

Adaía Montaño

Federico Libertun

Programación Avanzada para grandes volúmenes de datos

Informe Trabajo Práctico Final

Maestría en Ciencia de Datos

28-05-2025

Universidad de San Andrés

Informe de Implementación - TP Final

# Resumen General del Sistema

Para el trabajo final de la materia desarrollamos un sistema completo orientado a resolver el problema de recomendaciones de productos para anunciantes en un entorno AdTech. El sistema esta compuesto por una pipeline de procesamiento de datos Airflow, una API con FastAPI, y PostgreSQL como base de datos. Todo el sistema fue desplegado utilizando la infraestructura de Google Cloud Platform (GCP).  
La API, desarrollada con FastAPI, fue empacada en un contenedor Docker y desplegada en Google Cloud Run. Para ello se utilizó una imagen previamente subida al repositorio público de Docker Hub bajo el nombre adaiamontano/recom-api.

# Pipeline en Airflow

La pipeline de procesamiento de datos fue orquestada mediante Apache Airflow, desplegado en una instancia de máquina virtual (VM) en Google Compute Engine. Esta pipeline ejecuta diariamente el DAG dag\_recomendaciones, y puede monitorearse a través de la interfaz web de Airflow disponible en [http://34.41.50.150:8081](http://34.41.50.150:8081" \t "C:\\Users\\adaia\\AppData\\Local\\Temp\\_new) (usuario: admin, contraseña: admin).

Las tareas de la pipeline son las siguientes:

1. Filtrado de datos: Se descargan los archivos ads\_views.csv y product\_views.csv desde el bucket adtech-tp-data/raw, junto con el archivo advertisers.csv. Luego, se filtran los eventos correspondientes únicamente a la fecha anterior a la ejecución del DAG y a los advertisers activos definidos en advertisers.csv.
2. Generación de Top CTR: Con los datos de clics e impresiones (ads\_views.csv), se calcula el click-through-rate (CTR) para cada combinación de advertiser y producto. Se seleccionan los 20 productos con mayor score por advertiser, conformando el ranking top\_ctr.
3. Generación de Top Productos más vistos: Usando los datos de visualización (product\_views.csv), se determinan los 20 productos con mayor cantidad de vistas por advertiser, formando el ranking top\_product.
4. Carga de resultados a la base de datos: Los resultados generados en las dos tareas anteriores se insertan en las tablas top\_ctr y top\_product de una base de datos PostgreSQL alojada en Cloud SQL (adtech-sql).

# Base de Datos PostgreSQL

Utiliza Google Cloud SQL como servicio gestionado de base de datos, específicamente la instancia PostgreSQL llamada adtech-sql, con acceso mediante IP pública y autenticación integrada. Dentro de esta instancia, los resultados del procesamiento diario ejecutado por Airflow son almacenados en la base de datos postgres.

Las dos tablas principales utilizadas en la base de datos PostgreSQL son:

**top\_ctr**(advertiser\_id TEXT, product\_id TEXT, score FLOAT, fecha DATE)

**top\_product**(advertiser\_id TEXT, product\_id TEXT, views INTEGER, fecha DATE)

La tabla top\_ctr almacena los productos con mayor tasa de clics (CTR) por anunciante, calculada a partir de los eventos de clics e impresiones registrados en los logs. El campo score representa la tasa de clics estimada para cada producto.

La tabla top\_product contiene los productos más visualizados por cada anunciante, según la cantidad de vistas registradas para una fecha específica.

Ambas tablas usan como clave primaria la combinación de advertiser\_id, product\_id y fecha, lo cual garantiza la unicidad por día y por pareja anunciante-producto. Esta estructura permite registrar un ranking diario sin sobrescribir datos de días anteriores, y facilita las consultas históricas y la integración con servicios como la API de recomendaciones.

La validación de los resultados puede realizarse mediante Cloud SQL Studio, donde es posible ejecutar consultas SQL directamente. Por ejemplo, consultando SELECT \* FROM top\_product WHERE fecha = '2025-05-30';, se observa cómo el sistema carga datos correctamente cada día a partir del procesamiento automatizado.

# API en FastAPI

Se puede acceder desde este link: <https://recom-api-7120503241.us-central1.run.app/docs#/>

La API fue dockerizada utilizando un Dockerfile, y luego subida a un repositorio en Docker Hub bajo el nombre adaiamontano/recom-api. Posteriormente, se desplegó el contenedor desde Cloud Run, permitiendo el acceso público cumpliendo con los requisitos solicitados en el TP.  
  
Tiene los siguientes endpoints implementados:

**/recommendations/{adv\_id}/{model}:** devuelve hasta 20 productos recomendados.  
**/history/{adv\_id}:** muestra el historial de productos recomendados de los últimos 7 días.  
**/stats/:** devuelve métricas generales como cantidad de anunciantes, productos y registros.  
  
Probamos todos los endpoints desde Swagger y devuelven bien según los datos cargados.

