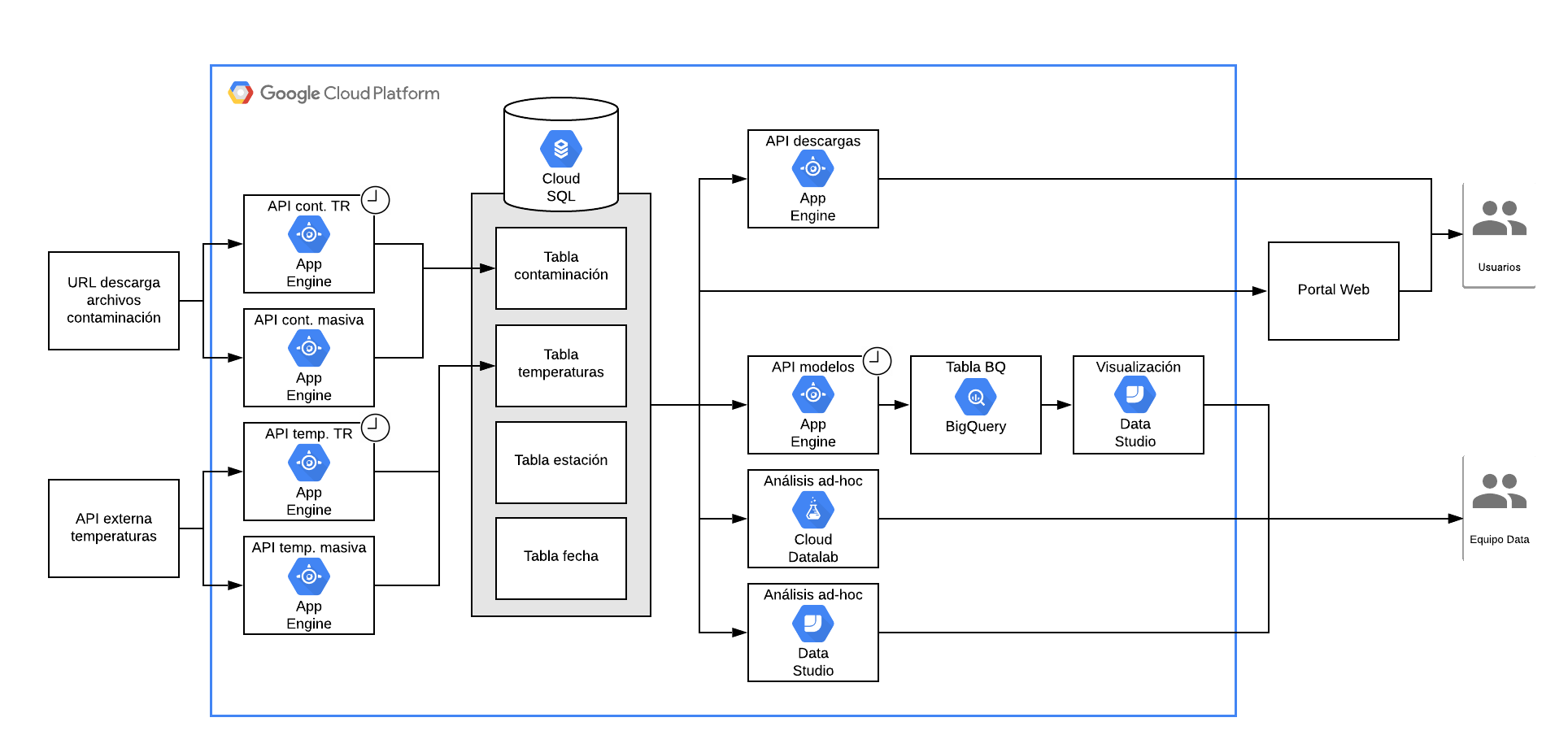
Propuesta de arquitectura 1:



Las fuentes de datos son dos: por un lado, la de contaminación, cuyos datos se pueden descargar a través de una url y, por otro, la de temperaturas, cuya información se puede extraer mediante llamadas get a una API.

Desarrollaríamos dos APIs propias para cada fuente de datos, ya que puede haber dos tipos de cargas: en tiempo real y masiva. Estas cuatro APIs estarían alojadas en instancias de Google App Engine, y usaríamos el framework Flask. Su cometido sería la ETL, es decir, la extracción, transformación y carga de datos.

