Universidad de las Americas

Integración de sistemas

Examen Progreso 1 Práctico

Martín Vargas

Repositorio GitHub, con todo lo necesario:
https://github.com/MartinVargas07/Examen-P1-IDS-Vargas.git

**Examen Práctico Progreso 1.**

Resuelva el caso presentado por el docente con visión de Arquitecto de Software.

Para la resolución del caso debe utilizar los Patrones de Integración que se han visto hasta ahora según considere necesarios.

**Caso BioNet**

**Contexto**

La empresa BioNet administra una red de laboratorios clínicos distribuidos en distintas ciudades del país. Debido a limitaciones de conectividad y diferentes proveedores de software, los sistemas de laboratorio no están unificados. La empresa ha solicitado una solución de integración para unificar los resultados de exámenes y sincronizarlos con el sistema central de gestión.

*Problemas por lo que atraviesa actualmente el laboratorio*

1. Cada laboratorio genera archivos .csv diarios con los resultados de los exámenes realizados.

2. Esos archivos se copian manualmente en un servidor FTP compartido.

3. El sistema central accede a una base de datos compartida donde se consolidan los resultados.

4. Existen inconsistencias frecuentes:

• Datos duplicados o sobrescritos.

• Errores de sincronización cuando se suben archivos incompletos.

• Problemas cuando varios procesos escriben al mismo tiempo en la base.

### 1.1 Identificación detallada del problema y riesgos del sistema actual

El flujo operativo actual de BioNet depende de **procesos manuales**, infraestructura heterogénea y ausencia de controles transaccionales. Esto genera una serie de riesgos técnicos y de negocio que amenazan la integridad de los resultados clínicos y la reputación de la empresa.

| # | Riesgo (categoría) | Descripción específica | Consecuencia operativa | Severidad |
|---|---|---|---|---|
| R-01 | **Calidad de datos** | **Duplicados y sobrescrituras** al cargar CSV sin claves únicas. | Informes inconsistentes; posibles diagnósticos equivocados. | Alta |
| R-02 | **Integridad transaccional** | **Archivos incompletos** se procesan parcialmente: filas huérfanas o truncadas. | Pérdida de resultados; reclamos de pacientes. | Alta |
| R-03 | **Concurrencia** | **Condición de carrera**: múltiples cargas escriben la misma tabla simultáneamente. | Bloqueos, deadlocks, corrupción lógica. | Media |
| R-04 | **Trazabilidad** | No existe **auditoría de cambios** ni quién-cuándo-qué se insertó o actualizó. | Imposible rastrear incidentes o fraudes. | Media |
| R-05 | **Seguridad** | Uso de FTP sin cifrado + credenciales genéricas. | Exposición de datos sensibles (HIPAA / LOPDP). | Alta |
| R-06 | **Escalabilidad** | Copia manual de archivos y polling ad-hoc. | Cuellos de botella al crecer el número de laboratorios. | Media |
| R-07 | **Gobernanza de esquemas** | Cada proveedor de LIS genera CSV con **variaciones** de columnas y formatos. | Fallas de parseo; altos costos de mapeo. | Media |
| R-08 | **Gestión de errores** | Errores silenciosos (p. ej., archivos se quedan en FTP). | Retrabajo y sobrecarga del equipo de TI. | Baja |

### 1.2 Justificación ampliada de los patrones seleccionados

**Transferencia de Archivos**

- **Compatibilidad inmediata:** los LIS ya producen CSV; no exige desarrollar APIs en cada laboratorio ni modificar sistemas legados.
- **Desacoplamiento temporal:** el envío asincrónico evita que la caída de la red detenga la operación local.
- **Bajo coste inicial:** basta con un servidor SFTP (+ hardening) y un watcher ligero.
- **Trazabilidad añadida:** al mover archivos a carpetas versionadas (/processed, /error, /archive) se conserva evidencia forense.

**Desventajas mitigadas**:
latencia y riesgo de corrupción, se controla con validación de completitud, hashing opcional y reintentos automáticos.

