codigo prueba

data_clientes.py

```
import pandas as pd
from dotenv import load_dotenv
import os
from pathlib import Path


class Read_Cliente:
    def __init__(self):
        load_dotenv()
        path_clientes = os.getenv('path_clientes')
        path_cuentas = os.getenv('path_cuentas')
        self.file_clientes_path = path_clientes
        self.file_cuentas_path = path_cuentas


    def read_data(self):
        try:
            data_clientes = pd.read_csv(self.file_clientes_path)
            data_cuentas = pd.read_csv(self.file_cuentas_path)

            return data_clientes, data_cuentas
        except FileNotFoundError:
            print(f"Error: The file at {self.file_clientes_path} was not found.")
            print(f"Error: The file at {self.file_cuentas_path} was not found.")
            return None
        except pd.errors.EmptyDataError:
            print("Error: The file is empty.")
            return None
        except pd.errors.ParserError:
            print("Error: There was a parsing error while reading the file.")
            return None
        except Exception as e:
            print(f"An unexpected error occurred: {e}")
            return None
```

cajero.py

clientes.py

```
from data.data_clientes import Read_Cliente
import pandas as pd
class new_client():
    def __init__(self, Nombres, Apellidos, Identificacion, Movil, Correo, tipo_cuenta):
        self.Nombres = Nombres
        self.Apellidos = Apellidos
        self.Identificacion = Identificacion
        self.Movil = Movil
        self.Correo = Correo
        self.tipo_cuenta = tipo_cuenta

    def create_client(self):
        reader = Read_Cliente()
        df_client, df_cuenta = reader.read_data()
        id_client = df_client['ID_cliente'].iloc[-1] + 1
        ID_cuenta = df_client['ID_cuenta'].iloc[-1] + 1


        new_row_client = {
            'Nombres': self.Nombres,
            'Apellidos': self.Apellidos,
            'Identificacion': self.Identificacion,
            'Movil': self.Movil,
            'Correo': self.Correo,
            'Tipo cuenta': self.tipo_cuenta,
            'ID_cliente' : id_client,
            'ID_cuenta': ID_cuenta
        }

        df_client = pd.concat([df_client, pd.DataFrame([new_row_client])], ignore_index=True)
        df_client.to_csv(reader.file_clientes_path, index=False)
        print('Se creo correctamete el cliente con identificacion:', self.Identificacion)


class manatgement_client():
    def __init__(self):
        self.read_data = Read_Cliente()
        self.data_clientes, self.data_cuentas = self.read_data.read_data()

    def consignation(self, identificacion):

        row_client = self.data_clientes[self.data_clientes['Identificacion'] == identificacion]

        if len(row_client ) > 0:
            id_cuenta = row_client['ID_cuenta'].values[0]

            row_cuenta = self.data_cuentas[self.data_cuentas['ID_Cuenta'] == id_cuenta]
            value = int(input("Valor a consignar:\n"))
            last_row = row_cuenta.sort_values('Fecha_operacion', ascending=False)
```

```
            if last_row.empty:
                print("Sin operaciones previas. Esta es la primera consignaciÃ³n.")
                new_operation = {
                'ID_Cuenta': id_cuenta,
                'Saldo' : value,
                'Tipo_movimiento': 1,
                'valor_movimiento': value,
                'Nuevo_saldo': value,
                'Fecha_operacion' : pd.Timestamp.now()
                }

            else:

                new_saldo = last_row['Nuevo_saldo'].values[0] + value
                new_operation = {
                    'ID_Cuenta': id_cuenta,
                    'Saldo' : new_saldo,
                    'Tipo_movimiento': 1,
                    'valor_movimiento': value,
                    'Nuevo_saldo': new_saldo,
                    'Fecha_operacion' : pd.Timestamp.now()
                }
            df_cuenta = pd.concat([self.data_cuentas, pd.DataFrame([new_operation])], ignore_index=True)
            df_cuenta.to_csv(self.read_data.file_cuentas_path, index=False)
            print(f'Se ha consignado {value} a la cuenta {id_cuenta}.')
        else:
            print(f"No se puede realizar la consignaciÃ³n. No se encontrÃ³ una cuenta asociada a la identificaciÃ³n {identificacion}.")
            pass


    def withdrawal(self, identificacion):
        value = int(input("Valor a retirar:\n"))
        row_client = self.data_clientes[self.data_clientes['Identificacion'] == identificacion]
        id_cuenta = row_client['ID_cuenta'].values[0]
        row_cuenta = self.data_cuentas[self.data_cuentas['ID_Cuenta'] == id_cuenta]

        if id_cuenta is not None:
            last_row = row_cuenta.sort_values('Fecha_operacion', ascending=False)
            if last_row.empty:
                print("Sin operaciones previas. No se puede retirar.")
                return

            new_saldo = last_row['Nuevo_saldo'].values[0] - value

            if new_saldo < 0:
                print("Saldo insuficiente para realizar el retiro.")
                return

            new_operation = {
                'ID_Cuenta': id_cuenta,
                'Saldo' : new_saldo,
                'Tipo_movimiento': 2,
                'valor_movimiento': value,
                'Nuevo_saldo': new_saldo,
                'Fecha_operacion' : pd.Timestamp.now()
            }

            df_cuenta = pd.concat([self.data_cuentas, pd.DataFrame([new_operation])], ignore_index=True)
            df_cuenta.to_csv(self.read_data.file_cuentas_path, index=False)
            print(f'Se ha retirado {value} de la cuenta {id_cuenta}.')
        else:
            print(f"No se puede realizar el retiro. No se encontrÃ³ una cuenta asociada a la identificaciÃ³n {identificacion}.")

    def query_client(self, identificacion):
        if self.data_clientes is not None:
            row_client = self.data_clientes[self.data_clientes['Identificacion'] == identificacion]
            row_cuenta = self.data_cuentas[self.data_cuentas['ID_Cuenta'] == row_client['ID_cuenta'].values[0]]
            if len(row_cuenta) == 0:
                print(f"La identificaciÃ³n {identificacion}, no registra movimientos.\n")
            else:

                row_client = row_cuenta.sort_values("Fecha_operacion", ascending=True)

                saldo = row_client['Saldo'].iloc[-1]
                print("El saldo de la cuenta es:", saldo)
        else:
            print("No se pudo recuperar la informaciÃ³n de los clientes.")
```