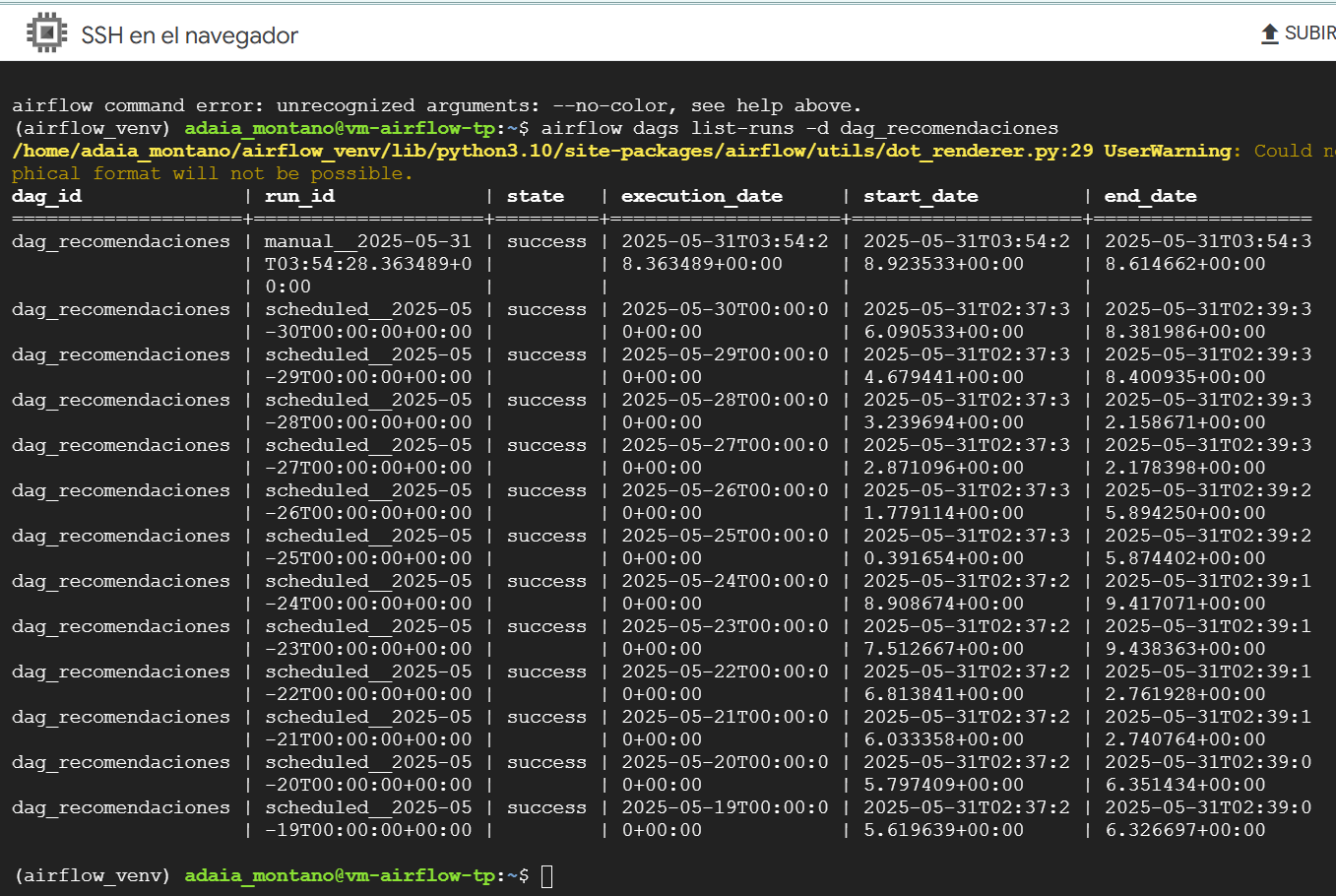
# Evidencia de Funcionamiento

Se incluye la evidencia de la implementación de la pipeline de datos, la API para exponer resultados y todo el despliegue en la nube.

Nota: Pueden encontrar los json actualizados disponibles en la base de datos en el repositorio de Github.

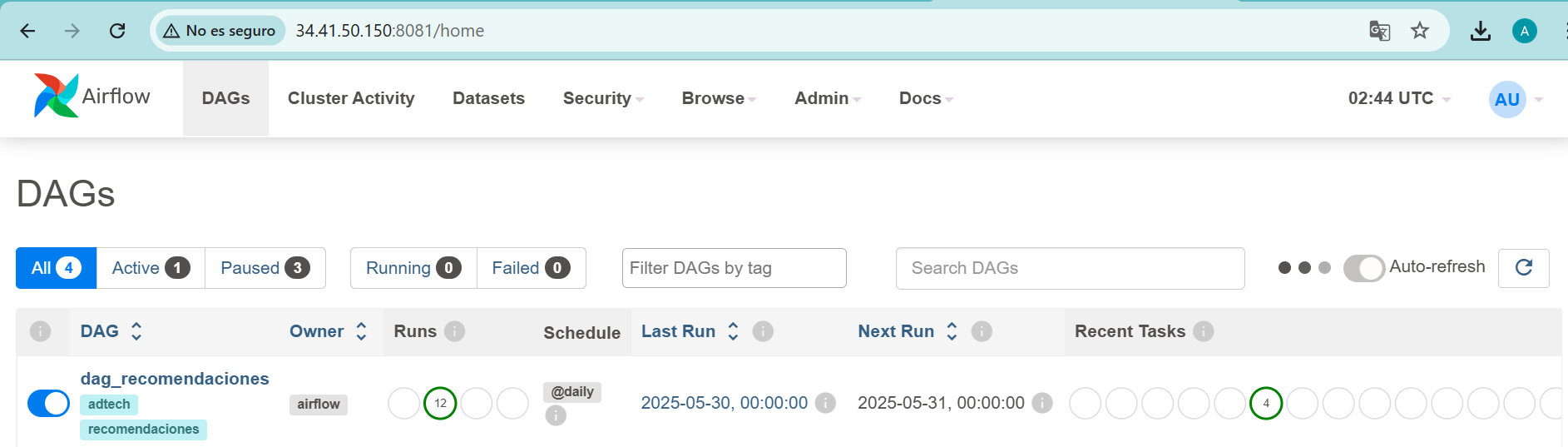
## 5.1 Ejecución del DAG en Airflow

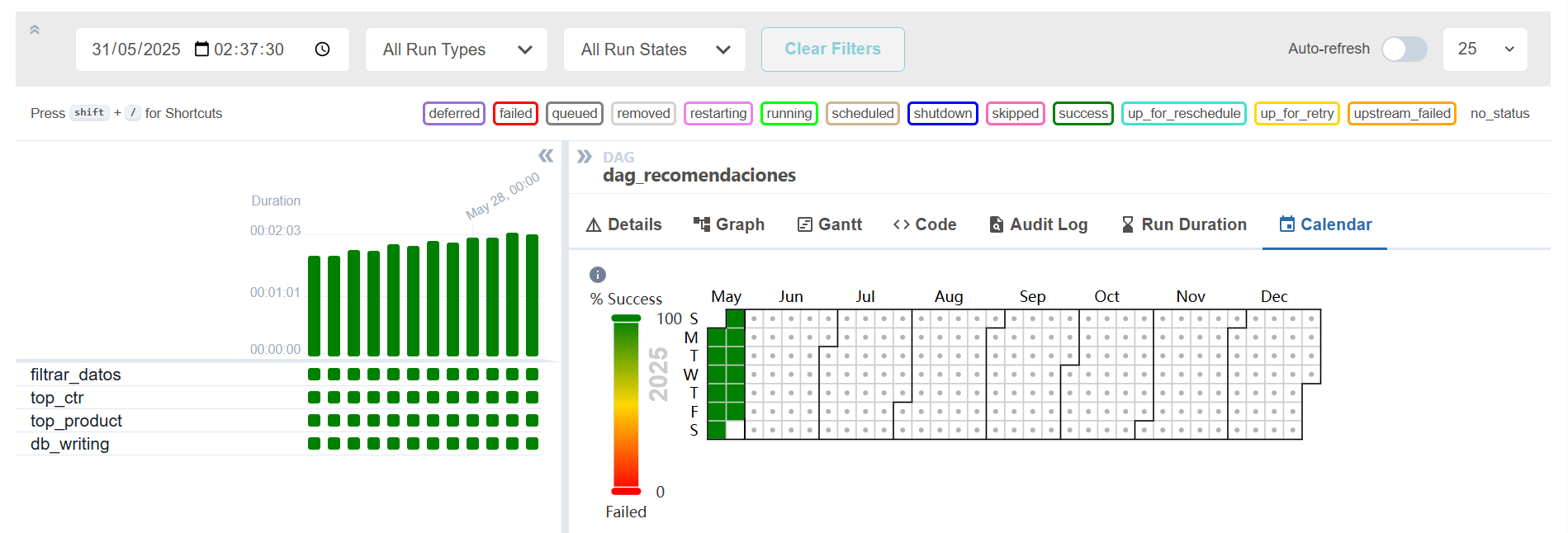
El siguiente registro muestra el historial de ejecuciones del DAG dag\_recomendaciones, ejecutado diariamente mediante Apache Airflow.

