ISM Bank

PROGRAMACION I

Adrián López
David Muñoz
Hugo López
Jordi Blasco

Descripción

- *El objetivo del proyecto era crear una aplicación de banco capaz de guardar información de los usuarios para luego poder hacer operaciones dentro de ella.*

- *El proyecto lo hemos dividido en dos grandes bloques, el bloque de Inicio de Sesion y el bloque de Crear Cuenta.*

  Empezaremos explicando cómo funciona la parte de Crear Cuenta. La función principal de Crear Cuenta está dividida en varias subfunciones:

```
#FUNCION PRINCIPAL PARA CREAR CUENTA

def crearCuenta():
    g.linea(y=2)
    g.linea(c='¯')

    nombre_usuario = pedirNombre()
    nombre_usuario = comprobarUsuario(nombre_usuario)

    g.linea(c='¯')

    passw = pedirPassw()
    passw = comprobarPassw(passw,nombre_usuario)

    g.linea()
    g.linea(c='_')
    g.bienvenida(nombre_usuario,passw)



    pedirDatos()
    saveUserPassw(nombre_usuario,passw)
    generaInfBancaria()


    return
```

1ª. Pide el nombre de usuario y la contraseña y comprueba que primero no se repita el nombre de usuario y que ambas tengan un mínimo de caracteres como otros requisitos mínimos.

```python
#PIDE UN NOMBRE DE USUARIO Y COMPRUEBA QUE CUMPLA CON LOS REQUISITOS

def pedirNombre():
    nombre_usuario = input('Introduce un nombre de usuario: ')
    return nombre_usuario

def comprobarUsuario(nombre_usuario):

    while len(nombre_usuario) < 3 or nombre_usuario in c.informacion_clientes['usuario']:

        if len(nombre_usuario) < 3:
            print('El nombre de usuario tiene que tener mas de 3 caracteres')
        else:
            print('Este nombre de usuario ya ha sido usado')

        g.linea(c='_')
        nombre_usuario = pedirNombre()

    return nombre_usuario
```

```python
#PIDE UNA CONTRASEÑA Y COMPRUEBA QUE CUMPLA CON LOS REQUISITOS


def pedirPassw():
    passw = input('Introduce una contrasena para la cuenta: ')
    return passw


def comprobarPassw(passw,user):
    while len(passw) < 8 or not any(i.isupper() for i in passw) or not any(i.isdigit() for i in passw) or user.lower() in passw.lower():
        g.linea(c='_')

        if len(passw) < 8:
            print ('La contrasena debe de tener 8 o mas caracteres')
        if not any(i.isupper() for i in passw):
            print('La contrasena debe de tener almenos una mayuscula')
        if not any(i.isdigit() for i in passw):
            print ('La contrasena debe de tener al menos un numero')
        if user.upper() in passw.upper():
            print ('La contrasena no puede contener tu nombre de usuario')
        g.linea(c='¯')
        passw = pedirPassw()


    return passw
```

2ª. Pide todos los datos personales tales como el nombre y apellidos, el número de teléfono y otros varios. Comprueba todas estas strings para que cumplan con requisitos como el no tener caracteres especiales y que sean lo más cercanos a la realidad posible, esto se hace gracias a la función ComprobarStrCarcEsp() que hablaremos de ella en el apartado de funciones genéricas. Por ejemplo en el DNI comprueba que sean 9 caracteres y obligatoriamente los 8 primeros números y el ultimo la letra. Después de pedir los datos los guarda en una variable diccionario que se llama usuario nuevo.

```
#PEDIR EL CONTENIDO DE LOS DATOS PERSONALES Y AUTOMATICAMENTE COMPROBAR QUE CUMPLEN CON LOS REQUISITOS

def pedirDatos():
    print('DATOS PERSONALES')
    g.linea(c='¯')

    pide (x='nombre',y='Nombre: ',z=[3,10])
    pide (x='apellidos',y='Apellidos: ',z=[6,25],carc='ap')
    pide(x='DNI',y='DNI: ',z=[9,9],carc='ln')
    #FUNCION MAS COMPLICADA YA QUE TENEMOS QUE COMPROBAR QUE LOS PRIMEROS 8 CARACTERES SEAN NUMEROS Y
    #QUE EL ULTIMO SEA UNA LETRA, ADEMAS DE QUE EN TOTAL TIENEN QUE SER 9 CIFRAS
    while not (g.comprobarStringCarcEsp(c.usuarioNuevo['DNI'][:8],tipo='n') and
               g.comprobarStringCarcEsp(c.usuarioNuevo['DNI'][8],tipo='l')):

        print('DNI no válido')

        c.usuarioNuevo['DNI'] = input('DNI: ')
        pide(x='DNI',y='DNI: ',z=[9,9],carc='ln',first=False)

    pide (x='telefono',y='Telefono: ',z=[9,9],carc='n')
    pide (x='email', y='Correo electronico: ', z=[5,35],carc='e')
    pide (x='CP', y='Codigo postal: ', z=[5,5],carc='n')
    pide (x='direccion', y='Direccion: ', z=[6,30], carc='ln')

    guardarUserNuevo()
```