main.py

```
from src.clientes import new_client, manatgement_client
import os

while True:
   # os.system('cls')  # Limpia la consola en Windows
    print("Bienvenido al sistema de gestiÃ³n de clientes.")
    Identificacion = int(input("CC:\n"))
    print("Indique que operacion desea realizar:")
    print("1. Crear cliente")
    print("2. Consultar saldo")
    print("3. Consignar dinero")
    print("4. Retirar dinero")

    opcion = input("Seleccione una opciÃ³n:\n")
    try:
        opcion = int(opcion)
    except ValueError:
```

```
    print("Por favor ingrese un nÃºmero vÃ¡lido.")
    continue

if opcion == 1:
    print("Crear cliente")

    nombre = input("Nombres:\n")
    apellido = input("Apellidos:\n")
    movil = int(input("Movil:\n"))
    correo = input("Correo:\n")
    tipo_cuenta = input("Tipo de cuenta:\n")

    client = new_client(
        Nombres=nombre,
        Apellidos=apellido,
        Identificacion=Identificacion,
        Movil=movil,
        Correo=correo,
        tipo_cuenta=tipo_cuenta
    )
    client.create_client()
elif opcion == 2:
    print("Consultar cliente")
    client = manatgement_client()
    client.query_client(Identificacion)

elif opcion == 3:
    print("Consignacion de dienero")
    client = manatgement_client()
    client.consignation(Identificacion)

elif opcion == 4:
    print("Retiro de dinero")
    client = manatgement_client()
    client.withdrawal(Identificacion)
```