**Manual Técnico**

**Drive Challenge MeLi**

**Agustin Fourcade**

**Julio 2020**

Índice

[**Requisitos Elementales** 2](#_Toc44890817)

[**¿Cómo ejecutar el software?** 3](#_Toc44890818)

[**Introducción** 4](#_Toc44890819)

[**Objetivo** 4](#_Toc44890820)

[**Objetivo Especifico** 4](#_Toc44890821)

[**Requerimientos mínimos de Hardware** 4](#_Toc44890822)

[**Requerimientos mínimos de Software** 5](#_Toc44890823)

[**Desarrollo** 5](#_Toc44890824)

# **Requisitos Elementales**

Para el correcto funcionamiento y desarrollo del software se recomienda tener la versión:

- Python 3.3.8

- Instalador de MySQL 8.0.20 ( <https://dev.mysql.com/downloads/file/?id=495322> )

- Visual Studio 2019

También se requiere tener instalado los siguientes paquetes de Python:

- pip install mysql-connector-python (8.0.20)

- pip install google-auth-oauthlib (0.4.1)

- pip install google-api-python-client (1.9.3)

En cuanto a las configuraciones, debemos crear una instancia de base de datos con los siguientes datos:

host = "127.0.0.1",

user = "root",

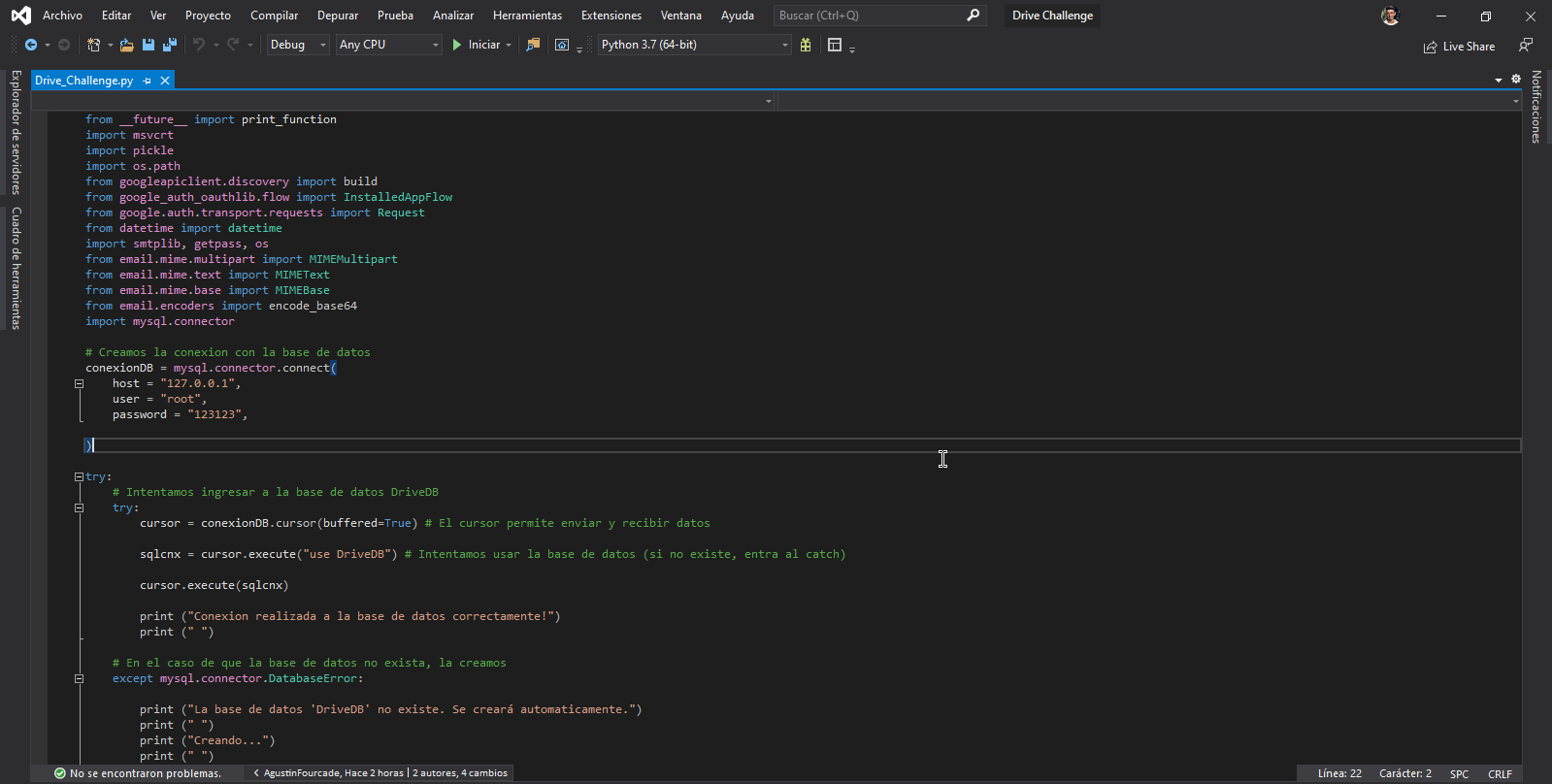
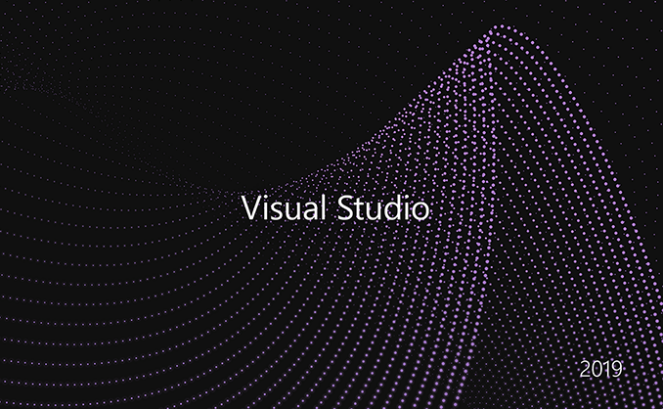
password = "123123"

