PROYECTO INTEGRADOR: DOCUMENTACIÓN DE ARQUITECTURA Y MODELO DE DATOS

LAURA ARBOLEDA GALLEGO

GIORDAN JESE RICARDO PARRA

INFRAESTRUCTURA DE BIG DATA

INGENIERÍA EN DESARROLLO DE SOFTWARE Y DATOS

ANDRÉS FELIPE CALLEJAS

IU DIGITAL DE ANTIOQUIA

MEDELLÍN

2025

Contenido

Introducción

La Big Data es una de las bases más importantes para el análisis, gestión y manipulación de grandes volúmenes de datos. En el siguiente proyecto integrador, se va a mostrar el procedimiento de ingestión, limpieza y enriquecimiento de datos a partir de una API de Google Books dentro de un entorno simulado de la nube. El objetivo de este proyecto es diseñar y documentar una arquitectura de Big Data que desglose los procesos desarrollados a lo largo del proyecto codificado, optimizando sus procesos de extracción, almacenamiento y transformación a partir de diversas fuentes.

Descripción General de la Arquitectura

Este proyecto integrador se divide en tres momentos clave: Ingestión, preprocesamiento y enriquecimiento. Estas fases, dentro del flujo de Big Data, permite el desarrollo de extracción, limpieza y transformación de los datos en un entorno predeterminado, en este caso, completamente simulado. Los datos almacenados se registran a través de SQLite3 y Scripts de Python que permiten el proceso automatizado de los datos. Estas fases se integran a través de un flujo de GitHub Actions, dividiendo el flujo en pequeñas tareas automatizadas.

Componentes principales

- **Base de datos analítica (SQLite3):** Se utiliza para contener los datos extraídos, limpios y transformados.
- **Scripts de procesamiento:**
  - **Ingesta.py:** Se encarga de vincularse a la base de datos a partir de SQLite3, a partir de esto carga los datos obtenidos de la API de Google Books, cargándolos a una tabla SQLite3.
  - **Limpieza.py:** Toma los datos almacenados en la base de datos analítica, los cuales contienen los datos puros de la API. Este proceso lo que hace es eliminar las

inconsistencias de la tabla original, datos duplicados y datos nulos, ordenando los datos de manera tal que estos queden más consistentes.

- **Transformacion.py:** Toma los datos reorganizados y procesados en la limpieza, vinculándose con un dataframe adicional que contiene datos que complementan, en este caso, una tabla principal que será Books, añadiendo claves compuestas y columnas adicionales.
- **GitHub Actions:** Permite la ejecución continua de determinados procesos paso por paso. En este punto, se automatizan los procesos ya sea instalación de dependencias, generar archivos en determinados formatos como CSV y TXT, generando también las respectivas auditorías con la información de cada archivo.

