PRÁCTICA FINAL: JTALENT

Diseño e Implementación de un Pipeline de Datos ELT en Google Cloud Platform

Iñigo Altube

JAKALA IBERIA

Septiembre de 2025

Índice general

Capítulo 1

Introducción y Objetivos

1.1.- CONTEXTO DEL PROYECTO

Este proyecto aborda el desafío de una entidad financiera que necesita consolidar su información, actualmente dispersa en una API interna, en un sistema robusto y escalable que permita la generación de reportes y el análisis de tendencias.

El desafío principal consiste en diseñar y construir un pipeline de datos completo en **Google Cloud Platform (GCP)**, abarcando desde la ingesta automática y resiliente de los datos hasta su transformación en un modelo analítico optimizado y su posterior explotación.

1.2.- OBJETIVOS DEL PROYECTO

El objetivo general es diseñar y documentar un flujo de ingeniería de datos en GCP. Para ello, se han definido los siguientes objetivos específicos:

- **Implementar un proceso de ingesta automática** de datos desde la API interna, garantizando la completitud y la resiliencia del proceso.
- **Diseñar un almacenamiento de datos multicapa** óptimo y escalable, siguiendo las mejores prácticas de la industria (Data Lake y Data Warehouse).
- **Definir y calcular KPIs** relevantes para el análisis de las operaciones financieras, tanto los obligatorios como los propuestos.
- **Facilitar el consumo de los datos** mediante un dashboard interactivo, asegurando la calidad y la consistencia de la información presentada.

Capítulo 2

Arquitectura de la Solución en GCP

La arquitectura ha sido diseñada siguiendo los principios de desacoplamiento, escalabilidad y seguridad, utilizando servicios gestionados de GCP para minimizar la sobrecarga operativa y los costes.

2.1.- DIAGRAMA DE ARQUITECTURA

El flujo de datos sigue un recorrido lógico orquestado por un workflow maestro que invoca a sub-workflows especializados, asegurando una clara separación de responsabilidades entre los componentes.

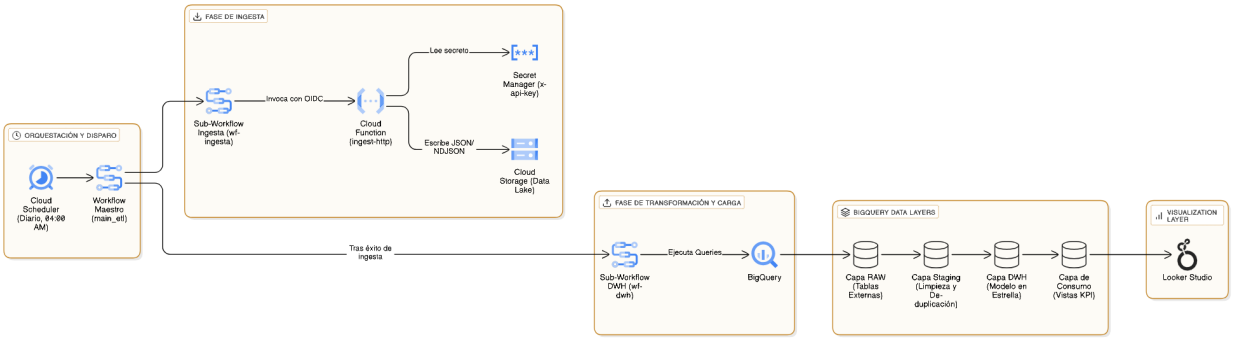

Figura 2.1: Diagrama de la arquitectura ELT implementada en GCP.

2.2.- JUSTIFICACIÓN DE SERVICIOS CLAVE

Cada servicio ha sido seleccionado para cumplir una función específica, buscando siempre la eficiencia, la escalabilidad y la seguridad.

Orquestación: Cloud Workflows Se eligió por su naturaleza serverless y su modelo de coste basado en el uso. A diferencia de soluciones como Cloud Composer, Workflows no requiere la gestión de infraestructura. Su capacidad para invocar otros servicios de GCP de forma nativa, paralelizar tareas y su manejo robusto de errores (‘try/except‘) son claves para la eficiencia y resiliencia del pipeline.

Procesamiento de Ingesta: Cloud Functions (2ª Gen) Se optó por una Cloud Function como motor de ingesta por su escalabilidad automática y su modelo de pago por uso.

Almacenamiento de Secretos: Secret Manager Imprescindible para almacenar de forma segura la `x-api-key` de la API. Con su uso se aumenta la seguridad, evitando exponer credenciales en el código o en variables de entorno.

