Escenarios de prueba API-TweetMaster y Mejoras:

1-Asegurarse de que se tienen instalados todos los requerimientos en el entorno virtual (recomendable su utilización):
→ **pip3 install -r requirements.txt**

2-Ir DataBaseConnector/configtables.py y modificar la linea 9 dependiendo del usuario y pass de su BD en MySQL
→ **engine = create_engine("mysql+pymysql://root:4236@localhost:3306/BDTweetMaster?charset=utf8",echo=True)**

3-Ahora vamos a probar nuestra API mediante **POSTMAN**. Para esto, primero iniciamos los servidores: (start_fetcher, start_reporter y start_manager). Debemos que abrir 3 ventanas de comandos distintas en la ubicación de nuestro root de la API-TweetMaster [Sin el entorno virtual activado]. Y en cada ventana poner: **./start_fetcher | ./start_manager | ./start_reporter**

4-Vamos a POSTMAN e importamos la colección "**RequestsAPITWEETMASTER.postman_collection.json**" ubicado en el root de la API. Primero vamos a probar los **POST Campaign** (al postear/insertar una campaña automáticamente se creará la BD con las tablas Campaign y Tweet):

- Probamos el **1-POST Campaign NO esta en curso** (nos devolverá un 201 indicando que la campaña se creó).
- Luego el **2-POST Campaign EN curso** (nos devolverá un 201 indicando que la campaña se creó).
- Y por último el **3-POST Campaign sin StartDate (**nos devolverá un 412 ya que falta un parámetro).

5-Ahora **probamos que se traigan los tweets de las Campañas activas** ejecutando el scheduler que llamará cada 2 minutos al Manager para obtener los
Para esto en la consola nos ubicamos en PDI-TweetMaster/Scheduler y ejecutamos:
→ **python3 easySchedule.py**

[De esta manera buscará cada 2 minutos y agregará a la BD todos los tweets de las campañas activas. Si el tweet ya está en la BD simplemente no lo ingresa. Como la 1er campaña ya finalizó no traerá nada, en cambio si trae los tweets de la 2da campaña]

6-Ahora vamos a probar en **POSTMAN** los **DELETE, GET y PATCH Campaign y los GET Reporter (Raw y JSON):** Para esto primero damos por finalizado el easySchedule (ctrl+z) y ahora nos ubicamos en la ubicación de nuestro root de la API-TweetMaster ejecutamos el siguiente Script para insertar 10 Campañas y 10 Tweets para hacer las pruebas. Estas campañas contemplan todos los casos: hay campañas que no empezaron, campañas activas con tweets y campañas activas sin tweets.
→ **python3 ScriptInsertsBD.py**

7-Vamos a **POSTMAN** y probarmos los **GET Campaign**:
- Probamos el **13-GET Campaign 1** (nos devolverá un 200 indicando que la campaña se devolvió exitosamente; y en el body response obtendremos dicha campaña).
- Probamos el **14-GET Campaign 30 NO existente** (nos devolverá un 404 indicando que la campaña no existe).

- Probamos el **15-GET Campaigns ID no especificado** (nos devolverá un 404 indicando que no se especificó el ID).

8- Vamos a **POSTMAN** y probarmos los **GET Reporter Raw:**

- Probamos el **16.1-Reporter RAW Campaign 30 NO existente** (nos devuelve un 404 indicando que la Campaña no existe).
- Probamos el **16.2-Reporter RAW Campaign 9 En curso** (nos devuelve un 412 indicando que la campaña todavía está en curso).
- Probamos el **16.3-Reporter RAW Campaign 5 Finalizada y Con Tweets** (nos devuelve 200 OK y el reporte en crudo de la Campaign 5).

9- Vamos a **POSTMAN** y probarmos los **GET Reporter JSON:**

- **17.1-Reporter JSON Campaign 30 no existente** (nos devuelve un 404 indicando que la Campaña no existe).
- **17.2-Reporter JSON Campaign 9 Existente y en curso** (nos devuelve un 412 indicando que la campaña todavía está en curso).
- **17.3-Reporter JSON Campaign 5 Finalizada y Con Tweets** (nos devuelve 200 OK y el reporte en resumen JSON de la Campaign 5).

10-Vamos a POSTMAN y probarmos los **DELETE Campaign**:

- Probamos el **7.1-DELETE Campaign por idC** (nos devolverá un 200 OK ya que la campaña que queremos eliminar, la 1, NO está activa, entonces se eliminó).
- Probamos el **7.2-DELETE Campaign por idC** (nos devolverá un 412 ya que la campaña que queremos eliminar, la 3, está activa).
- Probamos el **8-DELETE Campaign por email** (nos devuelve un 200 OK de que se eliminó la campaign 7 con el email “finalizada@gmail.com")
- Probamos el **9**-D**ELETE Campaign en curso por email** (nos devolverá un 412 ya que la campaña que queremos eliminar, la 3, está activa).
- Probamos el **10-DELETE Campaign multiple por email** (nos devuelve un 200 OK de que se eliminaron las campaigns con id 5 e id10 y sus tweets asociados) .
- Probamos el **11-DELETE Campaign multiple activas** (nos devolverá un 412 ya que las campañas que queremos eliminar están activas)
- Probamos el **12-DELETE Campaign multiple mezclado** (“mezclado@gmail.com” tiene campañas inactivas y activas. Al mandar este request sólo se elimina la campaña inactiva - id 11-, la activa – id 12 – no se elimina).

11-Vamos a **POSTMAN** y probarmos los **PATCH Campaign**:

- Probamos el **4-PATCH Campaign 3 Activa** (nos devuelve un 412 indicando que la campaña está activa y no se puede modificar).
- Probamos el **5-PATCH Campaign 6 NO Activa Email** (nos devuelve un 200 indicando que el email especificado de la campaña se modificó correctamente).
- Probamos el **6-PATCH Campaign 30 NO existente** (nos devuelve un 404 indicando que la campaña con id 30 no existe).

-Opcional: Ejecutar el servidor Swagger que contiene documentación de la API, para esto se debe ejecutar los siguientes comandos desde API-TweetMaster/API_DOC:
→ **pip3 install -r requirements.txt**
→ **python3 -m swagger_server**
Luego poner como URL del browser lo siguiente:
http://localhost:8080/FedericoCalonge/TweetMaster/1.0.0/ui/

-Mejoras/dificultades:

- Integrarlo la API con **TRAVIS**.
- Eliminar redundancia: Los objetos Tweet.py y Campaign.py no deberían existir ya que con SQLAlchemy tendríamos que trabajar con los objetos que creamos en configtables.
- Que ande el **Scheduler.py** mediante **crontab**. Ya que nosotros estamos utilizando el script de Python "**easySchedule.py"**. Sería mejor utilizar crontab ya que es un proceso maestro que ejecutaría el Scheduler.py sin necesidad de tener una ventana abierta por consola ejecutando el script (como en nuestro caso con **easySchedule.py**.), ya que lo ejecutaría en 2do plano.
- **No podemos pegarle por POSTMAN al FETCHER**. [Ver en POSTMAN **18-GET Fetcher (sin andar).** Le mandamos una campaña en el request body y nos debería devolver los tweets de esa campaña. En POSTMAN NO hacemos un GET ya que no se puede agregar nada en el body del request; por esto hacemos un POST. → Nos tira un **500 Server Error** y en el servidor de fetcher nos dice que el error es**"Input string must be text, not bytes"**.