Diagramas de arquitectura

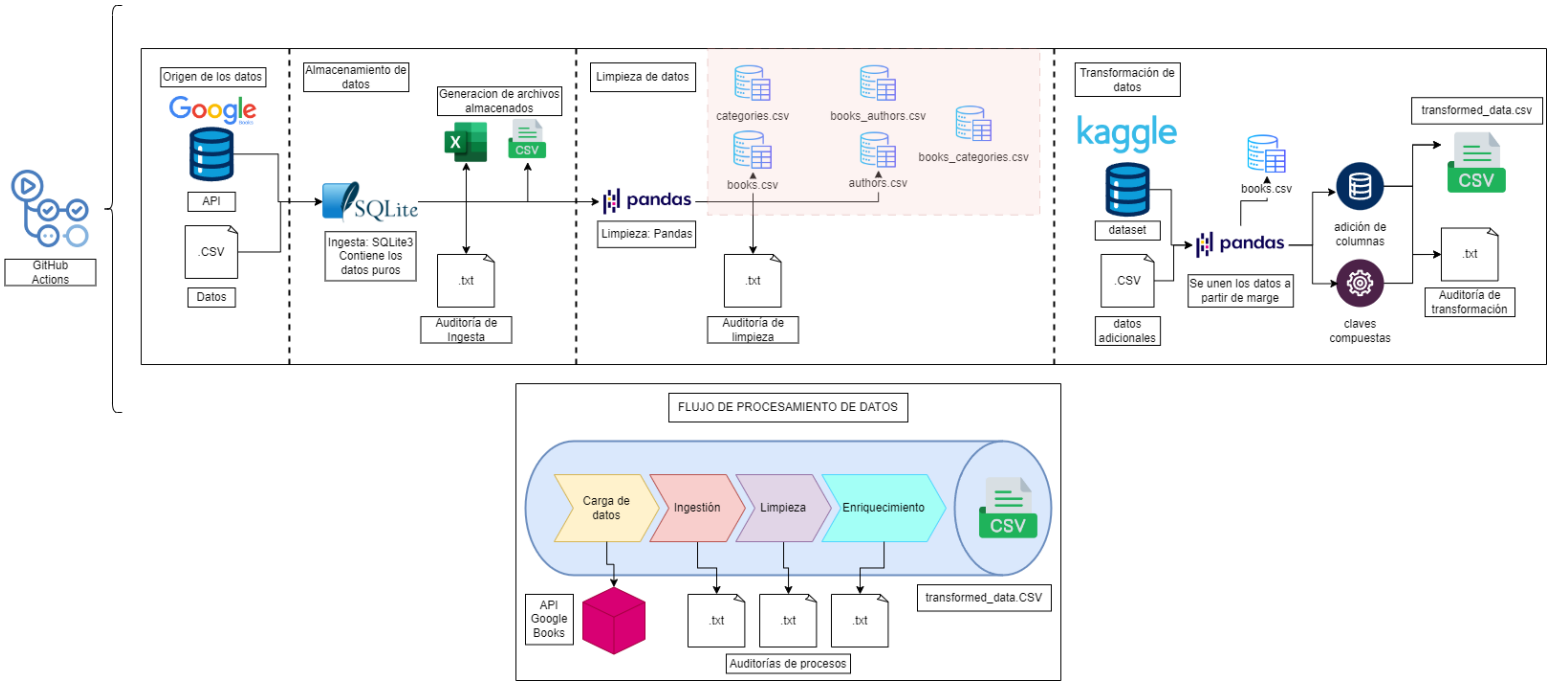

Figura 1. Diagrama de arquitectura

❖ **Modelo relacional**

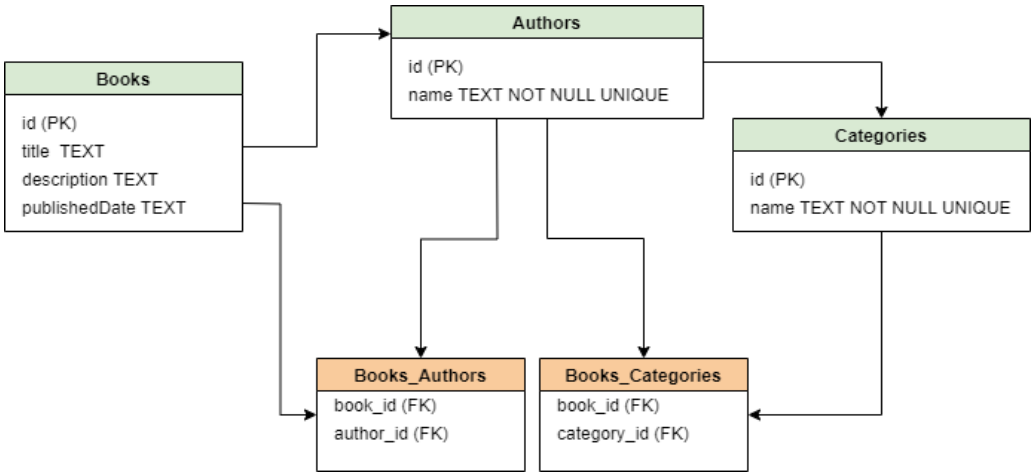

Figura 2. Modelo relacional

- **Diagrama de flujo**

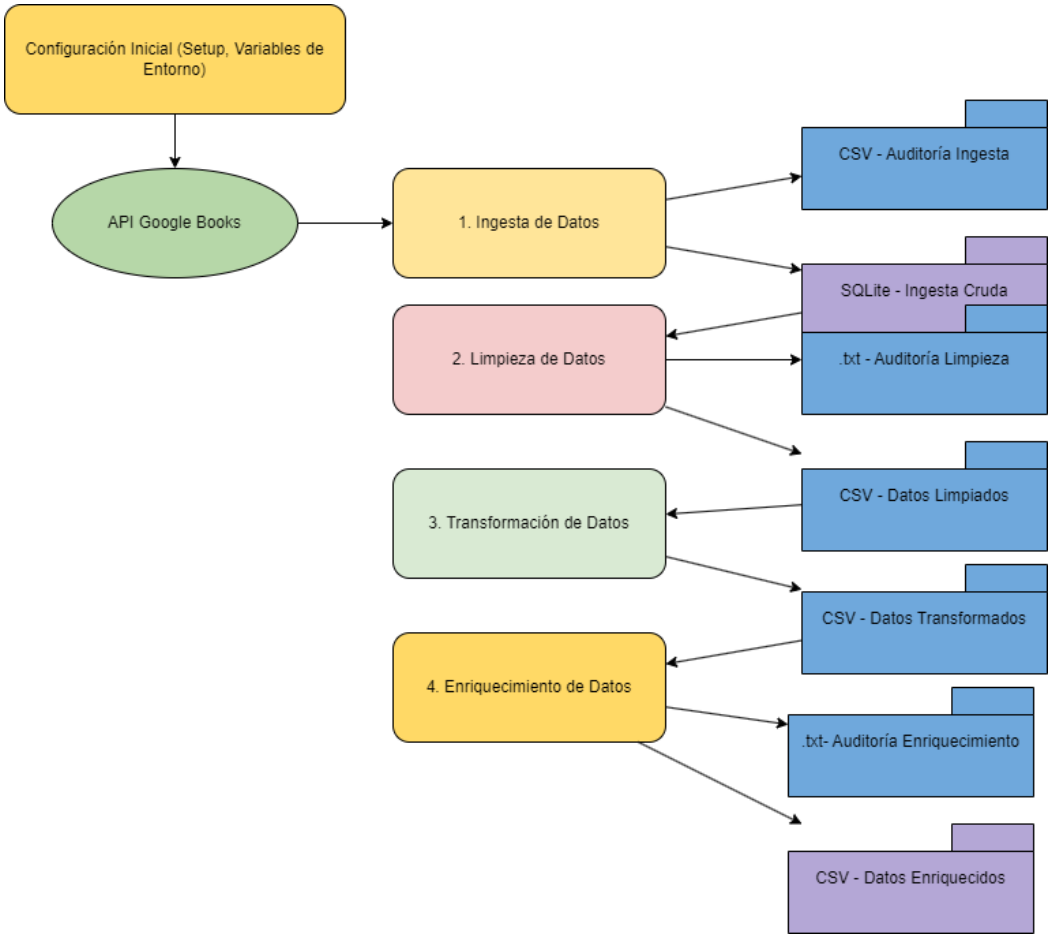

Figura 3. Diagrama de flujo

- **Modelo entidad relación**

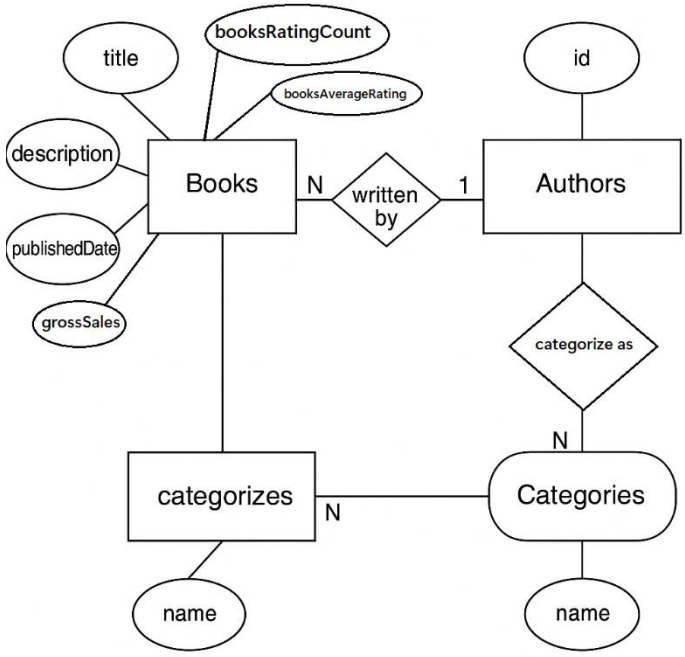

Figura 4. Modelo entidad relación

Modelo de datos

- **Descripción de modelo resultante:** El modelo de datos integra dos fuentes de información. Se tiene el principal que sería la API de Google Books, donde se desarrolla principalmente el proceso de ingesta y de limpieza de los datos. El proceso de enriquecimiento se desarrolla a partir de una vinculación con un dataset de Kaggle a partir de un token con formato.JSON, estos datos integran en este caso a la tabla Books, la cual es la que contiene la información más relevante para sus análisis. No obstante, el modelo se compone por cinco tablas: Books, categories, authors, books_authors y books_categories.
    - **Books:** Contiene los datos principales de los libros, ya sea su título, categoría, descripción, fecha de publicación, las calificaciones generales, las calificaciones promedio y el total de ingresos generados. Los datos numéricos como las

calificaciones generales y promedio; y los ingresos son datos integrados dentro del enriquecimiento de datos.