# **¿Cómo ejecutar el software?**

Si estamos interesados en acceder al código fuente de la aplicación o si queremos correr el software desde Visual Studio 2019 debemos abrir el archivo *Drive Challenge.sln* que está en la raíz de la carpeta descargada.

Si por algún motivo no aparece el código fuente, se debe ingresar a la carpeta *Drive Challenge* y abrir con Visual Studio 2019 el archivo llamado *Drive\_Challenge.py.*

Hecho esto, ya tenemos completo acceso al código fuente.



# **Introducción**

Este sistema ha sido analizado, diseñado, desarrollado y sometido a pruebas de diferente tipo con el fin de que brinde la mejor calidad de rendimiento posible a cualquier persona que pueda tener acceso al mismo.

# **Objetivo**

Este documento se ha creado con el propósito de mostrar cómo fue diseñado el sistema, y al mismo tiempo dar referencia de cómo interactuar con el mismo. Dicho proyecto tiene el objetivo de poder inventariar una base de datos todos los archivos pertenecientes a la unidad de Drive de un usuario.

# **Objetivo Especifico**

* Almacenar metadatos de los diferentes archivos dentro del Drive de un usuario (Nombre del archivo, la extensión, el owner, la visibilidad y la fecha de modificación).
* Debe tener lógica suficiente para poder detectar archivos repetidos para no llenar tuplas de mas en la base de datos.
* Debe mantener un inventario histórico de los archivos que, en algún momento, fueron públicos.
* Los archivos que en un principio son públicos, deben ser modificados y convertirlos en archivos privados. En el proceso, debe enviar un correo electrónico al dueño del archivo para notificarle el cambio que se realizó sobre su archivo.

# **Requerimientos mínimos de Hardware**

Procesador: intel inside 1.5 ghz o superiores

Memoria RAM: 512 mb o superiores

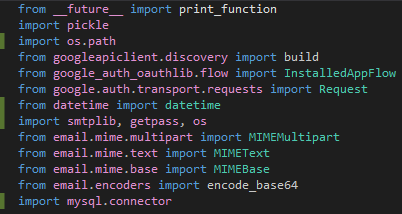
Disco duro: 64 gb o superiores

# **Requerimientos mínimos de Software**

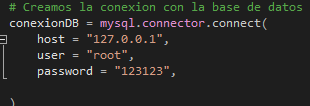
Privilegios de administrador: Si

Sistema operativo: Linux Ubuntu, Windows 7, OSX 10.13 (High Sierra)