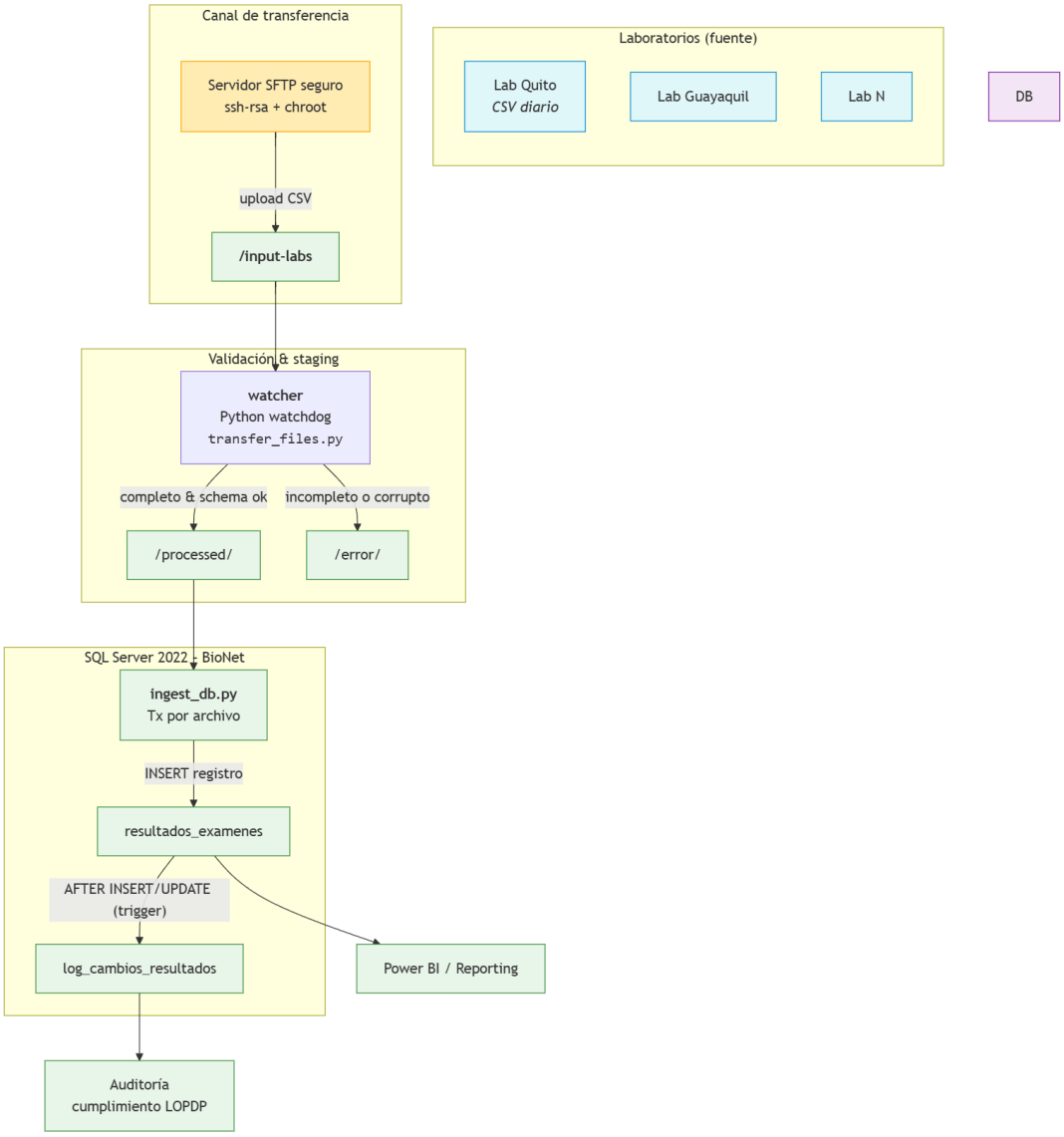
**Base de Datos Compartida**

- **Fuente de verdad única (SSOT):** consolida resultados para BI corporativo y reporte sanitario.
- **Lenguaje estándar (T-SQL):** facilita reporting, mantenimiento y conexión con plataformas analíticas existentes (Power BI, SSRS).
- **Controles ACID nativos:** índices únicos y transacciones evitan duplicados y garantizan atomicidad.
- **Trigger de auditoría:** genera un **log inmutable** para requisitos regulatorios (ISO 15189, GDPR/LOPDP).

