F25 - EA4. Documentación de la Arquitectura y Modelo de Datos

Infraestructura y arquitectura para Big Data - PREICA2501B010112

Programa

Ingeniería de Software y Datos

Profesor

ANDRES CALLEJAS

Estudiante

Miguel ángel Leal Beltrán

Universidad

IUDigital de Antioquia

Bogotá 2025

Contenido

Objetivo 3
1. Introducción y Descripción Global de la Arquitectura 3
2. Diagramas de Arquitectura 4
3. Modelo de Datos 6
4. Justificación de Herramientas y Tecnologías 7
5. Flujo de Datos y Automatización 10
6. Conclusiones y Recomendaciones 12
7. Referencias 15

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. Se espera que el documento evidencie la estructura, los flujos de datos, la selección de herramientas, la automatización y el modelo de datos (esquema, relaciones y estructura de la base de datos).

1. Introducción y Descripción Global de la Arquitectura

1.1 Visión Global

El presente proyecto implementa una solución de Big Data en un entorno simulado de nube, que comprende tres fases principales interconectadas: ingesta de datos, preprocesamiento y enriquecimiento. La arquitectura está diseñada para gestionar datos de anime provenientes de la API Jikan, transformarlos y enriquecerlos con información complementaria de diversas fuentes y formatos.

La solución implementa un flujo de trabajo completo que simula las operaciones típicas de ETL (Extracción, Transformación y Carga) en un entorno de Big Data, pero utilizando herramientas y tecnologías accesibles que permiten comprender los conceptos fundamentales sin la necesidad de una infraestructura cloud real.

1.2 Componentes Principales

La arquitectura consta de los siguientes componentes críticos:

Base de Datos Analítica (SQLite):

Almacena los datos originales de la API (anime_data.db)

Persiste los datos limpios (cleaned_data.csv/xlsx)

Aloja el dataset enriquecido final (enriched_data.csv/xlsx)

Scripts de Procesamiento:

Ingesta (ingesta.py): Extrae datos de la API Jikan y los almacena en SQLite

Limpieza (cleaning.py): Procesa los datos crudos, corrige errores y estandariza formatos

Enriquecimiento (transformation.py): Integra datos de múltiples fuentes para crear un dataset enriquecido

Mecanismo de Automatización:

GitHub Actions: Permite la ejecución automatizada del flujo completo mediante workflows definidos

Asegura la repetibilidad y consistencia del proceso

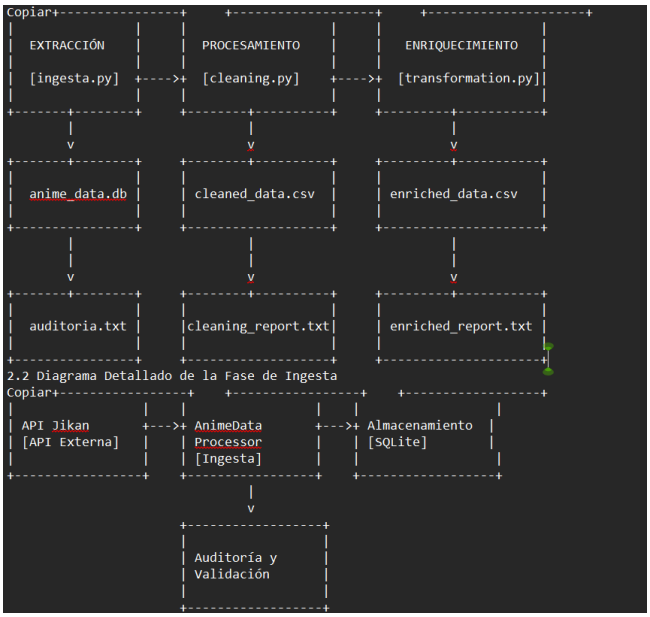
Evidencias y Auditoría:

