ingresados se guardan en la base de datos, el botón cancelar borra los datos.

Cuenta:

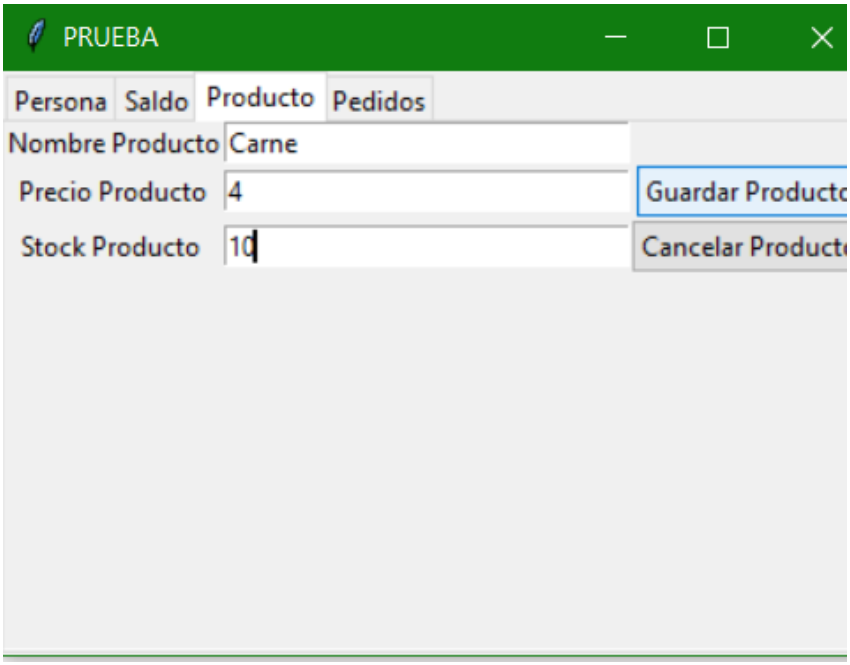
Al dar click en buscar cliente se llenarán la dirección, teléfono y el correo automáticamente del nombre que se escribió en el cuadro de texto, se debe primero buscar el cliente antes de crear la cuenta para que se genere la consulta en donde se podrá guardar la cuenta.

Al dar click en guardar cuenta se ingresan los datos en la base de datos



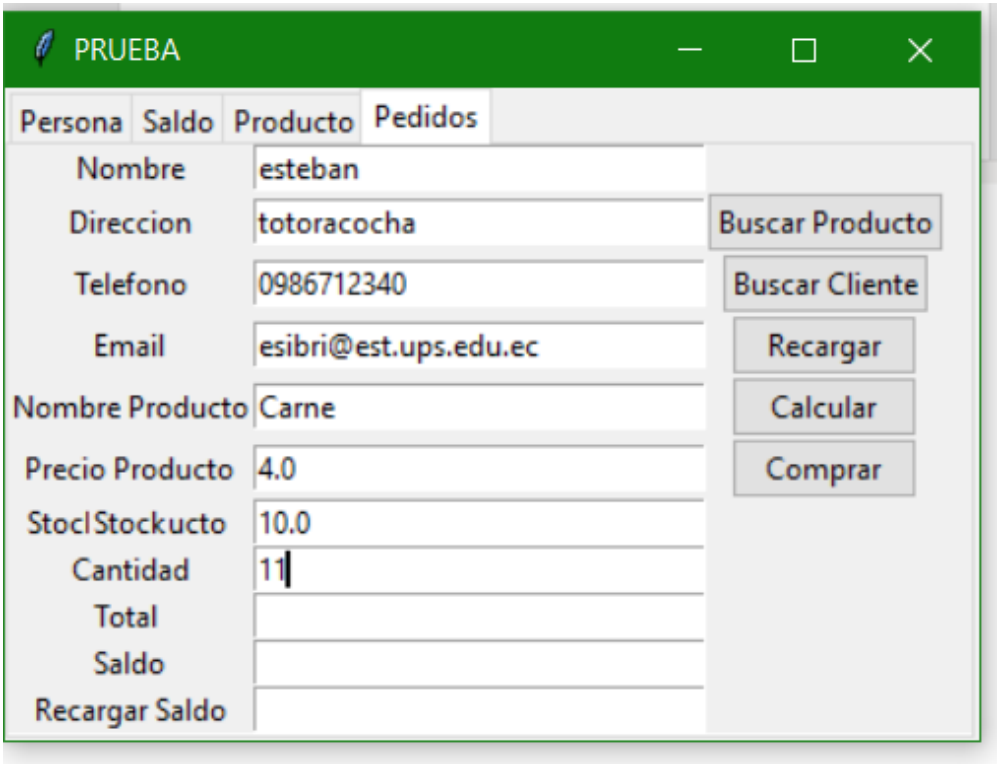
Productos:

Al dar click en el botón guardar producto, los datos ingresados se guardan en la base de datos, el botón cancelar borra los datos.

