



TEC. AZUAY
INSTITUTO UNIVERSITARIO

**TECNOLOGÍA SUPERIOR UNIVERSITARIA EN
DESARROLLO DE SOFTWARE**

ASIGNATURA:

... POO2 ...

TEMA:

... INFORME PROYECTO ...

ESTUDIANTE:

DEREK STEVEN VERGARA MOROCHO

DOCENTE:

... ING. DANIEL ORTIZ ...

CURSO:

N6A

FECHA:

21/8/2025

AÑO LECTIVO:

ABRIL 2025 – AGOSTO 2025



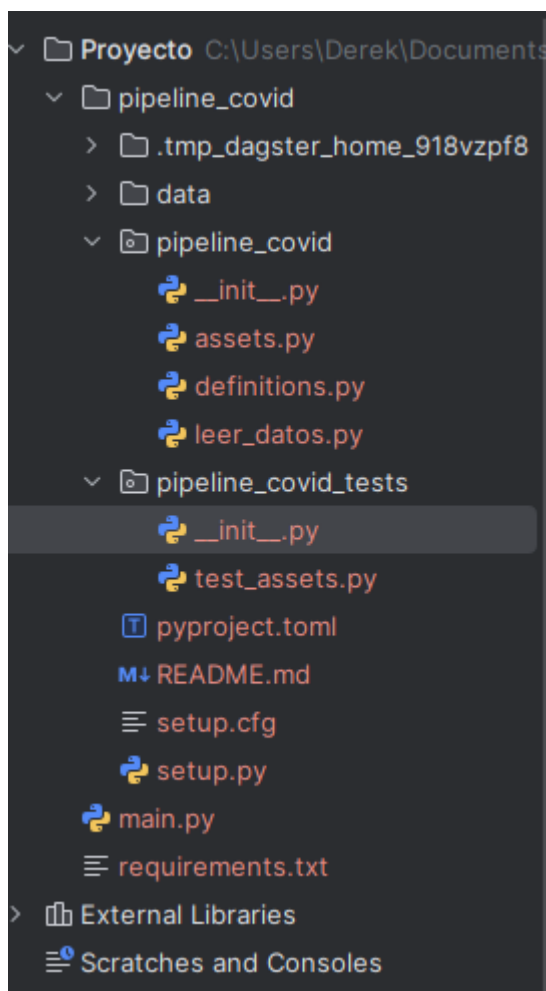
TEC. AZUAY
INSTITUTO UNIVERSITARIO

Introducción

El presente proyecto tiene como finalidad desarrollar un pipeline de procesamiento de datos utilizando la herramienta **Dagster**, con el objetivo de analizar información relacionada con casos de COVID-19. Este pipeline permite automatizar la ingesta, validación, transformación, generación de métricas y exportación de reportes, siguiendo una estructura modular y reutilizable. Se emplea el entorno de desarrollo **PyCharm**, con una organización de carpetas orientada a la mantenibilidad del código.

Estructura del proyecto

El proyecto está estructurado de la siguiente manera:



Descripción de los assets implementados

El desarrollo del pipeline incluye varios **assets**, los cuales representan distintas etapas del procesamiento de datos:

Asset: `leer_datos`

- **Propósito:** Carga el archivo `compactCSV.csv` desde la carpeta `data` y lo convierte en un `DataFrame`.
- **Entrada:** Archivo CSV
- **Salida:** `DataFrame` de `pandas`

Asset: `datos_procesados`

- **Propósito:** Realiza limpieza de datos, conversión de fechas, eliminación de nulos y normalización.
- **Transformaciones:** Conversión a `datetime`, normalización de nombres de columnas.

Asset: `metrica_incidencia_7d`

- **Propósito:** Calcula la incidencia acumulada de casos por cada 100.000 habitantes en los últimos 7 días por provincia.
- **Método:** Agrupación por provincia y fecha, cálculo de sumatorias y divisiones por población.