Archivos de registro que documentan cada etapa del proceso

Informes de validación de datos y métricas de calidad

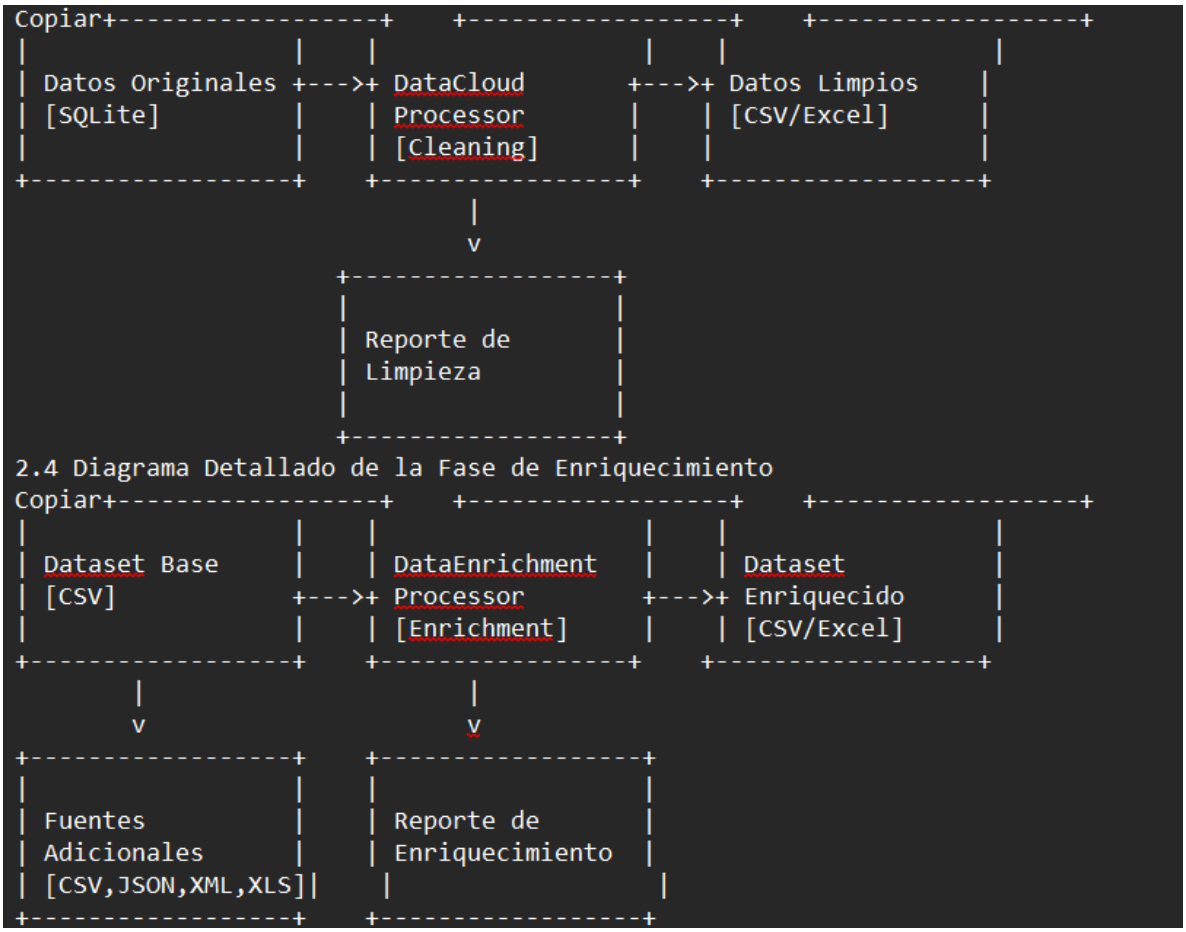
2. Diagramas de Arquitectura

2.1 Diagrama de Flujo General

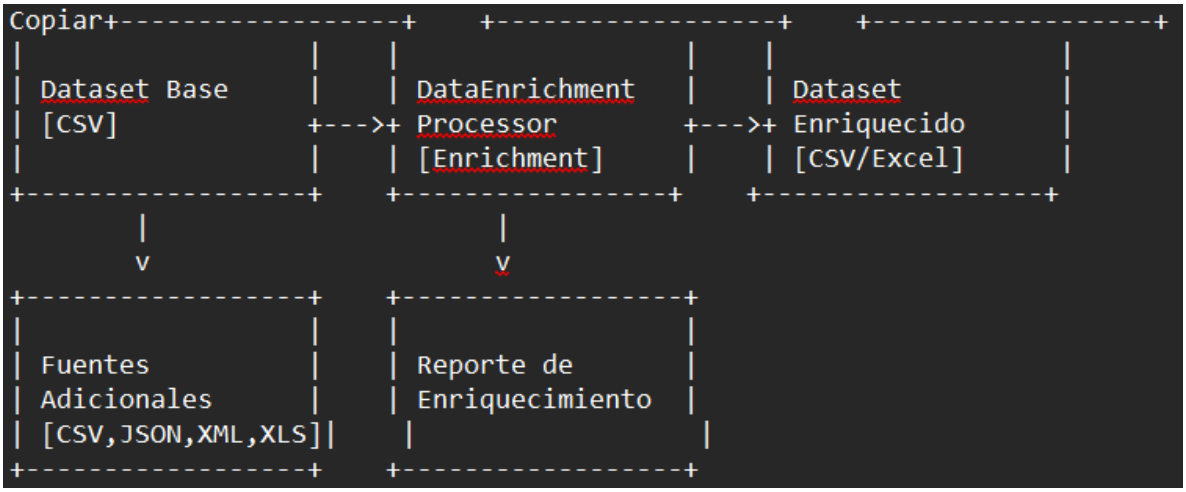


2.2 Diagrama Detallado de la Fase de Ingesta

2.3 Diagrama Detallado de la Fase de Limpieza



2.4 Diagrama Detallado de la Fase de Enriquecimiento



3. Modelo de Datos

3.1 Definición del Esquema

3.1.1 Modelo de Datos Original (Después de la Ingesta)

Tabla: personajes

CampoTipoDescripciónidINTEGERIdentificador único del personaje (PK)nombreTEXTNombre del personajenombre_kanjiTEXTNombre del personaje en kanji (japonés)roleTEXTRol del personaje (Protagonista, Antagonista, etc.)anime_idINTEGERID del anime al que perteneceimagen_urlTEXTURL de la imagen del personajefecha_extraccionTEXTFecha y hora de extracción de datos

Tabla: voice_actors

CampoTipoDescripciónidINTEGERIdentificador único autoincrementable (PK)person_idINTEGERID de la persona (actor de voz)person_nameTEXTNombre del actor de vozlanguageTEXTIdioma de la actuación de vozcharacter_idINTEGERID del personaje (FK referencia a personajes.id)image_urlTEXTURL de la imagen del actorfecha_extraccionTEXTFecha y hora de extracción de datos

