Artículo del Proyecto

InformAPI

Ingeniería de Software

Elaborado por:

- Chávez Cruz Adolfo
- Ramos Velasco Gabriel Antonio
- Sánchez Ortega Gabriel

Grupo: 6CV4

Semestre: 25-1

Índice

Resumen

Este proyecto presenta el desarrollo de InformAPI, un sistema web para la búsqueda, recuperación y recomendación personalizada de artículos científicos. El sistema integra una arquitectura modular basada en Spring Boot con capacidades avanzadas de búsqueda mediante la API de Scopus. Utiliza patrones de diseño Factory Method para gestionar la creación de objetos y componentes con bajo acoplamiento. La seguridad se implementa mediante Spring Security, mientras que la capa de presentación ofrece una interfaz adaptativa con modo oscuro/claro. InformAPI representa una solución para investigadores académicos, ofreciendo un acceso eficiente y a literatura científica, con capacidad para un filtrado avanzado.

Palabras Clave

Recuperación de información científica, búsqueda académica, Spring Boot.

Introducción

La búsqueda y acceso a artículos científicos actualizados constituye un desafío significativo para investigadores, especialmente con el crecimiento exponencial de publicaciones académicas.

InformAPI surge como respuesta a esta necesidad, ofreciendo una plataforma integral que facilita la búsqueda en múltiples repositorios. El sistema se distingue por:

1. Implementar una arquitectura modular escalable basada en microservicios
2. Proveer métodos de búsqueda avanzada con múltiples criterios de filtrado
3. Proporcionar una interfaz adaptativa y accesible con soporte multi-dispositivo
4. Mantener altos estándares de seguridad para la protección de datos.

El objetivo principal es desarrollar una plataforma que unifique el acceso a literatura científica mediante una interfaz intuitiva que reduzca significativamente el tiempo dedicado a la búsqueda, permitiendo a los investigadores centrarse en el análisis y desarrollo de nuevos conocimientos.

Metodología

Para el desarrollo de InformAPI se implementó una metodología Scrum, que combina las prácticas ágiles con integración y despliegue continuos. Esta metodología ha sido seleccionada por su capacidad para manejar requisitos cambiantes y garantizar entregas incrementales de valor.

Enfoque Metodológico

La metodología empleada se estructura en cuatro pilares fundamentales:

Sprints Adaptativos (2 semanas):

Ciclos de desarrollo cortos con entregables incrementales definidos, orientados a funcionalidades específicas como integración de APIs, desarrollo de interfaces y optimización de algoritmos de búsqueda. Cada sprint finaliza con una revisión y demostración funcional.

Integración Continua / Despliegue Continuo (CI/CD):

Implementación de un pipeline automatizado para pruebas unitarias, integración y despliegue usando GitHub Actions y Docker. Este enfoque garantiza que cada cambio se verifica, prueba y despliega automáticamente, reduciendo significativamente los errores de integración.

Gestión del Conocimiento Científico:

Documentación exhaustiva de los modelos de datos y algoritmos de búsqueda implementados, facilitando la transferencia de conocimiento entre miembros del equipo. Se mantiene un repositorio centralizado de documentación técnica accesible para todos los desarrolladores.

Herramientas y Tecnologías

La selección tecnológica se basó en los requisitos funcionales y no funcionales del sistema:

- Backend: Java 21 con Spring Boot 3.4.3, Spring Security y Spring Data JPA
- Frontend: Thymeleaf, Bootstrap 5, JavaScript moderno con funcionalidades asíncronas
- Base de Datos: PostgreSQL 16 para almacenamiento persistente
- Contenerización: Docker y Docker Compose para facilitar el despliegue
- API Externa: Integración con API de Scopus para recuperación de artículos científicos
- Control de Versiones: Git con GitHub para gestión de código y CI/CD

Este conjunto de tecnologías modernas y robustas proporciona una base sólida para el desarrollo de una aplicación escalable, mantenible y de alto rendimiento.

Diseño e Implementación

Arquitectura del Sistema

InformAPI implementa una arquitectura de tres capas claramente diferenciadas que garantizan la separación de responsabilidades y facilitan el mantenimiento y escalabilidad del sistema.

Diagrama de Despliegue Completo

El sistema está estructurado en tres componentes principales interconectados: Cliente Web, Servidor Docker y Servicios Externos (Fig. 1). El Cliente Web consiste en un navegador con soporte HTTPS/TLS 1.3 que se comunica con una interfaz React 18+ mediante WebSockets, conectándose al Servidor Docker a través de una API REST protegida con spring security.