Pide() llama a ComprobarStrCarcEsp() con las misma variables que esta tiene pero le añade el mensaje de valido y no valido además de guardarlo todo en la variable diccionario de Usuario Nuevo. Variables (str, tipo de comprobación: letras, números, emails etc., longitud que tiene que tener la str)

```
# COMPRUEBA SI EL ATRIBUTO CONTIENE UNA S AL FINAL PARA ESCRIBIR SI ES VALIDO O NO USANDO LA FUNCION DE COMPROBAR STRINGS

def pide(x,y,carc='l',z=None,first=True):

    if first:
        c.usuarioNuevo[x] = input(y).lower()

    while not g.comprobarStringCarcEsp(c.usuarioNuevo[x],long=z,tipo=carc):

        if x[-1] == 's':
            print(x.capitalize() + ' invalidos')
        else:
            print(x.capitalize() + ' no valido')
        g.linea(c='¯')

        c.usuarioNuevo[x] = input(y).lower()

    return c.usuarioNuevo[x]
```

3ª. Guarda la información que ya estaba guardada en la variable diccionario (Usuario Nuevo) en archivos de texto, cada valor del diccionario pertenece a la misma línea de archivos diferentes. Es decir, si tenemos el valor de nombre guardado, la función la almacena en la misma línea en la que almacena todos los demás datos pero en diferentes archivos. Tenemos tantos archivos como tipos de variables y tantas líneas en cada archivo como cliente. Lo hicimos así porque nos resultó más fácil que separar cada dato por comas modificarlos de esa forma.

También genera datos necesarios para una cuenta de banco como como el número de tarjeta o como la fecha de caducidad que añadirá a la fecha que se crea la cuenta 7 años

```python
#FUNCIONES PARA GUARDAR LA INFORMACION DE INICIO DE SESION

def saveUserPassw(user,passw):
    with open ('informacionUsuarios/contrasena.txt', 'ta') as passwtx:
        passwtx.write(passw + '\n')

    with open ('informacionUsuarios/user.txt', 'ta') as usertx:
        usertx.write(user + '\n')

#GENERA LA INFORMACION BANCARIA ALEATORIA COMO EL NUMERO DE LA TARJETA Y INICALIZA LOS VALORES DE SALDO Y BALANCE A 0,
# GUARDA TODO EN LOS FICHEROS

def generaInfBancaria():
    with open ('informacionUsuarios/saldo.txt', 'ta') as saldotx:
        saldotx.write('0' + '\n')
    with open ('informacionUsuarios/balance.txt', 'ta') as balancetx:
        balancetx.write('0' + '\n')

    #UTILIZA LA FECHA DE HOY PARA AÑADIRLE 7 AÑOS Y ASI DATAR EL AÑO DE CADUCIDAD DE LA TARJETA TAL COMO SE HACE EN LOS BANCOS

    fechaHoy = datetime.now()
    fechaAñoStr = (int(fechaHoy.strftime('%Y')) + 7)
    fechaMesStr = (fechaHoy.strftime('%m'))
    fechaMesAñoStr = fechaMesStr + '/' + str(fechaAñoStr)[2:]

    with open ('informacionUsuarios/fechaCaducidad.txt', 'ta') as fechCadtx:
        fechCadtx.write(fechaMesAñoStr + '\n')
    with open ('informacionUsuarios/numTarjeta.txt', 'ta') as numTartx:
        numTartx.write(g.aleatorio(16) + '\n')
    with open ('informacionUsuarios/IBAN.txt', 'ta') as IBANtx:
        IBANtx.write('ES' + g.aleatorio(22) + '\n')
```

```python
#GUARDA LA INFORMACION RESTANTE EN LOS FICHEROS
def guardarUserNuevo(user=c.usuarioNuevo):
    with open ('informacionUsuarios/nombreApellidos.txt', 'ta') as nApll:
        nApll.write(user['nombre'] + ' ' + user['apellidos'] + '\n')
    with open ('informacionUsuarios/DNI.txt', 'ta') as DNItx:
        DNItx.write(user['DNI'] + '\n')
    with open ('informacionUsuarios/telefono.txt', 'ta') as tel:
        tel.write(user['telefono'] + '\n')
    with open ('informacionUsuarios/email.txt', 'ta') as email:
        email.write(user['email'] + '\n')
```