**Compensaciones**

- Fuerte acoplamiento a un mismo RDBMS: aceptable porque todas las áreas consumen la misma semántica clínica.
- Riesgo de cuellos de botella: mitigado con buenas prácticas de índice y particionamiento si el volumen crece.

**1.3 Diseño de alto nivel**



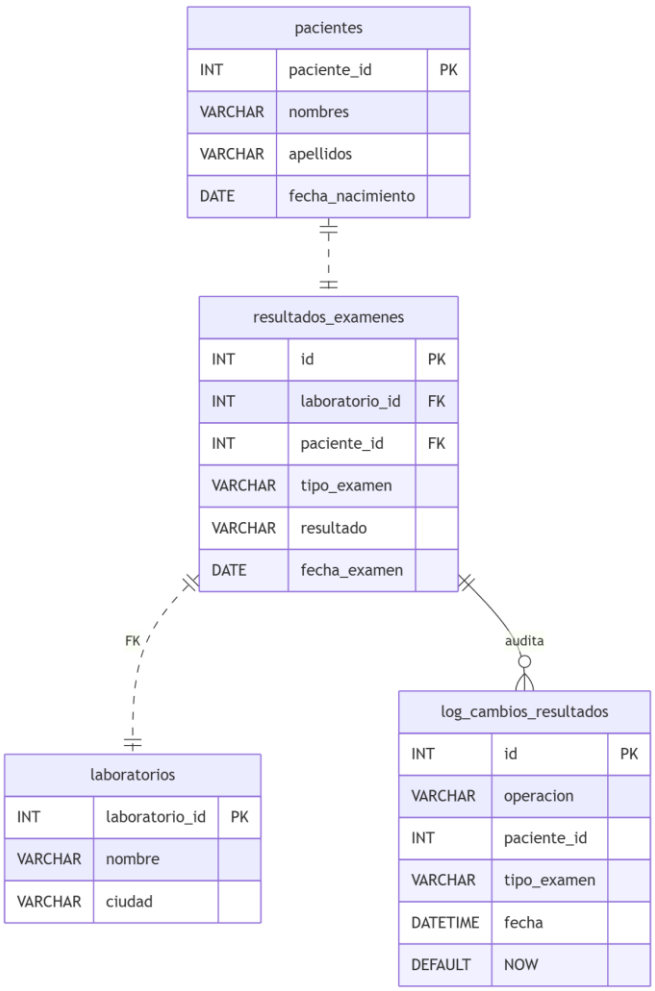

**Explicación textual del flujo**

1. **Generación de resultados**: Cada laboratorio exporta un archivo CSV diario firmemente estructurado (cabecera estándar).
2. **Transporte seguro**: El LIS se autentica por **SFTP (chroot + key-pair)** hacia el servidor central; la carga se ubica en /input-labs.
3. **Watcher**: Un demonio Python basado en *watchdog* detecta nuevos archivos, verifica:
    - Tamaño estable en 3 lecturas.
    - Encabezado exacto de columnas.
    - (Opcional) Checksum *MD5* vs. hash en archivo .md5.
4. **Clasificación**:
    - **OK → /processed**
    - **KO → /error** (genera alerta a soporte vía email o Teams Webhook).
5. **Ingesta transaccional**: ingest_db.py recorre /processed, abre **una transacción por archivo**:
    - Inserta filas en resultados_examenes.

