Evidencia De Aprendizaje 4

Documentación de la Arquitectura y Modelo de Datos

Adriana María Aguilar Viloria
Edwin BernardoVilla Sánchez
Nikol Tamayo Rúa

Docente:

Andrés Callejas

Curso:

Infraestructura y Arquitectura para Big Data

Programa Ingeniería de Software y Datos
Facultad Ingenierías y Ciencias Agropecuarias
Institución Universitaria Digital de Antioquia
2025

Tabla de Contenido

Introducción

El procesamiento de datos en entornos de Big Data requiere una serie de pasos estructurados para garantizar la calidad, coherencia y utilidad de la información. Este proyecto integrador simula un flujo de trabajo completo en un entorno de datos a gran escala, abarcando desde la ingesta de información hasta su limpieza y enriquecimiento.

El proyecto se estructura en tres etapas fundamentales:

1. **Ingesta de datos**: Se extraen datos desde una API pública y se almacenan en una base de datos local SQLite simulando un entorno en la nube.
2. **Preprocesamiento y limpieza**: Se normalizan, transforman y depuran los datos para eliminar inconsistencias y valores atípicos.
3. **Enriquecimiento de datos**: Se integran fuente adicional en formato CSV, para complementar la información base y generar un dataset más robusto.

Cada una de estas etapas ha sido desarrollada con herramientas como Python, Pandas y SQLite, y se ha automatizado el flujo de procesamiento mediante GitHub Actions para asegurar su ejecución sistemática y reproducible. Este documento detalla la arquitectura implementada, los componentes del sistema, el modelo de datos resultante y las herramientas utilizadas, proporcionando una visión integral del proyecto.

Objetivo

Documentar y explicar de forma detallada la arquitectura del proyecto integrador de Big Data, integrando la descripción de las fases (ingesta, preprocesamiento, enriquecimiento) en un entorno simulado de nube, y definir el modelo de datos resultante.

Descripción General de la Arquitectura

Visión Global

El proyecto sigue un enfoque estructurado para el procesamiento de datos en un entorno simulado de Big Data en la nube. Se han implementado tres fases clave:

1. **Ingesta de Datos**: La información se obtiene desde una API (*https://api.sampleapis.com/switch/games*) y se almacena en una base de datos SQLite (*ingestion.db*), y en formato XLSX (*ingestión.xlsx*) para facilitar su procesamiento posterior.
2. **Preprocesamiento y Limpieza**: Se eliminan valores nulos, duplicados, se cambia el tipo de dato de la fecha y se estructura la información en un dataset limpio (*cleaned_data.csv*). Además, se genera un reporte detallando los cambios aplicados (*cleaning_report.txt*).
3. **Enriquecimiento de Datos**: Se integran fuentes adicionales (*additional_info.csv*) para complementar los datos originales. Se realiza un cruce de información utilizando claves comunes y se genera un dataset enriquecido (*enriched_data.csv*), junto con un reporte de auditoría sobre la integración realizada (*enriched_report.csv*).

Cada etapa del proceso genera archivos de auditoría que documentan los cambios y mejoras aplicadas a los datos, asegurando transparencia y trazabilidad en todo el flujo de trabajo. Adicionalmente, se ejecutan de manera automatizada, garantizando la actualización.

Componentes Principales

1. **Base de Datos Analítica (SQLite):**
   - Es el almacenamiento central donde se guarda la información obtenida desde la API.
   - Se estructura en tablas relacionales optimizadas para facilitar consultas y análisis posteriores.

2. **Mecanismo de Auditoría:**

- Cada fase genera un reporte en *static/auditoria/*, documentando el estado de los datos antes y después de cada transformación:
  - ingestion.txt: Contiene detalles de la obtención de datos desde la API.
  - cleaning_report.txt: Registra las operaciones de limpieza realizadas.
  - enrichment_report.txt: Resume la integración con datos adicionales.