Campo	Tipo de Dato	Descripción
id	INTEGER (PK)	Identificador único del libro (Proporcionado por la API)
title	TEXT	Título del libro
	TEXT	Descripción del libro
publishedDate	TEXT	Fecha de publicación
clave_compuesta	TEXT	Datos en común con dataset adicional integrado en el enriquecimiento. Contiene el titulo del libro y su fecha
Book_ratings_count	INTEGER	Representa la cantidad total de calificaciones de un libro. Dato integrado en la transformación
Book_average_rating	REAL	Representa el promedio de calificaciones de cada libro. Dato integrado en la transformación
gross sales	REAL	Representa el total de ingresos generado por los libros en cuanto a ventas. Dato integrado en la transformación

Figura 5. Datos tabla Books

- **Authors:** Contiene los datos referenciales a los autores de los libros con su respectivo código de identificación, el cual es único para cada uno y puede variar según el libro solicitado a la API y el nombre del autor.

Campo	Tipo de Dato	Descripción
id	INTEGER (PK)	Identificador único del autor (Proporcionado por la API)
name	TEXT UNIQUE NOT NULL	Representa el nombre de la categoría de cada libro.

Figura 6. Datos tabla Authors

- **Categories:** Contiene los datos categóricos de los libros a partir del género que representan, estos datos pueden variar según el tipo de libro que se solicite a la API.

Campo	Tipo de Dato	Descripción