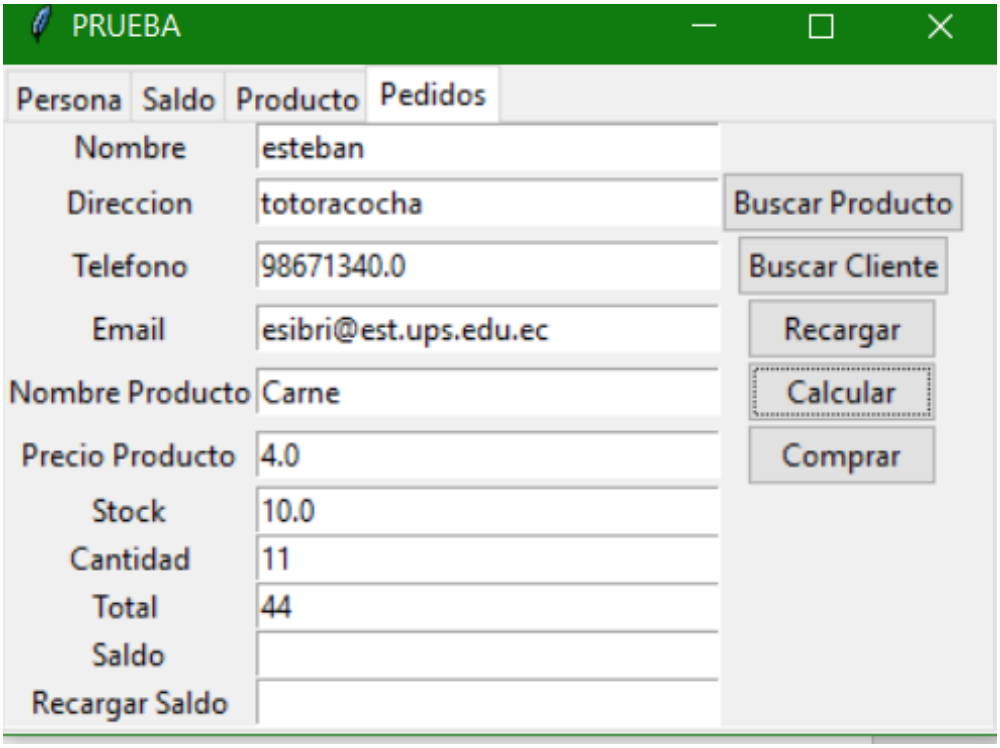


Realizar Pedido:

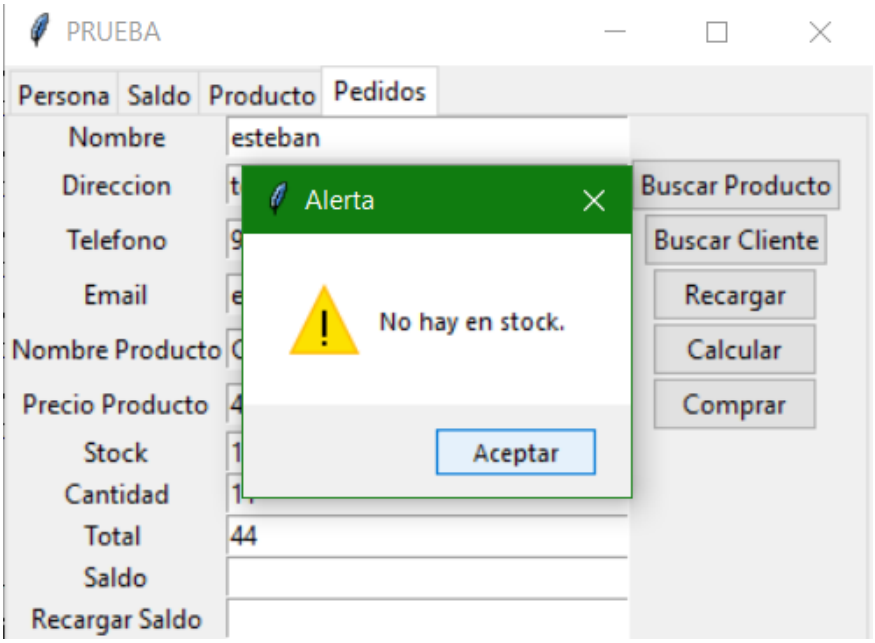
Para realizar el pedido se debe primero ingresar el nombre del cliente y presionar el botón buscar cliente el cua cargara los datos del usuario, luego se debe escribir el nombre del producto y presionar el botón buscar producto el cual cargara los datos del producto al ser presionado



