

Maestría en Inteligencia Artificial y Ciencia de Datos

García Muñoz, Marko D; Herrera Rivera, Jorge A; Torres Polanco, Cristhian A

Informe Técnico – Laboratorio 2 ETL

Repositorio GitHub: https://github.com/cristhianalextores/ETL_Lab2

1. Descripción del Proyecto

El laboratorio implementa una solución ETL académica en Python con SQLite. La arquitectura está compuesta por módulos de extracción, carga, logging y monitoreo. Los datos se gestionan a partir de tres fuentes en formato CSV:

- alumnos.csv
- matriculas.csv
- calificaciones.csv

El objetivo es cargar esta información en la base de datos lab2.db, asegurando trazabilidad mediante logs y monitoreo de métricas en la tabla etl_monitor.

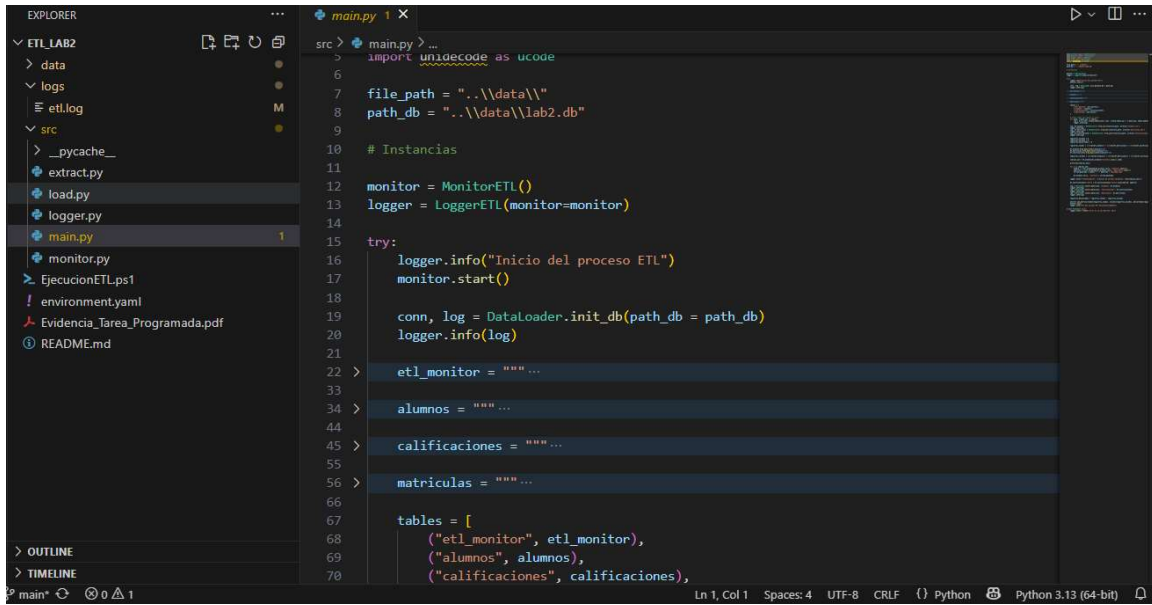
2. Preguntas del Laboratorio

a. ¿Qué hace el script `EjecucionETL.ps1`?

Este script automatiza la ejecución completa del ETL:

1. Activa el entorno conda definido en environment.yaml.
2. Ejecuta el flujo principal (main.py).
3. Al finalizar, desactiva el entorno.

De esta forma, se garantiza una corrida reproducible y libre de dependencias externas.

