Informe técnico – DataLab

Introducción

El objetivo de este proyecto es implementar un servicio REST en Spring Boot capaz de procesar trabajos de análisis de datos de forma concurrente.
Cada trabajo (*Job*) se divide en varias tareas (*Tasks*) que se ejecutan en paralelo sobre un dataset CSV, simulando un sistema real de procesamiento distribuido.

El proyecto no se limita a ejecutar tareas en paralelo, sino que incorpora control de estados, transacciones, auditoría, métricas y reintentos automáticos mediante AOP.

Diseño general y arquitectura

La arquitectura del sistema sigue una separación clara por capas:

- Controller: expone la API REST y valida entradas.
- Service: contiene la lógica de negocio y la concurrencia.
- Domain: define el modelo de datos (Job, Task, Result).
- Repository: acceso a datos mediante Spring Data JPA.
- Aspects: funcionalidades transversales (métricas y reintentos).
- Config: configuración del executor, async y perfiles.

Los controladores no contienen lógica de negocio. Toda la orquestación y ejecución se realiza en la capa de servicios.

Modelo de concurrencia

Cada Job se divide en N Tasks, una por cada *shard* del dataset.
Las Tasks se ejecutan en paralelo utilizando:

- ThreadPoolTaskExecutor configurable.
- @EnableAsync y @Async.
- CompletableFuture para coordinar la ejecución fan-out / fan-in.

El servicio JobProcessor lanza todas las Tasks de forma asíncrona y espera a su finalización mediante CompletableFuture.allOf().join().
Una vez completadas, se calcula el estado final del Job.

La cancelación se implementa de forma cooperativa mediante un flag en el Job, que las Tasks consultan antes de iniciar su ejecución.

Programación Concurrente – Feedback 1

Gestión transaccional

La creación del Job y de sus Tasks se realiza dentro de una transacción.
Las consultas de resultados se ejecutan en modo readOnly.

Para la auditoría de errores se utiliza un servicio específico anotado con @Transactional(REQUIRES_NEW), lo que permite registrar fallos sin afectar a la transacción principal de la Task.

AOP: métricas y reintentos

Se han implementado dos aspectos principales:

- PerformanceAspect: mide el tiempo de ejecución de métodos de service y repository, añadiendo información contextual mediante MDC y un traceId.
- RetryOnTransientErrorAspect: reintenta automáticamente la ejecución de una Task cuando se produce una excepción marcada como transitoria, usando una anotación personalizada @RetryableTransient y un backoff exponencial simple.

Esto permite separar completamente las preocupaciones transversales del código de negocio.

Estrategia de análisis (IoC / DI)

El análisis del dataset se implementa mediante el patrón Strategy:

- AnalyzerStrategy como interfaz común.
- SimpleAnalyzer como implementación por defecto.
- AdvancedAnalyzer activado únicamente en el perfil prod.

Esto permite cambiar el tipo de análisis sin modificar el resto del sistema, demostrando un uso real de IoC y DI.

Perfiles y persistencia

El proyecto define dos perfiles:

- dev: H2 en memoria, con creación automática de esquema.
- prod: PostgreSQL con pool de conexiones HikariCP y validación de esquema.

Ambos perfiles comparten el mismo código, cambiando únicamente la configuración.