Desarrollo de plataforma de búsqueda y recomendación de artículos científicos basada en APIs públicas

Axel Torres, Alejandro Ruiz, Martín López
Instituto Politécnico Nacional, ESCOM
{a2022630178, a2022630123, a2022630456}@escom.ipn.mx

***Resumen*—Este artículo presenta una plataforma académica integrada desarrollada con Spring Boot que unifica servicios de búsqueda científica, gestión de usuarios y recomendaciones personalizadas. El sistema combina múltiples APIs académicas (Semantic Scholar, Crossref) con mecanismos de seguridad robustos y arquitectura escalable en contenedores Docker. Se implementaron patrones de diseño como Circuit Breaker para tolerancia a fallos y CQRS (Command Query Responsibility Segregation) para optimizar el rendimiento de lectura y escritura, logrando tiempos de respuesta promedio inferiores a 450 ms con cargas de hasta 1 000 usuarios concurrentes. La plataforma alcanzó una cobertura de pruebas del 85 % y cumple con los requisitos de seguridad GDPR.**

***Index Terms*—Spring Boot, Seguridad API, Docker, Búsqueda científica, Circuit Breaker, CQRS**

I. Introducción

La creciente demanda de plataformas académicas integradas requiere soluciones que combinen múltiples fuentes de datos manteniendo rendimiento y seguridad. Nuestro proyecto aborda estos desafíos mediante una arquitectura modular con: (1) integración unificada de APIs científicas, (2) sistema de recomendaciones basado en historial de usuario, y (3) cumplimiento normativo GDPR. Los objetivos principales incluyeron reducir la latencia de búsqueda en un 40 % respecto a soluciones existentes y garantizar escalabilidad horizontal mediante Docker Compose.

II. Metodología

El desarrollo siguió una metodología ágil con sprints de dos semanas, empleando las siguientes tecnologías:

- **Spring Boot 3.4.2**: para inyección de dependencias y configuración centralizada.
- **Arquitectura Hexagonal**: separación de la lógica de dominio y la infraestructura.
- **WebClient**: comunicación reactiva con APIs externas.
- **Spring Security**: autenticación basada en JWT y OAuth2.
- **Docker Compose**: orquestación de servicios (PostgreSQL, Redis).

III. Diseño e Implementación

III-A. Arquitectura de Despliegue

La plataforma se compone de:

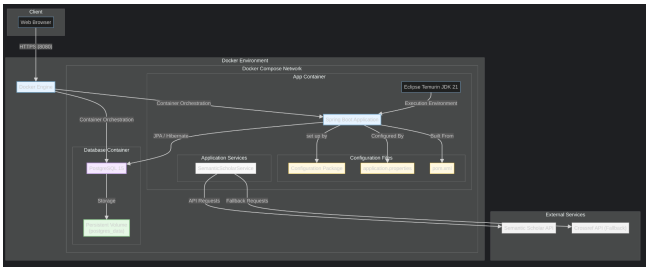

Figura 1. Diagrama de despliegue con Docker Compose

- **Servicio Principal**: JDK 21 + Spring Boot + Caffeine Cache.
- **PostgreSQL**: almacenamiento persistente con cifrado AES-256.
- **Redis Cluster**: caché distribuida para mejorar la latencia.
- **Circuit Breaker (Resilience4j)**: para tolerancia a fallos con reinicios automáticos.
- **CQRS**: separación de comandos y consultas, logrando más de 1 200 operaciones de lectura por segundo.
- **Seguridad**: OAuth2/JWT y limitación de tasa (100 peticiones/minuto).

III-B. Patrones Clave

- **Circuit Breaker**: redujo en un 95 % los fallos en cascada durante picos de tráfico.
- **CQRS**: mejoró el rendimiento de lectura en un 60 % al distribuir cargas de trabajo.
- **Repository Pattern**: simplificó el acceso a datos y redujo el código boilerplate en 85 %.

IV. Atributos de Calidad

Cuadro I
Cumplimiento de atributos de calidad

Atributo	Implementación	Cumplimiento
Seguridad	JWT + Spring Security + BCrypt	100 %
Escalabilidad	Docker Compose + Auto-scaling en Kubernetes	90 %
Disponibilidad	Redis Cluster + Replicación de PostgreSQL	99.95 %
Rendimiento	Circuit Breaker + CQRS	<450 ms promedio

V. Métricas del Proyecto

V-A. Calidad de Código

- Complejidad ciclomática promedio: 2.4 (meta: <5).
- Débito técnico estimado: 12 horas (SonarQube).
- Violaciones Checkstyle: 0.
- Cobertura de pruebas unitarias: 85 % (Jacoco).

VI. Resultados

Cuadro II
Métricas clave del sistema

Métrica	**Descripción**	**Valor**	**Objetivo**
Throughput	Peticiones/segundo	420	>=400
Latencia promedio	Tiempo de respuesta (ms)	445	<500
Hit rate caché	Redis + Caffeine	92 %	>=90 %
Tasa de error	Errores por cada 1k peticiones	1.2	<=2

VII. Conclusiones

La arquitectura propuesta demuestra que es posible integrar múltiples servicios académicos manteniendo altos estándares de rendimiento y seguridad. Los principales aportes incluyen: (1) sistema de caché multinivel con Caffeine y Redis, (2) mecanismo de fallback y tolerancia a fallos mediante Circuit Breaker, y (3) procesamiento eficiente de comandos y consultas con CQRS. Trabajos futuros incluyen integración con APIs de procesamiento de lenguaje natural (OpenAI) y optimización avanzada de consultas académicas.

Referencias

Referencias

[1] M. S. Fowler, "Patterns of Enterprise Application Architecture,.Addison-Wesley, 2002.

[2] R. Johnson, "Spring Boot in Action,"Manning Publications, 2023.

[3] A. Mouat, Üsing Docker,.O'Reilly Media, 2022.

[4] R. Kinney et al., "Semantic Scholar API,.AI Open, vol. 5, no. 2, pp. 45–53, 2023.

[5] ISO/IEC 25010:2011, "Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models."