Informe de funcionalidad y diseño - Entrega Final Python

1. En primer lugar se importan librerías, con las que se trabajará:

* Para conectar con la API (intermediario entre la aplicacion de python el servidor donde se encuentra la información que buscamos)
* Para conectar con la base de datos local, SQLITE 3
* Para poder hacer los gráficos

from polygon import RESTClient

import sqlite3

import matplotlib.pyplot as plt

2. El programa se conecta con la base de datos “archivo tutorial.db” y conectarse con la API que nos va a permitir obtener la información del servidor.

con = sqlite3.connect('C:/Users/feder/Desktop/guido\_juan\_tiscornia/base\_datos/tutorial.db')

cliente = RESTClient("P9tQYJ0HwyLYlx8a8Zx7m0xl\_9ScZMIB")

3. Luego, se utiliza un ***while*** (casi todo el código está dentro), que va a funcionar mientras el usuario no ingrese como opción la palabra “salir”.

*continuar = "seguir"*

*while(continuar != "salir"):*

4. Entonces se imprime un menú, con “1. Actualización de datos” y “2. Visualización de datos” como opciones.

while(continuar != "salir"):

print("1. Actualización de datos")

print("2. Visualización de datos")

opcion = input("Indique una opción del menú (salir para terminar): ")

5. En el caso de ingresar “1”, para la actualización de datos, se solicitan los datos de ticket, fecha de inicio y fecha de fin, y se utiliza la api para con esos datos obtener la información del servidor.

if (opcion == "1"):

ticker = input("Ingrese ticker a pedir: ")

fecha\_inicio = input("Fecha de inicio (formato aaaa-mm-dd): ")

fecha\_fin = input("Fecha de fin (formato aaaa-mm-dd): ")

datos\_formato\_lista = cliente.get\_aggs(ticker, 1, "day", fecha\_inicio, fecha\_fin)

6. Luego se hace un bucle ***for*** mediante el cual se recorre la información obtenida del servidor que viene un formato de lista de python y mediante una sentencia de sql se graban esos datos recibidos del servidor en la base de datos local.

for i in range(len(datos\_formato\_lista)):

variable = str(datos\_formato\_lista[i])[4:-1].split(",")

lista1 = []

for valor in variable:

lista1.append(valor.split("=")[1])

print(lista1)

open = float(lista1[0])

high = float(lista1[1])

low = float(lista1[2])

close = float(lista1[3])

volume = float(lista1[4])

vwap = float(lista1[5])

timestamp = int(lista1[6])

cur = con.cursor()

consulta = """INSERT INTO finanzas (ticker, open, high, low, close, volume, vwap, timestamp) VALUES(?, ?, ?, ?, ?, ?, ?, ?);"""

datos = (ticker, open, high, low, close, volume, vwap, timestamp)

cur.execute(consulta, datos)

con.commit()

7. En caso de que se seccione 2, para la visualización de datos, ese menú se subdivide en otras dos opciones: “1. Resumen de datos” y “2. Gráfico de in ticker”.

elif (opcion == "2"):

print("1. Resumen de datos")

print("2. Gráfico de un ticker")

opcion\_2 = input("Indique una opción del menú: ")

8. El primero lo que hace es utilizar nuevamente la conexión con la base de datos local para devolver toda la información que está alojada ahí (muestra toda la tabla en la consola).

if (opcion\_2 == "1"):

cur = con.cursor()

cur.execute("SELECT \* from finanzas")

datos = cur.fetchall()

for fila in datos:

print(fila)

9. El segundo lo que hace permitirnos ver un gráfico a partir de un ticker. El usuario ingresa un ticker y hace otra consulta a la base datos local mediante SQL, filtrando por ese ticker que ingresó el usuario. Se obtienen todos los datos al cierre de las acciones y con sus respectivos tiempos por orden temporal y en base a eso se grafica para ver como evolucionó a lo largo de tiempo.

elif (opcion\_2 == "2"):

ticker\_a\_visualizar = input("Ingrese el ticker que desea graficar: ")

consulta = """SELECT \* from finanzas where ticker=?"""

cur = con.cursor()

cur.execute(consulta, (ticker\_a\_visualizar,))

datos = cur.fetchall()

ticket\_al\_cierre = []

ticket\_timestamp = []

for fila in datos:

ticket\_al\_cierre.append(fila[4])

ticket\_timestamp.append(fila[-1])

plt.title("Gráfico de la acción " + ticker\_a\_visualizar)

plt.plot(ticket\_timestamp, ticket\_al\_cierre, "ro--")

plt.ylabel(ticker\_a\_visualizar)

plt.show()

10. Por último la opción salir nos permite salir de la aplicación, cortando el bucle while y así salimos de la aplicación.

elif (opcion == "salir"):

continuar = "salir"

ACLARACIÓN: Para una mayor comodidad al trabajar con la base de datos, se utilizó el programa “*DB Browser for SQLite*” (https://sqlitebrowser.org/)