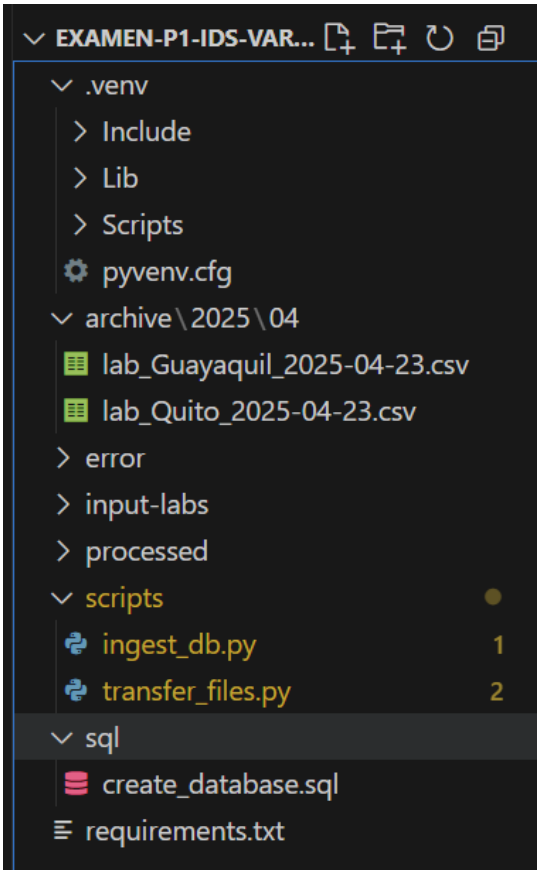
- Captura duplicados mediante índice único (uq_resultado).
- Confirma COMMIT; ante fallo realiza ROLLBACK.
- Mueve CSV a /archive/yyyy/mm/.

6. **Auditoría automática**: Trigger trg_audit_resultados graba INSERT/UPDATE en log_cambios_resultados (con timestamp y tipo de operación).
7. **Consumo corporativo**: ETL nocturno o vistas directas abastecen Power BI, portales médicos y reportes regulatorios.

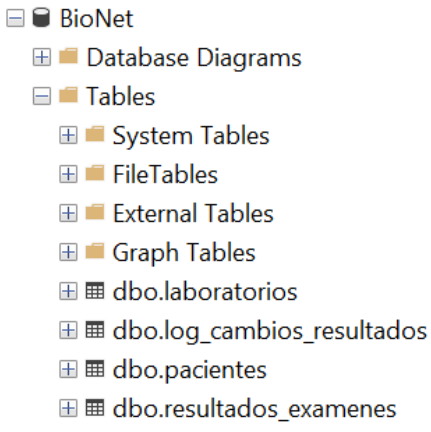
**Diagrama ER**




**Estructura final del proyecto**




**Base de datos:**



create_database.sq...OGSTRIX\marti (68))

