

# Paso a paso del Proceso de Ingesta y Análisis de Datos de red.cl

<b>Documento paso a paso del Proceso de Ingesta y Análisis de Datos de red.cl</b>	<b>1</b>
1. Ingesta de Datos desde API red.cl a Google Cloud Storage	1
2. Automatización de la Ingesta de Datos con Google Cloud Scheduler	2
3. Preparación de Datos con Google Cloud Dataprep	2
4. Análisis de Datos con Google BigQuery	2
5. Visualización de Datos con Looker Studio	3

## 1. Ingesta de Datos desde API red.cl a Google Cloud Storage

La primera parte del proceso implica recoger datos de la API de red.cl y almacenarlos en un bucket de Google Cloud Storage.

Para este proceso, creamos una Cloud Function de Google Cloud. Esta función fue escrita en Python y utilizó la biblioteca `requests` para hacer una solicitud GET a la API de red.cl. La respuesta se recibió en formato JSON, que luego fue convertido a un formato compatible con Google Cloud Storage y finalmente se almacenó en un bucket en formato JSON. Los datos recogidos incluyen detalles sobre los servicios de recorrido de los autobuses, así como información adicional tal como los horarios, las rutas, las paradas y los servicios disponibles en cada parada.

Aquí se puede ver un extracto de la Cloud Function que utilizamos para este proceso:

(...) Cloud Functions    ← Editar función

✓ Configuración — 2 Código

Entorno de ejecución  
Python 3.10

Código fuente  
Editor directo

main.py

requirements.txt

key.json

Punto de entrada \*  
main



Presiona Alt + F1 para ver las opciones de accesibilidad.

```
1 def main(requests):
2     # Librerías
3     import json
4     import pandas as pd
5     from pandas import json_normalize
6     from google.cloud import storage
7     import requests
8     import gcsfs
9     import fsspec
10
11     # Proceso de obtención de JSON de la api de red.cl
12     response = requests.get('https://www.red.cl/restservice_v2/rest/getservicios/all')
13     data = response.json()
14
15     # Configurar las credenciales de autenticación
16     client = storage.Client.from_service_account_json('key.json')
17
18     # Obtener el bucket en el que se almacenarán los datos
19     bucket = client.bucket('informacion-gob')
20
21     # Crear un objeto Blob en el bucket
22     blob = bucket.blob('recorridos/servicios_red.json')
23
24     # Convertir los datos a formato JSON y escribirlos en el objeto Blob
25     blob.upload_from_string(json.dumps(data), content_type='application/json')
```

## 2. Automatización de la Ingesta de Datos con Google Cloud Scheduler

Para asegurar que los datos estén siempre actualizados, decidimos automatizar el proceso de ingesta de datos utilizando Google Cloud Scheduler. Este servicio nos permitió programar la ejecución de nuestra Cloud Function cada 15 días, asegurándonos de que siempre tengamos los datos más recientes disponibles para el análisis.

La configuración de Google Cloud Scheduler implicó especificar la frecuencia de las tareas (en nuestro caso, cada 15 días) y la función a ejecutar (nuestra Cloud Function para la ingesta de datos). Una vez configurado, Google Cloud Scheduler se encarga de ejecutar la tarea en el intervalo especificado.

 Cloud Scheduler  Crea un trabajo

- Define el programa

Nombre \*

gtfs\_api

Debe ser único en los trabajos de la misma región.

Región \*

southamerica-east1 (São Paulo)

Descripción

Scheduler activador de Function-API

Frecuencia \*

\*\* 15 \*\*

?

Las programaciones se especifican con el formato cron de Unix. P. ej., cada minuto: `* * * * *`, cada 3 horas: `"B * /3 * * *`, todos los lunes a las 9:00 a.m.: `"B 9 * * 1"`. [Más información](#)

i

- Minuto:
  - Cada minuto
- y día:
  - Igual a 15 (día del mes)