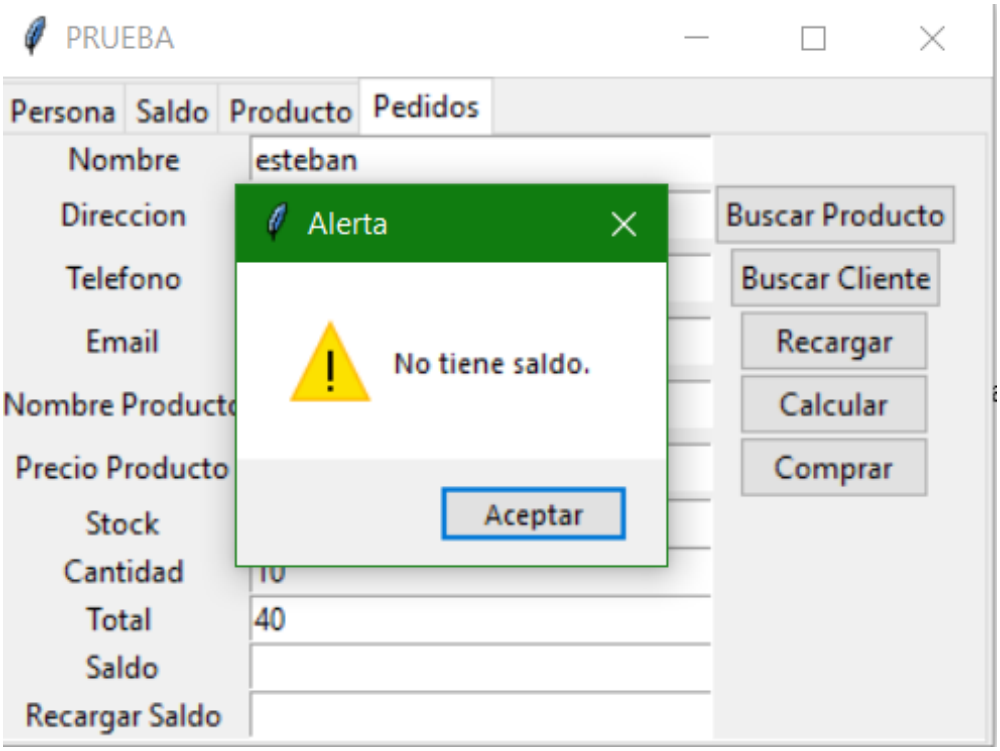
Luego debemos ingresar la cantidad a comprar, si presionamos el botón calcular este nos mostrara el valor total de la compra en el campo total



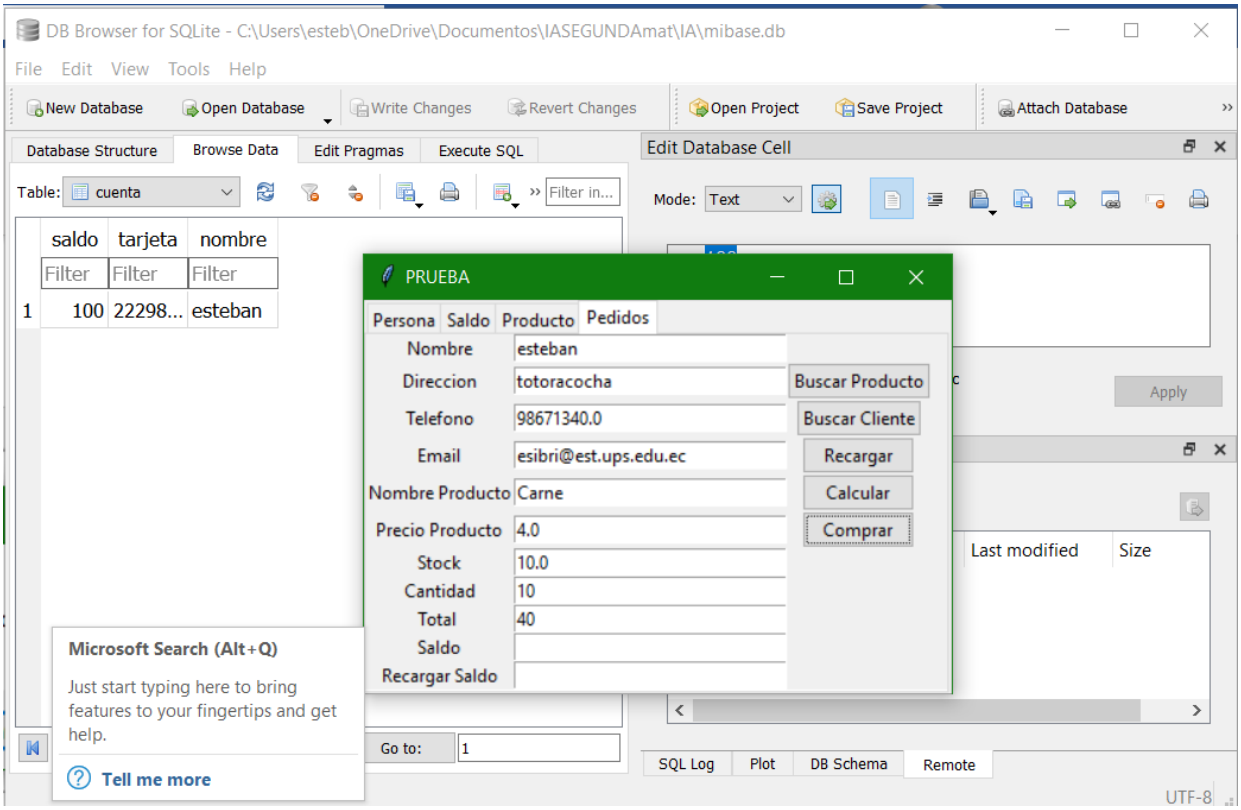
Luego se presiona el botón comprar, si excede el valor de stock se mostrará en pantalla un mensaje de advertencia.