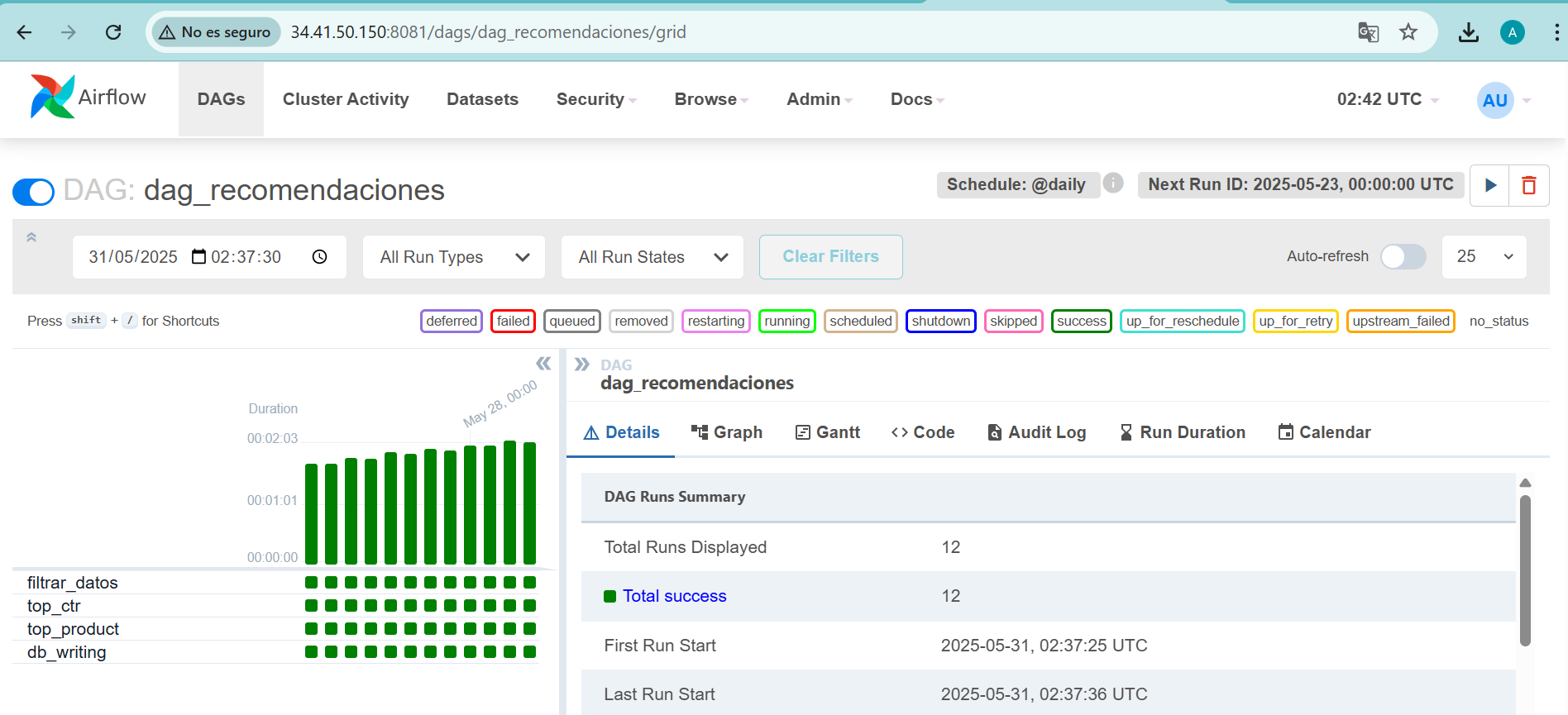


También podemos visualizar el funcionamiento desde AirflowUsuario/Contraseña admin/admin. Código disponible en el repositorio de Github.