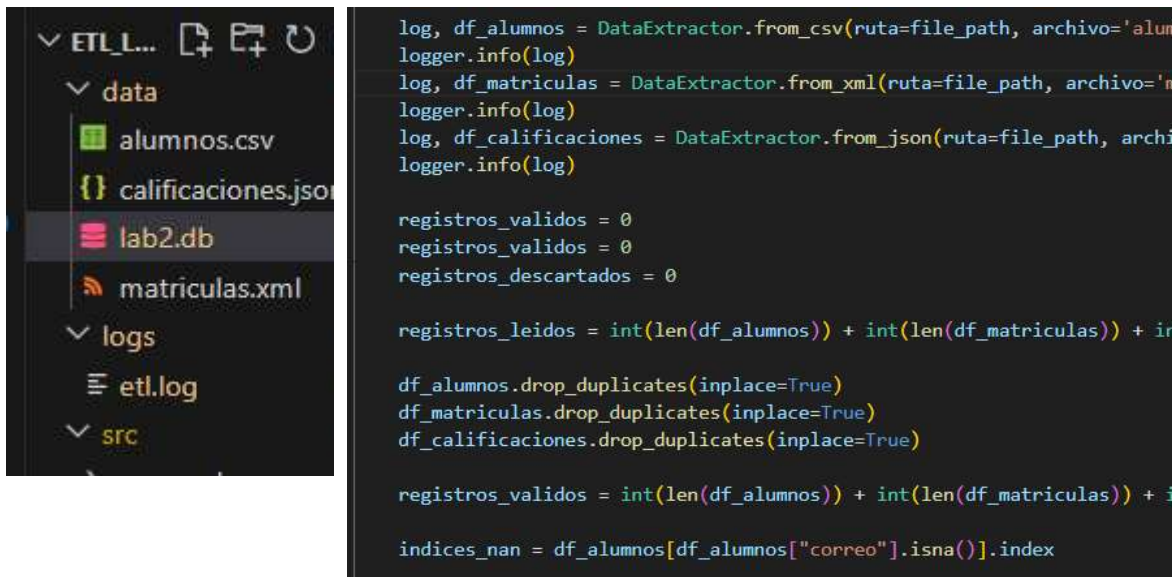


```
src > main.py 1 X
> import unicode as ucode
6
7 file_path = "..\\data\\"
8 path_db = "..\\data\\lab2.db"
9
10 # Instancias
11
12 monitor = MonitorETL()
13 logger = LoggerETL(monitor=monitor)
14
15 try:
16     logger.info("Inicio del proceso ETL")
17     monitor.start()
18
19     conn, log = DataLoader.init_db(path_db = path_db)
20     logger.info(log)
21
22 > etl_monitor = "" ...
23
24 > alumnos = "" ...
25
26 > calificaciones = "" ...
27
28 > matriculas = "" ...
29
30
31 tables = [
32     ("etl_monitor", etl_monitor),
33     ("alumnos", alumnos),
34     ("calificaciones", calificaciones),
35 ]
```

b. ¿Cómo funciona el proceso de extracción?

El módulo extract.py lee los archivos CSV ubicados en la carpeta data. Cada archivo corresponde a una entidad académica (alumnos, matrículas, calificaciones). La extracción incluye:

- Validación de estructura de los archivos.
- Conteo de registros leídos.
- Manejo de posibles errores durante la lectura.



```
log, df_alumnos = DataExtractor.from_csv(ruta=file_path, archivo='alumnos.csv')
logger.info(log)

log, df_matriculas = DataExtractor.from_xml(ruta=file_path, archivo='matriculas.xml')
logger.info(log)

log, df_calificaciones = DataExtractor.from_json(ruta=file_path, archivo='calificaciones.json')
logger.info(log)

registros_validos = 0
registros_validos = 0
registros_descartados = 0

registros_leidos = int(len(df_alumnos)) + int(len(df_matriculas)) + int(len(df_calificaciones))

df_alumnos.drop_duplicates(inplace=True)
df_matriculas.drop_duplicates(inplace=True)
df_calificaciones.drop_duplicates(inplace=True)

registros_validos = int(len(df_alumnos)) + int(len(df_matriculas)) + int(len(df_calificaciones))

indices_nan = df_alumnos[df_alumnos["correo"].isna()].index
```

c. ¿Qué sucede en la fase de carga?