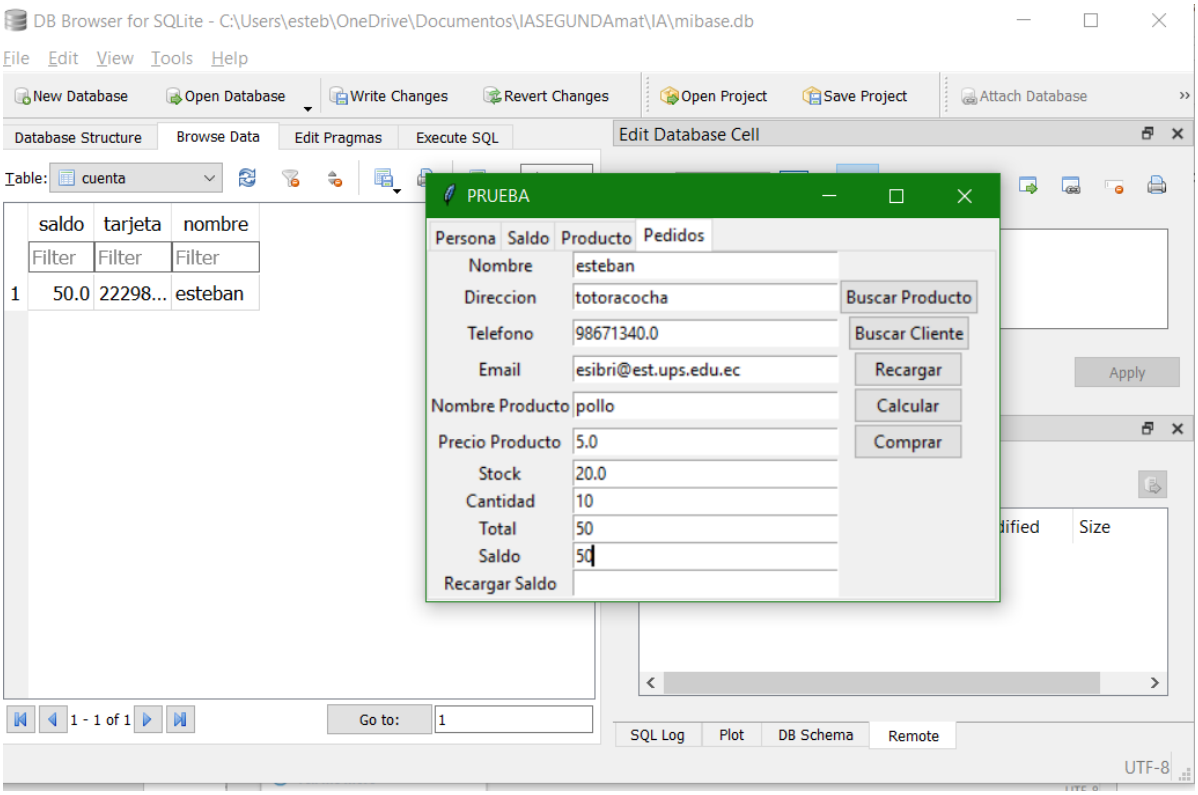
Si el producto se encuentra en stock y el usuario no tiene saldo en la tarjeta se mostrará el siguiente mensaje