Asset: `metrica_factor_crec_7d`

- **Propósito:** Calcula el factor de crecimiento entre las últimas dos semanas.
- **Cálculo:** División de casos semana actual entre semana anterior.

Asset: `reporte_excel_covid`

- **Propósito:** Genera un archivo de salida en formato Excel con las métricas procesadas.
- **Librería:** `pandas.ExcelWriter`

Descripción de los asset checks implementados

Check: `check_columnas_clave`

- **Valida:** Que existan las columnas necesarias (`provincia`, `fecha`, `casos`, etc.)

Check: `check_fecha_no_futura`

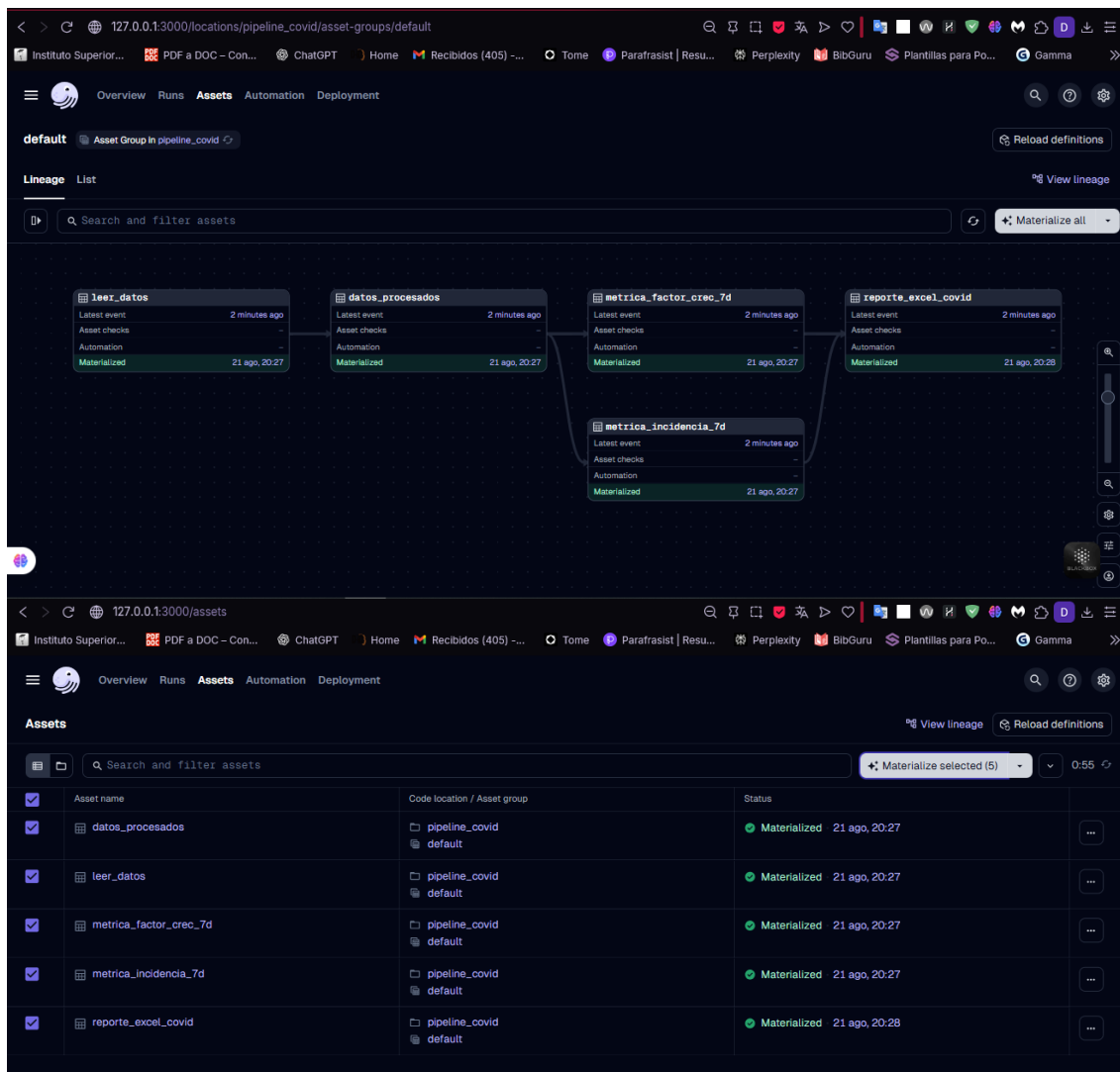
- **Valida:** Que ninguna fecha registrada sea mayor a la fecha actual.