Zona horaria \*

hora estándar de Chile (CLT)

i

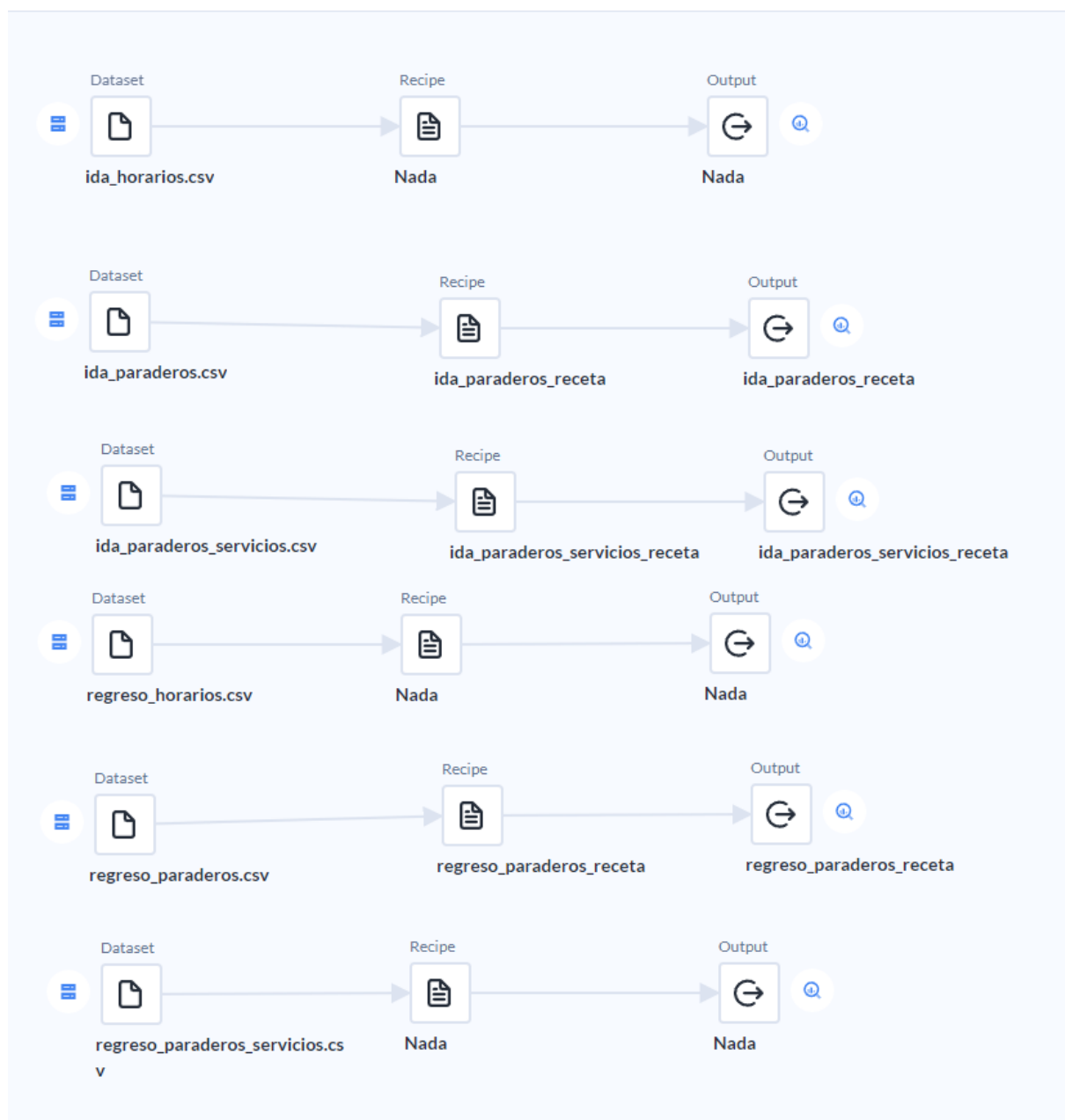
Los trabajos establecidos en zonas horarias afectadas por el horario de verano pueden ejecutarse fuera de la cadencia durante el cambio a DST. Si usas una zona horaria UTC, puedes evitar el problema. [Más información](#)

### 3. Preparación de Datos con Google Cloud Dataprep

Después de recoger los datos, el siguiente paso en nuestro proceso fue prepararlos para el análisis. Para hacer esto, utilizamos Google Cloud Dataprep, una herramienta de limpieza y preparación de datos que permite transformar, limpiar y explorar datos.

En Dataprep, importamos nuestros datos de Google Cloud Storage y creamos un flujo de trabajo que incluyó diversas transformaciones y limpiezas. Esto incluyó la eliminación de datos innecesarios, la normalización de los campos y la transformación de datos para facilitar el análisis posterior.