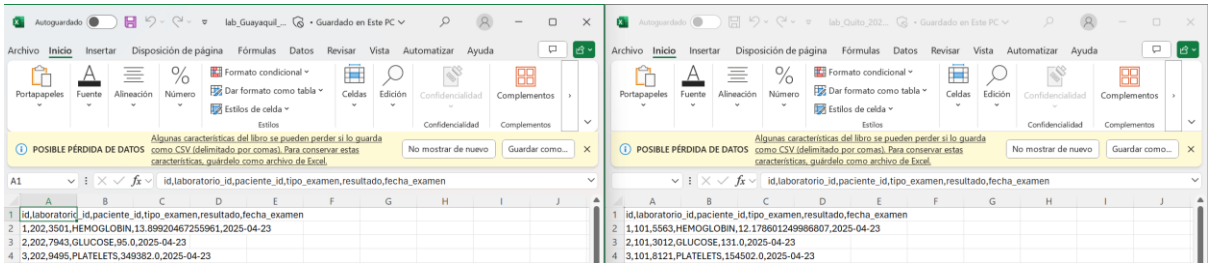
```sql
/* ------------------------------------------------------------
   BioNet - Vargas
------------------------------------------------------------*/
CREATE DATABASE BioNet;
GO
USE BioNet;
GO

/* ---------- tablas maestras ---------- */
CREATE TABLE laboratorios (
    laboratorio_id INT PRIMARY KEY,
    nombre         VARCHAR(100) NOT NULL,
    ciudad         VARCHAR(80)  NOT NULL
);

CREATE TABLE pacientes (
    paciente_id      INT PRIMARY KEY,
    nombres          VARCHAR(100),
    apellidos        VARCHAR(100),
    fecha_nacimiento DATE
);

/* ---------- tabla fact ---------- */
CREATE TABLE resultados_examenes (
    id              INT IDENTITY(1,1) PRIMARY KEY,
    laboratorio_id  INT         NOT NULL,
    paciente_id     INT         NOT NULL,
    tipo_examen     VARCHAR(50) NOT NULL,
    resultado       VARCHAR(50) NOT NULL,
    fecha_examen    DATE        NOT NULL,
    CONSTRAINT uq_resultado UNIQUE
        (laboratorio_id, paciente_id, tipo_examen, fecha_examen),
    CONSTRAINT fk_res_lab FOREIGN KEY (laboratorio_id)
        REFERENCES laboratorios(laboratorio_id),
    CONSTRAINT fk_res_pac FOREIGN KEY (paciente_id)
        REFERENCES pacientes(paciente_id)
);

/* ---------- auditoría ---------- */
CREATE TABLE log_cambios_resultados (
    id           INT IDENTITY(1,1) PRIMARY KEY,
    operacion    VARCHAR(10) NOT NULL,
    paciente_id  INT         NOT NULL,
    tipo_examen  VARCHAR(50) NOT NULL,
```

```
    paciente_id  INT          NOT NULL,
    tipo_examen  VARCHAR(50)  NOT NULL,
    fecha        DATETIME     NOT NULL DEFAULT SYSDATETIME()
);

GO
CREATE TRIGGER trg_audit_resultados
ON resultados_examenes
AFTER INSERT, UPDATE
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO log_cambios_resultados (operacion, paciente_id, tipo_examen)
    SELECT
        CASE WHEN EXISTS (SELECT * FROM deleted) THEN 'UPDATE' ELSE 'INSERT' END,
        i.paciente_id,
        i.tipo_examen
    FROM inserted i;
END
GO

/* ---------- datos básicos para demo ---------- */
INSERT laboratorios VALUES
(101,'Lab Quito'     ,'Quito'),
(202,'Lab Guayaquil' ,'Guayaquil');

-- Pacientes ficticios que coinciden con los CSV de ejemplo (sólo 6 filas)
INSERT pacientes (paciente_id,nombres,apellidos,fecha_nacimiento) VALUES
(1001,'Paciente','Uno','1990-01-01'),
(1002,'Paciente','Dos','1992-03-04'),
(1003,'Paciente','Tres','1995-05-06'),
(2001,'Paciente','Cuatro','1989-07-08'),
(2002,'Paciente','Cinco','1991-09-10'),
(2003,'Paciente','Seis','1993-11-12');
GO
```

Conexión a la base de datos:

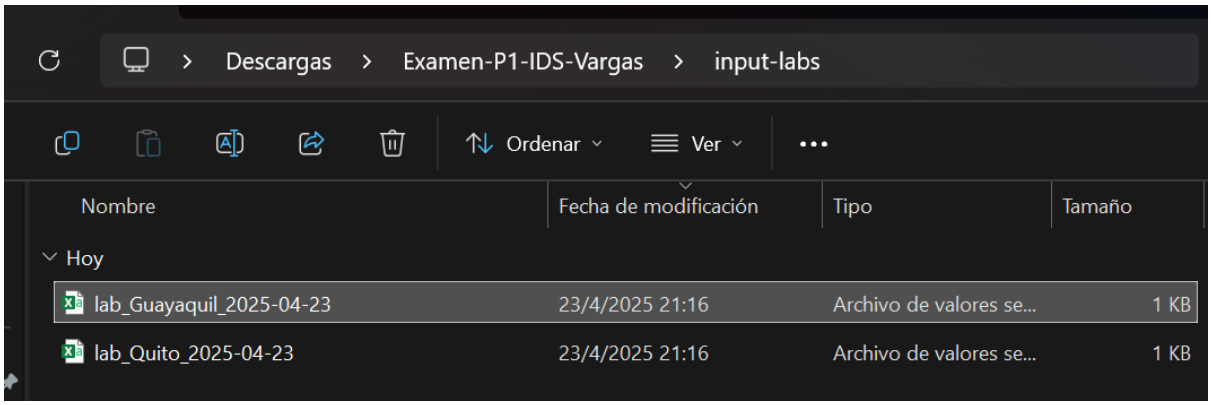
```
13  CONN_STR = (
14      "DRIVER={ODBC Driver 18 for SQL Server};"
15      "SERVER=MARTIN-ROGSTRIX\\SQLEXPRESS;"
16      "DATABASE=BioNet;"
17      "Trusted_Connection=yes;"
18  )
```

