Trabajo Final: Certificación en Python – ITBA 2022

Julián Eduardo Aguirre

aguirre.julian.eduardo@gmail.com

Repositorio:

https://github.com/JulianEduardoAguirre/ITBA\_ProyectoFinal.git

**Resumen**

Este trabajo consiste en la implementación de los módulos con las funciones necesarias para la consulta de datos sobre valores financieros de empresas que cotizan en bolsa, mediante peticiones a la API de <https://polygon.io/>. Con el paquete de datos obtenido por esta última, se realiza el procesamiento de los elementos del archivo JSON descargado y se los almacena en una base de datos local en la computadora del usuario. Por último, se implementa un menú con el que se dan las opciones de mostrar una tabla con el resumen de la información almacenada localmente, así como también se agrega la posibilidad de presentar un gráfico de velas japonesas para un ticker seleccionado.

Trabajo Final: Certificación en Python – ITBA 2022

# REQUISITOS

A continuación, se mencionan y describen los archivos y programas necesarios para ejecutar esta aplicación:

## Python 3

Este programa está implementado en el lenguaje Python 3, por lo que es necesario tener instalado esta versión del intérprete en el equipo.

## Archivos del programa

Para poder ejecutar el programa primero es necesario contar con los archivos alojados en el repositorio. El número de estos es 7 (siete), de los cuales el que debe ejecutarse es *main.py.*

La ejecución se debe realizar mediante terminal mediante el comando “python main.py”.

## Base de datos local

Para poder gestionar la base de datos en la que se almacenarán los datos solicitados a la API y sobre la cual trabaja este programa, es requisito necesario tener instalado SQLite en el equipo.

## Módulos de python

Una vez descargado o clonado el repositorio, es necesario instalar los siguientes módulos de python (pip install ‘nombre del módulo’)

* sqlite
* datetime
* requests
* pandas
* matplotlib

## API key

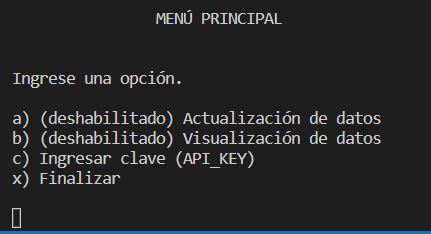
El programa solo se puede utilizar una vez obtenida una clave personal válida en la misma página de la API. Para ello, dirigirse a

Al ejecutar el programa, se podrá apreciar que los menús principales tienen agregada la leyenda “deshabilitado” desde el comienzo. Para ello, es necesario introducir la clave obtenida en la página de la API, con lo cual se habilita el uso de los menús de actualización de datos y de visualización.

# DESCRIPCIÓN DEL FUNCIONAMIENTO

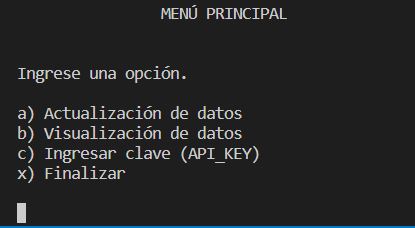
La primera vez que se ejecuta el programa, el mismo realiza las acciones necesarias para crear la base de datos, las tablas de la misma y resolver una conexión que queda habilitada durante la ejecución para hacer las gestiones necesarias de agregación de datos o consultas.

A continuación, se muestra el menú con opciones para que el usuario elija cómo proceder:



En un principio, las opciones a) Actualización de datos y b) Visualización de datos se encuentran deshabilitadas. Esto sucede porque el programa no se encuentra implementado con una clave API (API key) desde el comienzo, por lo que es necesario insertar una clave válida mediante el menu c) Ingresar clave (API\_KEY).

Una vez obtenida la habilitación para ingresar a los menus a) y b), el programa se encuentra funcional.



A continuación, se describe en detalle cada uno de estos:

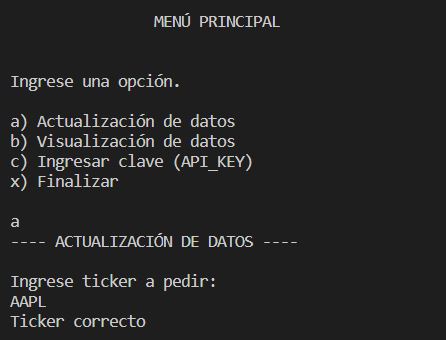
## a) Actualización de datos

Este es el menú que interactúa con la API, solicitando los datos pedidos por el usuario mediante terminal. Para ello, es necesario que el usuario indique el ticker y las fechas de inicio y de fin del periodo del que desea consultar la información. Se debe tener en cuenta que cada vez que se ingresa un parámetro en este menú, el mismo pasa por un proceso de validación (formato, existencia, coherencia) antes de ser transferido a la función correspondiente de petición a la API.