# **Desarrollo**

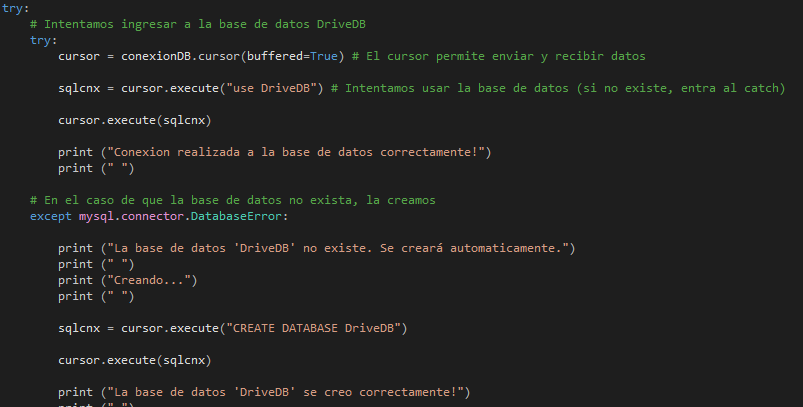


Creamos la conexión a una base de datos



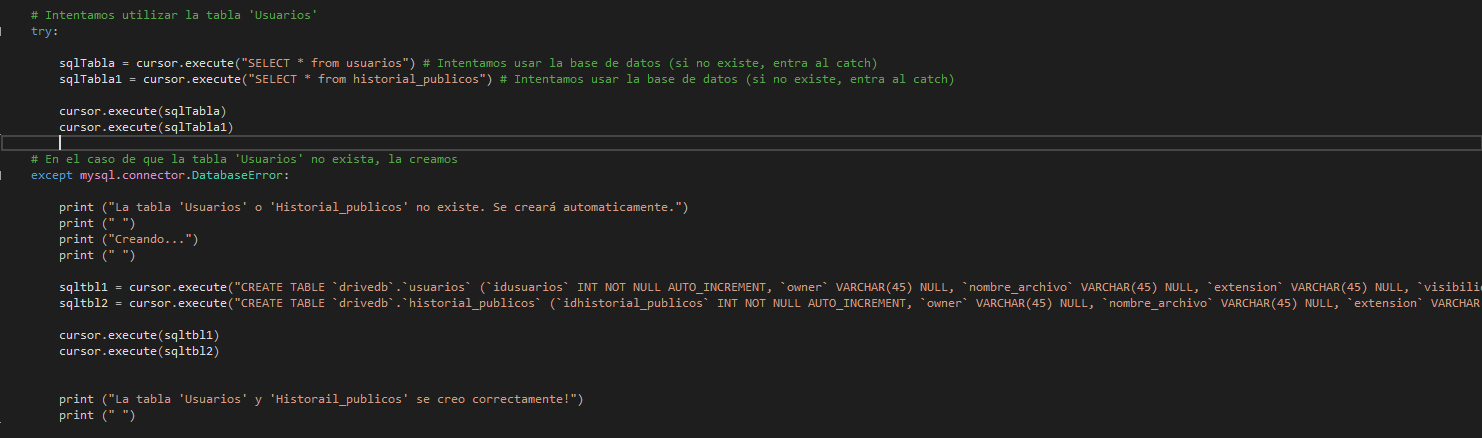
Creamos un cursor, el cual vamos a utilizar constantemente para la transferencia de datos entre el programa y la base de datos.

Seleccionamos la base de datos llamada “DriveDB”, si no existe, la creamos.



Intentamos leer datos de dos tablas específicas, *usuarios* y *historial\_publicos*

Si recibimos un error es que las tablas no están creadas, por lo tanto, procedemos a crearlas asignando todos los valores correspondientes a los requisitos del programa (nombre\_archivo, owner, etc).

