Certificación Profesional en Python (ITBA)
Trabajo Práctico Final

Matías Huenul

Índice

1. Introducción

En este trabajo práctico se desarrolló un programa de línea de comandos en Python para obtener información de una API de finanzas, almacenarlas en una base de datos local SQLite3 y graficarlos usando Matplotlib.

2. Diseño de la aplicación

La aplicación se estructuró en módulos reutilizables, y un script principal que se encarga del flujo del programa y la interacción con el usuario.

2.1. Base de datos

El módulo **database** contiene las funciones necesarias para realizar consultas sobre la base de datos. En este caso, se utilizó el motor **SQLite3**, y se definió una única base de datos almacenada en el archivo **tickers.db**. La misma posee una tabla, **tickers**, la cual almacena los valores por ticker y fecha, y tiene la siguiente estructura.

- **symbol (string):** Símbolo que identifica a un ticker.
- **name (string):** Nombre del ticker.
- **value (real):** Valor del ticker.
- **date (string):** Fecha correspondiente al dato.

2.2. Visualizaciones

El módulo **plot** contiene una función para realizar gráficos de tipo **line plot** utilizando la biblioteca **Matplotlib**.

2.3. Interacción con la API de Polygon

El módulo **polygon** contiene las funciones para comunicarse con la API de Polygon [1]. Se define la función **get_tickers**, la cual realiza un request de tipo **GET** al endpoint **/v3/reference/tickers/{ticker}** y devuelve un diccionario con los datos de un ticker para una fecha dada. En particular, se devuelven los campos que serán almacenados en la base de datos, siendo calculado el valor del ticker como $value = \frac{market\_cap}{weighted\_shares\_outstanding}$, de acuerdo a la documentación oficial de la API.

2.4. Utilidades

El módulo **utils** posee funciones útiles para el programa, como funciones de cálculo de fechas, escritura a archivos e interacción con el usuario.

2.5. Flujo del programa

La lógica principal del programa se encuentra en el script **main.py**. El mismo ejecuta un **wizard** que interactua con el usuario y lo guía a través de las distintas operaciones disponibles, las cuales se pueden categorizar en tres grupos: **actualización**, **visualización** y **exportación** de datos. Cabe destacar que, en cada paso de la interacción con el usuario, se valida que la opción ingresada sea válida, y en caso de que no lo sea, se vuelve a solicitar. Al finalizar la operación, se da la opción de realizar otra, y en caso de que la respuesta sea negativa, se finaliza la ejecución del programa.

2.5.1. Actualización de datos

Para esta operación, se debe ingresar el símbolo del ticker a consultar, y las fechas de inicio y fin de la consulta, en formato **yyyy-mm-dd**. Luego de recibir estos parámetros, el programa realiza una solicitud a la API y guarda los resultados en la base de datos, imprimiendo por salida estándar si la operación fue exitosa.

Es importante notar que, debido a que la API sólo permite solicitudes por fecha y no por intervalos, es necesario realizar una solicitud por cada día en el intervalo desde la fecha de inicio hasta la fecha de fin. Esto quiere decir, que si por ejemplo se solicitan datos entre **2022-01-01** y

2022-01-03, se realizarán **3** requests a la API. Esto puede traer complicaciones debido al **rate limit** que establece Polygon. En caso de exceder este límite, la API devuelve respuestas con código **429 Too Many Requests**, y en estos casos, se esperan 10 segundos y se reintenta hasta tres veces. Se observó experimentalmente que con estos tiempos de espera y número de reintentos, se logra evitar el problema la mayoría de las veces, pero no siempre. Por ello se implementó un manejo de errores inteligente de forma tal que aún si ciertos datos no pudieron ser recuperados, los que sí se hayan podido recuperar se guardan de todas formas y se imprime un mensaje al usuario informando de esto.

Adicionalmente, si la información de un ticker para una fecha dada ya existe en la base de datos, el programa no volverá a solicitarlo en próximas operaciones. De esta manera, si en una ejecución se pidió un cierto intervalo, del cual sólo unos cuantos no pudieron ser recuperados, en una próxima ejecución sólo estos datos faltantes serán solicitados a la API.

2.5.2. Visualización de datos

Dentro de esta categoría se tiene dos opciones.

- **Resumen:** Esta operación imprime por línea de comandos resultados agregados por ticker (primera y última fecha de datos, cantidad de registros y valor promedio).
- **Gráfico de ticker:** Esta operación recibe un símbolo de ticker y muestra un gráfico del mismo según lo almacenado en la base de datos.

2.5.3. Exportación de datos

Esta operación permite generar un archivo de texto plano con los datos de la base de datos. Recibe un nombre de archivo a generar, un formato (csv o json), un símbolo de ticker y las fechas a exportar. En caso de no ingresar un ticker, se tomarán los datos de todos los tickers. En caso de no especificar fecha de inicio, se tomarán los datos desde la primera fecha disponible, y en caso de no especificar fecha de fin, se tomarán los datos hasta la última fecha disponible.

3. Conclusiones

En el desarrollo de este programa se utilizaron diversas tecnologías y se aprendió a modularizar un proyecto de forma tal que cada componente pueda interactuar para realizar una tarea en específico. Estos conocimientos adquiridos se pueden resumir en la siguiente lista.

- Creación e interacción con una base de datos **SQLite3**.
- Realización de requests HTTP a una API.
- Interacción con un usuario por línea de comandos.
- Manejo de errores en tiempo de ejecución.
- Creación de visualizaciones con **Matplotlib**.
- Escritura de información en archivos de texto en formatos **CSV** y **JSON**.

Referencias

[1] Stocks API - Polygon.io
Ticker Details v3
https://polygon.io/docs/stocks/get_v3_reference_tickers__ticker