INFORME DE FUNCIONALIDAD Y DISEÑO DEL TRABAJO PRACTICO FINAL

Certificación Profesional Python

Integrantes:
Matias Carpinacci
Martin Vázquez

1. **Planificación de tareas e investigación del trabajo:**
   Al finalizar la última clase del curso, emprendimos la primera etapa del trabajo, el armado del equipo y distribución de tareas. Estuvimos viendo los primeros días que la mayoría de los estudiantes del curso tenían conocimientos previos de programación y en algunos casos manejaban varios lenguajes. En nuestro caso, los 2 integrantes de este grupo no manejamos ningún tipo de lenguaje, no tenemos experiencia previa en programación, y tampoco hemos estudiado carreras afines. Fue por ello por lo que nos pareció lo mejor realizar reuniones vía Google meet y trabajar juntos en cada parte del trabajo. Poder realizar este trabajo nos ha llevado muchas horas de lectura e investigación para cada uno de los renglones del programa, pero el objetivo era el aprendizaje.

2. **Desarrollo del programa:**

```
from datetime import datetime
import requests
import json
import pandas as pd
from pandas import DataFrame
import sqlite3
from sqlalchemy import create_engine
import matplotlib.pyplot as plt
import seaborn as sns

#Inicializamos variable "menu" para diferenciar primera corrida del
 programa
menu = 0

#Iniciamos loop infinito del programa
while True:
  print("Menu Principal\n\n"
          "1. Actualización de datos\n"
          "2. Visualización de datos\n"
          "3. Cancelar\n\n")

  menu = int(input("Ingrese un valor: "))

  if menu == 1:
      StocksTicker=str(input("Ingrese el stock ticker a actualizar:
 "))
      while True:
        try:
```

```
            d1 = input("\n Ingrese fecha de inicio en formato (YYYY-
MM-DD) :")
            #Confirmamos que la fecha ingresada tenga el formato requ
erido
            d1 = datetime.strptime(d1,'%Y-%m-%d')
            break
          except ValueError:
            print("\n Debe ingresar una fecha de inicio válida con el
 formato YYYY-MM-DD")
        while True:
          try:
            d2 = input("\n Ingrese fecha de cierre en formato (YYYY-
MM-DD) :  ")
            #Confirmamos que la fecha ingresada tenga el formato requ
erido
            d2 = datetime.strptime(d2,'%Y-%m-%d')
            break
          except ValueError:
            print("\n Debe ingresar una fecha de cierre válida con el
 formato YYYY-MM-DD")

        #Pasamos fechas de formato datetime a STRING para poder extra
er información de la API.
        fecha_inicio = d1.strftime('%Y-%m-%d')
        fecha_cierre = d2.strftime('%Y-%m-%d')

        # Formato de la respuesta del cliente
        RCliente = str(StocksTicker) + "/range/1/day/" + fecha_inicio
 + "/" + fecha_cierre
        URL = "https://api.polygon.io/v2/aggs/ticker/" + RCliente + "
?adjusted=true&sort=asc&limit=120&apiKey=io8O10fpLNbNm3NnJEGbH629Qn
KtP84b"
        api = requests.get(URL)
        apipars = api.json()

        # Crear tabla con columnas "Valores de Acción" y Filas "días"
        tabla = pd.DataFrame(apipars["results"])
        #Agregamos columna de fechas con formato YYYY-MM-DD
        tabla['date'] = pd.to_datetime(tabla['t'], unit='ms').dt.strf
time('%#Y/%#m/%d')
        #Eliminar columnas que no requerimos para el gráfico
        tabla = tabla.drop(["t", "v", "vw", "o", "h", "l", "n"], axis
=1)
        #Agregamos columna que indique el stocksticker
        tabla = tabla.assign(Acción = StocksTicker)
        tabla = tabla.rename({'c':'Valor Cierre'}, axis=1)

        #PASAMOS DATAFRAME A SQL
        cadena_conexion = 'sqlite:///DBAcciones.db'
```

```
      conexion = create_engine(cadena_conexion)
      #Creo/conecto a base de datos y genero una conexión entre pyt
hon y SQLite
      con = sqlite3.connect("DBAcciones.db")
      #Creo tabla "ACCIONES" en SQL y le cargo la información del D
ataFrame (el "append" agrega datos a la BD si ya esta creada)
      tabla.to_sql(name='ACCIONES', con=conexion, if_exists="append
")
      #Leo la tabla de la base de datos y elimino columna "index" q
ue se creo automáticamente
      sql_tabla = pd.read_sql("SELECT * FROM ACCIONES", con=conexio
n).drop(columns=["index"])
      #Eliminamos las filas duplicadas
      sql_tabla = sql_tabla.drop_duplicates()
      sql_tabla = sql_tabla.sort_values(by=["Acción", "date"])
      #Cargo la tabla modificada a la BD
      sql_tabla.to_sql(name='ACCIONES', con=conexion, if_exists="re
place")
      print(sql_tabla)
      #Cerramos la conexión con la base de datos
      con.close()
      print("\n\nLa base de datos se ha actualizado correctamente\n
\n")
  elif menu == 2:
      print("1. Resumen\n"
          "2. Gráfico de Ticker\n\n")

      menu = int(input("Ingrese un valor: "))

      if menu == 1:
        # Conectamos a la base de datos
        cadena_conexion = 'sqlite:///DBAcciones.db'
        conexion = create_engine(cadena_conexion)
        con = sqlite3.connect("DBAcciones.db")
        # Creamos un Data Frame a partir de la base de datos
        resumen_df = pd.read_sql("SELECT * FROM ACCIONES", con=cone
xion).drop(columns=["index","Valor Cierre"],axis=1)


        # Tomamos fecha minima y maxima, y armamos un data frame RE
SUMEN para imprimir
        min_df = resumen_df.groupby("Acción").min()
        max_df = resumen_df.groupby("Acción").max()
        resumen_df2 = pd.merge(min_df,max_df,how="right",on="Acción
")
        print(resumen_df2)

        # Cerramos conexión
        con.close()
```

```
      if menu == 2:
         # Conectamos a la base de datos
         cadena_conexion = 'sqlite:///DBAcciones.db'
         conexion = create_engine(cadena_conexion)
         con = sqlite3.connect("DBAcciones.db")
         # Creamos un Data Frame a partir de la base de datos
         datos_df = pd.read_sql("SELECT * FROM ACCIONES", con=conexi
on).drop(columns=["index"],axis=1)

         # Solicitamos el ticker a graficar
         Ticker=str(input("Ingrese el stock ticker a graficar: "))

         # Creamos un Data Frame a partir de la base de datos
         datos_df2 = datos_df.query("Acción==@Ticker")
         print(datos_df2)

         #Creo el Grafico
         datos_df2.plot(x="date",y="Valor Cierre", xlabel='Fecha', y
label='Valor Acciones', title=Ticker)

         # Cerramos conexión
         con.close()
         break
   else:
      break
```