### Ingreso de ticker.

Se le solicita al usuario el nombre del ticker con el que la empresa cotiza en bolsa.

Se realiza validación para evitar que se ingrese una cadena de texto vacía como también se consulta a la API si el ticker ingresado existe dentro de sus registros.



En cualquier caso que no se apruebe la validación, se solicita nuevamente al usuario el ingreso del ticker.

### Ingreso de fechas.

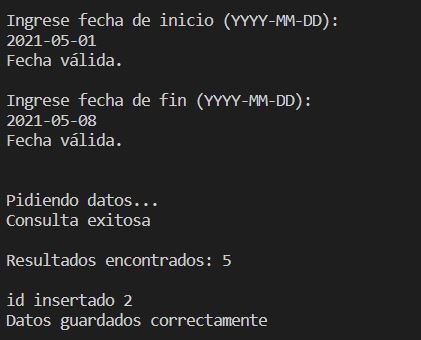
Las fechas deben escribirse en formato **AAAA-MM-DD** (ejemplo: 1989-10-04). La primera validación que se realiza es que el valor insertado posea el formato correspondiente.

Una vez pasada la primera validación, se ejecuta una función en la que se corrobora que la fecha ingresada efectivamente se corresponda con una real. Es decir, el valor 2022-12-40, correspondiente al 40 de diciembre de 2022 será rechazado, como también lo sería el 29 de febrero de un año no biciesto.

Por último, una vez ingresadas y validadas tanto fecha de inicio y fecha de fin del periodo a buscar, se comprueba que la segunda no sea más antigua que la primera. Se admite, sin embargo, que ambas fechas coincidan, ante lo cual se realiza la solicitud de datos para ese día particular.

Finalizado el ingreso de los valores de ticker y fechas, los mismos se transfieren a la función que realiza la consulta a la API. Con la respuesta de la API, se tienen dos opciones:

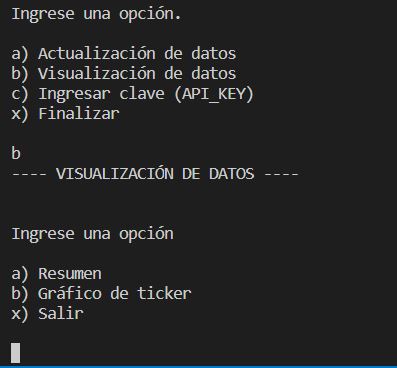
* Se encontraron registros: Se procesa el archivo recibido y se guarda la información en la base de datos.



* No se encontraron registros: Teniendo en cuenta que la validación de ticker impide pedir a la API un ticker inexistente, este resultado se da cuando dicha empresa no tiene información en los días solicitados.

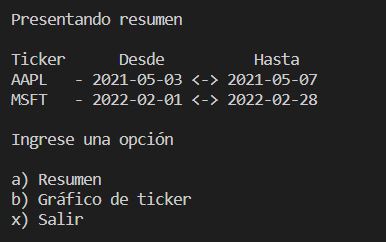
## b) Visualización de datos

Dentro de este menú, se cuenta con dos submenús para presentación de la información almacenada en la base de datos:



### Resumen.

Consulta sobre todos los datos almacenados en la base de datos y presenta una tabla donde se indican el nombre de los tickers almacenados y el periodo de tiempo comprendido entre el registro más antiguo y el más actual con el que se cuenta.



Se debe tener en cuenta que el periodo mencionado no implica, de manera necesaria, que se dispone de información de todos los días hábiles comprendidos en el mismo.

En el caso que aún no se disponga de información de ningún ticker en la base de datos, la elección de esta opción devuelve el mensaje “Sin registros en la base de datos”

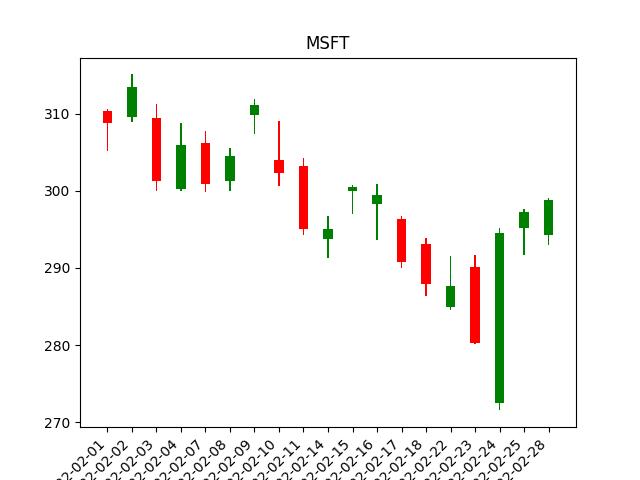
### Gráfico de ticker.