3. **Mecanismo de Automatización (GitHub Actions):**

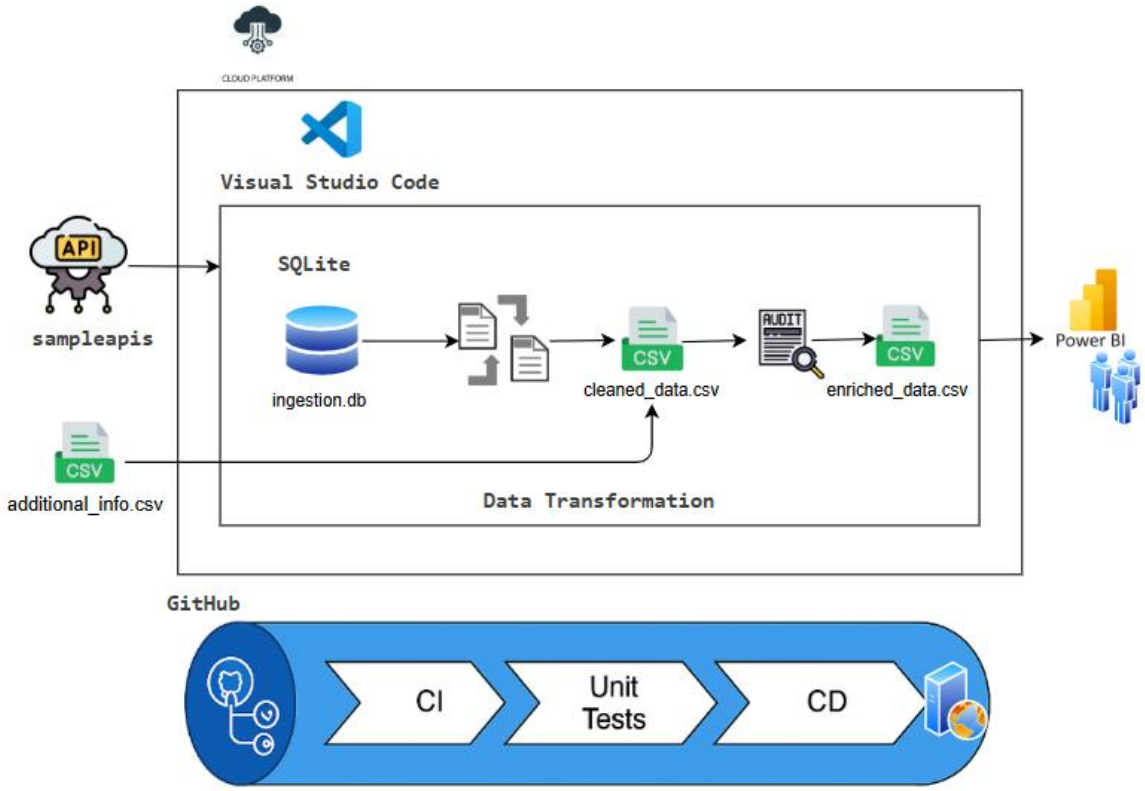
• Se ha configurado un workflow en GitHub Actions (*test_proyecto.yml*) que permite ejecutar de forma automatizada los scripts de ingesta, limpieza y enriquecimiento de datos.

• Esta automatización garantiza que los datos estén actualizados y procesados sin intervención manual, mejorando la eficiencia del sistema.

Estructura del Proyecto

```
[proyecto_integrador_edwin_sanchez_nikol_tamayo_adriana_aguilar]
├── .github
│   └── workflows
│       └── test_proyecto.yml  # Workflow para la ingesta, limpieza, enriquecimiento y auditoría
├── src
│   └── bigdata
│       ├── static
│       │   ├── auditoria
│       │   │   ├── cleaning_report.txt      # Reporte de limpieza
│       │   │   ├── exploratory_analysis.txt # Análisis exploratorio
│       │   │   ├── ingestion.txt            # Registro de ingesta
│       │   │   └── enrichment_report.txt    # Reporte de auditoría del enriquecimiento
│       │   ├── csv
│       │   │   ├── cleaned_data.csv         # Datos limpios
│       │   │   ├── dirty_data.csv           # Datos originales
│       │   │   ├── additional_info.csv      # Datos adicionales para enriquecer
│       │   │   └── enriched_data.csv        # Dataset final enriquecido
│       │   ├── db
│       │   │   └── ingestion.db             # Base de datos SQLite con los datos procesados
│       │   └── xlsx
│       │       └── ingestion.xlsx           # Datos en formato Excel
│       ├── cleaning.py                      # Script de limpieza de datos
│       ├── enrichment.py                    # Script de enriquecimiento de datos
│       └── ingestion.py                     # Script de ingesta de datos
├── docs
│   ├── arquitectura.drawio                  # Diagrama de arquitectura del sistema
│   ├── modelo_datos.drawio                  # Diagrama del modelo de datos
├── .gitignore
├── README.md
└── setup.py
```