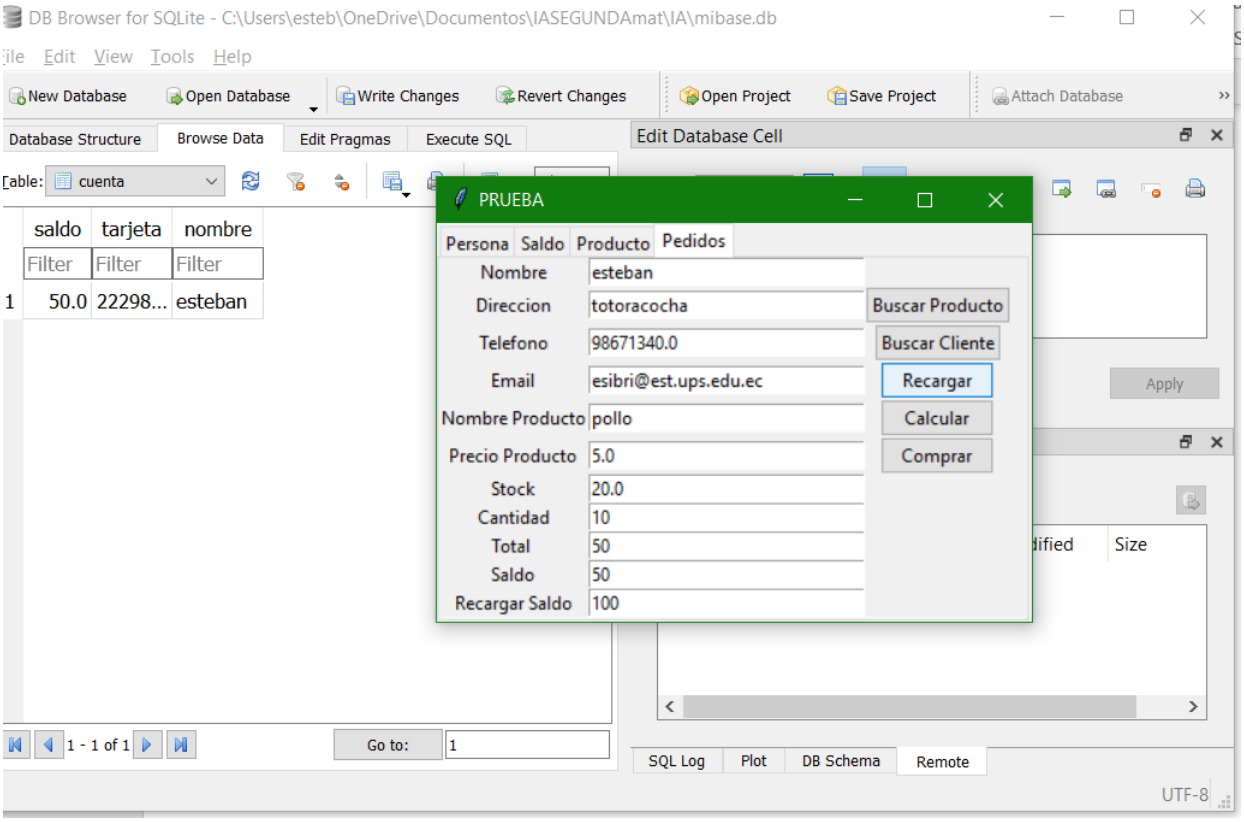
Si cumple con todos los requisitos se puede realizar la compra y se mostrara la disminución del saldo en la base de datos.



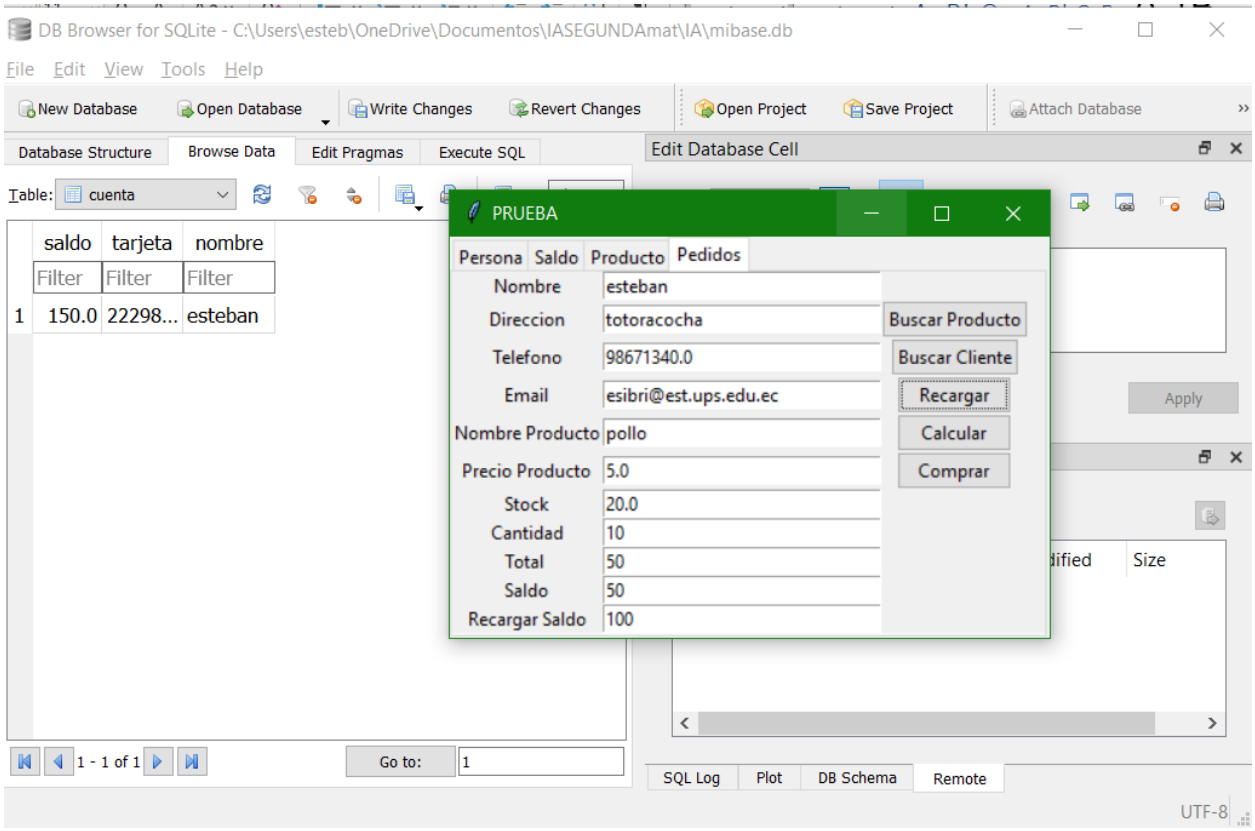
Simulación Tema: Inteligencia Artificial 1.