El módulo load.py:

- Crea las tablas necesarias en lab2.db si no existen.

- Inserta los registros extraídos en las tablas correspondientes (alumnos, matriculas, calificaciones).
- Controla duplicados o inconsistencias para asegurar integridad.

d. ¿Cómo se implementa el monitoreo?

El módulo monitor.py inserta métricas de ejecución en la tabla etl_monitor. Entre ellas:

- Fecha y hora de ejecución.
- Registros leídos.
- Registros válidos y descartados.
- Tiempo total de ejecución.
- Errores ocurridos.

	registros_leidos	registros_válidos	registros_descartados	duracion	error
70	20:00:03	30	29	1	0.1
71	20:15:03	30	29	1	0.11
72	20:30:03	30	29	1	0.11
73	20:45:03	30	29	1	0.09
74	21:00:03	30	29	1	0.12
75	21:15:03	30	29	1	0.1
76	21:30:05	30	29	1	0.4
77	21:45:04	30	29	1	0.18
78	22:00:03	30	29	1	0.1
79	18:03:58	30	29	1	0.32
80	18:15:06	30	29	1	0.23
81	18:30:04	30	29	1	0.31
82	18:45:06	30	29	1	0.46
83	19:00:03	30	29	1	0.09
84	19:15:04	30	29	1	0.21
85	19:24:43	0	0	0	0.03 [ERROR] Error en la corrida E'
86					

Esto permite auditar y evaluar la eficiencia del ETL a lo largo de múltiples corridas.

e. ¿Cómo se gestiona el logging?

El módulo logger.py escribe eventos en dos lugares:

1. Consola → para retroalimentación inmediata al usuario.
2. Archivo logs/etl.log → registro persistente de todas las corridas.

```

main.py 1, M  etl.log M X  extract.py
logs > etl.log
945 2025-09-26 19:00:03,104 [INFO] Fin del proceso ETL Satisfactoriamente
946 2025-09-26 19:15:04,645 [INFO] Inicio del proceso ETL
947 2025-09-26 19:15:04,653 [INFO] La BD ya existe: ..\data\lab2.db
948 2025-09-26 19:15:04,660 [INFO] La tabla 'etl_monitor' ya existe.
949 2025-09-26 19:15:04,661 [INFO] La tabla 'alumnos' ya existe.
950 2025-09-26 19:15:04,661 [INFO] La tabla 'calificaciones' ya existe.
951 2025-09-26 19:15:04,662 [INFO] La tabla 'matriculas' ya existe.
952 2025-09-26 19:15:04,676 [INFO] CSV cargado correctamente: alumnos.csv
953 2025-09-26 19:15:04,761 [INFO] XML cargado correctamente con ElementTree: matriculas.xml
954 2025-09-26 19:15:04,775 [INFO] JSON cargado correctamente: calificaciones.json
955 2025-09-26 19:15:04,796 [INFO] Transformación - Creación de Correos Faltantes: 3
956 2025-09-26 19:15:04,817 [INFO] 8 registros insertados en tabla 'alumnos'
957 2025-09-26 19:15:04,833 [INFO] 13 registros insertados en tabla 'calificaciones'
958 2025-09-26 19:15:04,851 [INFO] 8 registros insertados en tabla 'matriculas'
959 2025-09-26 19:15:04,861 [INFO] Fin del proceso ETL Satisfactoriamente
960 2025-09-26 19:24:43,023 [INFO] Inicio del proceso ETL
961 2025-09-26 19:24:43,039 [INFO] La BD ya existe: ..\data\lab2.db
962 2025-09-26 19:24:43,039 [INFO] La tabla 'etl_monitor' ya existe.
963 2025-09-26 19:24:43,039 [INFO] La tabla 'alumnos' ya existe.
964 2025-09-26 19:24:43,039 [INFO] La tabla 'calificaciones' ya existe.
965 2025-09-26 19:24:43,039 [INFO] La tabla 'matriculas' ya existe.
966 2025-09-26 19:24:43,054 [INFO] CSV cargado correctamente: alumnos.csv
967 2025-09-26 19:24:43,054 [ERROR] Error en la corrida ETL: too many values to unpack (expected 2)
968

```

El log incluye mensajes de info, warning y error, útiles para depuración y control de calidad.