Diagrama de Arquitectura



Modelo de Datos

Definición del Esquema

El modelo de datos resultante del proceso de integración está compuesto por dos fuentes principales:

1. **cleaned_data.csv (Datos principales desde la API)**
    - Contiene la información base sobre los videojuegos obtenida desde la API https://api.sampleapis.com/switch/games.
    - Se aplicaron transformaciones como limpieza de datos y conversión de tipos de datos (por ejemplo, fecha_lanzamiento a datetime).
2. **additional_info.csv (Datos adicionales)**
    - Proporciona información complementaria sobre los videojuegos, como la plataforma en la que están disponibles, la calificación y el tamaño en GB.
    - Se une con cleaned_data.csv a través del campo id.
3. **enriched_data**: Resultado de la integración entre las dos tablas anteriores.

Transformaciones aplicadas:

- Unión de datasets por columna 'id'.
- Integración de columnas: plataforma, calificacion, tamaño_gb

Tabla 1 cleaned_data

Metadata Información videojuegos.

Atributo	Descripción	Tipo dato
id	Identificador único del videojuego.	Int(PK)
nombre	Nombre del videojuego.	string
Genero	Género del videojuego.	string
desarrolladores	Nombre del estudio o empresa desarrolladora.	string
publicadores	Empresa responsable de la publicación del juego.	string
fecha_lanzamiento	Fecha en la que el juego fue lanzado.	datetime

Nota. Esta tabla representa la metadata básica de los videojuegos.

Tabla 2 additional_info

Metadata Información adicional de los videojuegos.

Atributo	Descripción	Tipo dato
id	Identificador del videojuego (relación con cleaned_data.id)	Int(PK)
plataforma	Plataforma en la que está disponible el videojuego.	string
calificacion	Calificación promedio del	float

Atributo	Descripción	Tipo dato
	videojuego.	
Tamaño_gb	Tamaño del juego en GB	float

Nota. Esta tabla representa la metadata con información adicional de los videojuegos.