id	INTEGER (PK)	Identificador único de la categoria (Proporcionado por la API)
name	TEXT UNIQUE NOT NULL	Representa el nombre de la categoria a la que pertenece cada libro

Figura 7. Datos tabla Categories

- **Books_Authors:** Formula la relación entre la tabla Books con Authors,

Campo	Tipo de Dato	Descripción
book_id	INTEGER (FK)	Identificador único del libro (Proporcionado por la API)
author_id	INTEGER (FK)	Identificador único del autor (Proporcionado por la API)

englobando los libros con sus respectivos autores a través de sus id.

Figura 8. Datos tabla Books_Authors

- **Book_categories:** Formula la relación entre la tabla Books con Categories, englobando los libros con sus respectivas categorías o géneros a través de sus id.

Campo	Tipo de Dato	Descripción
book_id	INTEGER (FK)	Identificador único del libro (Proporcionado por la API)
categorie_id	INTEGER (FK)	Identificador único de la categoria (Proporcionado por la API)

Figura 9. Datos tabla Books_Categories

Justificación:

El modelo de datos fue diseñado con el objetivo de facilitar una estructura relacional clara, normalizada y optimizada para la integración, transformación y análisis de información relacionada con libros, autores y categorías. Esta estructura nos permite mantener la integridad referencial y garantiza una organización eficiente de los datos provenientes de múltiples fuentes, incluyendo una API y datasets complementarios.