Data Lake: Cloud Storage Es la base del Data Lake. Su bajo coste, alta durabilidad y su perfecta integración con BigQuery lo convierten en la opción ideal.

Data Warehouse: BigQuery Su arquitectura serverless, columnar y su motor de procesamiento masivo en paralelo lo hacen ideal para el análisis de grandes volúmenes de datos. Permite la separación lógica en datasets (RAW, Staging, DWH), facilitando una gobernanza clara del dato.

Capítulo 3

Implementación del Pipeline de Datos

3.1.- FASE DE INGESTA: EXTRACT & LOAD

El proceso de ingesta está diseñado para ser robusto y flexible, manejando tanto las cargas diarias como la necesidad de realizar cargas históricas.

3.1.1.- Análisis de la API y Estrategia de Ingesta

La API de JTALENT no soporta filtros de fecha, lo que significa que cada llamada a un endpoint devuelve un **volcado completo** de los datos. Ante esta característica, se ha diseñado una estrategia de ingesta que convierte cada volcado diario en un **snapshot histórico inmutable** en nuestro Data Lake. Esta decisión de diseño proporciona:

- **Trazabilidad Completa:** Podemos auditar el estado exacto de los datos en cualquier día del pasado.
- **Reprocesabilidad:** Si se detecta un error en la lógica de transformación, podemos regenerar el DWH desde cero utilizando los snapshots históricos.
- **Detección de Cambios (CDC):** Al comparar el snapshot de un día con el del día anterior, podemos inferir cambios, como los registros eliminados, que la API no notifica explícitamente.

3.1.2.- Lógica de Extracción (Cloud Function)

El código Python de la Cloud Function es el encargado de interactuar con la API. Su lógica incluye:

- **Lectura segura de secretos** desde Secret Manager en cada ejecución.

- **Iteración por rango de fechas:** Procesa día a día el rango solicitado (`start_date`, `end_date`), generando un snapshot diario en el Data Lake, lo que otorga gran flexibilidad para cargas históricas.
- **Manejo de paginación:** Un bucle `while` recorre todas las páginas de cada endpoint hasta extraer el 100 % de los datos.
- **Control de calidad en origen:** Conversión de valores `NaN` a `null` para garantizar la compatibilidad con el formato JSON estricto que requiere BigQuery.
- **Resiliencia:** Implementa una lógica de reintentos con *exponential backoff* para manejar fallos temporales de la API.

3.1.3.- Estructura en el Data Lake (Cloud Storage)

Los datos se almacenan en formato NDJSON en el bucket `raw-jtalent` siguiendo un **particionado tipo Hive**:

```
gs://raw-jtalent/raw/<endpoint>/ingestion_date=YYYY-MM-DD/
```

Este esquema permite a BigQuery escanear únicamente los datos de las fechas relevantes, lo que reduce drásticamente los costes y mejora el rendimiento.

3.2.- FASE DE TRANSFORMACIÓN: TRANSFORM

Esta es la fase donde los datos crudos se convierten en información de valor para el negocio, aplicando una robusta lógica de limpieza y modelado.

3.2.1.- Arquitectura de Datos Multicapa en BigQuery

Hemos implementado una arquitectura de tres capas, una práctica estándar que garantiza la calidad y la mantenibilidad:

Capa RAW Compuesta por **Tablas Externas** que apuntan a Cloud Storage. Son una representación inmutable del origen.

Capa Staging (‘stg‘) Tablas nativas intermedias donde se realiza la primera limpieza, el casteo de tipos y la estandarización. Se vacían y recargan en cada ejecución (‘Truncate and Load‘).

Capa DWH (‘dwh‘) El modelo final, optimizado para consulta y análisis.

3.2.2.- Diseño del Data Warehouse: Modelo en Estrella

Se ha diseñado un **Modelo en Estrella** Este modelo evita la redundancia masiva de datos, optimiza el rendimiento de las consultas en un motor columnar como BigQuery, y facilita enormemente el mantenimiento al tener una única fuente de verdad para cada concepto de negocio.

