# Arranque del sistema.

Para iniciar los contenedores correctamente, en la raíz del proyecto hay que ejecutar el comando “docker-compose up -d” en terminal.

Una vez iniciados todos los contenedores, abrir el navegador y escribir en la barra de búsqueda “localhost:8080” y se abrirá una ventana con la aplicación web de Apache Airflow. Credenciales:

* Usuario: “airflow”
* Contraseña: “airflow”

Nos aparecerá listados los dags disponibles, en este caso el único dag que hay, y podemos iniciarlo manualmente.  
Podremos ver los logs dentro de la misma aplicación o en la carpeta del proyecto, “logs”.

# Especificaciones técnicas

Para esta prueba han sido utilizados las siguientes características software:

* Lenguaje Python para el desarrollo del código del dag y todas las tareas que este conlleva.
* Para la tarea de generación de datos, he usado la librería Faker de Python. Esta librería permite una generación rápida y sencilla de datos de prueba, siendo estos datos realistas y variados teniendo buena compatibilidad con múltiples formatos (CSV, JSON, SQL, etc.). Utiliza datos completamente ficticios.
* Para la limpieza de los datos he usado la librería Pandas de Python. Esta librería permite el manejo de los datos, así como la extracción como la carga de estos en ficheros de múltiples formatos de manera sencilla.
* Respecto a la subida de los datos a la base de datos, he hecho una conexión desde Apache Airflow a un contenedor de PostgreSQL. Utilizando SQLAlchemy recupero el motor de base de datos utilizado en la conexión para conectarme y subir los datos limpios a la base de datos.
* El flujo de datos del dag, es el siguiente:  
  Interfaz de usuario gráfica

  El contenido generado por IA puede ser incorrecto.

Generate\_data: tarea de generación de datos con operador de Python.

Clean\_data: tarea de limpieza de datos con operador de Python

Grupo de tareas Datos\_procesados:

* + Drop\_table\_g1: elimina, si existe, la tabla “usuarios\_validos”.
  + Create\_table\_g1: crea una nueva tabla de datos “usuarios\_validos”.
  + Load\_to\_postgres\_g1: carga los datos en la tabla “usuarios\_validos” de la base de datos postgres.

Grupo de tareas Datos\_sin\_procesar:

* + Drop\_table\_g2: elimina, si existe, la tabla “usuarios\_invalidos”
  + Create\_table\_g2: crea una nueva tabla de datos “usuarios\_invalidos”
  + Load\_to\_postgres\_g2: carga los datos en la tabla “usuarios\_invalidos” de la base de datos postgres.

La tabla creada de “usuarios\_invalidos” no tiene asignada PRIMARY KEY ya que los datos introducidos pertenecen al fichero “messy\_data.csv” y existen datos duplicados. Esta tabla es creada para comprobar la diferencia entre los datos correctos e incorrectos.

# Problemas de implementación

Para generar los datos, el primer camino que tome fue crear un contenedor de Ollama el cual permite ejecutar modelos de lenguaje (LLM) en tu máquina local. Para obtener los datos necesitas realizar una petición al servicio Ollama del contenedor con un prompt que te devuelva una respuesta válida. El inconveniente, es que al estar en un contenedor limitado por su tamaño de capacidad de almacenamiento el modelo de lenguaje no podía ser de mucho tamaño por lo que es algo más limitado. Al hacer la petición de los datos no enviaba el formato correcto y no generaba más de ciertos tokens, limitando así la cantidad de filas requeridas.

La alternativa que he escogido ha sido la librería Faker de Python que permite una generación rápida y sencilla de datos ficticios y aleatorios, pudiendo ser introducidos más tarde en un fichero CSV con la librería Pandas de Python.

# Configuraciones especiales

Para poder probar el uso de Ollama es necesario ejecutar el siguiente comando tras levantar el Docker Compose:

**docker exec -it ollama\_container ollama pull llama3.1**

Tras ejecutar el comando, tenemos hay que descomentar el código que hay en el fichero “dag.py” dentro de la función “ejecutar\_generate\_data”.

Cuando ejecutemos el dag de nuevo, se creara un nuevo fichero llamado “prueba.csv” en el directorio /data, que incluirá la respuesta del LLM.