Bajo esta opción, se le ofrece al usuario la posibilidad de presentar un gráfico del tipo “candlesticks” o “velas japonesas”, mediante el cual se muestra información sobre el precio de apertura y cierre de una acción, junto con el máximo y mínimo de una determinada fecha.

Nuevamente, se solicita al usuario que ingrese el nombre del ticker que desea graficar y las fechas de inicio y cierre del periodo. Así también, se realiza validación de fechas, tanto para el formato de ingreso (AAAA-MM-DD) como la comparación entre ambas.

Teniendo en cuenta la posibilidad de contar, o no, con la información solicitada, se pueden dar dos resultados:

* En caso de encontrar registros en la base de datos para la información solicitada, se desplegará una nueva ventana con la figura correspondiente:



(Para continuar con la ejecución del programa, es necesario cerrar la ventana desplegada)

* Si no se cuenta con información sobre lo ingresado por el usuario (no existe el ticker o no existen registros en el período buscado), se presentará un mensaje correspondiente en la terminal.

## c) Ingresar clave (API KEY)

Este menú es importante puesto que es en dónde se debe ingresar la clave obtenida en el sitio web de la API.



**Es importante destacar que, en caso de contar con una clave válida ingresada correctamente, si se vuelve a ingresar a este menú, se reseteará la clave y habrá que introducirla nuevamente (o introducir otra).**

# DESCRIPCIÓN DE LA IMPLEMENTACIÓN

En esta sección, se listan y describen los módulos constituyentes del programa, junto a las funciones en cada uno:

## main.py

Archivo en el cual solamente se invoca a la función *menu\_principal()* del módulo funciones\_menu.py

## para\_fechas.py

En este archivo, se agrupan las funciones usadas para el procesamiento de la información ingresada por el usuario cuando se solicitan valores de fechas

#### validar\_formato\_fecha(fecha)

Permite corroborar que el formato utilizado para el ingreso de fechas por teclado (AAAA-MM-DD) es correcto. Para ello, intenta dividir la palabra ingresada, separándola sobre el carácter ‘-‘ y luego verifica que los tres fragmentos pertenezcan a un intervalo numérico (en el caso que se puedan convertir en números enternos).

Retorna True o False dependiendo de la validez del formato ingresado.

#### convertir\_a\_datetime (fecha)

Recibe una fecha en formato AAAA-MM-DD válido, también validado como fecha convertible a DATETIME y retorna la variable correspondiente de este tipo de datos.

#### comparar\_fechas (fecha1, fecha2)

Recibe dos variables DATETIME y retorna el resultado de la comparación

#### fecha\_2 >= fecha\_1

#### ingresar\_fecha (fecha)

Función que recibe una fecha ingresada por teclado e implementa las funciones validar\_formato\_fecha y validar\_fecha.

Retorna True en caso de superar ambas comprobaciones.

## conexion\_api.py

Archivo en el que se encuentran las funciones utilizadas para la interacción con la API

#### validar\_ticker (ticker, api\_key)

Recibe un ticker ingresado por el usuario junto a la clave de la API.

Consulta a la API por información sobre dicho ticker en la dirección correspondiente.

El resultado se almacena en una variable interna y se procesa. De este procesamiento se determina si el ticker ingresado es válido (función retorna True) o no. Además, previamente se impide al usuario el ingreso de una palabra vacía.

#### API\_solicitar\_datos\_ticker (ticker, fecha\_inicio, fecha\_cierre, api\_key)

Recibe el nombre del ticker y las fechas de inicio y de cierre del periodo a consultar, ingresados por el usuario. Además, recibe la clave API para hacer la consulta.

En base al archivo retornado por la API, se determina si se encontraron o no registros para ese periodo. En caso afirmativo, la función retorna el archivo (como un diccionario) meintras que en caso contrario, retorna None

#### prueba\_api\_key (api\_key)

Realiza una consulta, normalmente válida, a la API, cuando la clave es correcta. Por tanto, si la clave almacenada en el programa es válida, la función retorna True.