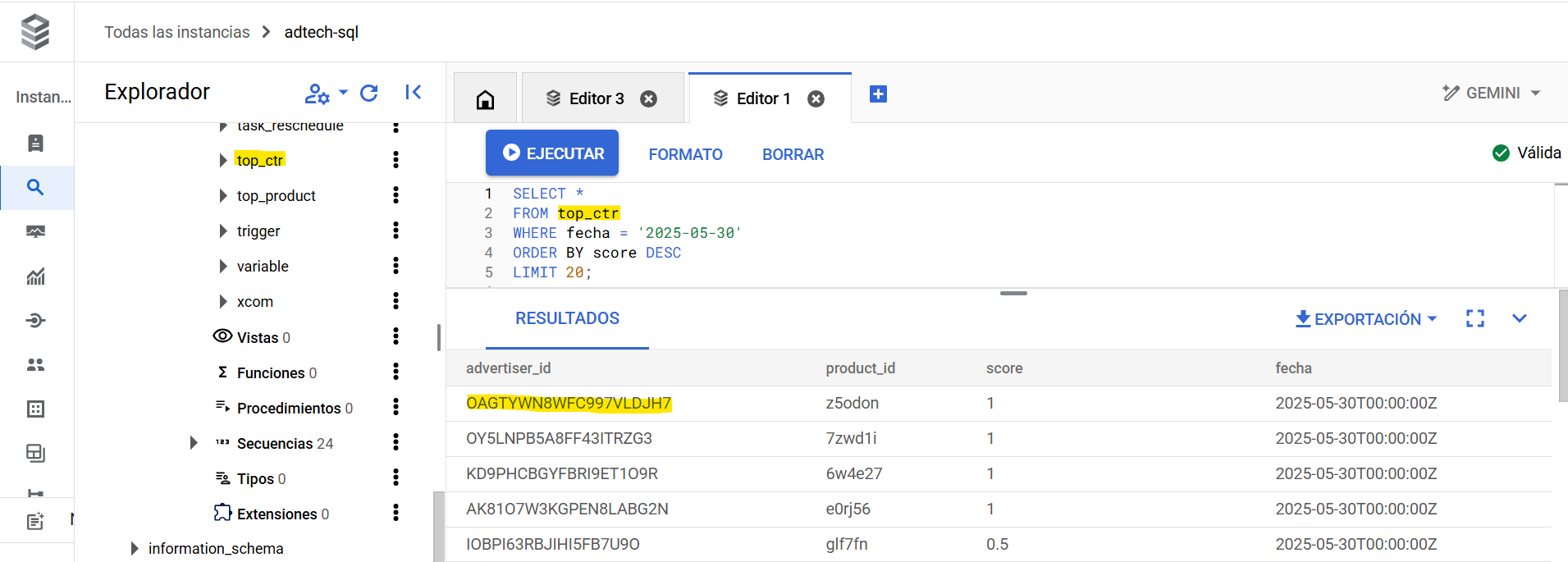
Link: <http://34.41.50.150:8081/home>

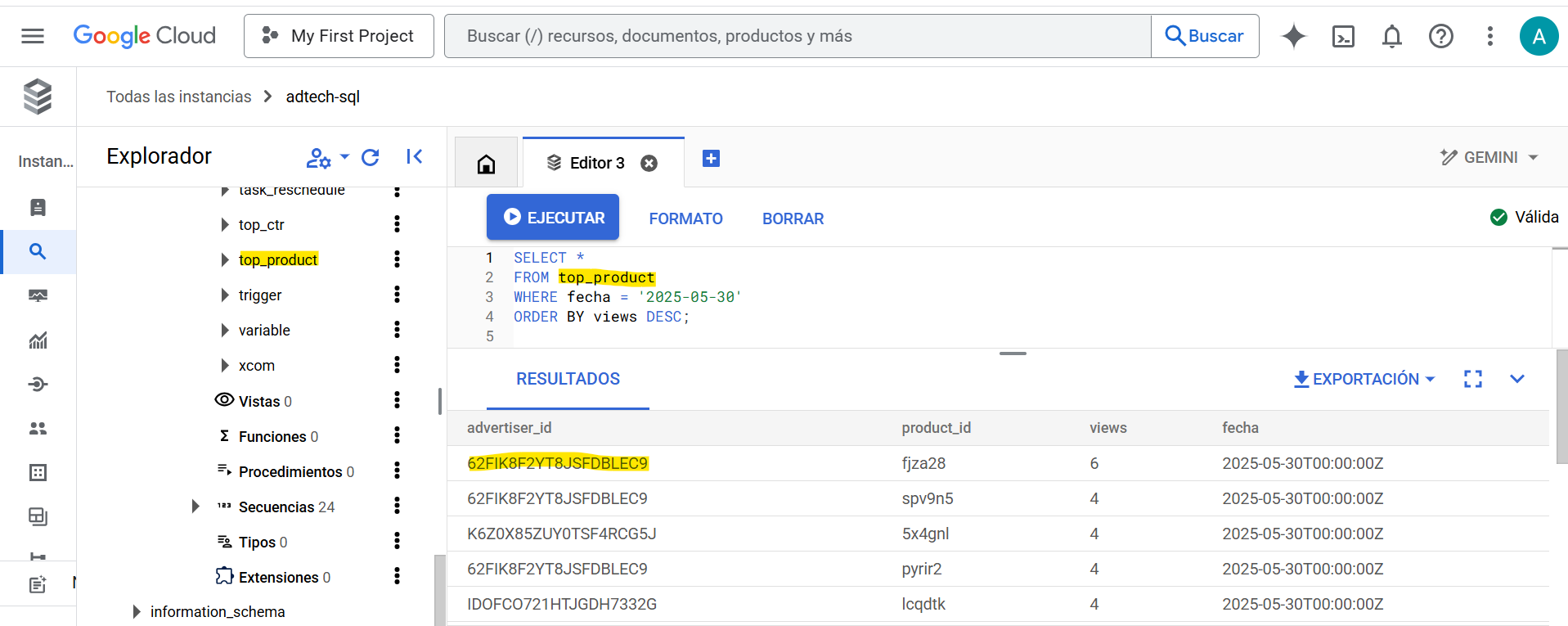




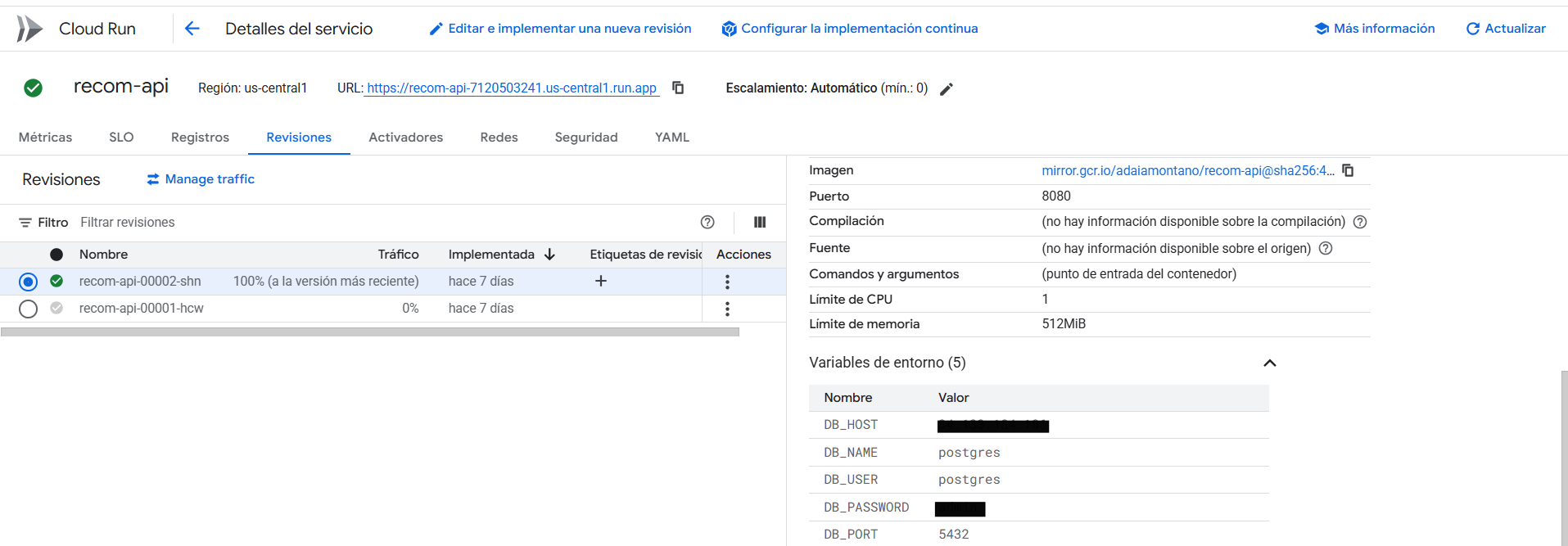


## 5.2 Carga y validación en PostgreSQL

