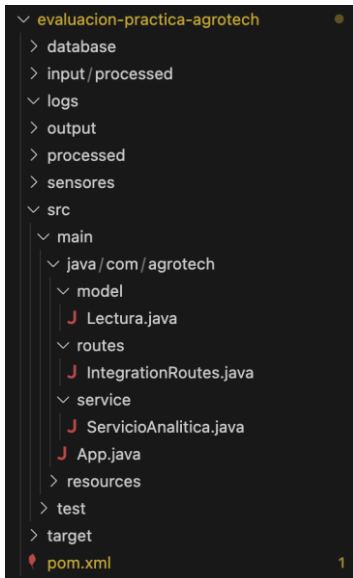


## PRUEBA PROGRESO 1

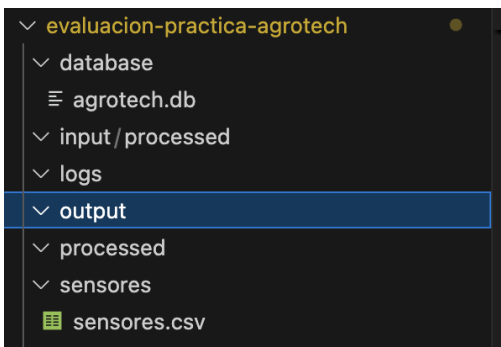
Hayland Montalvo

Repositorio: <https://github.com/hsmg777/Examen-P1-Intergacion-Sistemas-3-patrones.git>

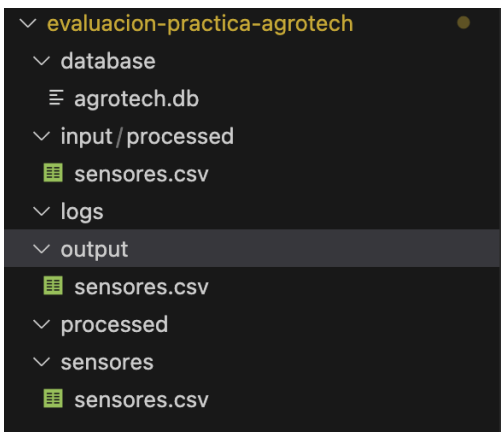
### ARQ DEL PROYECTO:



### Antes del proceso:



### Luego del proceso:



## BD SQLITE:

```
haychis@MacBook-Pro-de-Haychis evaluacion-practica-agrotech % sqlite3 database/agrotech.db 'SELECT * FROM lecturas;'  
S001|2025-05-22T10:00:00Z|45.0|26.4  
S002|2025-05-22T10:05:00Z|50.0|25.1  
S003|2025-05-22T10:10:00Z|47.0|27.3  
haychis@MacBook-Pro-de-Haychis evaluacion-practica-agrotech %
```

## LOGS EVIDENCIA:

```
1363 [Camel (camel-1) thread #2 - timer://bootstrap] INFO route1 - [BOOT] Tabla 'lecturas' lista.  
3173 [Camel (camel-1) thread #3 - timer://rpcTest] INFO rpc-cliente - [CLIENTE] Solicitando S001  
3173 [Camel (camel-1) thread #3 - timer://rpcTest] INFO rpc-servidor - [SERVIDOR] Para S001  
3178 [Camel (camel-1) thread #3 - timer://rpcTest] INFO rpc-cliente - [CLIENTE] Respuesta: {"id":"S001","mensaje":"Sin datos"}  
3178 [Camel (camel-1) thread #3 - timer://rpcTest] INFO route2 - [TEST] RPC S001 OK  
54656 [Camel (camel-1) thread #1 - file://./input] INFO file-transfer - [FILE] Recibido: sensores.csv  
54685 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insertando: {"id_sensor":"S001","fecha":"2025-05-22T10:00:00Z","humedad":45.0,"temperatura":26.4}  
54698 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insert OK  
54698 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insertando: {"id_sensor":"S002","fecha":"2025-05-22T10:05:00Z","humedad":50.0,"temperatura":25.1}  
54699 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insert OK  
54699 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insertando: {"id_sensor":"S003","fecha":"2025-05-22T10:10:00Z","humedad":47.0,"temperatura":27.3}  
54700 [Camel (camel-1) thread #1 - file://./input] INFO agroanalyzer-insert - [ANALYZER] Insert OK
```

## Reflexion individual:

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

1. File Transfer para leer sensores.csv, copiarlo/moverlo y transformar cada fila a JSON; es simple, auditable y mantiene desacoplados al productor y al consumidor.
2. Shared Database guarda las lecturas en lecturas (SQLite) para que distintos módulos consulten el mismo estado sin integraciones punto a punto.
3. RPC simulado con `direct:solicitarLectura` ↔ `direct:rpc.obtenerUltimo` para pedir el último dato “como si fuera local” y obtener una respuesta inmediata, útil cuando el cliente necesita decidir en el momento.

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

Hay alto acoplamiento al esquema, así que cualquier cambio de columnas o constraints puede romper a otros equipos; se generan problemas de contención y rendimiento por locks y competencia de lecturas/escrituras; la gobernanza y seguridad suele ser difusa (no siempre está claro quién puede leer, escribir o borrar) y, además, el versionado entre equipos se vuelve difícil porque no hay una interfaz formal como en una API, lo que complica evolucionar sin afectar a los consumidores.

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

Porque mantiene el modelo solicitud–respuesta en el mismo hilo lógico: el cliente envía una petición y bloquea hasta recibir la respuesta del “servidor”. Además, podemos configurar timeouts, propagar errores y aplicar back-pressure (no saturar al servidor), lo

que lo hace muy parecido a una invocación remota real, pero controlado dentro de Camel para probar lógica de negocio sin exponer servicios externos.

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

En resumen, los patrones clásicos funcionan pero se quedan cortos cuando el sistema crece: con File Transfer suele haber procesamiento por lotes (no en tiempo real), riesgo de duplicados y poca trazabilidad. Con Shared DB todos quedan amarrados al mismo esquema (cualquier cambio rompe a otros), el versionado es difícil y el escalado se complica y en RPC hay dependencia temporal, si el servicio está caído, la llamada falla y la resiliencia es baja. En cambio, enfoques modernos basados en eventos, APIs versionadas y colas permiten desacoplar mejor a los equipos, observar flujos con métricas y trazas, reintentar de forma segura, logrando sistemas más tolerantes a fallos y cercanos al tiempo real.