Csv's Simples de Ejemplo:

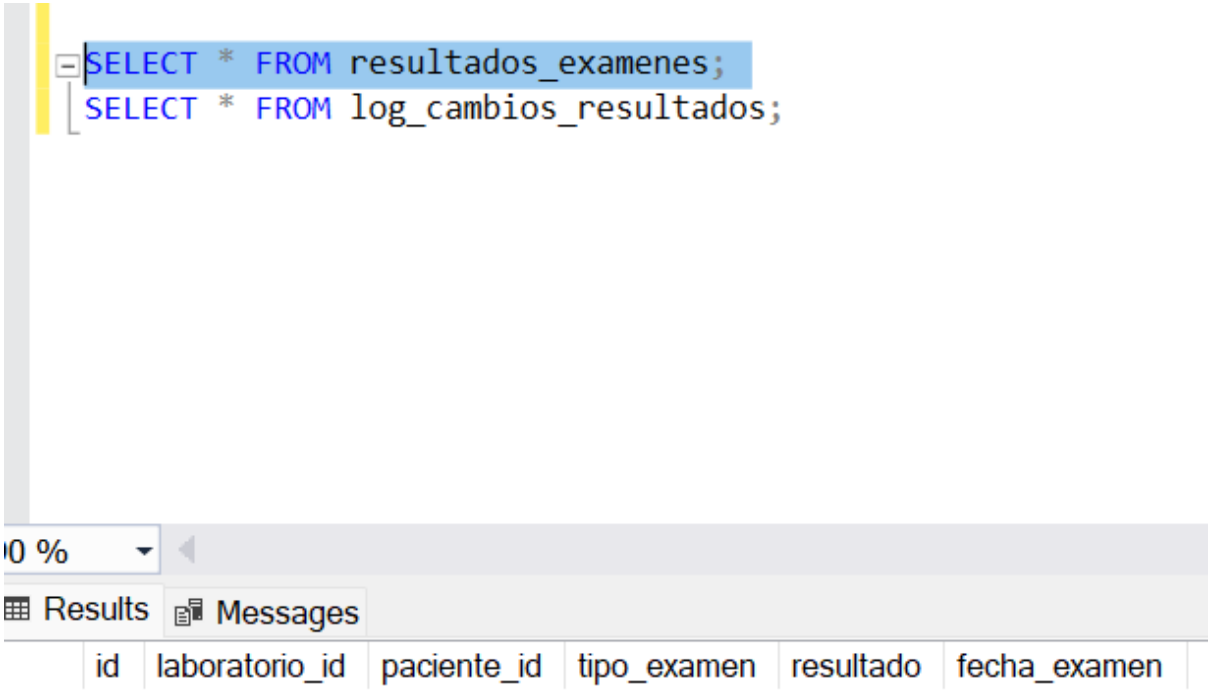


**Pruebas:**

Se empieza desde input-labs:




Se evidencia que no hay registros de momento:



Se instalan los requerimientos:

```
[spyder](base) PS C:\Users\marti\Downloads\Examen-P1-IDS-Vargas> python -m venv .venv
[spyder](base) PS C:\Users\marti\Downloads\Examen-P1-IDS-Vargas> . .venv\Scripts\activate
(.venv) [spyder](base) PS C:\Users\marti\Downloads\Examen-P1-IDS-Vargas> pip install -r requirements.tx
t
Collecting watchdog==4.0.0 (from -r requirements.txt (line 1))
  Downloading watchdog-4.0.0-py3-none-win_amd64.whl.metadata (37 kB)
Collecting pandas==2.2.2 (from -r requirements.txt (line 2))
  Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl.metadata (19 kB)
Collecting pyodbc==5.1.0 (from -r requirements.txt (line 3))
  Downloading pyodbc-5.1.0-cp311-cp311-win_amd64.whl.metadata (2.8 kB)
Collecting python-dotenv==1.0.1 (from -r requirements.txt (line 4))
  Downloading python_dotenv-1.0.1-py3-none-any.whl.metadata (23 kB)
Collecting numpy>=1.23.2 (from pandas==2.2.2->-r requirements.txt (line 2))
  Downloading numpy-2.2.5-cp311-cp311-win_amd64.whl.metadata (60 kB)
     ---------------------------------------- 60.8/60.8 kB 1.6 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.2 (from pandas==2.2.2->-r requirements.txt (line 2))
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl.metadata (8.4 kB)
Collecting pytz>=2020.1 (from pandas==2.2.2->-r requirements.txt (line 2))
  Downloading pytz-2025.2-py2.py3-none-any.whl.metadata (22 kB)
Collecting tzdata>=2022.7 (from pandas==2.2.2->-r requirements.txt (line 2))
  Downloading tzdata-2025.2-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting six>=1.5 (from python-dateutil>=2.8.2->pandas==2.2.2->-r requirements.txt (line 2))
  Downloading six-1.17.0-py2.py3-none-any.whl.metadata (1.7 kB)
Downloading watchdog-4.0.0-py3-none-win_amd64.whl (82 kB)
     ---------------------------------------- 82.9/82.9 kB 1.2 MB/s eta 0:00:00
Downloading pandas-2.2.2-cp311-cp311-win_amd64.whl (11.6 MB)
     ---------------------------------------- 5.3/11.6 MB 1.2 MB/s eta 0:00:06
```