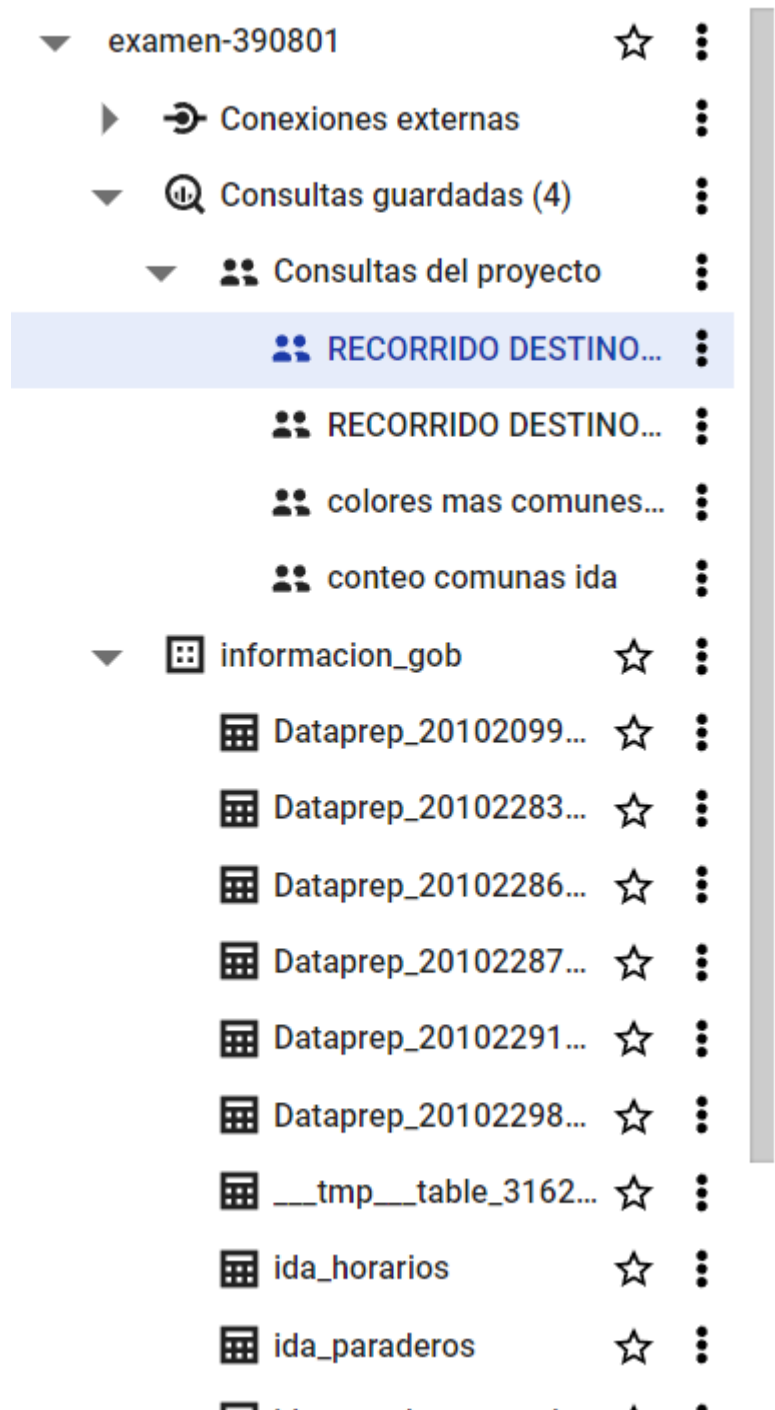
Flow Api



## 4. Análisis de Datos con Google BigQuery

Una vez que los datos estuvieron limpios y preparados, los cargamos en Google BigQuery para analizarlos. BigQuery es un almacén de datos empresarial que permite el análisis rápido de grandes conjuntos de datos.