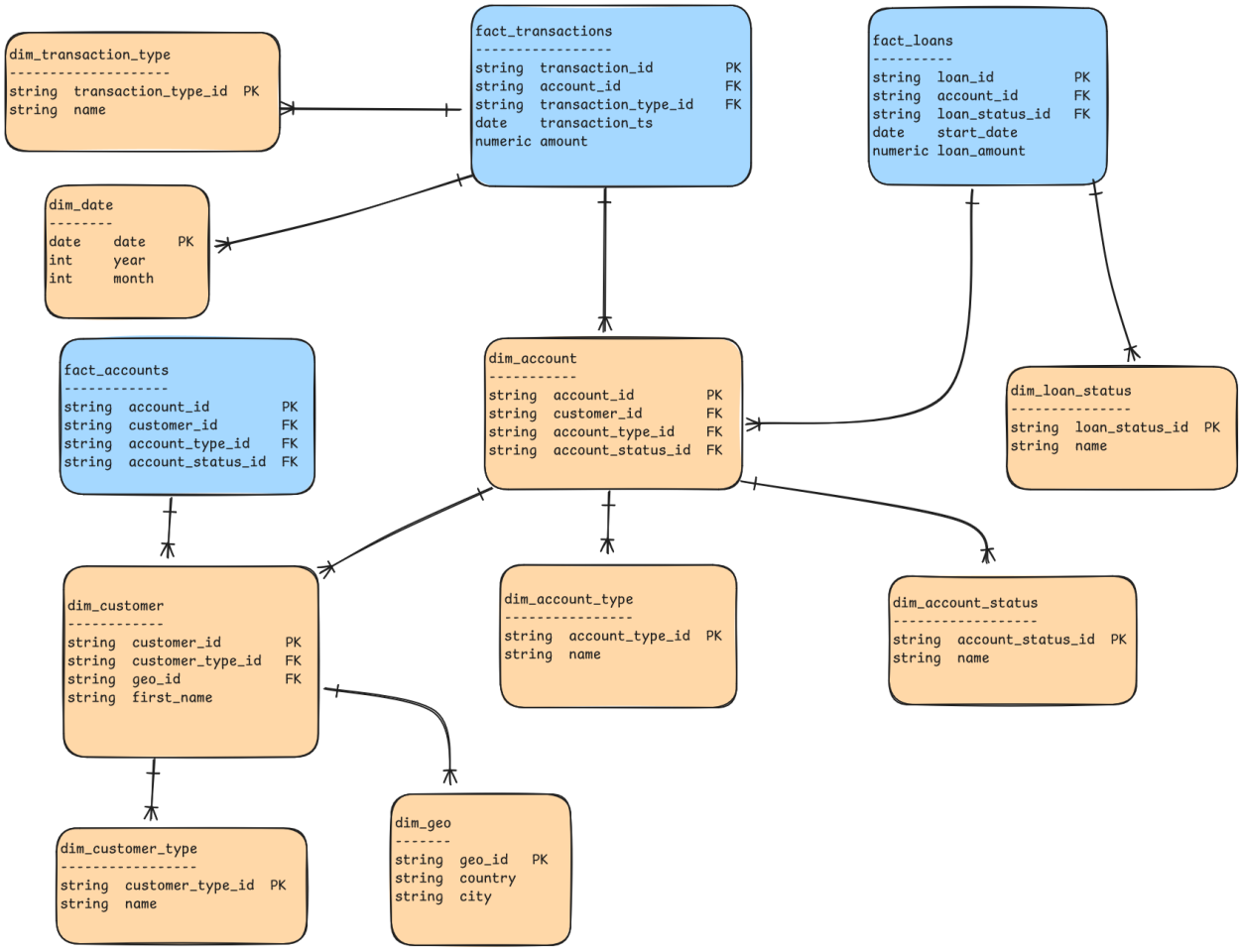

Figura 3.1: Diagrama del Modelo en Estrella implementado.

3.2.3.- Manejo de la Calidad de Datos y Técnicas Avanzadas

El pipeline no solo mueve datos, sino que los limpia y enriquece activamente:

- **De-duplicación de Datos en Origen:** Durante el análisis, se detectó que la API de origen envía registros con claves primarias duplicadas (ej. `CustomerID`, `LoanID`) dentro de la misma carga diaria. Para blindar el DWH contra esta inconsistencia, el workflow incluye pasos explícitos que utilizan la función de ventana `ROW_NUMBER()` para eliminar estos duplicados en la capa de Staging, protegiendo así la integridad.

- **Enriquecimiento de Datos Geográficos:** Se identificó una inconsistencia crítica en los datos: el campo `country` era estático y erróneo. Para solucionarlo y cumplir con el KPI obligatorio de morosidad por región, se implementó una **estrategia de enriquecimiento de datos**. Se creó una tabla de referencia local (‘dwh.ref_city_country_map‘) a partir de un fichero JSON de mapeo, y el workflow la utiliza para inferir el país correcto a partir de la ciudad, corrigiendo el error de la fuente.
- **Sincronización Idempotente:** El DWH se sincroniza con el estado de la fuente usando una estrategia de `MERGE` y `DELETE`, asegurando que el modelo final refleja fielmente la realidad actual, incluyendo las eliminaciones, y que el proceso puede re-ejecutarse sin causar duplicados ni inconsistencias.
- **Slowly Changing Dimension Type 2 (SCD2):** La `dim_customer` guarda un historial completo de los cambios de los clientes. Esta lógica compleja se ha encapsulado en un **Stored Procedure (`sp_upsert_dim_customer`)** que maneja inserciones, actualizaciones y borrados lógicos (*soft deletes*).