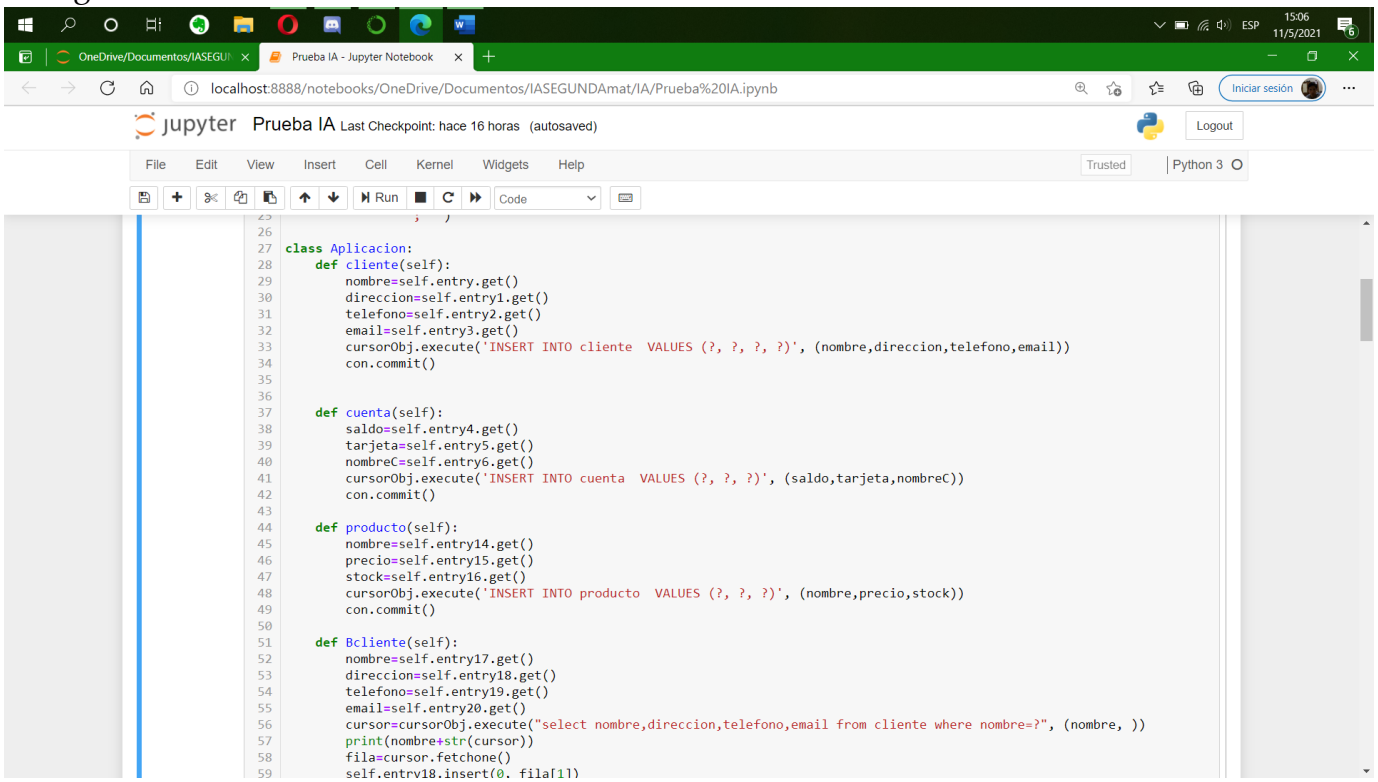
Para recargar saldo en la tarjeta del cliente se debe ingresar el nombre del cliente y rellenar el campo recargar saldo