Seguiremos explicando como funciona el apartado de Inicio de Sesión, su función principal la hemos dividido también en subfunciones:

```python
#FUNCION PRINCIPAL DE INICIAR SESION, COMPRUEBA QUE LA CONTRASEÑA COINCIDA CON EL USUARIO Y
#DEVUELVE EL NUMERO DE CLIENTE QUE CORRESPONDERÁ AL HUECO DE LA LISTA DE LOS VALORES DEL DICCIONARIO EN LA QUE
# ESTARAN GUARDADOS LOS DATOS DE LOS USUARIOS

def iniciarSesion():

    cliente = comprobarCredenciales()
    if cliente == None:
        return

    g.linea()
    g.bienvenida2(c.informacion_clientes['usuario'][cliente])
    infoCuenta(cliente)

    g.linea()
    menu(cliente)
```

1ª Comprueba que las credenciales sean las que tenemos guardadas en el nuevo diccionario que hemos creado a partir de los archivos de texto que tendrá la siguiente forma:

Información_clientes={‘usuario’:[‘sara’,’mateo’,…],’contraseña’:[‘Sarita33’,’Chuete8’,…],

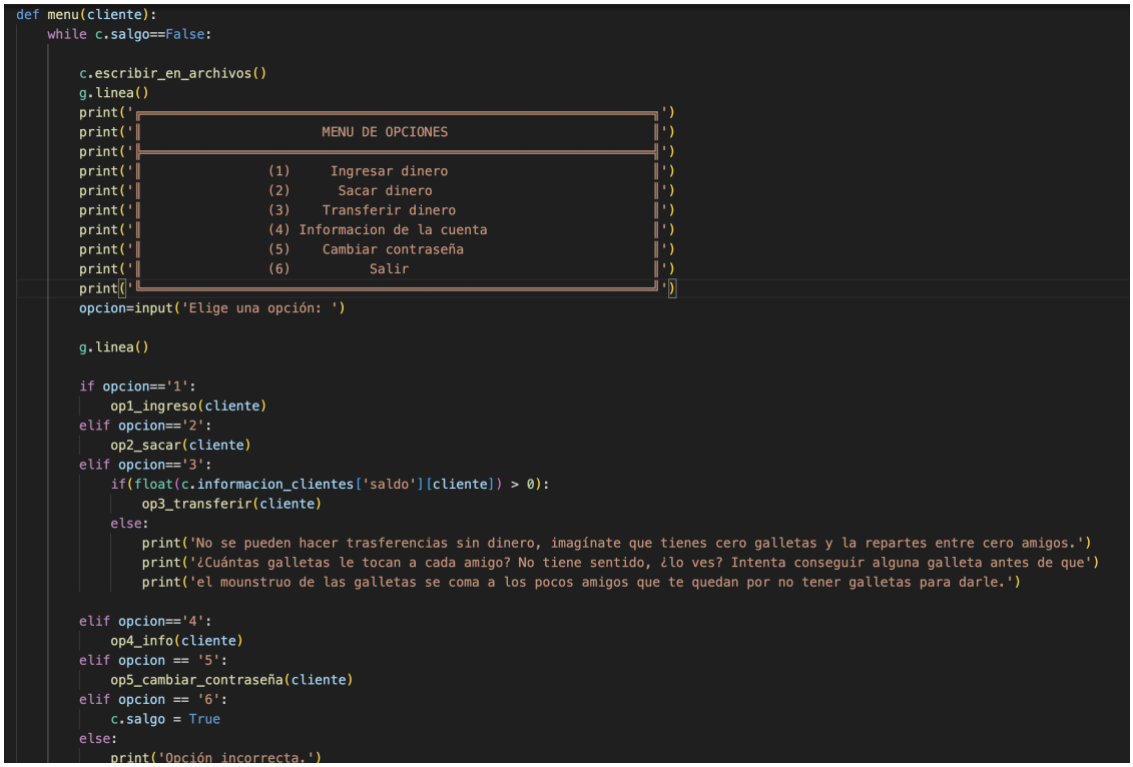
’numTel’:[’6666666666’,’633456875’,…],…:[…,]}