3.1.2 Modelo de Datos Limpio (Después del Preprocesamiento)

El modelo de datos después de la limpieza mantiene la estructura original, pero con las siguientes mejoras:

Tipos de datos consistentes (id y anime_id como INTEGER)

Valores nulos tratados (imputados con valores por defecto)

Estandarización de texto (mayúsculas para roles, eliminación de caracteres especiales)

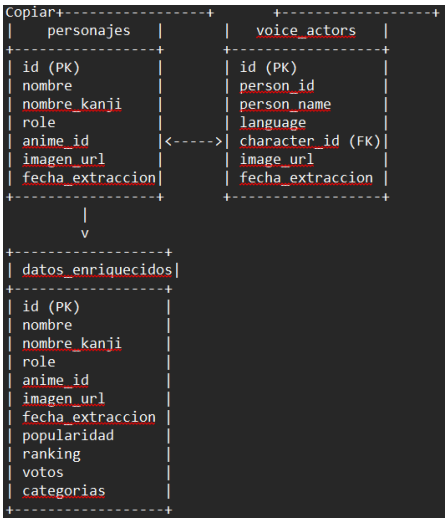
Eliminación de duplicados

3.1.3 Modelo de Datos Enriquecido (Final)

El modelo final añade las siguientes columnas al dataset limpio:

CampoTipoOrigenDescripciónpopularidadINTEGERExcelÍndice de popularidad del personaje (1-100)rankingINTEGERExcelRanking del personaje entre todos los personajesvotosINTEGERExcelNúmero de votos recibidos por el personajecategoriasTEXTXMLCategorías o géneros asociados al personaje/anime

3.2 Diagrama del Modelo de Datos



3.3 Justificación del Modelo de Datos

El modelo de datos fue diseñado siguiendo estos principios:

Normalización controlada: Se utilizó un nivel básico de normalización para separar entidades principales (personajes y actores de voz), pero manteniendo una estructura accesible para análisis posteriores.