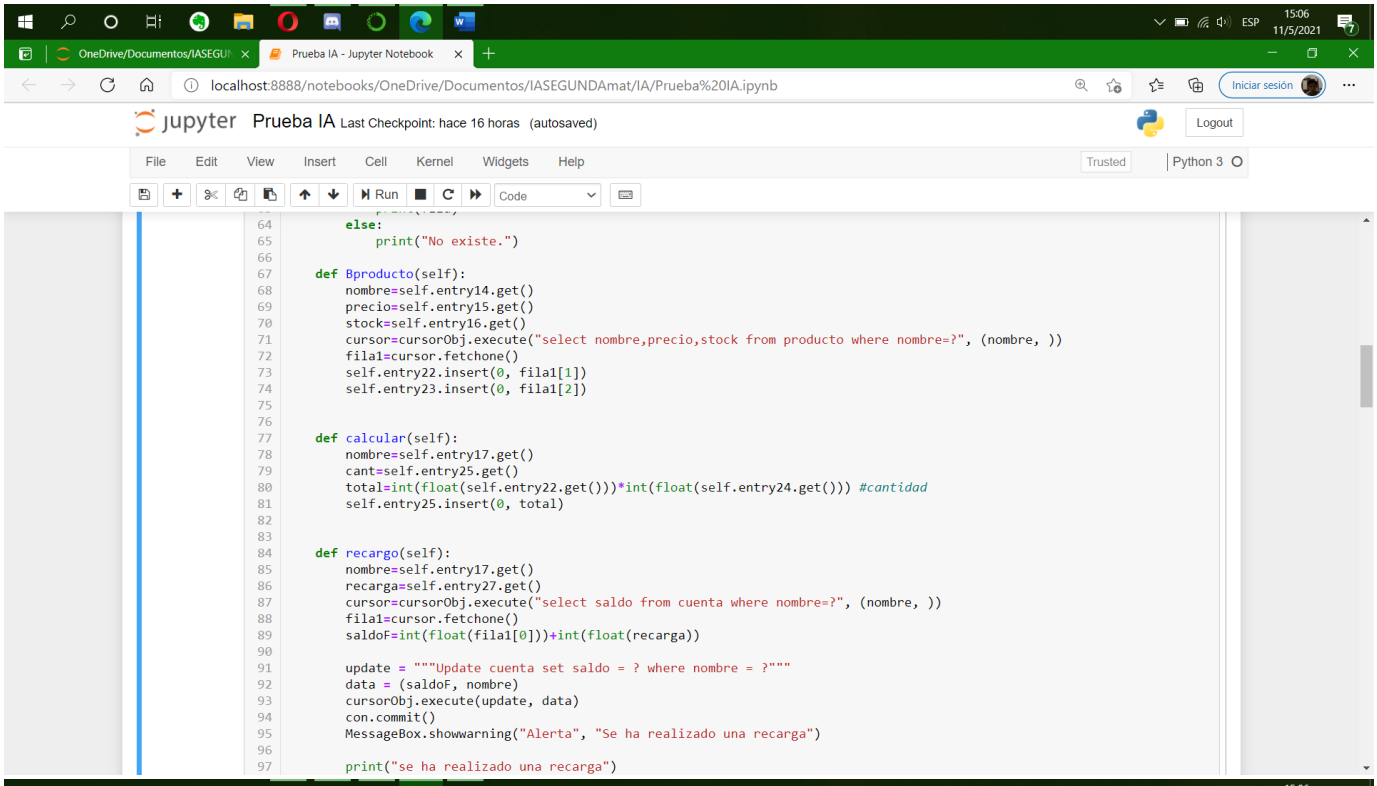


Simulación Tema: Inteligencia Artificial 1.