## armado\_diccionario.py

Se implementan las funciones para procesar el resultado devuelto por la API y preparar los diccionarios con los datos para las funciones de comunicación con la base de datos y de graficación (generación de dataframe de pandas)

#### crear\_diccionario\_desde\_json (datos)

Recibe un diccionario de python (previamente se importaba el diccionario desde un archivo JSON local) y se procesa. Se genera un nuevo diccionario resumido en el que se almacena el nombre del ticker y una lista de diccionarios con los datos financieros y la fecha correspondiente. Este diccionario es retornado al finalizar.

En caso que el archivo no posea los datos financieros usados, la función retorna None e imprime un mensaje en terminal.

#### datos\_para\_dataframe (datos\_ddbb)

Recibe un diccionario de python (obtenido mediante consulta a la base de datos) y se genera un nuevo diccionario de listas, el cual se retorna al finalizar.

## db\_tickers.py

En este archivo se implementan las funciones para la creación e interacción con la base de datos.

#### conexion\_db ()

Conecta el programa con la base de datos “stocks.db”. En caso que no exista, la crea.

Retorna un objeto con la conexión a la base de datos y un objeto cursor para interacción.

#### crear\_tablas (conexion, cursor)

Crea las dos tablas (en caso de no existir) usadas por este programa:

* ticker: Almacena el nombre de los tickers válidos retornados en el paquete de datos por la API. Les asigna un id.
* datos: Almacena la fecha y los parámetros financieros de un ticker (open, close, high, low). Se relaciona con el ticker correspondiente mediante la clave foránea “ticker\_id”

#### ticker\_consultar\_id (ticker, cursor)

Recibe el objeto conexión con la base de datos y consulta el id de un determinado ticker. Retorna el valor del id en caso que exista en la tabla “ticker”. En caso contrario, retorna None

#### datos\_consultar\_entrada (id\_ticker, fecha, cursor)

Consulta si existen datos para un determinado ticker en una determinada fecha.

Retorna True o False dependiendo del resultado.

#### ticker\_insertar (ticker, cursor)

Inserta un ticker en la tabla correspondiente y retorna el id asignado (utiliza la función *ticker\_consultar\_id*)

#### insertar\_datos (mi\_dicc, cursor, conexion)

Recibe un diccionario de Python, obtenido por la función crear\_datos\_desde\_json(), lo procesa e inserta la información en las tablas de la base de datos.

Primero consulta el id asignado al ticker ingresado (o lo inserta en caso que sea uno nuevo). Luego, prepara la sentencia para el ingreso de datos en entradas individuales por fecha. La inserción del registro se realiza si no existe previamente.

#### consultar\_datos (ticker, fecha\_inicio, fecha\_cierre, cursor)

Consulta en la base de datos por la información de un ticker especificado por el usuario, junto al período de búsqueda. Si se encuentran registros, los almacena en una variable y los retorna al final de la función. En caso contrario, retorna None.

#### informacion\_ticker (cursor)

Realiza una consulta de todas las entradas a la base de datos y presenta los resultados agrupados por ticker, indicando el periodo de tiempo entre el registro más antiguo y el más actual.

Si la base de datos aún no cuenta con información almacenada, se presenta un mensaje en la terminal.

## graficacion.py

En este módulo se implementa una función para la conversión en dataframe de los datos retornados tras consulta a la base de datos y otra para generar el gráfico de velas correspondiente.

#### generar\_dataframe (datos\_df)

Genera una variable DATAFRAME del módulo pandas con el diccionario recibido de la función *datos\_para\_dataframe().*

#### graficar\_candlesticks(ticker, datos\_dataframe)

Utiliza el submódulo pyplot de matplotlib para generar una figura en la que se presentan las distintas entradas (por fecha) para un ticker almacenado en la base de datos.

Los datos a graficar se pasan como parámetro en *datos\_dataframe* y corresponden a una consulta hecha a la base de datos, por lo que el periodo que se muestra depende de lo solicitado por el usuario y no todo lo que se encuentra almacenado.

## funciones\_menu.py

Se implementa al función del menú principal, el cual es una estructura de sentencias WHILE, IF y ELSE, con el que se controla la ejecución y el estado del programa. Se agregan, además, otras funciones para evitar repetición de código.

#### ingreso\_ticker (api\_key)

En esta función se realiza el pedido del nombre del ticker al usuario y se invoca a la función que conecta con la API para determinar si existe o no.

Dado que se encuentra implementada con un bucle WHILE, se pedirá al usuario el ingreso reiterado hasta tanto no se escriba un ticker correcto.

#### agregar\_deshabilitado (palabra)

Agrega la palabra “(deshabilitado)” delante de otra.