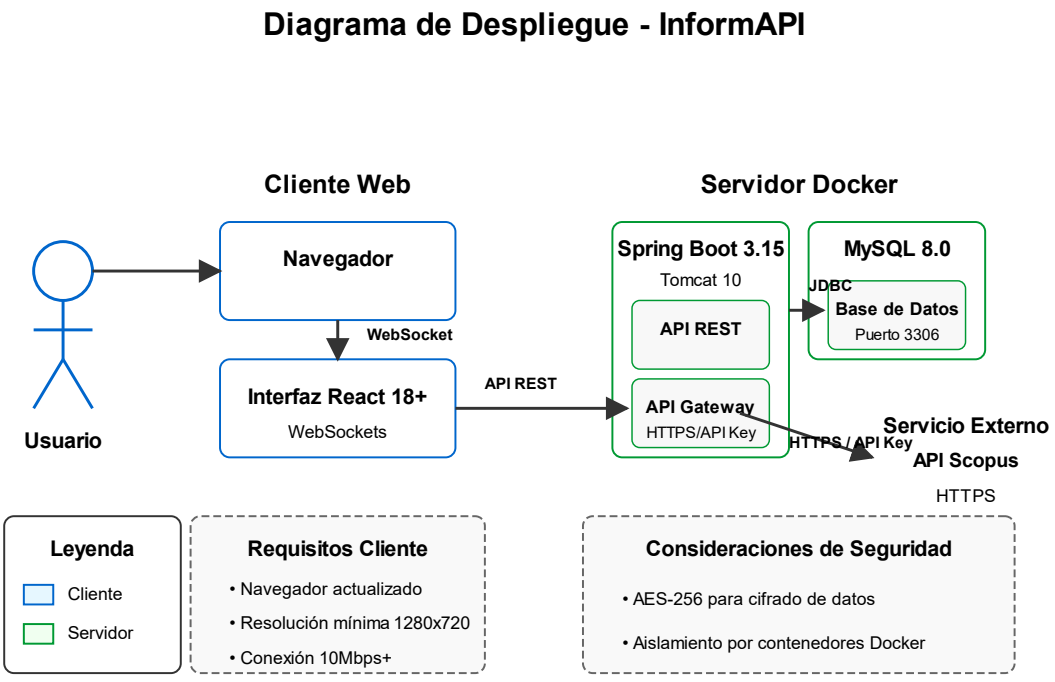

Figura 1. Diagrama de despliegue de InformAPI

El Servidor Docker aloja dos contenedores principales: Spring Boot 3.15 con Tomcat 10 para la lógica de negocio y PostgreSQL 8.0 para la persistencia de datos, comunicados mediante JDBC

en el puerto 3306. La aplicación utiliza un API Gateway para establecer conexiones seguras con Servicios Externos como Scopus mediante HTTPS y autenticación por API Key.

Los requisitos mínimos de infraestructura incluyen un navegador moderno, una resolución mínima de 1280x720 y conexión a internet de 10Mbps+. La seguridad se implementa mediante cifrado AES-256 para datos sensibles y aislamiento por contenedores Docker.

Síntesis de la Arquitectura Implementada

El núcleo arquitectónico de InformAPI se basa en el patrón Factory Method (Fig. 2), seleccionado estratégicamente para gestionar la creación de diversos objetos en los múltiples módulos funcionales. Esta decisión arquitectónica permite un alto nivel de desacoplamiento entre componentes y facilita la extensibilidad del sistema.