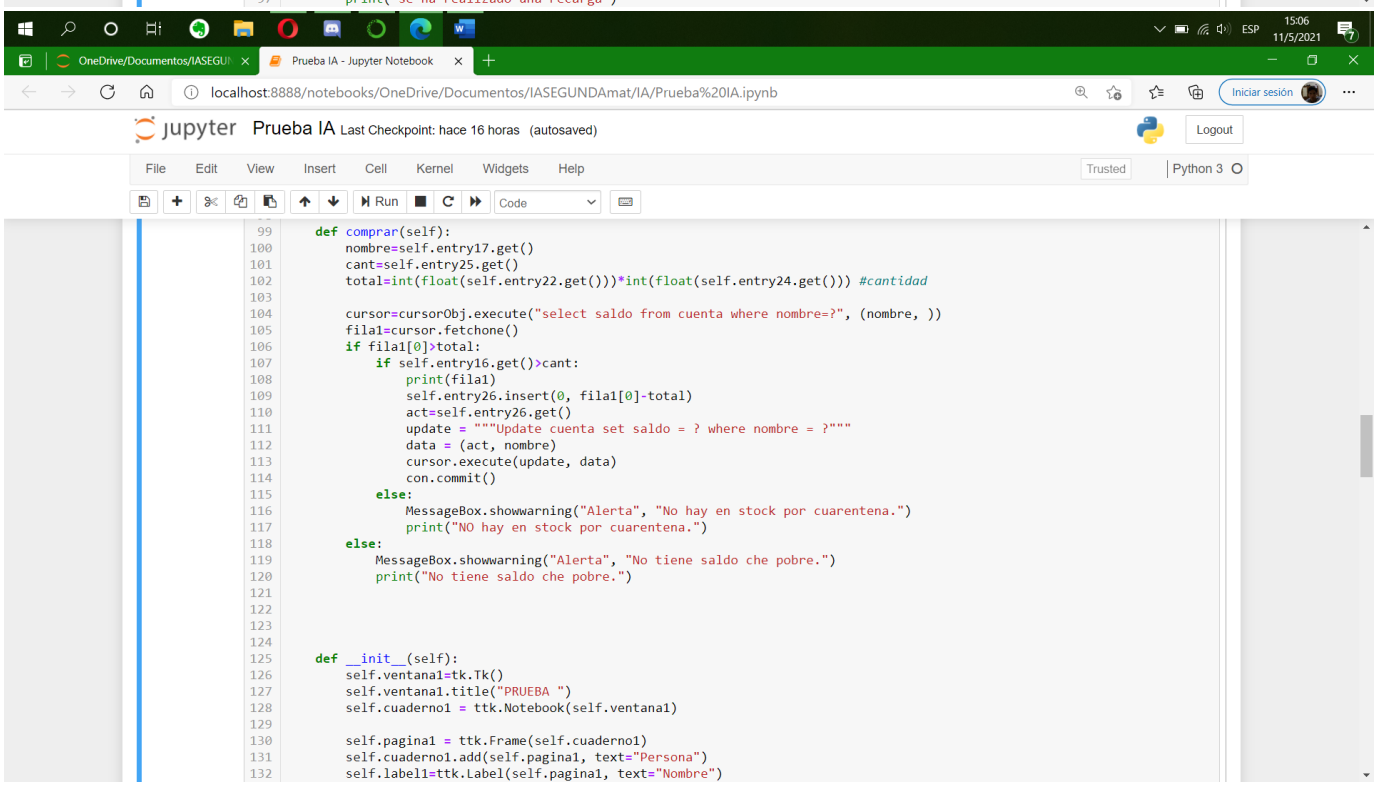


Codigo Fuente:






```
        else:
            print("No existe.")

    def Bproducto(self):
        nombre=self.entry14.get()
        precio=self.entry15.get()
        stock=self.entry16.get()
        cursor=cursorObj.execute("select nombre,precio,stock from producto where nombre=?", (nombre, ))
        fila1=cursor.fetchone()
        self.entry22.insert(0, fila1[1])
        self.entry23.insert(0, fila1[2])


    def calcular(self):
        nombre=self.entry17.get()
        cant=self.entry25.get()
        total=int(float(self.entry22.get()))*int(float(self.entry24.get())) #cantidad
        self.entry25.insert(0, total)


    def recargo(self):
        nombre=self.entry17.get()
        recarga=self.entry27.get()
        cursor=cursorObj.execute("select saldo from cuenta where nombre=?", (nombre, ))
        fila1=cursor.fetchone()
        saldoF=int(float(fila1[0]))+int(float(recarga))

        update = """Update cuenta set saldo = ? where nombre = ?"""
        data = (saldoF, nombre)
        cursorObj.execute(update, data)
        con.commit()
        MessageBox.showwarning("Alerta", "Se ha realizado una recarga")

        print("se ha realizado una recarga")
```




```
    def comprar(self):
        nombre=self.entry17.get()
        cant=self.entry25.get()
        total=int(float(self.entry22.get()))*int(float(self.entry24.get())) #cantidad

        cursor=cursorObj.execute("select saldo from cuenta where nombre=?", (nombre, ))
        fila1=cursor.fetchone()
        if fila1[0]>total:
            if self.entry16.get()>cant:
                print(fila1)
                self.entry26.insert(0, fila1[0]-total)
                act=self.entry26.get()
                update = """Update cuenta set saldo = ? where nombre = ?"""
                data = (act, nombre)
                cursor.execute(update, data)
                con.commit()
            else:
                MessageBox.showwarning("Alerta", "No hay en stock por cuarentena.")
                print("NO hay en stock por cuarentena.")
        else:
            MessageBox.showwarning("Alerta", "No tiene saldo che pobre.")
            print("No tiene saldo che pobre.")



    def __init__(self):
        self.ventana1=tk.Tk()
        self.ventana1.title("PRUEBA ")
        self.cuaderno1 = ttk.Notebook(self.ventana1)

        self.pagina1 = ttk.Frame(self.cuaderno1)
        self.cuaderno1.add(self.pagina1, text="Persona")
        self.label1=ttk.Label(self.pagina1, text="Nombre")
```