Sean las que corresponda con el usuario y contraseña correspondiente y si no te volvera a pedir las credenciales hasta 3 veces

```python
def comprobarCredenciales():
    n = 0


    while n < 3:
        print('╔═══════════════════════════════════════════════════════════')
        usuario_iniciando = input('║ Introduce el usuario para iniciar sesión: ')
        contrasena_iniciando = input('║ Introduce la contraseña para iniciar sesión: ')
        print('╚═══════════════════════════════════════════════════════════')

        for i in range (len(c.informacion_clientes['usuario'])):
            if c.informacion_clientes['usuario'][i] == usuario_iniciando and c.informacion_clientes['contraseña'][i] == contrasena_iniciando:
                cliente = i




                return cliente

        print('Usuario o contraseña incorrectos.')
        n = n + 1

    return None
```

2ª. Te muestra la información de la cuenta del cliente que ha iniciado sesión

3ª. Entra en el menú dividido en 6 subfunciones más, en las que modifican los datos de la cuenta sacando o ingresando dinero, haciendo transferencias a través de bizum, cambiando la contraseña o viendo la información de la cuenta.



```
def menu(cliente):
    while c.salgo==False:

        c.escribir_en_archivos()
        g.linea()
        print('╔═══════════════════════════════════════════════════╗')
        print('║                    MENU DE OPCIONES               ║')
        print('╠═══════════════════════════════════════════════════╣')
        print('║              (1)    Ingresar dinero                ║')
        print('║              (2)     Sacar dinero                 ║')
        print('║              (3)   Transferir dinero              ║')
        print('║              (4) Informacion de la cuenta         ║')
        print('║              (5)   Cambiar contraseña             ║')
        print('║              (6)         Salir                    ║')
        print('╚═══════════════════════════════════════════════════╝')
        opcion=input('Elige una opción: ')

        g.linea()

        if opcion=='1':
            op1_ingreso(cliente)
        elif opcion=='2':
            op2_sacar(cliente)
        elif opcion=='3':
            if(float(c.informacion_clientes['saldo'][cliente]) > 0):
                op3_transferir(cliente)
            else:
                print('No se pueden hacer trasferencias sin dinero, imagínate que tienes cero galletas y la repartes entre cero amigos.')
                print('¿Cuántas galletas le tocan a cada amigo? No tiene sentido, ¿lo ves? Intenta conseguir alguna galleta antes de que')
                print('el mounstruo de las galletas se coma a los pocos amigos que te quedan por no tener galletas para darle.')

        elif opcion=='4':
            op4_info(cliente)
        elif opcion == '5':
            op5_cambiar_contraseña(cliente)
        elif opcion == '6':
            c.salgo = True
        else:
            print('Opción incorrecta.')
```