Enfoque en entidades clave: El modelo se centra en los personajes como entidad principal, permitiendo expandirse con relaciones a otras entidades como actores de voz.

Flexibilidad para enriquecimiento: La estructura permite fácilmente añadir atributos adicionales durante la fase de enriquecimiento sin comprometer la integridad.

Equilibrio entre normalización y rendimiento analítico: El modelo final (dataset enriquecido) está parcialmente desnormalizado para facilitar consultas analíticas rápidas sin necesidad de múltiples joins.

Este modelo facilita tanto la integración de datos de múltiples fuentes como el análisis posterior, ya que proporciona una visión completa de los personajes de anime con información complementaria que permite realizar análisis multidimensionales.

4. Justificación de Herramientas y Tecnologías

4.1 Elección de Herramientas

SQLite

Justificación: Se seleccionó SQLite como base de datos por su naturaleza embebida, que no requiere configuración de servidor adicional, facilitando la portabilidad del proyecto.

Contribución: Proporciona persistencia de datos estructurados con capacidades SQL completas pero sin la sobrecarga de un DBMS cliente-servidor.

Escalabilidad: Aunque no es la opción más escalable para entornos de Big Data reales, es perfecta para simulación y prototipado, permitiendo entender los conceptos sin infraestructura compleja.

Pandas

Justificación: Pandas ofrece estructuras de datos eficientes y una rica API para manipulación y análisis de datos.

Contribución: Simplifica enormemente tareas complejas como limpieza, transformación y unión de datasets de diferentes fuentes.

Eficiencia: Sus operaciones vectorizadas permiten procesar grandes volúmenes de datos eficientemente en memoria.

NumPy

Justificación: Proporciona el backend computacional para operaciones numéricas rápidas utilizadas por Pandas.

Contribución: Permite cálculos estadísticos eficientes necesarios durante el análisis exploratorio y la transformación de datos.

Python (ecosistema completo)

Justificación: Lenguaje de alto nivel con excelente soporte para ciencia de datos y procesamiento de datos.

Contribución: Facilita la integración entre componentes y permite explotar un vasto ecosistema de bibliotecas de análisis y visualización.

Mantenibilidad: Su sintaxis clara y enfoque en legibilidad mejora la mantenibilidad del código.

GitHub Actions

Justificación: Sistema de CI/CD integrado con GitHub que permite automatizar workflows.

Contribución: Permite la ejecución automática de las tres fases del proyecto (ingesta, limpieza, enriquecimiento).

Reproducibilidad: Garantiza que el proceso sea reproducible y consistente en cada ejecución.

4.2 Simulación del Entorno Cloud

La simulación del entorno cloud se logra mediante:

Abstracción de almacenamiento:

SQLite simula una base de datos en la nube

El sistema de archivos local actúa como un almacenamiento de objetos

Procesamiento por etapas:

Cada script (ingesta.py, cleaning.py, transformation.py) actúa como un servicio independiente

Los datos se pasan entre etapas a través de archivos persistentes (similar a buckets en la nube)

Automatización:

GitHub Actions simula la orquestación de servicios en la nube

Los workflows definidos representan pipelines de datos automatizados

Monitoreo y auditoría:

Los archivos de registro generados simulan los logs de servicios cloud

Los reportes de métricas actúan como dashboards de monitoreo

Esta aproximación permite comprender los conceptos de procesamiento distribuido y pipelines de datos en la nube sin necesidad de infraestructura compleja, facilitando el aprendizaje y la experimentación.

5. Flujo de Datos y Automatización

5.1 Explicación del Flujo de Datos

El flujo de datos a través del sistema sigue las siguientes etapas:

Fase de Ingesta:

Los datos se extraen de la API Jikan mediante peticiones HTTP

Se aplican transformaciones básicas para adaptar el formato

Se introducen errores controlados para simular problemas reales de calidad

Los datos se almacenan en una base de datos SQLite (anime_data.db)

Se genera un archivo de auditoría (auditoria.txt) que documenta el proceso

Fase de Preprocesamiento:

Los datos crudos se cargan desde la base de datos SQLite

Se realiza un análisis exploratorio para identificar problemas de calidad

Se aplican operaciones de limpieza (tratamiento de nulos, corrección de tipos, estandarización)

