Evaluación Práctica Progreso 1 – Caso BioNet

Anthony Cochea

Integración de Sistemas

Facultad de Ingeniería y Ciencias Aplicadas, Universidad De Las Américas

23 de abril 2025

Informe: Evaluación Práctica Progreso 1 – Caso BioNet

1. Identificación del Problema: Riesgos del Sistema Actual

El sistema actual de BioNet presenta varias limitaciones y riesgos que afectan la integración y gestión de los resultados de exámenes clínicos:

- **Procesos Manuales**: Los laboratorios generan archivos .csv que se copian manualmente a un servidor FTP compartido. Esto introduce errores humanos, como olvidar subir un archivo o subir archivos incompletos.
- **Inconsistencias en los Datos**:
    - **Datos Duplicados o Sobrescritos**: No hay un mecanismo para evitar que los mismos resultados se inserten varias veces o se sobrescriban, lo que puede corromper la información en el sistema central.
    - **Errores de Sincronización**: Los archivos incompletos subidos al FTP pueden ser procesados, causando datos corruptos o incompletos en la base de datos.
    - **Acceso Concurrente**: Varios procesos escribiendo simultáneamente en la base de datos compartida pueden generar conflictos y pérdida de datos.
- **Falta de Trazabilidad**: No hay un registro claro de las operaciones realizadas (inserciones o actualizaciones), lo que dificulta auditar y depurar problemas.

Estos riesgos afectan la confiabilidad de los datos y la capacidad de BioNet para gestionar los resultados de manera eficiente.

2. Justificación de Uso de los Patrones Transferencia de Archivos y Base de Datos Compartida

- **Patrón Transferencia de Archivos**:
    - **Razón de Uso**: Los laboratorios ya generan archivos .csv como método estándar para compartir resultados. Usar el patrón de Transferencia de Archivos permite aprovechar este formato existente, eliminando la necesidad de cambios drásticos en los sistemas de los laboratorios. Además, este patrón es ideal para entornos donde la conectividad puede ser limitada (como lo menciona el caso), ya que los archivos pueden procesarse de forma asíncrona.
    - **Beneficio**: Automatizar la lectura, validación y movimiento de archivos elimina los errores humanos asociados al proceso manual de copiar archivos al FTP. También permite clasificar los archivos (válidos e inválidos) y moverlos a carpetas específicas para un mejor control.
- **Patrón Base de Datos Compartida**:
    - **Razón de Uso**: El sistema central necesita consolidar los resultados de todos los laboratorios en un único repositorio para facilitar el acceso y la gestión.

Una base de datos compartida es la solución más directa para lograr esto, ya que permite a todos los sistemas acceder a los mismos datos de manera estructurada.

- **Beneficio**: Al usar una base de datos relacional (PostgreSQL en este caso), podemos implementar mecanismos como índices únicos para evitar duplicados y triggers para registrar operaciones, mejorando la consistencia y trazabilidad de los datos.

3. Diseño de Alto Nivel de la Solución

3.1 Estructura de Carpetas

La solución utiliza un sistema de carpetas para organizar y procesar los archivos .csv:

- **input-labs**: Carpeta donde los laboratorios depositan los archivos .csv generados diariamente. El sistema monitorea esta carpeta para procesar los archivos automáticamente.
- **processed**: Carpeta donde se mueven los archivos .csv que fueron procesados exitosamente y cuyos datos se insertaron en la base de datos.
- **error**: Carpeta donde se mueven los archivos .csv que no cumplen con las validaciones (por ejemplo, archivos con encabezados incorrectos o datos incompletos).

Diagrama de Estructura de Carpetas:

3.2 Flujo de Integración

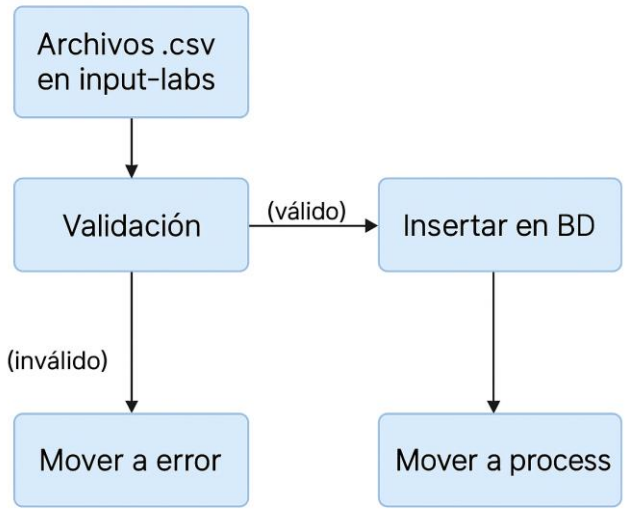
El flujo de integración describe cómo se procesan los archivos desde que se depositan en input-labs hasta que se consolidan en la base de datos:

1. **Lectura de Archivos**: Un componente (implementado con Apache Camel) monitorea la carpeta input-labs y detecta nuevos archivos .csv.
2. **Validación**: Cada archivo se valida para asegurar que:

- Tiene el encabezado correcto (laboratorio_id,paciente_id,tipo_examen,resultado,fecha_examen).
- No tiene campos vacíos.