Toda la información modificada la vuelve a guardar en los archivos. Explicar cada una de las subfunciones sería muy tedioso (aún que en el código fuente ya están explicadas) explicaremos la más interesante (op3_transferir()). Consiste en preguntar al cliente si quiere hacer la transferencia mediante bizum o mediante la cuenta de banco y pregunta por el numero te teléfono o por el IBAN dependiendo del caso. Una vez haya localizado al cobrador de la transferencia (que siempre tiene que ser cliente nuestro) preguntará la cantidad que se le quiera transferir y si supera o iguala el 50 por ciento del saldo del cliente se le pedirá la contraseña de nuevo. Cuando tanto el cobrador como el tipo de transferencia y la cantidad de esta estén definidos se realizaran las operaciones necesarias para transferir el dinero. Y te volverá a mostrar un pequeño cuadro con información relevante.

```python
#FUNCION QUE SIRVE PARA COMPARAR Y BUSCAR POR NUMERO O POR IBAN PARA HACER UN TRASPASO DE DINERO A OTRA CUENTA

def bizumTransferencia(cliente):

    encontrado = False
    tipo=''

    while tipo != 'b' and tipo != 't':
        tipo=input('Que desea realizar, bizum o transferencia? (b/t): ')

    if tipo == 'b':
        while encontrado == False:
            tel=input('Introduce el numero de telefono del cobrador del bizum: ')
            for i in range (len(c.informacion_clientes['telefono'])):

                if tel == c.informacion_clientes['telefono'][i] and c.informacion_clientes['telefono'][cliente] != c.informacion_clientes['telefono
                    cobrador = i
                    encontrado = True


    elif tipo == 't':

        while encontrado == False:
            IBAN=input('Introduce el numero IBAN (de ISM) de la cuenta del cobrador de la transferencia: ')
            for i in range (len(c.informacion_clientes['IBAN'])):

                if IBAN == c.informacion_clientes['IBAN'][i] and c.informacion_clientes['IBAN'][cliente] != c.informacion_clientes['IBAN'][i]:
                    cobrador = i
                    encontrado = True


    return cobrador
```

```python
def op3_transferir (cliente):

    cobrador = bizumTransferencia(cliente)
    s = input('Desea transferir dinero a ' + c.informacion_clientes['nombreApellidos'][cobrador].upper() + ' ? (s/n): ')
    if s.upper() == 'S':

        cantidad = float(input('Introduce el dinero que se va a transferir: '))

        g.linea('_')

        while (cantidad >= float(c.informacion_clientes['saldo'][cliente]) or cantidad < 0):
            if cantidad < 0:
                print('no puedes introducir una cantidad negativa')
            else:
                print('Fondos insuficientes')

            cantidad = float(input('Vuelva a introducir el importe, siempre menor a su saldo de ' + str(c.informacion_clientes['saldo'][cliente])+


        if cantidad >= float(c.informacion_clientes['saldo'][cliente])/2:
            print('Disculpa pero al intentar transferir una cantidad que iguala o supera el 50% de dinero')
            print('de tu cuenta necesitamos una confirmación.')
            g.linea(c='¯')

            if falloContraseñas(cliente):
                return
            else:
                traspaso(cliente,cobrador,cantidad)

        else:
            traspaso(cliente,cobrador,cantidad)
    return
```

```python
def traspaso (cliente,cobrador,cantidad):

    c.informacion_clientes['saldo'][cliente] = str(float(c.informacion_clientes['saldo'][cliente]) - cantidad)
    c.informacion_clientes['balance del mes'][cliente]= str(float(c.informacion_clientes['balance del mes'][cliente]) - cantidad)


    c.informacion_clientes['saldo'][cobrador]= str(float(c.informacion_clientes['saldo'][cobrador]) + cantidad)
    c.informacion_clientes['balance del mes'][cobrador]= str(float(c.informacion_clientes['balance del mes'][cobrador]) + cantidad)


    infoCuenta(cliente,operacion=True)
```

Falta por explicar las funciones genéricas y la forma en la que el programa guarda los datos y genera el diccionario (información_clientes) a partir de las funciones escribir_en_archivos() y leerArchivos().

Son funciones inversas es decir escribir_en_archivos() lee cada clave del diccionario que ha creado leerArchivos(), la variable diccionario (información_usuarios) y a partir de esa lectura sobreescribe línea por línea los valores del diccionario según la posición que ocupen en la lista. Y la funcion leerArchivos() se recorre cada una de las líneas de cada uno de los archivos y las va ingresando según las claves que correspondan en los valores en forma de lista del diccionario.

```python
#FUNCION QUE RECORRE CADA LINEA DE CADA ARCHIVO Y GUARDA LAS STRINGS CONFORME A LAS CLAVES QUE HAYA GUARDADAS EN EL DICCIONARIO
#CON LOS VALORES EN FORMA DE LISTA
def leerArchivos():

    for archivo, clave in zip(archivos, claves):
        with open(archivo, 'r') as archivo_actual:
            lineas = archivo_actual.readlines()
            informacion_clientes[clave] = [linea.strip() for linea in lineas]

#FUNCION QUE ESCRIBE CADA VALOR DEL DICCIONARIO DE LISTAS DE CLIENTES Y LAS SOBRE ESCRIBE EN LOS FICHEROS LINEA POR LINEA
def escribir_en_archivos():
    for clave, archivo in zip(informacion_clientes.keys(), archivos):
        with open(archivo, 'w') as archivo_actual:
            for elemento in informacion_clientes[clave]:
                archivo_actual.write(f"{elemento}\n")
```