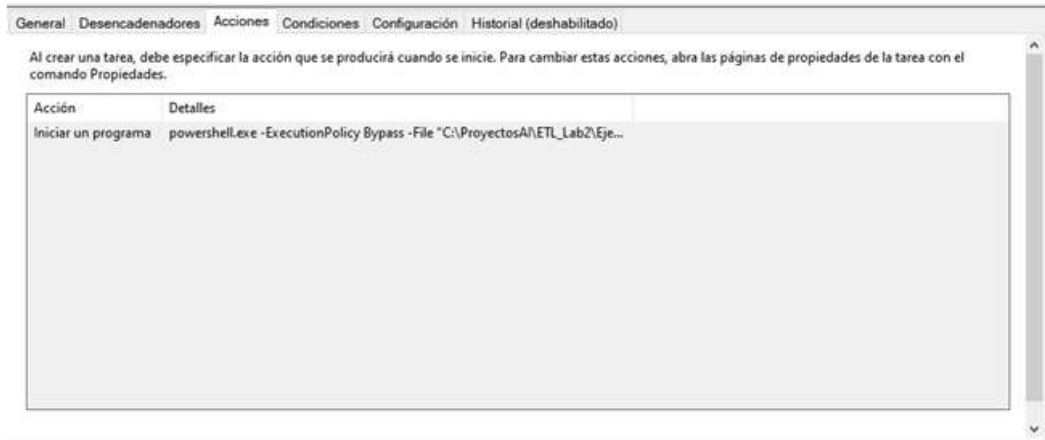
f. ¿Qué requisitos técnicos son necesarios?

Se requiere:

- Tener instalado Conda/Miniconda.
- Crear el entorno con:
conda env create -f environment.yaml
conda activate LabETL
- Ejecutar el proceso con:
./EjecucionETL.ps1

3. Automatización

Nombre	Estado	Desencadenadores	Hora próxima ejecución
EjecucionETL_Lab2	Listo	A las 9:00 p. m. todos los días - Tras desencadenarse, repetir cada 15 minutos durante 1 día.	23/09/2025 9:15:00 p. m.
MicrosoftEdgeUp...	Listo	Se definieron varios desencadenadores	24/09/2025 10:07:16 a. m.
MicrosoftEdgeUp...	Listo	A las 9:37 a. m. todos los días - Tras desencadenarse, repetir cada 1 hora durante 1 día.	23/09/2025 9:37:16 p. m.
OneDrive Per-Mac...	Listo	A las 2:00 p. m. el 1/05/1992 - Tras desencadenarse, repetir cada 1.00:00:00 indefinidamente.	24/09/2025 5:13:03 p. m.
OneDrive Reportin...	Listo	A las 3:21 p. m. el 21/09/2025 - Tras desencadenarse, repetir cada 1.00:00:00 indefinidamente.	24/09/2025 3:21:53 p. m.
OneDrive Startup ...	Listo	Cuando DESKTOP-7185RTM\Ctorres inicie sesión	



4. Conclusiones

- El proyecto demuestra una implementación modular y escalable de ETL usando Python y SQLite.
- Se garantiza trazabilidad y control de calidad gracias a los módulos de logging y monitoreo.
- La solución es fácilmente automatizable en Windows mediante PowerShell y Task Scheduler.
- La arquitectura puede extenderse a otros motores de base de datos o a flujos de datos más complejos.