Se eliminan duplicados y valores atípicos

Los datos limpios se almacenan en archivos CSV y Excel (cleaned_data.csv/xlsx)

Se genera un informe detallado del proceso (cleaning_report.txt)

Fase de Enriquecimiento:

El dataset limpio se carga como base

Se divide en dos partes para simular fuentes distintas (CSV y JSON)

Se generan fuentes adicionales en diferentes formatos (Excel y XML)

Se integran todas las fuentes mediante operaciones de unión (joins) por ID

Los datos enriquecidos se almacenan en archivos CSV y Excel (enriched_data.csv/xlsx)

Se genera un informe de auditoría del proceso (enriched_report.txt)

5.2 Automatización con GitHub Actions

El proceso completo se automatiza mediante un workflow de GitHub Actions definido en .github/workflows/bigdata.yml:

El workflow se activa por:

Push a la rama main

Pull requests a la rama main

Activación manual (dispatch)

El pipeline de ejecución:

Configura un entorno Python 3.9

Instala las dependencias necesarias

Ejecuta el script de enriquecimiento

Verifica que los artefactos esperados se hayan generado correctamente

Almacena los artefactos generados como evidencia

Esta automatización garantiza que:

El proceso se ejecute de manera consistente

Las dependencias estén correctamente instaladas

Los artefactos se generen en las ubicaciones esperadas

Exista evidencia de cada ejecución

6. Conclusiones y Recomendaciones

6.1 Conclusiones

La arquitectura implementada para este proyecto de Big Data logra simular efectivamente un entorno de procesamiento de datos en la nube, consiguiendo:

Flujo completo de ETL: La solución abarca todo el ciclo desde la extracción hasta el enriquecimiento y análisis de datos.

Simulación efectiva: Se logra simular conceptos de Big Data complejos usando herramientas accesibles.

Arquitectura modular: El diseño por módulos independientes permite entender el rol de cada componente y facilita el mantenimiento.

Análisis de calidad: Los procesos incorporan verificaciones y métricas de calidad que aseguran la integridad de los datos.

Automatización: La implementación demuestra cómo la automatización mejora la reproducibilidad y consistencia.

6.2 Limitaciones

La solución también presenta algunas limitaciones inherentes a su naturaleza simulada:

Escalabilidad limitada: SQLite y Pandas no están diseñados para volúmenes realmente grandes de datos.

Procesamiento secuencial: No se implementa procesamiento paralelo o distribuido real.

Persistencia local: El almacenamiento se realiza localmente, sin las ventajas de sistemas distribuidos.

6.3 Recomendaciones para Implementación Real

Para llevar esta arquitectura a un entorno de Big Data real, se recomienda:

Migración a bases de datos distribuidas:

Reemplazar SQLite por soluciones como Apache Cassandra, MongoDB o Amazon DynamoDB

Implementación de procesamiento distribuido:

Incorporar Apache Spark o Hadoop para procesamiento paralelo

Utilizar PySpark en lugar de Pandas para análisis escalable

Adopción de servicios cloud reales:

Migrar a servicios como AWS S3 para almacenamiento

Utilizar AWS Glue, Azure Data Factory o Google Dataflow para ETL

Implementar Amazon EMR o Databricks para procesamiento

Mejora de la orquestación:

Reemplazar GitHub Actions por Apache Airflow o Oozie para orquestación más robusta

Implementar monitoreo con servicios como Datadog o Prometheus

Implementación de streaming:

Incorporar procesamiento en tiempo real con Kafka o Kinesis

Desarrollar análisis de streaming con Spark Streaming o Flink

Estas mejoras permitirían escalar la solución para manejar volúmenes reales de Big Data manteniendo la arquitectura conceptual desarrollada en este proyecto.

7. Referencias

Python Software Foundation. (2023). Python Documentation. https://docs.python.org/

SQLite. (2023). SQLite Documentation. https://www.sqlite.org/docs.html

Pandas Development Team. (2023). Pandas Documentation. https://pandas.pydata.org/docs/

NumPy Development Team. (2023). NumPy Documentation. https://numpy.org/doc/

GitHub. (2023). GitHub Actions Documentation. https://docs.github.com/en/actions

Jikan API. (2023). Jikan API Documentation. https://jikan.moe/