Se definieron entidades principales como books, authors y categories, cada una con identificadores únicos y atributos relevantes, lo cual permite una representación precisa y extensible de cada elemento del dominio. Tambien, se incluyeron las tablas puente book_author y book_category. Para modelar las relaciones muchos a muchos, asegurando así la flexibilidad en los vínculos entre libros, autores y categorías. Además, el modelo incorpora campos derivados del proceso de enriquecimiento y transformación de datos, como las calificaciones book_ratings_count, book_average_rating y las ventas, que son clave para los análisis posteriores. La inclusión del campo clave_compuesta, que contiene una combinación de título y fecha, permite una integración efectiva con datasets externos.

En conjunto, este modelo se justifica porque facilita los siguientes cuatro puntos:

- **La integración de datos heterogéneos** provenientes de diversas fuentes.
- **La trazabilidad de relaciones complejas** mediante claves foráneas.
- **La optimización de consultas analíticas**, al contar con métricas directamente almacenadas y normalizadas.
- **La escalabilidad del sistema**, permitiendo incorporar nuevas entidades o atributos sin comprometer la integridad del diseño.

Este enfoque sienta una base sólida para las siguientes fases del proyecto, tales como el análisis exploratorio, visualización de datos y automatización que veremos más adelante.

Justificación de herramientas y tecnologías

- **SQLite3:** Fue seleccionada debido a su ligereza y compatibilidad con Python, además de funcionar como una herramienta bastante accesible, ni necesidad de hacer uno de herramientas intermediaras para hacer efectiva su conexión con los datos. En cuanto a escalabilidad, permite prototipar modelos de datos de una manera mucho más eficiente.

Funciona como una buena alternativa para proyectos no tan robustos y facilita la replicación, copia y migración de datos.

- **Pandas:** Al ser una librería que desarrolla de forma eficiente el análisis y la manipulación de datos, fue elegida mayormente por su integración nativa con estructuras de tipo dataframe, la cual es ideal para las etapas de transformación y limpieza. Permitiendo una mejor documentación y descripción de los datos al momento de analizarlos y ejercer ciertas funciones con ellos.
- **GitHub Actions:** Sirve como una herramienta de automatización de procesos a través de flujos de trabajo a partir de pequeñas tareas que va ejerciendo a través de pequeños pasos. Este procedimiento garantiza la trazabilidad del proyecto, así también como el control de versiones y mantiene la consistencia de los procesos automatizados. Además de que puede estar en una constante actualización en cuanto a los pasos, esto en función al crecimiento del proyecto.

Simulación de entorno cloud

Pese a que este proyecto no tuvo la oportunidad de desplegarse en una plataforma en la nuble como AWS o Azure, tiene la habilidad de simular un entorno en la nube bastante parecido de manera local, utilizando herramientas que replican comportamientos esenciales en los entornos distribuidos y automatizados caracterizados del cómputo en la nube. Estos elementos toman las siguientes funciones:

SQLite3: Siendo una base de datos local, toma el rol de simular un almacenamiento estructurado de datos que, en un entorno de la nube real, serían Amazon RDS o Google Cloud SQL.

Scripts de extracción y transformación (ETL): Simulan un procesamiento de ingesta, transformación y enriquecimiento a través de Python y Pandas, las cuales son las fases esenciales dentro de una arquitectura en la nube. Python y Pandas simulan el trabajo que en un contexto real lo harían Google Dataflow o Apache Spark.

GitHub Actions como Orquestador Automatizado: Se emplea para simular un orquestador tipo Airflow o Cloud Functions que ejecuta flujos automáticos cuando se detectan cambios en el repositorio. Esto garantiza una ejecución completamente autónoma y repetible, como lo haría un pipeline en la nube.

Auditorías: Emulan la trazabilidad y monitoreo propios de las plataformas cloud mediante dashboards o logs almacenados en buckets o servicios de monitoreo.

Flujo de datos y Automatización.

Tomando como base el flujo de datos desde su ingesta hasta su transformación a través de su automatización por Github Actions se presenta la siguiente explicación detallada.

1. Ingesta de Datos.

Fuentes de Datos:

API de Google Books: Se consulta mediante llamadas automatizadas.