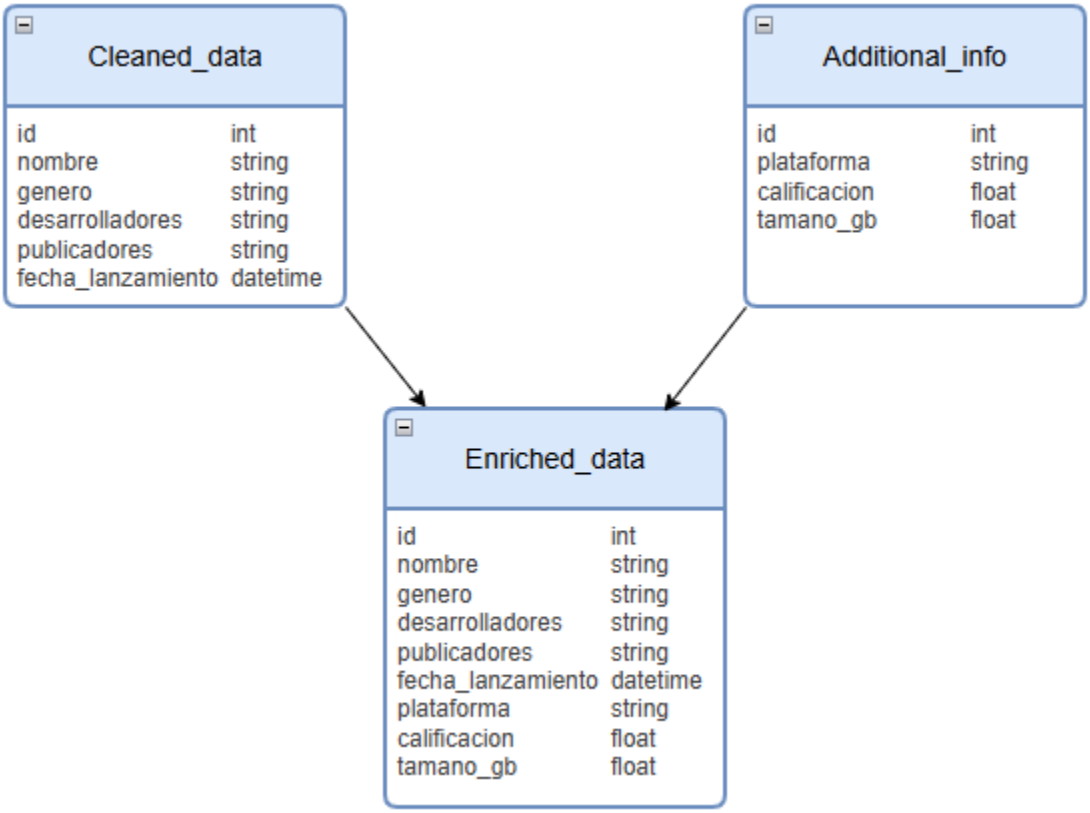
Tabla 3 enriched_data

Unión de las tablas cleaned_data y additional_info

Atributo	Descripción	Tipo dato
id	Identificador único del videojuego.	Int(PK)
nombre	Nombre del videojuego.	string
Genero	Género del videojuego.	string
desarrolladores	Nombre del estudio o empresa desarrolladora.	string
publicadores	Empresa responsable de la publicación del juego.	string
fecha_lanzamiento	Fecha en la que el juego fue lanzado.	Datetime
plataforma	Plataforma en la que está disponible el videojuego.	string
calificacion	Calificación promedio del	float
Tamaño_gb	Tamaño del juego en GB	float

Nota. Esta tabla representa la unión de las dos tablas anteriores

Diagrama de Datos



Justificación del Modelo de Datos

El diseño de este modelo se basa en la necesidad de integrar datos limpios con información adicional y mantener un historial de auditoría del proceso de enriquecimiento.

1. **Estructura clara**: Separa los datos base, los datos adicionales y la versión enriquecida.

   - La tabla cleaned_data → almacena los datos limpios obtenidos tras el preprocesamiento.
   - La tabla additional_info → enriquece los datos con información complementaria, relacionada a través del id.
   - La tabla enrichment_data → es la tabla final con la unión de las dos tablas anteriores.

- La tabla enrichment _report → guarda el registro de auditoría con información sobre las transformaciones realizadas.

2. **Optimización para Consultas y Análisis**:
   - Se mantiene una estructura clara donde los datos limpios son la base del análisis.
   - La información adicional se vincula para proporcionar detalles adicionales sin modificar la estructura base.
   - La auditoría asegura trazabilidad y permite evaluar el impacto de las transformaciones aplicadas.
3. **Trazabilidad**: La tabla de auditoría permite rastrear el impacto del enriquecimiento y evaluar la calidad de los datos fusionados.

- El modelo permite agregar nuevas fuentes de enriquecimiento sin alterar la estructura base.

Justificación de Herramientas y Tecnologías

Elección de Herramientas:

Para el desarrollo del proyecto, se han seleccionado herramientas clave que permiten la correcta integración, procesamiento y análisis de datos. A continuación, se justifica la selección de cada una:

1. **Visual Studio Code (VS Code):**

- Utilizado como entorno de desarrollo principal para escribir, ejecutar y depurar los scripts de integración de datos.
- Permite la ejecución de consultas SQLite y manipulación de datos con Pandas.

2. **SQLite**

- **Justificación:** SQLite es una base de datos ligera y de fácil implementación, ideal para el almacenamiento y gestión de datos sin requerir un servidor adicional.
- **Contribución:** Facilita la persistencia de los datos intermedios y finales en un formato estructurado, optimizando el acceso y la manipulación de información durante el procesamiento.

