DOCUMENTO TÉCNICO DE FLUJO DE DATOS

PRUEBA DATA ENGINEER JR

PAULA CHAMORRO

Noviembre 2025

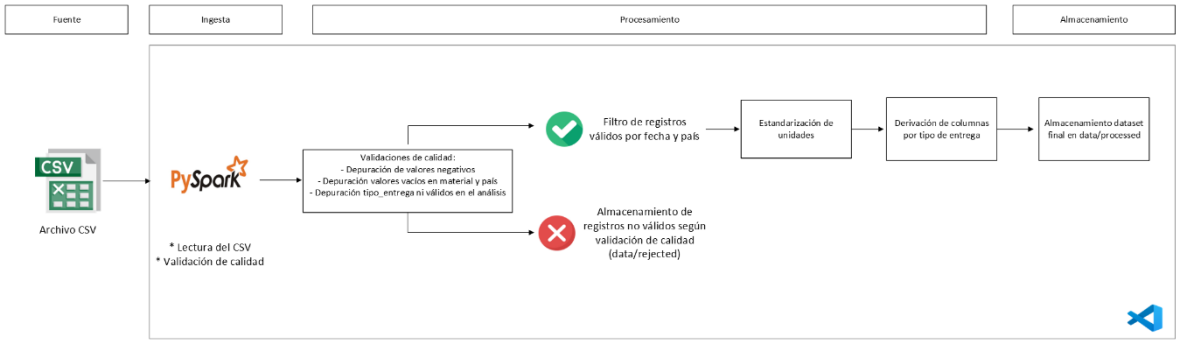
OBJETIVO GENERAL

Construir un flujo de procesamiento de datos para datos que no fueron procesados en los primeros 6 meses del año, utilizando PySpark que permita:

- Leer un archivo CSV de la información de los últimos 6 meses
- Aplicar validaciones y calidad de datos
- Configurable para filtrar por rangos de fecha y país
- Estandarizar unidades de medida
- Clasificación de tipos de entrega
- Generar salida de la información final procesada

ARQUITECTURA Y DESCRIPCIÓN DEL FLUJO

La arquitectura propuesta presenta el diseño de la solución implementada:



Es importante aclarar que, dado que la máquina en la que se realizó el proceso tiene Sistema Operativo Windows, se creó una imagen desde Docker para poder trabajar mucho mejor con PySpark ya que es mucho más compatible con un Sistema Operativo Linux.

Los siguientes pasos describen las etapas del procesamiento reflejado en el diagrama:

Lectura archivo CSV

1. El desarrollo se implementó en Visual Studio Code, almacenado en un repositorio de GitHub
2. Se trabajó en al ambiente develop
3. Se lee el dataset original (archivo CSV) usando PySpark con autodetección de tipos de datos

4. Se implementó una tabla de manejo de errores para dejar los logs de todos aquellos registros que no lograron entrar al análisis por calidad de datos
5. Se incorporó una columna adicional para la tabla de manejo de errores llamada 'tipo_error' para dejar especificada la razón por la cual no entraron en los datos de análisis

Validación calidad de datos

1. Se realizaron las siguientes validaciones de calidad de datos:
   i) Se asume obligatoriedad en *material* y *país* por lo que los registros que tengan vacíos y/o NULL se excluyen del análisis
   ii) Para el campo *tipo_entrega* solo se aceptan los valores ZPRE, ZVE1, Z04, Z05, se marcan como error todos aquellos que sean distintos a los valores mencionados. (Los valores permitidos están configurados en *config.yaml*)
   iii) Se excluyen lo valores *negativos* en precio y *cantidad*

Filtrado configurable por fecha y país

1. Los registros inválidos se almacenan por *fecha_proceso* en *data/rejected*
2. Se filtran los registros válidos por fecha y país, estos valores se leen desde *config.yaml*

Estandarización de unidades

1. La parametrización de la equivalencia de unidades está configurada en *config.yaml*
2. Se genera la columna *cantidad_estandarizada* para almacenar los nuevos valores de cantidad según la equivalencia (valor original * valor CS-ST). CS: 20 y ST: 1

Clasificación tipo de entrega

1. El tipo de entrega puede indicar:
   i) De rutina: ZPRE, ZVA1
   ii) Con bonificación: Z04, Z05

   Estas categorías están configuradas en *config.yaml*

2. Se crean nuevas columnas binarias que faciliten la analítica y las agregaciones:

- is_zpre
- is_zve1
- is_z04
- is_z05
- is_rutina
- is_bonificacion

Escritura de resultados

1. La ruta de escritura de archivos está parametrizada en *config.yaml*
2. Los registros válidos finales se almacenan *data/processed/{fecha_proceso}* en formato parquet particionado por fecha
3. Los registros rechazados se almacenan en *data/rejected/{fecha_proceso}* en formato parquet particionado por fecha, de modo que se pueda tener la auditoría de calidad de datos e identificar los registros que no pudieron ser cargados

ESTRUCTURA DEL PROYECTO

El diseño de la solución mantiene la siguiente estructura en la que se relaciona el detalle posterior al nombre del archivo:

caso-de-uso-datos/

––– src/

- main.py: ejecutable final

- transformations.py: elaboración de lectura, transformación y lectura

––– config/

- config.yaml: archive con los parámetros requeridos

––– data/

- raw/: almacenamiento archivo CSV
- processed/: almacenamiento particionado del dataset final procesado
- rejected/: almacenamiento particionado de los registros que no entraron en el análisis según la validación y calidad de datos

––– docker/

- Dockerfile: configuración máquina de Docker para ejecutar proyecto PySpark

––– docs/

- diagrama_flujo.png: diagrama gráfico del proceso
- arquitectura_flujo.pdf: documentación técnico del proceso

––– README.md

CONSIDERACIONES TÉCNICAS

El flujo implementado controla todos los parámetros operativos a través de un archivo YAML gestionado con OmegaConf, lo que permite una configuración flexible del código. PySpark se encarga de la lectura, transformación y escritura de los datos, garantizando eficiencia en el proceso, además, la escritura particionada por fecha optimiza el consumo posterior en herramientas de BI y análisis requerido.