3.3.- ORQUESTACIÓN Y AUTOMATIZACIÓN

La automatización del pipeline es gestionada por un workflow maestro que asegura la correcta secuenciación y ejecución de las tareas.

3.3.1.- Workflow Maestro (‘main_etl‘)

Este workflow es el punto de entrada único para todo el proceso y es invocado diariamente por **Cloud Scheduler**. Su diseño sigue una estrategia de ejecución por fases para optimizar el rendimiento y gestionar las dependencias:

1. **Fase de Ingesta:** Invoca al sub-workflow `wf-ingesta`, que ejecuta la Cloud Function para extraer los datos del día.
2. **Fase de DWH:** Tras el éxito de la ingesta, invoca al sub-workflow `wf-dwh`, que a su vez ejecuta sus fases de transformación:
   - Carga de Staging en **paralelo**.
   - Carga de Dimensiones en **paralelo** (con pasos secuenciales para dependencias).
   - Carga de Hechos en **paralelo**.
   - Cálculo del Snapshot de forma secuencial.

Esta estrategia de paralelización reduce drásticamente el tiempo total de ejecución del pipeline, optimizando el uso de recursos y, por tanto, el coste.

3.4.- SEGURIDAD Y GOBERNANZA (IAM)

Se ha seguido el **Principio de Mínimo Privilegio** en todo el diseño, definiendo un conjunto de Cuentas de Servicio (SA) con responsabilidades claras y separadas para minimizar la superficie de riesgo.

Cuadro 3.1: Resumen de Cuentas de Servicio y sus Permisos Clave.

Cuenta de Servicio	**Responsabilidades y Justificación**
`sa-ingestion`	Usada por la Cloud Function. Solo puede acceder a Secret Manager y escribir en el bucket RAW. No tiene acceso a BigQuery.
`sa-workflows`	Usada por los Workflows. Puede invocar otros workflows y la Cloud Function de ingesta, y ejecutar jobs en BigQuery.
`sa-scheduler`	Usada por Cloud Scheduler. Su único permiso es invocar al workflow maestro, minimizando su alcance.

Capítulo 4

Análisis de Negocio: KPIs

El modelo de datos ha sido diseñado para dar soporte a los siguientes indicadores clave de rendimiento.

4.1.- KPIS OBLIGATORIOS

Usuarios activos por tipo

Definición: Número de clientes únicos con al menos una cuenta en estado ”Activo”, agrupado por el tipo de cliente.

Ingreso medio por cliente (IMPC)

Definición: Promedio del balance neto de transacciones (ingresos menos retiradas) por cada cliente.

Tasa de crecimiento anual (TCA)

Definición: Crecimiento porcentual en el número de aperturas de cuentas distintas en comparación con el año anterior.

Tasa de morosidad por región

Definición: Porcentaje de préstamos en estado ”Overdue” sobre el total de préstamos, agrupado por el **país enriquecido** de los clientes para corregir las inconsistencias del origen.

Importe promedio de préstamo (IPP)

Definición: Valor medio de los importes de todos los préstamos concedidos.

4.2.- KPIS PROPUESTOS Y JUSTIFICACIÓN

% de Cuentas Activas por Tipo de Cuenta

Definición: Mide qué porcentaje de las cuentas de un tipo específico se encuentran en estado "Activo" frente al total.

Pregunta de Negocio: ¿Qué productos (tipos de cuenta) tienen mayor retención y uso por parte de los clientes?

Decisiones que Apoya: Permite al equipo de producto decidir qué tipos de cuenta potenciar o rediseñar, y al equipo de marketing crear campañas para reactivar cuentas de un tipo específico.

Frecuencia Media de Transacciones por Cliente

Definición: Número promedio de transacciones realizadas por cada cliente en un periodo determinado.

Pregunta de Negocio: ¿Cuál es el nivel de *engagement* o vinculación de nuestros clientes con nuestros servicios?

Decisiones que Apoya: Ayuda a segmentar clientes para acciones de fidelización o venta cruzada. Un descenso en este KPI puede ser un indicador temprano de riesgo de fuga de clientes (*churn*).