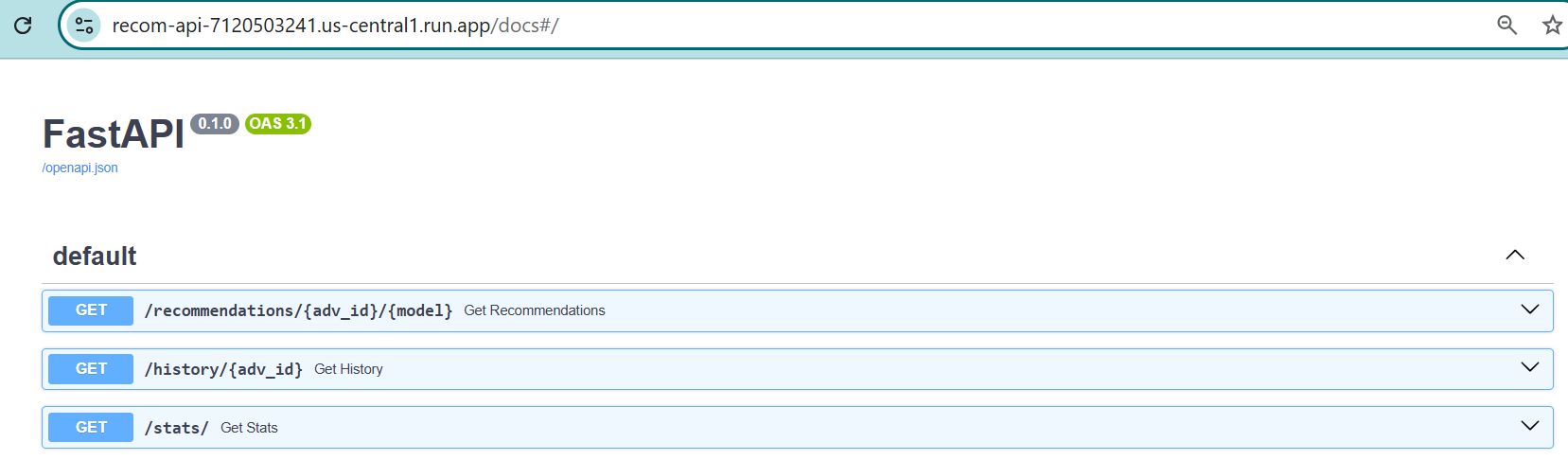




## 5.3 Despliegue en Cloud Run

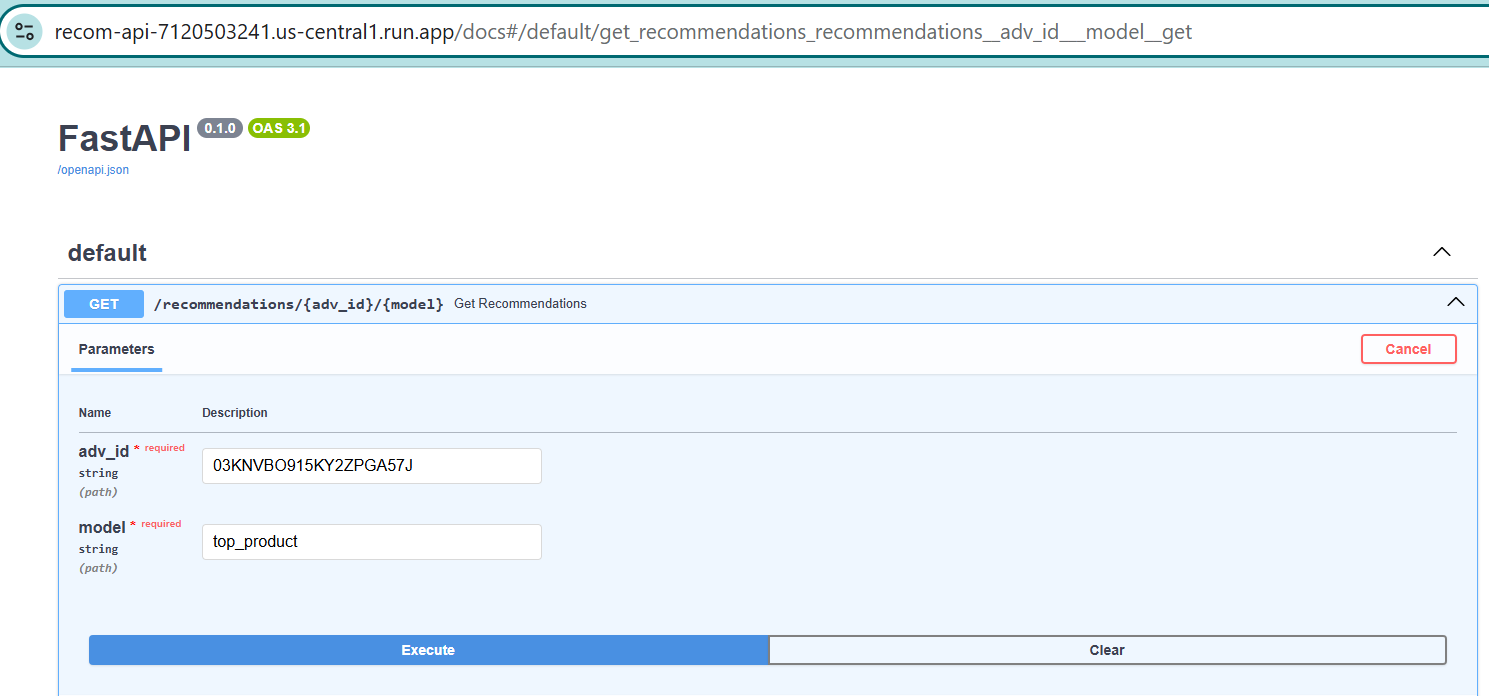


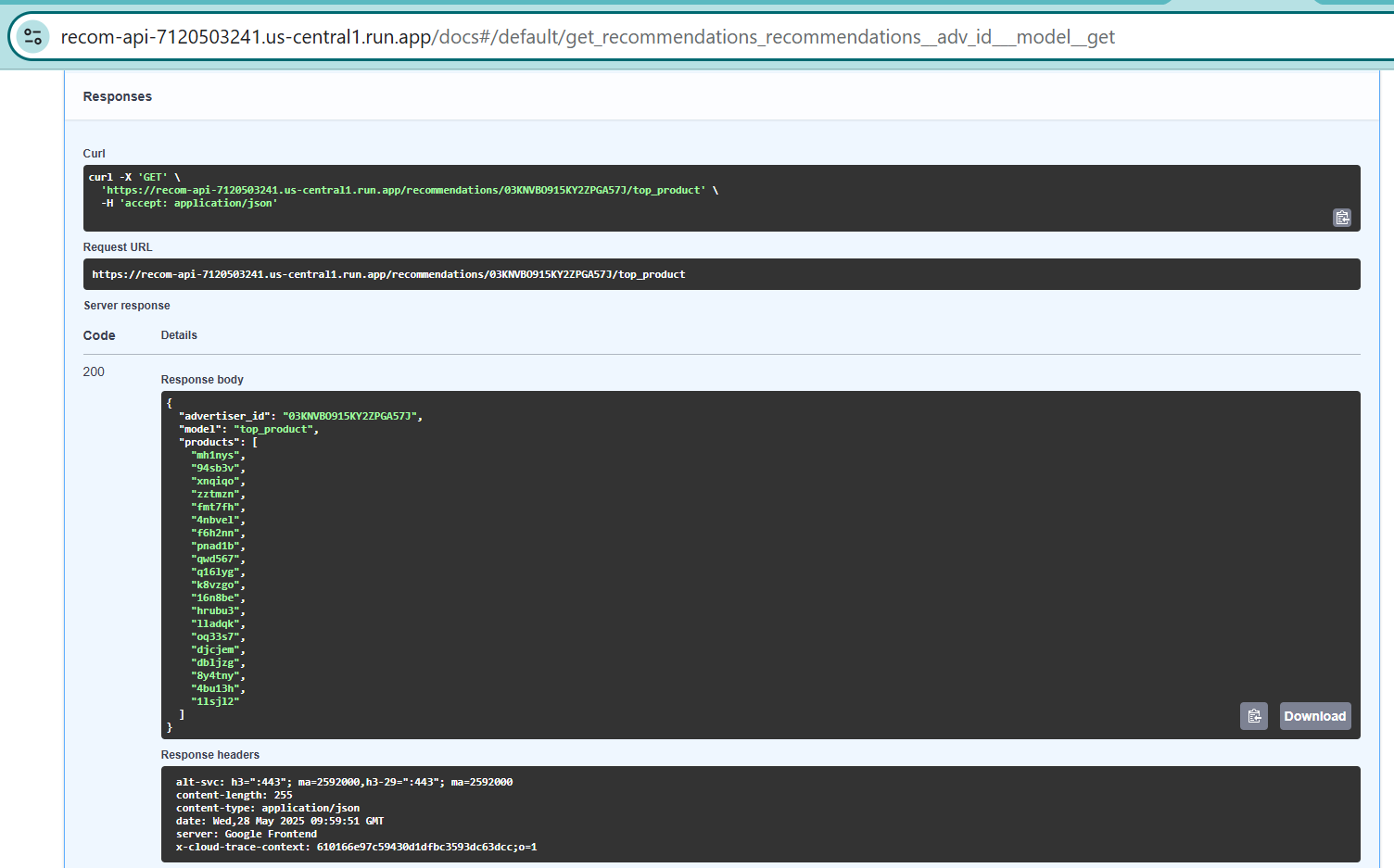
## 6.4 Verificación de la API (Swagger)