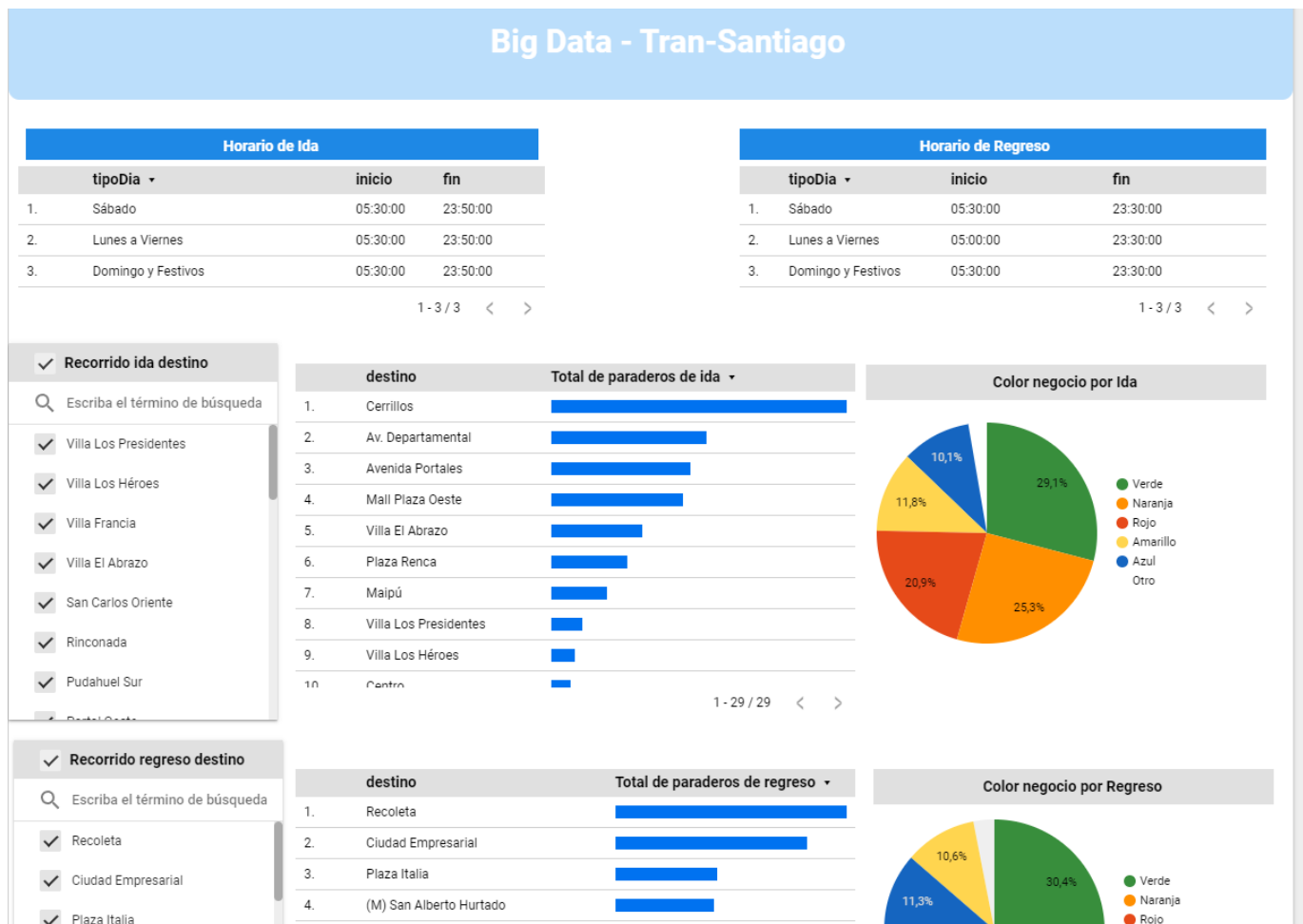
Con los datos en BigQuery, pudimos realizar consultas SQL para extraer información valiosa. Por ejemplo, pudimos identificar los servicios de autobús más utilizados, las rutas más populares, las horas pico de viaje, entre otros análisis.



## 5. Visualización de Datos con Looker Studio

Finalmente, utilizamos Looker Studio para visualizar los resultados de nuestro análisis. Looker es una plataforma de análisis y visualización de datos que permite crear tableros de control interactivos y atractivos.

Con Looker, pudimos crear visualizaciones que mostraban de manera clara y comprensible los hallazgos de nuestro análisis. Esto permitió a los usuarios finales entender mejor los datos y tomar decisiones basadas en ellos.



Este documento es un resumen del proceso que seguimos para la ingesta, preparación, análisis y visualización de los datos de la API de red.cl. La integración de diversas herramientas de Google Cloud Platform nos permitió crear un flujo de trabajo automatizado y eficiente que transformó los datos en información valiosa y fácilmente comprensible.