3. **Procesamiento**:
   - Si el archivo es válido, sus datos se extraen y se insertan en la base de datos.
   - Si el archivo es inválido, se mueve a la carpeta error.
4. **Inserción en la Base de Datos**:
   - Los datos se insertan en la tabla resultados_examenes.
   - Un mecanismo de validación evita duplicados (usando una constraint única).
   - Un trigger registra las operaciones (INSERT o UPDATE) en la tabla log_cambios_resultados.
5. **Confirmación**: Los archivos válidos se mueven a la carpeta processed.

Diagrama del Flujo de Integración:



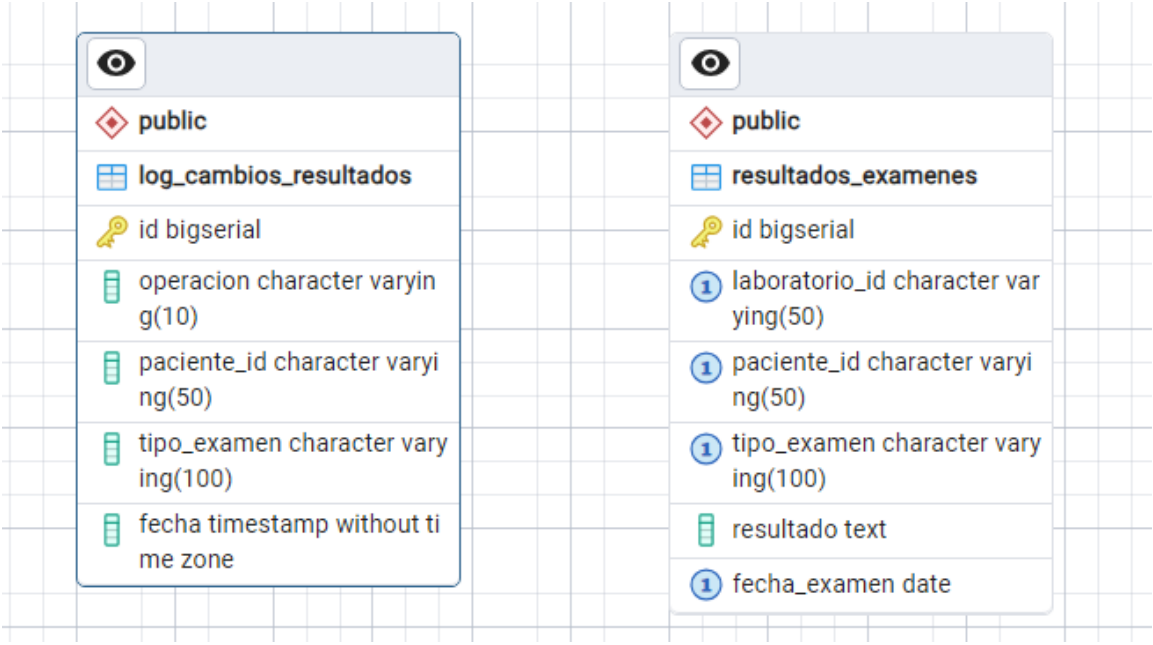
3.3 Esquema de Base de Datos Sugerido

La solución utiliza una base de datos relacional (PostgreSQL) con dos tablas:

- **resultados_examenes**:
  - Almacena los resultados de los exámenes clínicos.
  - Campos:

        - id: Identificador único (autoincremental).
        - laboratorio_id: Identificador del laboratorio (texto).
        - paciente_id: Identificador del paciente (texto).
        - tipo_examen: Tipo de examen realizado (texto).
        - resultado: Resultado del examen (texto).
        - fecha_examen: Fecha del examen (fecha).
    - Restricción: Un índice único sobre laboratorio_id, paciente_id, tipo_examen, y fecha_examen para evitar duplicados.
- **log_cambios_resultados**:
    - Registra las operaciones realizadas en resultados_examenes.
    - Campos:
        - id: Identificador único (autoincremental).
        - operacion: Tipo de operación (INSERT o UPDATE).
        - paciente_id: Identificador del paciente (texto).
        - tipo_examen: Tipo de examen (texto).
        - fecha: Fecha y hora de la operación.

Diagrama del Esquema de Base de Datos:



Script:

Query | Query History

```
CREATE TABLE resultados_examenes (
    id BIGSERIAL PRIMARY KEY,
    laboratorio_id VARCHAR(50) NOT NULL,
    paciente_id VARCHAR(50) NOT NULL,
    tipo_examen VARCHAR(100) NOT NULL,
    resultado TEXT NOT NULL,
    fecha_examen DATE NOT NULL,
    CONSTRAINT unique_result UNIQUE (laboratorio_id, paciente_id, tipo_examen, fecha_examen)
);

CREATE TABLE log_cambios_resultados (
    id BIGSERIAL PRIMARY KEY,
    operacion VARCHAR(10) NOT NULL,
    paciente_id VARCHAR(50) NOT NULL,
    tipo_examen VARCHAR(100) NOT NULL,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION log_resultados_trigger_function()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO log_cambios_resultados (operacion, paciente_id, tipo_examen, fecha)
    VALUES (TG_OP, NEW.paciente_id, NEW.tipo_examen, CURRENT_TIMESTAMP);
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER log_resultados_trigger
AFTER INSERT OR UPDATE ON resultados_examenes
FOR EACH ROW
EXECUTE FUNCTION log_resultados_trigger_function();
```