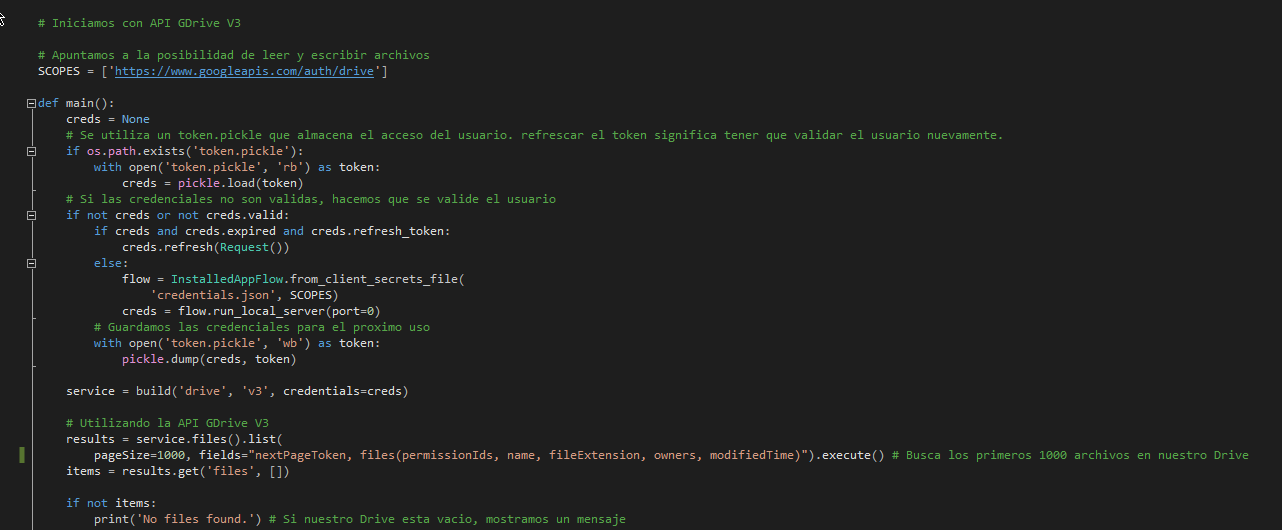


Definimos el scope que va a tener la API (en este caso, escribir y leer archivos)

Utilizamos un token, el cual almacena el acceso del usuario, la modificación de este significa tener que acreditar el acceso nuevamente.

También guardamos las credenciales del token para futuros accesos.

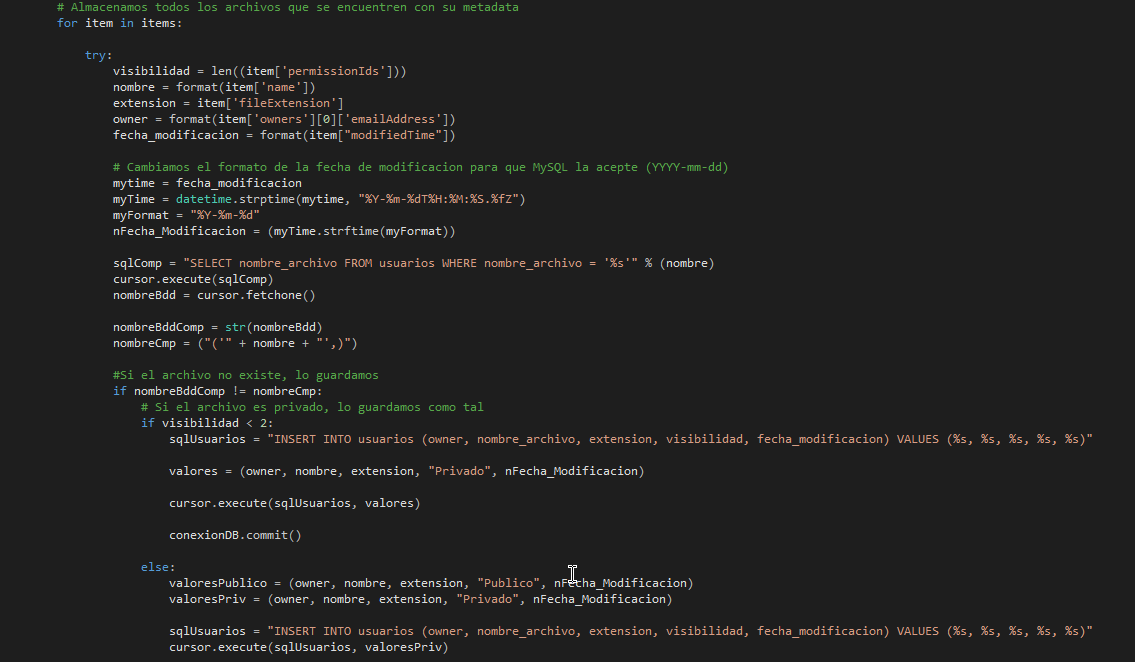
Al final, definimos los metadatos que vamos a solicitar de cada archivo que este alojado en Google Drive, si el Drive del usuario esté vacío, mostramos un mensaje anunciándolo.