En el fichero de funciones genéricas tenemos las funciones que vamos a ir usando en todo el programa:

Bienvenida 1 y 2: Dan la bienvenida al usuario

Linea(): genera la cantidad de líneas que se desee y del carácter y la longitud que se desee

ocultarString(): oculta una str con el carácter que se desee en la posición que se desee con los caracteres ocultos que se deseen

comprobarStrCarcEsp(): comprueba que una str cumpla con una serie de requisitos, ((longitud mínima y máxima) y el tipo de str(si quieres que sean letras y números o solo letras, etc)) te devuelve un bool de si es valida o no la str

aleatorio(): te genera la cantidad de números aleatorios que desees y te la guarda en una str

stringEsp(): te separa una str en los saltos que desees

Para acabar hemos unido las dos funciones de iniciar sesión y crear cuenta en otra funcion que seria la pantalla de inicio en la que se le pregunta al usuario si quiere iniciar sesión o no y cada vez que se sale a este primer menú llama a leerArchivos() para que los datos de un usuario nuevo que acaba de crear una cuenta se lean junto a las demás cuentas.

```python
#FUNCION PRINCIPAL DEL PROGRAMA, ES LA PANTALLA PRINCIPAL


def iniciarSesion_crearCuenta():

    salir = False

    while not salir:
        c.salgo = False
        c.leerArchivos()

        g.linea(y=2)


        print('╔═══════════════════════════════════════════════════════════════════════════')
        s = input("╠ ¿Desea Iniciar sesión, Registrarse o Salir? (i/r/s): ")
        print('╚═══════════════════════════════════════════════════════════════════════════')

        if s=='i':

            IS.iniciarSesion()

        elif s=='r':

            CC.crearCuenta()
            c.leerArchivos()

        elif s=='s':

            g.linea()
            salir = True
```

Diseño

Hemos divido el proyecto de la siguiente forma:

- *Crear Cuenta:*
  - *Comprobar usuario y contraseña*
  - *Pedir datos personales*
  - *Guardar los datos*

- *Iniciar Sesion:*
  - *Leer datos*
  - *Comprobar credenciales*
  - *Menú y guardar datos*
    - *Ingresar*
    - *Sacar*
    - *Transferir*
    - *Información de la cuenta*
    - *Cambiar contraseña*
    - *Salir*
- *Salir*

- *Hemos usado variables de numero flotante, diccionarios y un diccionario con listas dentro de las claves*

Distribución del trabajo

Integrante	Labor realizada
Adrián López	Integracion de los datos de constantes y variables en el código y funciones graficas
David Muñoz	Diseño de funciones genéricas y de datos
Hugo López	Desarrollo de las operaciones y funciones
Jordi Blasco	Desarrollo de la parte estructural del programa, unificación del código y funciones

Uso que se ha hecho de inteligencias artificiales generativas (chatGPT, Bard, Github Copilot, etc.)

Hemos usado chatGPT para las funciones de escribir_en_archivos y leerArchivos. Ya que queríamos hacer unas funciones muy concretas sobre los archivos y no sabíamos como leer línea por línea ni meter las líneas en los huecos de las listas de la forma en la que lo queríamos, así que estuvimos un rato preguntándole y encontramos trozos de la solución que buscábamos:

conversacion

Problemas y dificultades encontradas y como se ha abordado su solución

Conclusiones

- *Se han cumplido todos los objetivos que nos marcamos, si que es verdad que queríamos hacer un programa aun mas complejo con cuentas de ahorro y mas tarjetas pero decidimos no complicarnos tanto la vida porque si no, no llegábamos a la fecha límite.*

- *El proyecto ha merecido la pena en todos sus aspectos, hemos aprendido a como usar ficheros y funciones de la manera más óptima, así como solucionar errores más rápido y el como trabajar mediante un repositorio de GitHub donde periódicamente íbamos haciendo commits y Pull Recuests para ir actualizando el código todos a la vez.*

  Hemos usado tanto GitHub Desktop para hacer los commits a este repositorio:
  https://github.com/JBLOZ/InteligentStockMarket
  como Visual Studio para programar y tener mejor visión de todos los archivos.

Referencias bibliográficas

- **(Alberto Real Fernández):** LECTURA/ESCRITURA DE FICHEROS.pdf
- **(ChatGPT)**: *conversacion*

No hemos usado más que el pdf de lectura y escritura de ficheros y la conversación previamente descrita con chat gpt