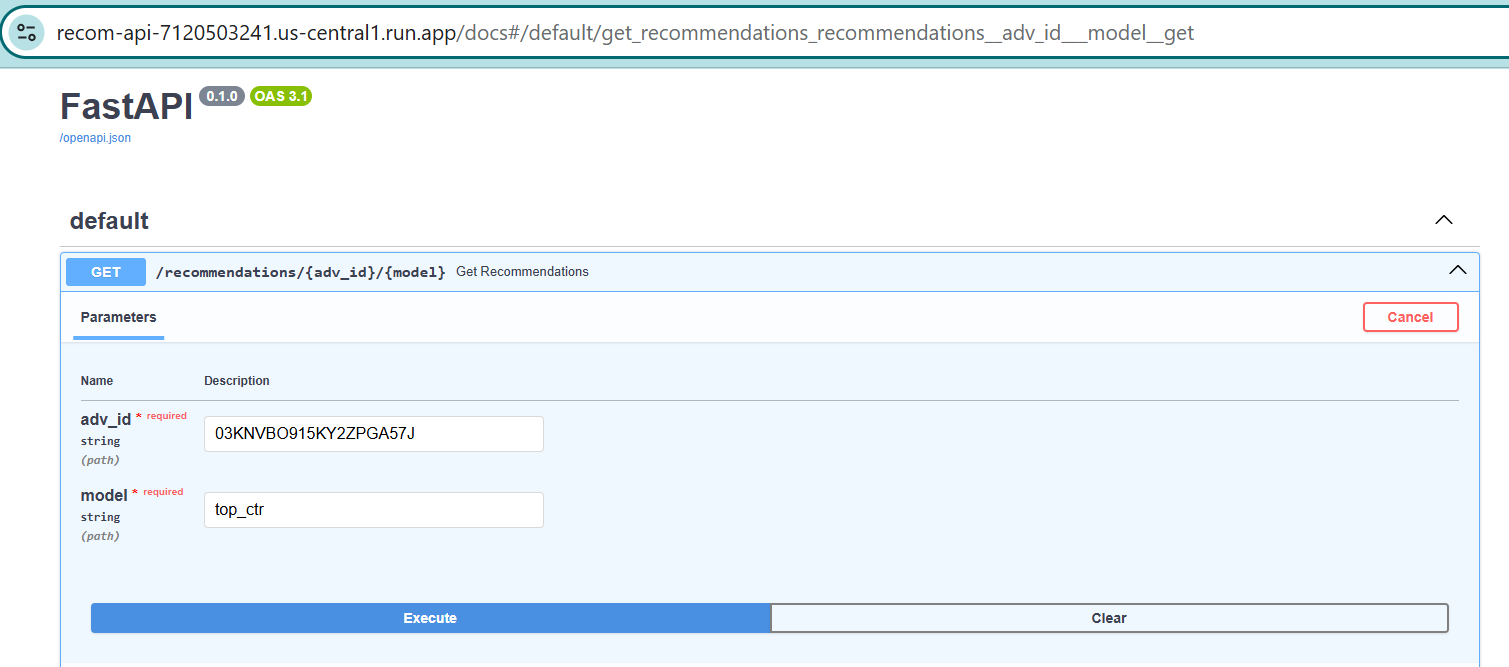
## 6.4.1 Respuestas de la API

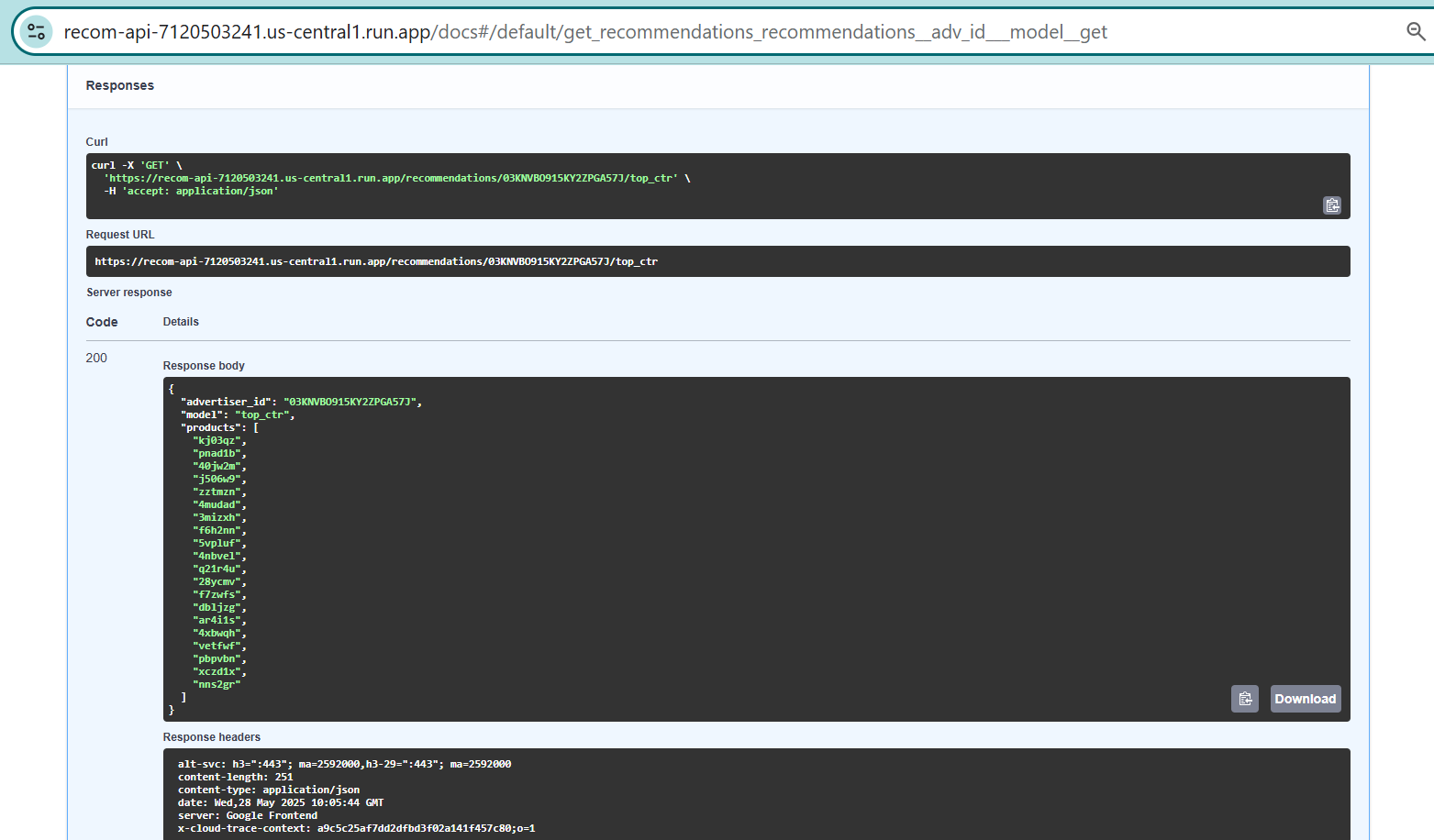
Modelo top\_product para el advertiser 03KNVBO915KY2ZPGA57J



