

# codigo prueba

## data\_clientes.py

```
"""
data_clientes.py
=====

Este módulo gestiona la lectura de datos de clientes y cuentas bancarias desde archivos CSV.

Funcionalidades:
-----
- Leer los datos de clientes y cuentas desde rutas especificadas en variables de entorno.
- Manejar errores comunes al leer archivos CSV (archivo no encontrado, archivo vacío, errores de parseo).

Clases:
-----
- Read_Cliente:
    - Métodos:
        - read_data(): Devuelve dos DataFrames de pandas con los datos de clientes y cuentas.

Entradas:
-----
- Rutas de archivos CSV definidas en variables de entorno:
    - path_clientes
    - path_cuentas

Dependencias:
-----
- pandas
- python-dotenv
- os
- pathlib

Autor: Tu Nombre
Fecha: 2024

"""

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

```
"""
clientes.py
=====

Este módulo implementa la lógica para la gestión de clientes y operaciones bancarias.
```

Funcionalidades:

-----

- Crear un nuevo cliente y su cuenta asociada.
- Realizar consignaciones y retiros en cuentas existentes.
- Consultar el saldo de una cuenta por identificaci3n.

Clases:

-----

- new\_client:
  - Permite crear un nuevo cliente y guardar sus datos en el archivo correspondiente.
- manatgement\_client:
  - M3todos:
    - consignation(identificacion): Consigna dinero en la cuenta asociada.
    - withdrawal(identificacion): Retira dinero de la cuenta asociada.
    - query\_client(identificacion): Consulta el saldo de la cuenta asociada.

Entradas:

-----

- Datos personales del cliente.
- Identificaci3n del cliente.
- Valor a consignar o retirar.

Dependencias:

-----

- pandas
- data\_clientes.Read\_Cliente

"""

```
from data.data_clientes import Read_Cliente
import pandas as pd
from src.operation import Operacion
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 cre3 correctamente 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 realizar_operacion(self, identificacion, operacion: "Operacion"):
        operacion(self.data_clientes, self.data_cuentas, self.read_data.file_cuentas_path).ejecutar(identificacion)
```

## main.py

"""

main.py  
=====

Este m3dulo implementa la interfaz principal para el sistema de gesti3n de clientes de un cajero bancario.

Funcionalidades:

-----

- Crear un nuevo cliente y su cuenta asociada.
- Consultar el saldo de una cuenta por identificaci3n.
- Consignar dinero en una cuenta existente.
- Retirar dinero de una cuenta existente.

Uso:

----

Al ejecutar el script, se muestra un menú interactivo en consola donde el usuario puede seleccionar la operación deseada.

Clases utilizadas:

-----

- new\_client: Permite crear un nuevo cliente.
- management\_client: Permite consultar saldo, consignar y retirar dinero.

Entradas:

-----

- Identificación del cliente (CC)
- Datos personales para la creación de cliente
- Valor a consignar o retirar

"""

```
from src.clientes import new_client, management_client
import os
from src.operation import ConsultaSaldo, Consignacion, Retiro

while True:
    # os.system('cls') # Limpia la consola en Windows
    print("Bienvenido al sistema de gestión 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 opción:\n")
    try:
        opcion = int(opcion)
    except ValueError:
        print("Por favor ingrese un número válido.")
        continue
    client = management_client()
    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:
        client.realizar_operacion(Identificacion, ConsultaSaldo)

    elif opcion == 3:
        client.realizar_operacion(Identificacion, Consignacion)

    elif opcion == 4:
        client.realizar_operacion(Identificacion, Retiro)
```