Imagen que contiene Forma

Descripción generada automáticamente

**PROYECTO AGORA**

**DESCRIPCIÓN GENERAL REPOSITORIO ETL**

**DOCUMENTO INICIAL**

**Septiembre 2024**

**Control de cambios**

|  |  |  |  |
| --- | --- | --- | --- |
| **VERSIÓN** | **FECHA** | **MODIFICACIÓN** | **RESPONSABLE** |
| 1.0 | 30 de septiembre de 2024 | Creación documento descripción general repositorio ETL | Pedro Fabián Pérez Arteaga |

# Documento descripción general repositorio ETL

## Introducción

Este manual describe el proceso general ETL (Extracción, Transformación y Carga) implementado en el proyecto AGORA. Este proceso se encarga de manejar fuentes de datos diversas dentro de un DataLake, utilizando Apache Spark y HDFS (Hadoop Distributed File System). El objetivo principal es proporcionar una visión clara y sencilla del flujo de trabajo ETL, adecuada para personas no técnicas, pero con explicaciones técnicas básicas. Es importante mencionar que el código se encuentra comentado para facilidad de modificaciones.

## Glosario

* ETL: Proceso que implica la Extracción de datos de una fuente, su Transformación para cumplir ciertos estándares, y su Carga en un destino final.
* DataLake: Repositorio centralizado que permite almacenar grandes cantidades de datos en su formato original o procesados.
* HDFS: Sistema de archivos distribuido utilizado para almacenar datos de manera escalable.
* Apache Spark: Herramienta de procesamiento de datos que permite manejar grandes volúmenes de información de manera eficiente.
* Fuente de Datos: Archivos o sistemas de los cuales se extrae información para el proceso ETL.

## Flujo del Proceso ETL

### 1. Extracción

La extracción implica la lectura de archivos de datos desde el sistema HDFS. En el proyecto AGORA, cada fuente tiene archivos ubicados en carpetas específicas dentro del DataLake. Se utilizan formatos como CSV o TXT, y se maneja una codificación adecuada para caracteres especiales.

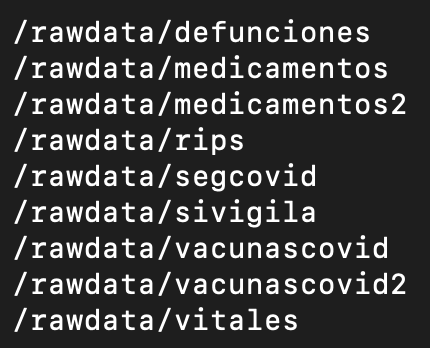


Imagen del DataLake con la estructura de las fuentes disponibles

### 2. Transformación

En esta etapa, los datos extraídos son procesados para adaptarse a los estándares del proyecto. Las transformaciones incluyen cambios de nombres de columnas, filtrado de registros erróneos, y ajustes de formato. Por ejemplo, los archivos de nacimientos transforman la columna 'PersonaID' en 'personaid'. También se corrigen errores como datos desplazados debido a comas internas.

### 3. Carga

Finalmente, los datos transformados son cargados en el DataLake en formato Parquet. Este formato permite una lectura y escritura más eficientes. Los datos se almacenan en rutas específicas para cada fuente, asegurando un acceso organizado y escalable.



Imagen del DataLake con la estructura de salida del ETL

## Detalles Técnicos por Fuente

Como ejemplo detallo una fuente:

### Fuente: Defunciones

sources:

- name: "defunciones"

input\_path: "hdfs:///rawdata/vitales/defunciones/"

output\_path: "hdfs:///stagedata/vitales/defunciones/"

input\_files: ["ConsultarDefunciones2008-2023\_V4.csv"]

input\_columns: ['PersonaID', 'Sexo', 'RegimenAfiliacionSGSSS', 'Administradora', 'CodigoPertenenciaEtnica', 'municipioResidencia', 'SemanasGestacion', 'EdadMadreMomentoDiagnostico', 'FechaDefuncionAAAAMM', 'IpsDondeRealizaParto', 'Peso', 'AnoID']

output\_columns: ["personaid", "sexo", "regimenafiliacion", "administradora", "codigoetnia", "municipioresidencia", "semanasgestacion", "edadmadre", "fechadefuncion", "ips", "peso", "anoid"]

column\_renames:

- original: "PersonaID"

new: "personaid"

- original: "Sexo"

new: "sexo"

- original: "RegimenAfiliacionSGSSS"

new: "regimenafiliacion"

- original: "Administradora"

new: "administradora"

- original: "CodigoPertenenciaEtnica"

new: "codigoetnia"

- original: "municipioResidencia"

new: "municipioresidencia"

- original: "SemanasGestacion"

new: "semanasgestacion"

- original: "EdadMadreMomentoDiagnostico"

new: "edadmadre"

- original: "FechaDefuncionAAAAMM"

new: "fechadefuncion"

- original: "IpsDondeRealizaParto"

new: "ips"

- original: "Peso"

new: "peso"

- original: "AnoID"

new: "anoid"

process: true

error\_file: "/home/agora/DataLake/ETL/errors/vitales/defunciones/defunciones\_errados.txt"

encoding: "latin1"

Los datos se transforman según el esquema definido y se almacenan en la ruta de salida en formato Parquet.

## Estructura del Repositorio

El repositorio del proyecto AGORA se organiza de la siguiente manera:  
  
1. Archivo principal: main2.py  
2. Directorio de configuración: Contiene archivos YAML para la configuración de fuentes y errores:  
 - config.yaml: Configuración general de las fuentes y parámetros del ETL.  
 - config\_errors.yaml: Configuración específica para el manejo y corrección de errores.  
3. Directorio src/etl: Contiene los módulos principales del ETL:  
 - reader.py: Maneja la lectura de datos desde HDFS.  
 - transformer.py: Realiza las transformaciones necesarias en los datos.  
 - writer.py: Escribe los datos transformados en el destino especificado.  
 - error\_handler.py: Detecta y maneja errores en los registros.

## Descripción de Funciones por Archivo

### Archivo: main2.py

Este es el punto de entrada principal para ejecutar el ETL. Realiza las siguientes tareas:  
1. Carga los archivos de configuración.  
2. Itera sobre las fuentes definidas y ejecuta las etapas de extracción, transformación y carga.  
3. Registra métricas como el número de registros procesados y errores encontrados.

### Archivo: reader.py

Funciones principales:  
- read\_from\_hdfs: Lee archivos desde rutas definidas en HDFS.  
- validate\_headers: Verifica que los encabezados de los archivos coincidan con los esperados.

### Archivo: transformer.py

Funciones principales:  
- transform\_columns: Renombra columnas y realiza ajustes en los datos.  
- filter\_errors: Identifica registros con errores y los separa para su manejo posterior.

### Archivo: writer.py

Funciones principales:  
- write\_to\_hdfs: Escribe datos transformados en formato Parquet en HDFS.  
- clean\_output\_directory: Elimina datos previos en la ruta de salida para evitar duplicados.

### Archivo: error\_handler.py

Funciones principales:  
- log\_error: Registra errores encontrados durante el proceso.  
- correct\_errors: Intenta corregir errores comunes según reglas definidas.

## Secuencia para Ejecutar main2.py

La ejecución del archivo main2.py sigue la siguiente secuencia:  
1. Se carga la configuración desde config.yaml y config\_errors.yaml.  
2. Por cada fuente activada en la configuración:  
 - Se leen los datos desde HDFS usando reader.py.  
 - Los datos son transformados con transformer.py, renombrando columnas y filtrando errores.  
 - Los datos válidos son escritos en HDFS en formato Parquet mediante writer.py.  
 - Los registros con errores son manejados y registrados usando error\_handler.py.  
3. Al finalizar, se generan estadísticas del proceso, como registros totales, válidos y con errores.

## Conclusiones

Este manual proporciona una visión general del proceso ETL implementado en el proyecto AGORA. El uso de Apache Spark y HDFS garantiza la escalabilidad y eficiencia en el manejo de grandes volúmenes de datos. Para mayor información técnica, consulte la documentación del código fuente.