Se utiliza mientras no se ingrese una clave API válida en el menú correspondiente y en reemplazo de otra función que modifica una palabra agregando un efecto de tachado.

Retorna la nueva cadenás de caracteres.

#### ingreso\_fechas ()

Función donde se alojan las sentencias de input y de comprobación (mediante funciones del módulo para\_fechas) para automatizar el proceso de ingreso de las fechas solicitadas por el usuario.

Una vez aprobadas las validaciones correspondientes, se retornan las variables con las fechas como las ingresó el usuario.

#### menu\_principal()

Es la función que se invoca cuando se ejecuta el programa en la terminal.

Procede a llamar a las funciones que interconectan con la base de datos, tras lo cual se entra en los distintos flujos del programa dados por las sentencias WHILE, IF y ELSE, según lo que consulte el usuario.

Páginas de utilidad

FECHAS

Conversión de epoch a datetime

<https://www.delftstack.com/howto/python/python-convert-epoch-to-datetime/>

<https://pythonguides.com/python-epoch-to-datetime/>

Comparar fechas

<https://www.geeksforgeeks.org/comparing-dates-python/>

Validación de fechas

<https://stackoverflow.com/questions/25465144/how-to-validate-a-date-in-python>

<https://www.codevscolor.com/date-valid-check-python>

<https://www.tutorialspoint.com/How-to-do-date-validation-in-Python>

Insertando fechas en SQL

<https://www.tutorialspoint.com/sql/sql-date-functions.htm>

GRAFICACIÓN

Gráfico de velas

<https://www.geeksforgeeks.org/how-to-create-a-candlestick-chart-in-matplotlib/?ref=rp>

JSON

Manejo de archivos JSON

<https://www.geeksforgeeks.org/read-json-file-using-python/>

LISTAS

Manejo de listas

<https://www.geeksforgeeks.org/python-check-if-list-empty-not/>

DICCIONARIOS

Manejo de diccionarios

<https://www.geeksforgeeks.org/handling-missing-keys-python-dictionaries/>

<https://stackoverflow.com/questions/29336243/pythonic-way-to-handle-non-existant-python-dict-keys>

SQL

Evitar repetición de datos

<https://stackoverflow.com/questions/17442901/how-to-avoid-duplicate-values-for-insert-in-sql>

<https://codingsight.com/sql-insert-into-select-5-easy-ways-to-handle-duplicates/>

<https://ubiq.co/database-blog/how-to-avoid-inserting-duplicate-records-in-mysql/>

<https://www.geeksforgeeks.org/sql-insert-ignore-statement/>

Obtener un diccionario en consulta SQL

<https://stackoverflow.com/questions/3300464/how-can-i-get-dict-from-sqlite-query>

Tablas

Tabla 1

[Título de tabla]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Encabezado de columna | Encabezado de columna | Encabezado de columna | Encabezado de columna | Encabezado de columna |
| Encabezado de fila | 123 | 123 | 123 | 123 |
| Encabezado de fila | 456 | 456 | 456 | 456 |
| Encabezado de fila | 789 | 789 | 789 | 789 |
| Encabezado de fila | 123 | 123 | 123 | 123 |
| Encabezado de fila | 456 | 456 | 456 | 456 |
| Encabezado de fila | 789 | 789 | 789 | 789 |

Nota: [Coloque todas las tablas del artículo en una sección de tablas, después de las referencias (y, si corresponde, de las notas al pie). Use una página nueva para cada tabla e incluya un número de tabla y un título de tablapara cada una, como se muestra en esta página. Todo el texto explicativo aparece en una nota de tabla después de la tabla, como en esta. Use el estilo de tabla o ilustración, disponible en la galería de estilos de la pestaña Inicio, para agregar el espaciado entre la tabla y la nota Las tablas en el formato de APA pueden usar un interlineado de una línea o de 1,5 líneas. Incluya un título para cada fila o columna, incluso si el contenido parece obvio. Se configuró un estilo de tabla predeterminado para esta plantilla que cumple con las normas del estilo APA. Para insertar una tabla, en la pestaña Insertar, haga clic en Tabla.]

Título de ilustraciones

Ilustración 1. [Incluya todas las ilustraciones en su propia sección, después de las referencias (y, si corresponde, de las notas al pie y las tablas). Incluya un título numerado para cada ilustración. Use el estilo de tabla o ilustración para agregar fácilmente espaciado entre la ilustración y el título.]

Para obtener más información sobre todos los elementos del formato de estilo APA, vea el Manual de estilo de la APA, 6.ª edición.