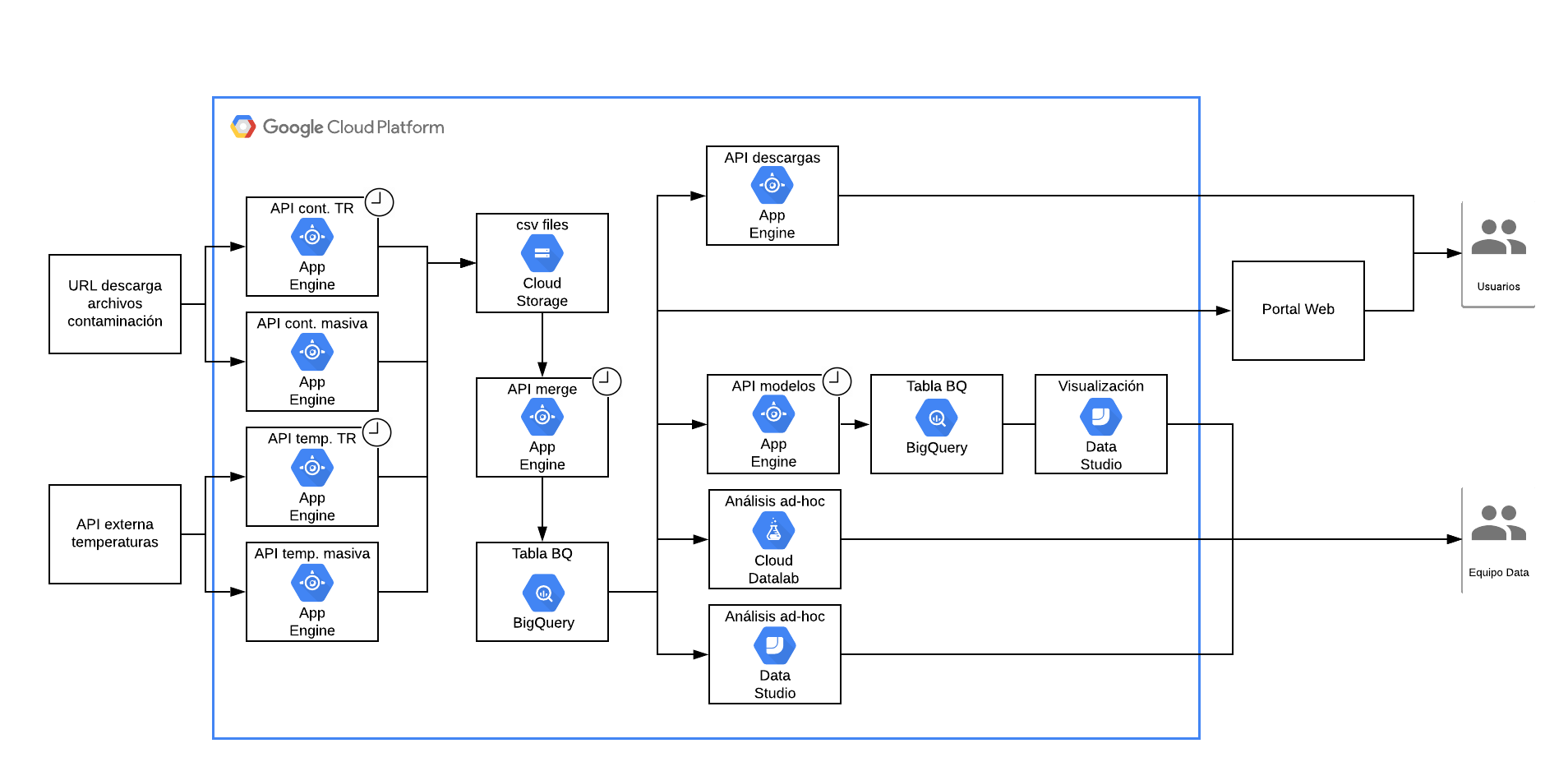
En este primer caso, la carga de datos sería en una base de datos PostgreSQL alojada en un servicio de Cloud SQL, formada por una tabla de contaminación, una de temperaturas, una de las estaciones de sensores para captar la contaminación y otra de fechas. Además, generaríamos otra tabla que contuviera información tanto de contaminación como de temperaturas, para que los usuarios que consulten esta base de datos puedan tener información conjunta.