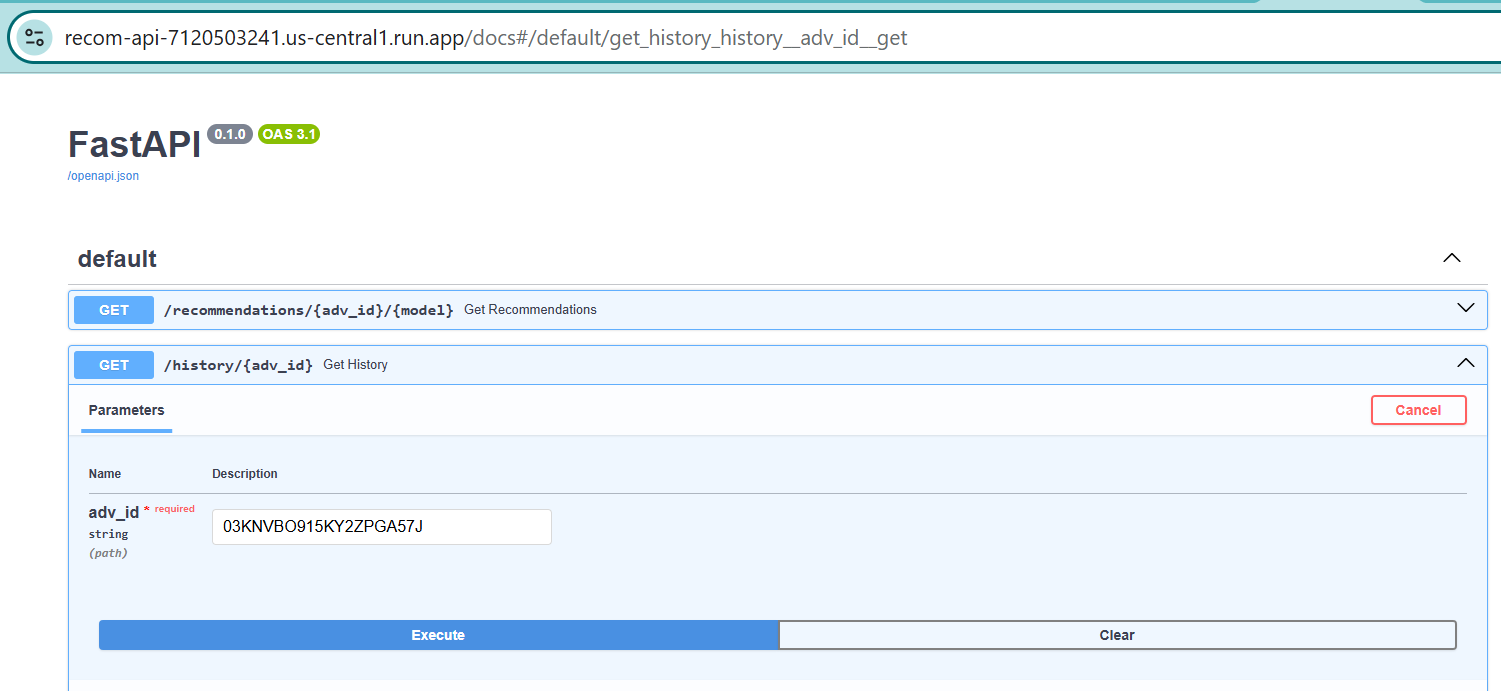


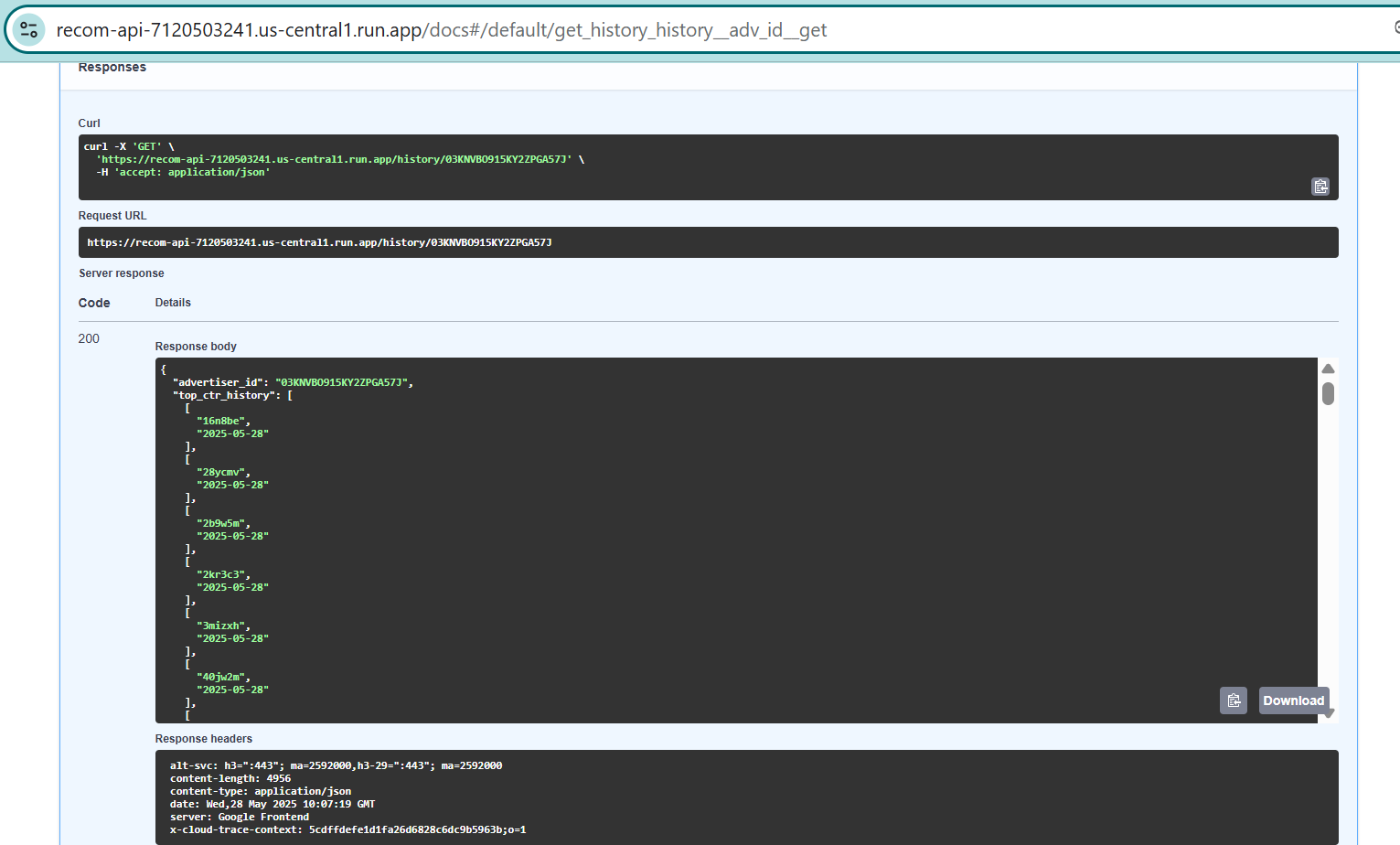
Modelo top\_ctr para el advertiser 03KNVBO915KY2ZPGA57J