Archivos CSV/Excel locales: Generados o descargados como soporte al dataset principal.

Datos complementarios de Kaggle: Aportan información adicional para el modelado.

Automatización - Paso5 del pipeline:

Se ejecuta el script ingesta.py:

1. Recupera los libros usando la API de Google Books.
2. Inserta los datos crudos en una base de datos SQLite (almacenamiento intermedio).
3. Exporta la información a formatos CSV y Excel.

Artefactos generados:

1. libros.sqlite
2. books.csv y books.xlsx.

3. auditoria_ingesta.txt

2. Limpieza de Datos

Objetivo con los datos:

Eliminar inconsistencias y garantizar integridad estructural.

Automatización - Paso7 del pipeline:

Se ejecuta un script que elimina duplicados, nulos, columnas innecesarias. Estandariza formatos de fecha, texto, identificadores.

Divide los datos en archivos intermedios:

1. books.csv
2. authors.csv
3. categories.csv

Relaciones: books_authors.csv y books_categories.csv.

Artefactos generados:

1. Archivos .csv intermedios.

2. auditoria_limpieza.txt

3. Transformación de Datos

Integrar datos externos desde Kaggle, para construir relaciones, crear estructuras compuestas y enriquecer las entidades.

Automatización - Paso9 del pipeline:

Se hace la unión de datasets externos con los previamente limpiados, se crean claves compuestas Y se añaden nuevas columnas calculadas: book_ratings_count, book_average_rating y gross_sales.

Los datos finales son consolidados en transformed_data.csv.

Artefactos generados:

1. transformed_data.csv

2. auditoria_transformacion.txt

4. Enriquecimiento de Datos

Incorporación de atributos adicionales y preparar los datos para visualización, modelado o análisis. Se realiza el join final entre los libros y sus relaciones, para incluir nombres de autores, categorías en texto, popularidad, etc.

Artefactos generados:

1. CSV con columnas enriquecidas para modelado.
2. Auditoría final.

Auditoría del Proceso

En cada etapa se generan archivos .txt con descripciones detalladas del proceso ejecutado:

1. Número de registros leídos/escritos.
2. Errores o registros nulos detectados.
3. Estadísticas del procesamiento.

Conclusiones y Recomendaciones.

Este proyecto logró simular exitosamente un entorno de computación en la nube desde un entorno local. A través de herramientas como SQLite3, scripts en Python y Pandas, GitHub Actions y procesos de auditoría, se replicaron funciones clave de servicios como Amazon RDS y Apache Spark.

Los principales beneficios identificados de la arquitectura propuesta incluyen:

Portabilidad y simplicidad: El entorno local es de fácil despliegue, bajo costo y accesible para pruebas, lo cual resulta ideal en fases iniciales o educativas del desarrollo.

Automatización eficiente: Gracias a GitHub Actions, se logra una orquestación automática que simula la ejecución de pipelines reales, garantizando trazabilidad y reproducibilidad.

Escalabilidad conceptual: La estructura modular del proyecto permite una migración futura a servicios reales en la nube sin necesidad de rediseñar completamente la arquitectura.

Simulación realista de procesos ETL: La lógica de procesamiento de datos se implementa con herramientas ampliamente utilizadas en la industria, facilitando su futura integración con soluciones en la nube.

Sin embargo, también se identifican ciertas limitaciones:

Falta de escalabilidad real: Aunque se simula el comportamiento de la nube, el entorno local no puede replicar la capacidad de procesamiento, tolerancia a fallos y escalamiento horizontal que ofrece una plataforma como AWS o GCP.

Seguridad limitada: La gestión de credenciales, políticas de acceso y cifrado no alcanza los niveles robustos que ofrecen los servicios cloud.

Monitoreo simplificado: Las auditorías implementadas ofrecen trazabilidad básica, pero no incluyen alertas, métricas o dashboards integrados como lo haría un Stackdriver o CloudWatch.

Recomendaciones.

- Migrar gradualmente los componentes a servicios cloud reales.
- Usar contenedores como Docker para facilitar despliegues futuros.
- Mejorar la trazabilidad y seguridad con herramientas especializadas.
- Estimar costos para evaluar la viabilidad de operar en la nube.