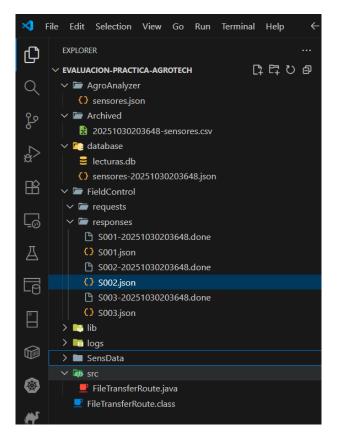
Informe del Taller – Integración AgroTech

1) Descripción general

Se implementó una solución de integración entre **SensData**, **AgroAnalyzer** y **FieldControl** usando **Apache Camel** y **SQLite**, aplicando tres patrones:

- 1. **File Transfer**: mover y transformar datos desde SensData hacia AgroAnalyzer.
- 2. **Shared Database**: persistencia común en SQLite (lecturas.db) para consultas de ambos módulos.
- 3. **RPC simulado**: solicitud síncrona desde FieldControl a AgroAnalyzer para obtener el último valor por sensor.

Estructura de carpetas (resumen):



2) Evidencia de ejecución

Fase 1

```
rte.java C) sensores.json X 🕒 S001:req 🕒 S002:req
   <sub>C</sub>
                                                                                                                                                                                                ttech> # (opcional) limpiar clase previa
tech> Remove-Item -ErrorAction SilentlyContinue .\FileTransferRoute.class
                                                              racionsistemus\evaluacion-practica-agrotecho # compilar
racionsistemus\evaluacion-practica-agrotecho # compilar
racionsistemus\evaluacion-practica-agrotecho javac -encoding UTF-8 -cp "llb/*" -d . .\src\FileTransferRoute.java
racionsistemus\evaluacion-practica-agrotecho java -op ";llb/*" FileTransferRoute
racionsistemus\evaluacion-practica-agrotecho java -op ";llb/*" FileTransferRoute
ache.came.lan.Mainisupport - Apache Camel (wini) ed.14.1 is starting
ache.came.lan.Mainisupport - Apache Camel (wini) ed.14.1 is starting
ache.came.lan.puport.LifecycleStrategySupport - Autowired property: dataSource on component: sql as exactly one instance of type: javax.sql.DataSource (org.sqlite.SQLiteDataSource) foun
                                                                che camel. impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.4) is starting che camel. impl. engine. AbstractCamelContext - Repache Camel 4.14.1 (camel.4) is starting che camel. impl. engine. AbstractCamelContext - Repache Camel 4.14.1 (camel.4) is starting che camel. impl. engine. AbstractCamelContext - Started int.db (timer//init.db) che camel. impl. engine. AbstractCamelContext - Started sensors-csv-to-db-and-json (file://c:/Curssos/IntegracionSistemss/evaluacion-practica-agrotech/SensData) che camel. impl. engine. AbstractCamelContext - Started pre-cliente (file://c:/Curssos/IntegracionSistemss/evaluacion-practica-agrotech/FieldControl/requests) che camel. impl. engine. AbstractCamelContext - Started pre-agroundyzer-server (direct://pcc.handle) che camel. impl. engine. AbstractCamelContext - Started pre-agroundyzer-server (direct://pcc.handle) che camel. impl. engine. AbstractCamelContext - Started pre-agroundyzer-server (direct://pcc.handle) che camel. impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490ss) context of the camel. impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490ss) context of the camel. impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490s) context of the camel impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490s) context of the camel impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490s) context of the camel impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) started in 150s (huildioss initioss start:150ss boot;1490s) context of the camel impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) context of the camel impl. engine. AbstractCamelContext - Apache Camel 4.14.1 (camel.1) context of th
                                    mel-1) thread #4 - timer://init-db] INFO init-db - === [FASE #0/3] INIT-DB -> verificando/creando tabla 'lecturas' ===
mel-1) thread #2 - file://c:/cursos/Integracionsistemas/evaluacion-practica-agrotech/SensData] INFO sensors-csv-to-db-and-json - Se archivará como 'Archived/20251830218000-sensores.csv
mel-1) thread #2 - file://c:/cursos/Integracionsistemas/evaluacion-practica-agrotech/SensData] INFO sensors-csv-to-db-and-json - === [FASE 1/3] CSV->JSON: filas leidas = 3 ===
        amed (camed-1) thread #2 - file://c:/cursos/integracionsistemas/evaluacion-practica-agrotech/sensData] INFO sensors-csv-to-db-and-json - === [FASE 1/3] CSV->25CNN: COMPLETA (JSON escrito y CSV archiva
Fase 2
       camel (camel-1) thread #2 - file://c:/Cursos/IntegracionSistemas/
camel (camel-1) thread #2 - file://c:/Cursos/IntegracionSistemas/
s, temperature26.4)
camel (camel-1) thread #2 - file://c:/Cursos/IntegracionSistemas/
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                [DB] Insert -> {id sensor=S002, fecha=2025-05-22, humedad
       	imes File Edit Selection View Go Run Terminal Help ullet 	o
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            Q evaluacion-practica-agrotech
```

```
··· ■ FileTransferRoute.java 			 { } sensores-20251030210802.json ×
凸
       V EVALUACION-PRACTICA-AGROTECH
                                                   日日日日
                                                                      database > () sensores-20251030210802.json > ...
                                                                              [ {
| "id_sensor" : "S001",
| "3025-05-22
Q

✓ ■ AgroAnalyzer

           () sensores.json
                                                                                 "fecha" : "2025-05-22",
"humedad" : "45",
"temperatura" : "26.4 "
وړ

✓ Image: Archived
           20251030210802-sensores.csv

✓ ► database

                                                                               }, {
| "id_sensor" : "S002",
| "2025-05-22"
₽
        ≡ lecturas.db
                                                                                 "fecha" : "2025-05-22",
"humedad" : "50",
"temperatura" : "25.1 "
           () sensores-20251030210802.json
田

✓ I requests

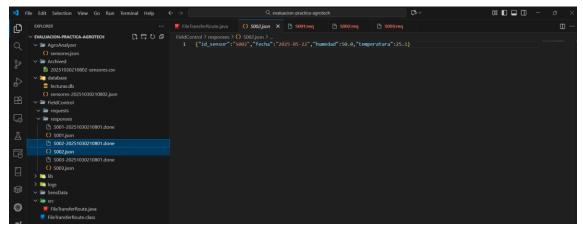
                                                                               }, {
    "id_sensor" : "S003",

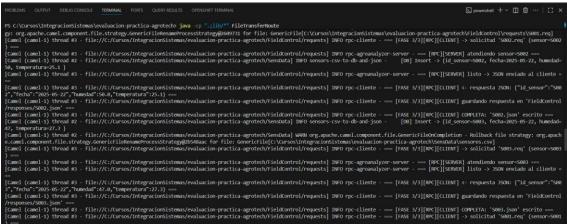
✓ responses

             🖺 S001-20251030210801.done
                                                                                 "fecha": "2025-05-22",
                                                                                 "humedad" : "47",
"temperatura" : "27.3 "
Д
             € $3001.json
             🖺 S002-20251030210801.done
同
              🖺 S003-20251030210801.done
> 📴 lib
         > 📭 logs
```