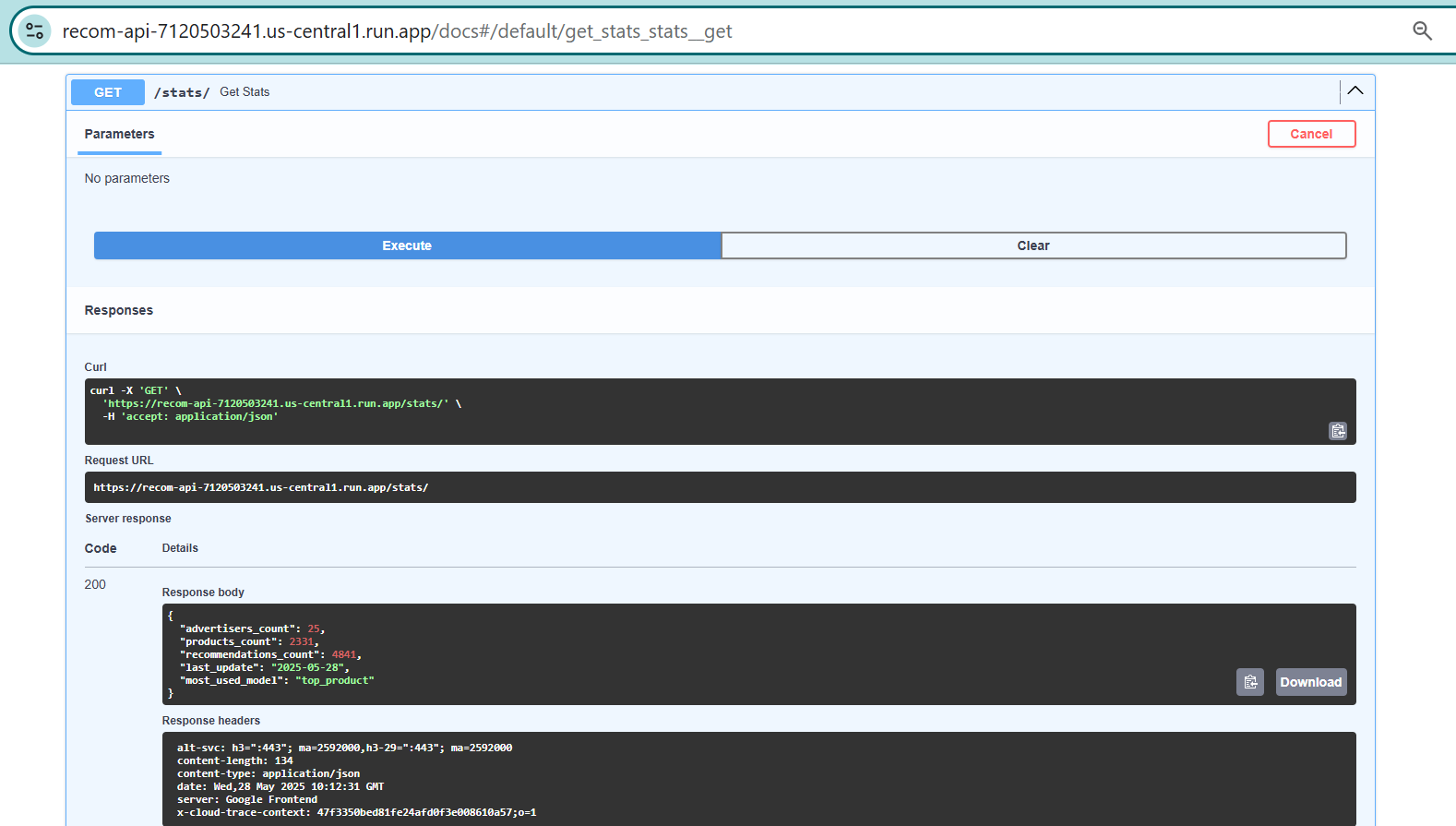


History para el advertiser 03KNVBO915KY2ZPGA57J

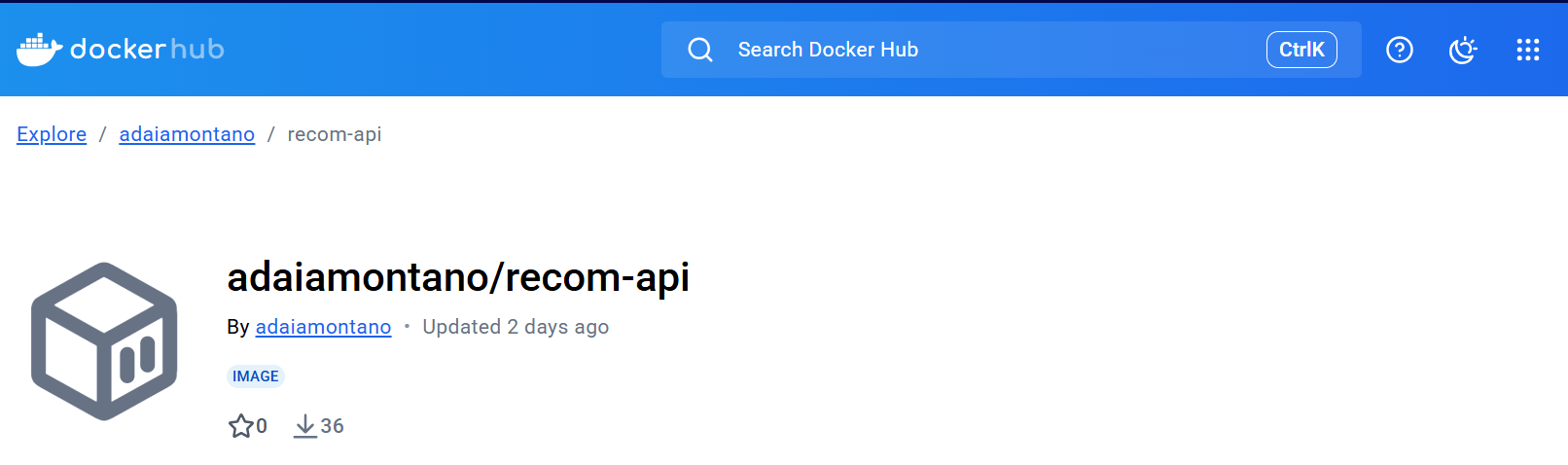




Stats



## 6.5 Docker



# 7. Dificultades Encontradas

* Tuvimos que ajustar manualmente las credenciales y parámetros de conexión para que Airflow pueda conectarse a Cloud SQL. Esto implicó asegurarse de usar la IP pública, configurar los puertos correctamente y declarar bien los datos de conexión dentro del DAG.
* Al momento de hacer el push de la imagen Docker a Docker Hub, fue necesario autenticarse manualmente con docker login, ya que sin eso daba error de permisos y no subía nada. Además, había que asegurarse de tener bien el tag latest y el nombre del repositorio.
* La configuración de Airflow en la VM nos llevó tiempo. Hubo que crear una regla de firewall específica en GCP, y además habilitar el acceso público a través de la IP externa de la instancia.
* En SSH (dentro de la VM de Compute Engine), tuvimos que trabajar con rutas absolutas para mover o descargar archivos (como logs, DAGs, etc.), ya que si no, las rutas relativas fallaban o apuntaban a ubicaciones incorrectas dentro del entorno virtual de Airflow.
* A veces el DAG aparecía con tareas fallidas aunque los archivos o credenciales estaban correctos. Para solucionarlo hubo que revisar los logs directamente en la interfaz de Airflow y hacer varias limpiezas manuales (clear task, reiniciar scheduler, etc.).
* También hubo problemas con la persistencia de datos en Cloud SQL cuando se reiniciaban algunas configuraciones, lo que llevó a tener que re-crear las tablas manualmente y verificar que estuvieran bien creadas antes de volver a correr el DAG.
* Durante la integración final, fue necesario modificar el archivo airflow.cfg para que Airflow pudiera conectarse correctamente a la base de datos PostgreSQL de Cloud SQL. Esto implicó cambiar el parámetro sql\_alchemy\_conn, reemplazando la configuración por defecto (sqlite) con una cadena de conexión PostgreSQL completa, incluyendo usuario, contraseña, IP pública, puerto, y nombre de la base.
* El DAG principal (dag\_recomendaciones) fue ajustado para incluir las 4 tareas solicitadas en la consigna: filtrado de datos, cálculo de top productos por CTR, cálculo de top productos por views, y escritura final en las tablas correspondientes. Además, fue necesario modificar la lógica interna de cada tarea para asegurar que se utilicen correctamente las dependencias entre outputs (XCom) y se respeten los 20 productos esperados por advertiser.
* Otro problema crítico fue que, al cerrar la terminal SSH, el scheduler y el webserver de Airflow dejaban de funcionar, lo que impedía validar el DAG o mostrar resultados. Para resolverlo, se implementaron dos archivos .service (airflow-scheduler.service y airflow-webserver.service) en systemd. Esto permitió dejar ambos procesos activos como servicios del sistema, ejecutándose en segundo plano aún después de cerrar sesiones SSH, y garantizando disponibilidad continua para correcciones o revisiones posteriores.

# Conclusión

El TP integró varias tecnologías y nos permitió poner en práctica muchos de los conceptos vistos en clase. Pudimos automatizar todo el flujo: desde la ingesta de datos, pasando por el procesamiento diario en Airflow, hasta la carga final en una base de datos PostgreSQL en Cloud SQL. Además, desarrollamos y publicamos una API funcional en FastAPI, dockerizada y desplegada en Google Cloud Run, lo que también nos sirvió para entender cómo documentar y exponer endpoints correctamente.

Más allá de las dificultades técnicas que fuimos resolviendo, el resultado fue un sistema completo y funcional que resuelve el problema planteado de forma eficiente y escalable. Fue clave poder ver cómo se conectan todos los componentes (storage, DAG, base de datos, API) dentro de la infraestructura de GCP y cómo se puede monitorear y validar todo desde interfaces como Cloud SQL y Airflow.