Tasa de Activación (N30) por Mes de Apertura

Definición: Mide, para cada **corte mensual** de nuevos clientes, qué porcentaje de ellos realizó su primera transacción dentro de los primeros 30 días desde la apertura de su cuenta.

Pregunta de Negocio: ¿Nuestra capacidad para activar a los nuevos clientes está mejorando o empeorando con el tiempo? ¿Las campañas de marketing de un mes específico atrajeron a clientes de mayor calidad (más activos)?

Decisiones que Apoya: Es una herramienta para **medir el impacto** de los cambios en el proceso de *onboarding*. Permite al negocio visualizar tendencias (ej. "la tasa de activación ha caído un 5 % en el último trimestre") y correlacionar el rendimiento de cada cohorte de clientes con las acciones de marketing o producto realizadas en ese mes.

Capítulo 5

Consumo de Datos y Visualización

El último eslabón del pipeline es la capa de consumo, implementada en **Looker Studio**. El objetivo es proporcionar al área de negocio una herramienta visual e interactiva para explorar los datos y monitorizar los KPIs definidos.

5.1.- DISEÑO Y CONEXIÓN

El dashboard se conecta directamente a las **vistas de KPI (`kpi_...`)** y **tablas de hechos (`facts_...`)** creadas en el dataset 'dwh' de BigQuery. Este enfoque de "capa semántica" garantiza que la lógica de negocio reside en el DWH, asegurando consistencia y un rendimiento óptimo, ya que Looker Studio solo consultará datos pre-calculados y agregados.

5.2.- ESTRUCTURA DEL DASHBOARD

El informe se estructuró en varias páginas para facilitar la navegación:

1. **Resumen Ejecutivo:** Contiene tarjetas de resultados con los KPIs más importantes y un gráfico de columnas para la evolución anual de aperturas.
2. **Análisis de Clientes y Riesgo:** Profundiza en los KPIs de "Usuarios activos por tipo" y visualiza la "Tasa de morosidad por país" en un mapa geográfico interactivo.
3. **Análisis de Productos:** Muestra el rendimiento y la salud de los diferentes tipos de cuenta a través de gráficos y tablas de desglose.

Se implementaron **controles interactivos**, como filtros por rango de fechas y tipo de cliente, para permitir a los usuarios explorar los datos de forma autónoma y descubrir sus propios insights.

Capítulo 6

Conclusión y Trabajos Futuros

6.1.- CONCLUSIONES

Se ha construido con éxito un pipeline de datos completo, desde la ingesta hasta la transformación, utilizando servicios serverless de GCP y intentando seguir las mejores prácticas de la industria, para conseguir una arquitectura robusta, escalable, segura.

6.2.- LIMITACIONES Y LECCIONES APRENDIDAS

Un componente clave de cualquier proyecto técnico es el análisis crítico de sus limitaciones y de los puntos que podrían mejorarse en futuras iteraciones.

- **Dependencia de una API de Volcado Completo:** La principal limitación del sistema viene impuesta por la propia API de origen. Al no soportar cargas incrementales (ej. por fecha de modificación), nos vemos obligados a descargar y procesar el conjunto completo de datos cada día. Aunque la arquitectura intenta manejar esto de forma eficiente, en un escenario con miles de millones de registros, esta estrategia podría volverse costosa y lenta.
- **Análisis Exploratorio y Pruebas de Calidad de Datos:** Durante el desarrollo, la validación de los datos de la API (como la detección de duplicados o inconsistencias en los esquemas) se realizó de forma manual mediante consultas SQL exploratorias. Si bien este enfoque fue suficiente para identificar y corregir problemas clave (como los duplicados en las tablas de `accounts` y `transactions`), no se implementó un sistema de tests de calidad automatizado y registrado.
- **Despliegue y Pruebas Manuales:** Todo el despliegue de la infraestructura y los workflows se ha realizado mediante comandos manuales de `gcloud`. Para un entorno productivo, el

siguiente paso sería la implementación de un flujo de **Integración Continua y Despliegue Continuo (CI/CD)**.

6.3.- TRABAJOS FUTUROS

El pipeline actual sienta unas bases sólidas sobre las que se pueden construir futuras mejoras:

- **Monitorización y Alertas Avanzadas:** Implementar un dashboard en Cloud Monitoring con métricas personalizadas (ej. filas procesadas por día, latencia de la ingesta) y configurar alertas en Cloud Logging para notificar proactivamente sobre fallos en el pipeline.
- **Análisis Predictivo:** Utilizar los datos históricos del DWH para entrenar modelos de Machine Learning en BigQuery ML, como por ejemplo un modelo de predicción de riesgo de fuga de clientes (churn) o un modelo de propensión a la morosidad.