3. **Pandas**

- **Justificación:** Pandas es una biblioteca de Python especializada en el manejo y análisis de datos estructurados.
- **Contribución:** Permite la manipulación eficiente de los datasets, incluyendo la limpieza, transformación y enriquecimiento de datos, asegurando calidad y consistencia en los resultados.

4. **GitHub Actions**

- **Justificación:** GitHub Actions permite la automatización de flujos de trabajo, incluyendo integración y despliegue continuo (CI/CD).
- **Contribución:** Facilita la ejecución automatizada de los scripts de procesamiento y transformación de datos, asegurando que cada cambio en el código se valide de manera eficiente.

5. **API externa (SampleAPIs):**

- Fuente de datos que proporciona información cruda, la cual es almacenada en SQLite y procesada posteriormente.

Cómo cada herramienta contribuye a la escalabilidad, eficiencia y mantenibilidad del proyecto:

- **Eficiencia:** SQLite permite consultas rápidas y almacenamiento local sin configuración adicional. Pandas optimiza la manipulación de datos en memoria.
- **Escalabilidad:** La estructura modular con GitHub permite ampliar el flujo de datos y su procesamiento sin afectar el código base.
- **Mantenibilidad:** La integración con GitHub Actions permite automatizar la ejecución de tareas, reduciendo errores manuales y asegurando la consistencia de los datos.

Simulación del Entorno Cloud:

El entorno cloud es simulado a través de la combinación de las herramientas seleccionadas y los datos generados en las actividades previas:

1. **Base de datos SQLite:**

- Actúa como repositorio local de datos, similar a una base de datos en la nube.
- Se almacena y consulta en Visual Studio Code, emulando un servicio de almacenamiento gestionado.

2. **Automatización con GitHub Actions:**

- Simula un pipeline de procesamiento en la nube ejecutando los scripts automáticamente.
- Cada vez que se actualiza el código en el repositorio, se activa el workflow que ejecuta la transformación de datos.

3. **Uso de Archivos CSV como Inputs/Outputs:**

- Los archivos se procesan localmente, pero su flujo imita el ingreso y salida de datos en un entorno de Big Data.
- Se usan como insumos de entrada y se generan datasets enriquecidos, como en un pipeline de datos en la nube.

El proyecto está diseñado para ser portable a un entorno cloud real sin modificaciones significativas. El uso de SQLite, GitHub Actions y Pandas permite manejar datos de forma eficiente y automatizada, con un flujo de trabajo estructurado y replicable en cualquier infraestructura cloud.

Flujo de Datos y Automatización

Explicación del Flujo

El proceso de integración de datos sigue una serie de etapas bien definidas, asegurando la transformación, almacenamiento y enriquecimiento de los datos antes de su análisis final.

1. **Ingesta de Datos:**

   Fuente: https://api.sampleapis.com/switch/games
   Tecnologías: Python, Pandas, SQLite

   - Se consume información desde una API externa (SampleAPIs) que proporciona datos en formato JSON.
   - Se almacena en una base de datos SQLite (ingestión.db) con una tabla videojuegos y se guarda un archivo en Excel con la información (ingestión.xlsx)
   - Se crea auditoría (ingestión.txt) para comparar los datos obtenidos del API con los registros almacenados en la base de datos.
   - Adicionalmente, se carga un archivo CSV (additional_info.csv) que contiene información complementaria.

2. **Almacenamiento y Transformación:**

   - Los datos de la API se transforman en estructuras tabulares con Pandas, asegurando la correcta tipificación de los datos.
   - Se realiza la limpieza de los datos tipificando los datos correctamente (por ejemplo, fecha_lanzamiento se convierte a datetime), se eliminan valores nulos, duplicados o inconsistentes y se guarda en cleaned_data.csv
   - Se crea archivo de auditoría (cleaning_report.txt) para registrar la información antes y después de la limpieza.

3. **Enriquecimiento de Datos:**

   - Se cruzan los datos de cleaned_data.csv con los de additional_info.csv, utilizando la columna ‘id’ como clave de unión.