Fase 3

∨ 📨 SensData





3) Preguntas

1. ¿Qué patrón aplicaste en cada fase del flujo y por qué?

En la Fase 0 (INIT-DB) apliqué un Timer con SQL Gateway para crear/verificar la tabla antes de cualquier proceso. En la Fase 1 (CSV→DB→JSON+Archivo) usé File Inbound y el patrón Pipes & Filters: deserializo CSV, hago split por fila, inserto en BD, vuelvo a marshal a JSON y escribo a disco con archivado del CSV para trazabilidad. En la Fase 3 (RPC simulado) implementé Request-Reply sobre direct: para que el "cliente" pida un sensorld y el "servidor" responda de inmediato con el último registro.

2. ¿Qué riesgos observas al usar una base de datos compartida?

En una BD compartida veo riesgos claros: arranque desordenado si la tabla aún no existe; contención y bloqueos (peor con SQLite) que degradan rendimiento; punto único de falla; posibles inconsistencias por cambios de esquema sin versionar; exposición de datos si no hay controles de acceso finos; y backups/restauraciones más frágiles con múltiples escritores.

3. ¿Cómo ayuda el RPC simulado a representar un flujo síncrono?

El RPC simulado representa un flujo síncrono porque el cliente queda a la espera del resultado en la misma interacción (una solicitud produce una respuesta inmediata), con contrato explícito (input: sensorld; output: JSON) y trazabilidad completa en los logs y en el archivo responses/<sensor>.json. Eso refleja cómo, en un servicio real, el consumidor depende temporalmente del proveedor.

4. ¿Qué limitaciones tienen los patrones clásicos frente a arquitecturas modernas?

Los patrones clásicos se quedan cortos frente a arquitecturas modernas porque escalan peor y son más frágiles ante fallos (no hay colas/eventos ni *back-pressure*), fomentan acoplamiento temporal (cualquier caída rompe la cadena), y carecen de prácticas distribuidas como *Saga/Outbox*, idempotencia o *circuit breakers*. Además, tienen menor observabilidad (sin *tracing* ni métricas estandarizadas) y una gobernanza de contratos más débil frente a enfoques event-driven o microservicios bien instrumentados.