Si hay archivos iteramos la cantidad de veces correspondiente a la cantidad de archivos, para que, en cada iteración, guardemos en variables los metadatos y los vayamos insertando en la base de datos.

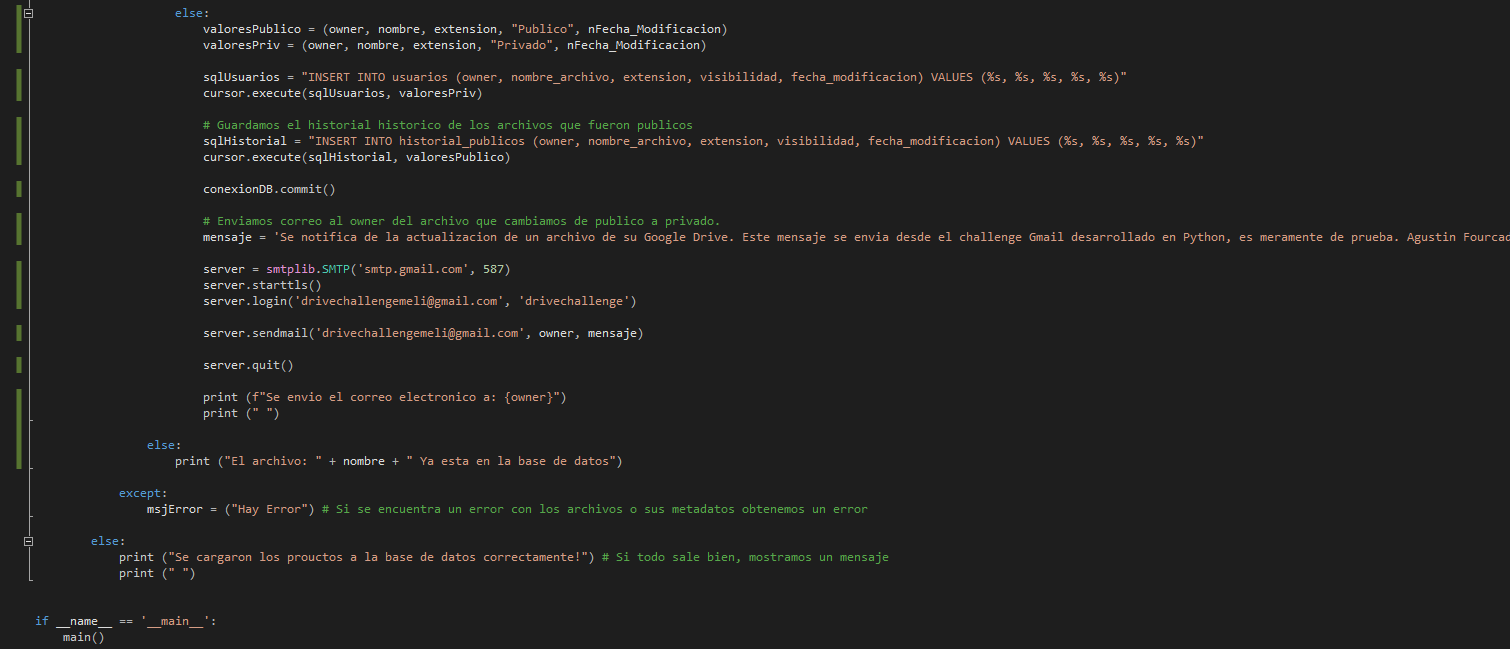
También vemos una función *datetime,* la utilizamos para cambiar el formato de la fecha de modificación del archivo para que MySQL la acepte sin problemas (YYYY-mm-dd).

Preguntamos si el archivo ya existe en la base de datos, en el caso de que no, lo agregamos. Caso contrario, mostramos un mensaje diciendo que el archivo ya se encuentra alojado en la BDD.



Si el archivo se encuentra en público, lo hacemos privado y enviamos un correo electrónico el dueño del archivo para que este al tanto de la modificación realizada.

(Como posibilidad de mejora, la contraseña del correo electrónico podría estar alojada en una variable de entorno (.env) y ser accedida mediante la librería *decoupe*)



Desarrollado en Julio 2020.

Challenge para Mercado Libre.

Agustin Fourcade.