- Se integran plataforma, calificacion y tamaño_gb a la tabla original.
- Se genera la tabla final enriched_data.csv que contiene la información consolidada.
- Se crea archivo de auditoría (enrichment_report.txt) con los registros originales y enriquecidos

4. **Generación del Reporte de Auditoría:**

   Cada una de las etapas anteriores registra su propio informe de auditoría, asegurando la trazabilidad de los datos en todo el flujo:

   - **ingestion.txt** → Auditoría de la ingesta.
   - **cleaning_report.txt** → Auditoría de la limpieza.
   - **enrichment_report.txt** → Auditoría del enriquecimiento.

5. **Automatización con GitHub Actions:**

   - Cada vez que se actualiza el código en el repositorio, GitHub Actions ejecuta un pipeline que:
     - Descarga los datos de la API.
     - Procesa y transforma los datos.
     - Genera los archivos finales y el reporte de auditoría.

6. **Salida y Consumo de Datos:**

   - La tabla enriched_data puede ser exportada a herramientas como Power BI para su análisis y visualización.

Conclusiones

La arquitectura implementada sigue un flujo claro y ordenado desde la ingesta hasta el enriquecimiento de datos, asegurando que cada paso sea reproducible y estandarizado.

Automatización y Reproducibilidad: El uso de GitHub Actions permite una ejecución automatizada del flujo de trabajo, asegurando que cada actualización en el código genere nuevos datasets sin intervención manual.

Trazabilidad y Control de Calidad: Cada etapa del proceso genera un reporte de auditoría, lo que facilita la supervisión del flujo de datos **y** permite detectar errores o inconsistencias a tiempo.

Escalabilidad y Flexibilidad: El uso de SQLite como base de datos local permite la manipulación eficiente de datos.

Integración con Herramientas de Análisis: Los datos finales pueden ser utilizados en **Power BI** u otras plataformas de visualización para generar análisis avanzados y reportes interactivos.

Simplicidad y Eficiencia: Se emplearon herramientas ampliamente conocidas como Pandas y Visual Studio Code, lo que hace que el flujo de trabajo sea comprensible y fácil de mantener.

Auditoría en Cada Etapa: El sistema registra informes de auditoría en cada fase del pipeline (ingesta, limpieza y enriquecimiento), lo que permite identificar problemas y asegurar la calidad de los datos en todo momento.

Recomendaciones

Migración a una Base de Datos Escalable: SQLite es una solución ligera y efectiva, pero para entornos con mayor volumen de datos, se recomienda migrar a una base de datos escalable como PostgreSQL o MySQL.

Implementación en un Entorno Cloud Real: Actualmente, la arquitectura se ejecuta en un entorno local simulado. Para mayor disponibilidad y rendimiento, se podría implementar en plataformas como AWS, Azure o Google Cloud, utilizando servicios como Amazon RDS o BigQuery.

Automatización con Pipelines de Datos Avanzados: Se podría mejorar el pipeline actual utilizando herramientas más especializadas como Apache Airflow para programar y monitorear las tareas de ingesta, limpieza y enriquecimiento.

Mejoras en la Calidad de los Datos: Se recomienda agregar validaciones más estrictas durante la limpieza de datos, detectando valores atípicos o inconsistencias antes de la integración en la base de datos final.

Optimización del Proceso de Enriquecimiento: Se podría ampliar el enriquecimiento con más fuentes de datos externas, como APIs adicionales o datasets públicos, para proporcionar una visión más completa.

Manejo de Versionamiento de Datos: Se podría incluir un mecanismo de versionamiento de datos en la base de datos, para llevar un control sobre cambios y actualizaciones en cada iteración del pipeline.

Mejor Documentación y Testing del Código: El código puede beneficiarse de mayor documentación y la implementación de pruebas automatizadas para asegurar que cada etapa funcione correctamente antes de ejecutarse en producción.

Bibliografía

OpenAI. (2024). *ChatGPT: Language model documentation*. Recuperado el 3 de abril de 2025, de https://openai.com/research/

SampleAPIs. (2024). *Nintendo Switch Games API*. Recuperado el 3 de abril de 2025, de https://api.sampleapis.com/switch/games

GitHub. (2024). *GitHub Actions Documentation*. Recuperado el 3 de abril de 2025, de https://docs.github.com/en/actions