Check: `check_unicidad`

- **Valida:** Que no existan duplicados en base a las columnas clave (`provincia`, `fecha`).

Ejecución y resultados

El pipeline fue ejecutado de forma satisfactoria utilizando la interfaz web de Dagster (dagster dev). Cada asset fue materializado correctamente y los checks fueron validados.



Assets Table Data:

Asset name	Code location / Asset group	Status
datos_procesados	pipeline_covid / default	Materialized 21 ago, 20:27
leer_datos	pipeline_covid / default	Materialized 21 ago, 20:27
metrica_factor_crec_7d	pipeline_covid / default	Materialized 21 ago, 20:27
metrica_incidencia_7d	pipeline_covid / default	Materialized 21 ago, 20:27
reporte_excel_covid	pipeline_covid / default	Materialized 21 ago, 20:28

Capturas de pantalla:

A continuación se muestran algunas evidencias del pipeline funcionando:

- Asset leer_datos:

```
@asset 2 usages
def leer_datos() -> pd.DataFrame:
    ruta_archivo = Path(__file__).resolve().parents[1] / "data" / "compact.csv"
    df = pd.read_csv(ruta_archivo)
    print("Columnas disponibles en el CSV:", df.columns.tolist())

    # Renombrar 'country' a 'location' para que el resto del pipeline funcione
    df = df.rename(columns={"country": "location"})

    columnas = ['location', 'date', 'new_cases', 'people_vaccinated', 'population']
    df = df[columnas]
    df['date'] = pd.to_datetime(df['date'])
    return df
```

- Asset metrica_incidencia_7d:

```
@asset
def metrica_incidencia_7d(datos_procesados: pd.DataFrame):
    incidencia = datos_procesados.groupby("provincia")["nuevos_casos"].rolling(window=7).sum().reset_index()
    incidencia.rename(columns={"nuevos_casos": "incidencia_7d"}, inplace=True)
    return incidencia
```

- Asset reporte_excel_covid:

```
@asset
def reporte_excel_covid(metrica_incidencia_7d: pd.DataFrame, metrica_factor_crec_7d: pd.DataFrame, datos_procesados: pd.DataFrame) -> M:
    output_dir = Path(__file__).resolve().parents[1] / "outputs"
    os.makedirs(output_dir, exist_ok=True) # Crear la carpeta si no existe
    ruta_excel = output_dir / "reporte_covid_final.xlsx"

    with pd.ExcelWriter(ruta_excel) as writer:
        datos_procesados.to_excel(writer, index=False, sheet_name="Datos Procesados")
        metrica_incidencia_7d.to_excel(writer, index=False, sheet_name="Incidencia 7d")
        metrica_factor_crec_7d.to_excel(writer, index=False, sheet_name="Factor Crec 7d")

    print(f"Reporte generado en: {ruta_excel}")
```

8. Conclusión

Este proyecto demuestra el uso efectivo de **Dagster** para la creación de pipelines modulares y controlados para el procesamiento de datos. Se validaron los principios de calidad de datos mediante asset checks y se generó un reporte final en Excel que puede ser compartido con stakeholders. El uso de PyCharm y una buena organización del proyecto permitió una experiencia de desarrollo eficiente y reproducible.

El resultado es un flujo de trabajo robusto, reutilizable y adaptable para futuros conjuntos de datos relacionados a salud u otros dominios.