A partir de aquí, se producen varias salidas de la información:

* API descargas: desarrollaríamos otra API a la que los usuarios pudieran hacer llamadas para descargar los datos con las fechas como parámetro.
* Portal web: el equipo de Front se podría conectar a esta base de datos para permitir que los usuarios visualicen mapas de calor, gráficas de evolución, etc.
* API modelos: esta API contendría el código de los modelos de ML que se utilizaran para prever la contaminación de los 3 días siguientes. Los resultados de esos modelos se podrían alojar en una tabla de BigQuery (o en otra tabla de la base de datos), a la cual podríamos conectar un informe de Google Data Studio para visualizar los resultados.
* Análisis ad-hoc: podríamos hacer este análisis de dos maneras:
  + Cloud Datalab: esta herramienta de consiste en unos notebooks que se conectan a cualquier otro servicio de nuestro proyecto de GCP, de manera que podríamos hacer consultas a nuestra base de datos para conocer información que se necesite de manera puntual rápidamente.
  + Data Studio: con esta herramienta de visualización fácil de usar (aunque algo limitada) podríamos analizar los datos de manera ágil.

Nota: El icono del reloj en la esquina superior de las APIs “Contaminación Tiempo Real”, “Temperatura Tiempo Real”, y “Modelos”, significa que estas se ejecutarían de manera programada, en este caso, cada hora, para extraer, tratar, cargar y visualizar los dato en tiempo real. Esto se conseguiría gracias al servicio de “Tareas Cron” de Google Cloud, servicio que permite programar tareas.

Propuesta de arquitectura 2:



Las fuentes de datos son dos: por un lado, la de contaminación, cuyos datos se pueden descargar a través de una url y, por otro, la de temperaturas, cuya información se puede extraer mediante llamadas get a una API.

Desarrollaríamos dos APIs propias para cada fuente de datos, ya que puede haber dos tipos de cargas: en tiempo real y masiva. Estas cuatro APIs estarían alojadas en instancias de Google App Engine, y usaríamos el framework Flask. Su cometido sería la ETL, es decir, la extracción, transformación y carga de datos.

En este segundo caso, el primer paso sería cargar los datos ya tratados en Cloud Storage como ficheros csv, tanto de contaminación como de temperaturas. Después, una API (API merge) se encargaría de extraer de esos segmentos de Cloud Storage los ficheros horarios, haría un merge de contaminación y temperatura, y cargaría en una tabla de BigQuery los registros.

A partir de aquí, se producen varias salidas de la información:

* API descargas: desarrollaríamos otra API a la que los usuarios pudieran hacer llamadas para descargar los datos con las fechas como parámetro.
* Portal web: el equipo de Front se podría conectar a esta base de datos para diseñar un portal para permitir que los usuarios visualicen mapas de calor, gráficas de evolución, etc.
* API modelos: esta API contendría el código de los modelos de ML que se utilizaran para prever la contaminación de los 3 días siguientes. Los resultados de esos modelos se podrían alojar en una tabla de BigQuery (o en otra tabla de la base de datos), a la cual podríamos conectar un informe de Google Data Studio para visualizar los resultados.
* Análisis ad-hoc: podríamos hacer este análisis de dos maneras:
  + Cloud Datalab: esta herramienta de consiste en unos notebooks que se conectan a cualquier otro servicio de nuestro proyecto de GCP, de manera que podríamos hacer consultas a nuestra base de datos para conocer información que se necesite de manera puntual rápidamente.
  + Data Studio: con esta herramienta de visualización fácil de usar (aunque algo limitada) podríamos analizar los datos de manera ágil.

Nota: El icono del reloj en la esquina superior de las APIs “Contaminación Tiempo Real”, “Temperatura Tiempo Real”, “Merge” y “Modelos”, significa que estas se ejecutarían de manera programada, en este caso, cada hora, para extraer, tratar, cargar y visualizar los dato en tiempo real. Esto se conseguiría gracias al servicio de “Tareas Cron” de Google Cloud, servicio que permite programar tareas.