Se inicia el watcher, y se deja activo en la terminal:

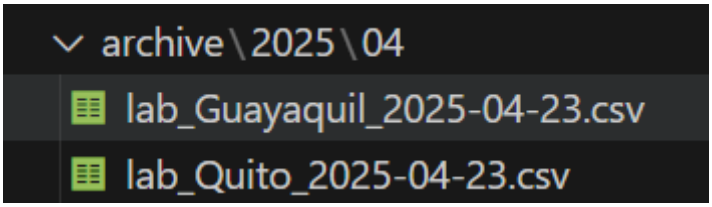
```
(.venv) [spyder](base) PS C:\Users\marti\Downloads\Examen-P1-IDS-Vargas> python scripts/transfer_files.
py
>>
▶ Watcher activo en C:\Users\marti\Downloads\Examen-P1-IDS-Vargas\input-labs
```

```
▶ Watcher activo en C:\Users\marti\Downloads\Examen-P1-IDS-Vargas\input-labs
[✓] lab_Guayaquil_2025-04-23.csv validado y enviado a processed/
[✓] lab_Quito_2025-04-23.csv validado y enviado a processed/
```

Como ya teniamos los archivos dentro de input-labs, muestra esos mensajes que se envian a processed

Ahora abrimos otra terminal y ejecutamos ingest_db:

```
(.venv) [spyder](base) PS C:\Users\marti\Downloads\Examen-P1-IDS-Vargas> python scripts/ingest_db.py
>>
⏳ Procesando lab_Guayaquil_2025-04-23.csv …
✓  lab_Guayaquil_2025-04-23.csv ingresado OK
⏳ Procesando lab_Quito_2025-04-23.csv …
✓  lab_Quito_2025-04-23.csv ingresado OK
```




Pasan a la carpeta archive, con el mes y año.

Verificamos en la base de datos:

```
SELECT COUNT(*) AS total FROM resultados_examenes;
SELECT * FROM log_cambios_resultados ORDER BY id;
```

90 %

Results | Messages

| | total |
|---|---|
| 1 | 6 |

```
SELECT COUNT(*) AS total FROM resultados_examenes;
SELECT * FROM log_cambios_resultados ORDER BY id;
```

90 %

Results | Messages

| | id | operacion | paciente_id | tipo_examen | fecha |
|---|---|---|---|---|---|
| 1 | 7 | INSERT | 2003 | PLATELETS | 2025-04-23 22:02:41.497 |
| 2 | 8 | INSERT | 2002 | GLUCOSE | 2025-04-23 22:02:41.497 |
| 3 | 9 | INSERT | 2001 | HEMOGLOBIN | 2025-04-23 22:02:41.497 |
| 4 | 10 | INSERT | 1003 | PLATELETS | 2025-04-23 22:02:41.497 |
| 5 | 11 | INSERT | 1002 | GLUCOSE | 2025-04-23 22:02:41.497 |
| 6 | 12 | INSERT | 1001 | HEMOGLOBIN | 2025-04-23 22:02:41.497 |