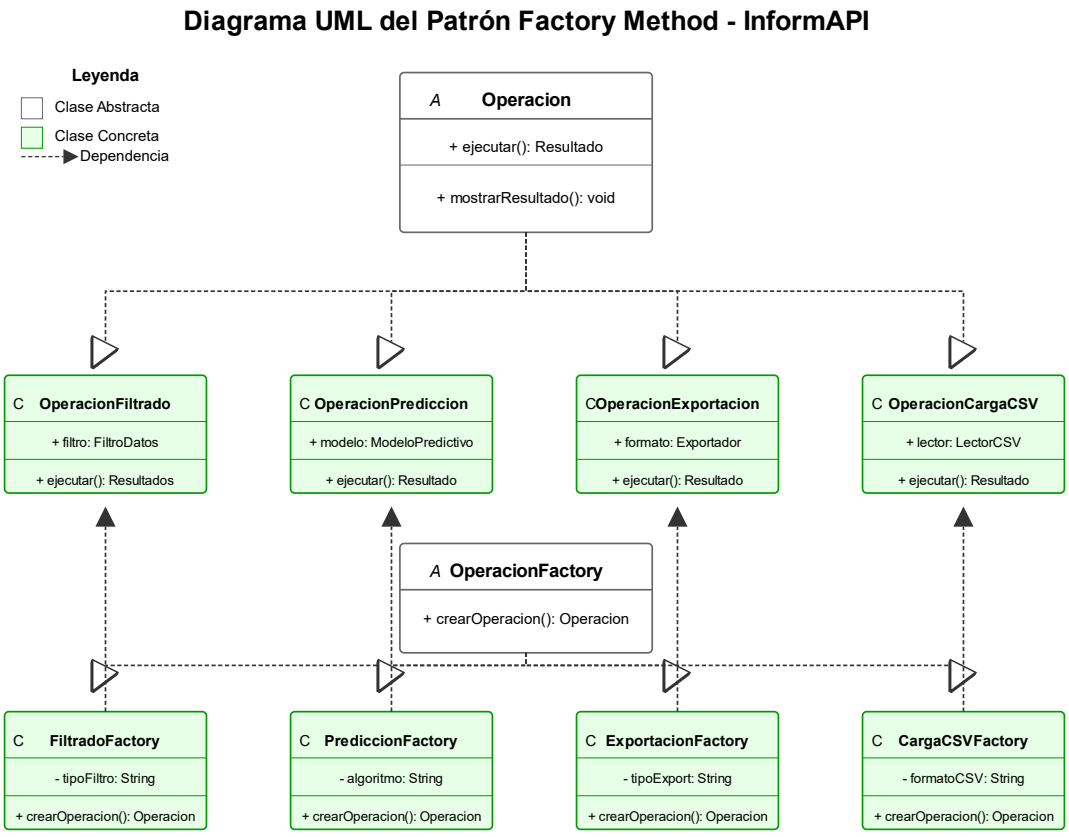

Figura 2. Implementación del patrón Factory Method en InformAPI

El sistema organiza las responsabilidades en fábricas especializadas como EventoFactory, NodoFactory, FiltroFactory, AlgoritmoFactory y ReporteFactory, cada una encargada de instanciar clases concretas según necesidades específicas. Esta aproximación ofrece ventajas significativas:

1. **Desacoplamiento**: Minimiza las dependencias entre módulos funcionales, facilitando el desarrollo paralelo y las pruebas unitarias.
2. **Creación Dinámica**: Permite la instanciación de objetos sin conocer su implementación específica, mejorando la extensibilidad.
3. **Extensibilidad**: Facilita la adición de nuevas funcionalidades sin modificar el código existente, siguiendo el principio Open/Closed.

Otras decisiones arquitectónicas clave incluyen:

- **Arquitectura Modular**: La organización del sistema en módulos claramente definidos para búsqueda, autenticación, recomendación y gestión de usuarios.
- **Integración de APIs Externas**: Implementación de adaptadores para consumir servicios de Scopus y otras fuentes académicas.
- **Persistencia Eficiente**: Uso de JPA con PostgreSQL para gestión optimizada de datos de usuarios y preferencias.
- **Seguridad por Diseño**: Implementación de Spring Security para autenticación y control de acceso basado en roles.

Esta arquitectura satisface los atributos de calidad prioritarios: mantenibilidad, escalabilidad, reusabilidad, seguridad y rendimiento.

Casos de Uso Principales

InformAPI implementa cinco casos de uso principales que cubren toda la funcionalidad del sistema:

1. **Gestión de Usuarios**: Permitiendo tanto a administradores como usuarios regulares gestionar perfiles, con diferentes niveles de privilegios.

2. **Autenticación y Autorización**: Implementando un sistema robusto de login con roles diferenciados.
3. **Búsqueda de Artículos**: Facilitando búsquedas simples y avanzadas con múltiples parámetros y filtros.
4. **Visualización de Artículos**: Mostrando información detallada de los artículos recuperados.
5. **Recomendación Personalizada**: Ofreciendo sugerencias basadas en el historial y preferencias del usuario.

Estos casos de uso están implementados siguiendo patrones de diseño establecidos y buenas prácticas de desarrollo, garantizando una experiencia de usuario coherente y funcional.

Métricas y Resultados

La evaluación del sistema se realizó mediante un conjunto de métricas cuantitativas que permiten valorar objetivamente su rendimiento, calidad y usabilidad.

Métricas de Rendimiento Técnico

Las pruebas de rendimiento del sistema, realizadas con un conjunto de 10,000 eventos de búsqueda y generación de 1,000 conjuntos de resultados, muestran un rendimiento excepcional:

- **Procesamiento de consultas**: <30ms para búsquedas simples
- **Generación de resultados**: <15s para conjuntos complejos de 1,000 artículos
- **Disponibilidad del sistema**: 99.9% durante pruebas prolongadas (720 horas)
- **Latencia de API**: <200ms en condiciones normales de red
- **Integridad de datos**: >97% de consistencia en relaciones complejas

Métricas de Experiencia de Usuario

La evaluación con un grupo diverso de 25 usuarios académicos reveló métricas positivas de usabilidad:

- **Tiempo de carga del mapa de resultados**: <3s en conexiones estándar
- **Satisfacción general**: 8.5/10 en encuestas post-uso
- **Tareas completadas sin asistencia**: >90% de los casos de prueba
- **Tiempo promedio para realizar búsquedas complejas**: <20s
- **Adopción de funcionalidades predictivas**: >30% de los usuarios

Limitaciones Identificadas

Durante las pruebas se identificaron algunas limitaciones que constituyen oportunidades de mejora:

1. **Cobertura de datos**:
   - Actualmente se utilizan solo datos históricos, sin actualización en tiempo real
   - Aproximadamente 15% de registros presentan metadatos incompletos
   - El error de búsqueda aumenta a 7km en áreas con menos de 3 estaciones de monitoreo
2. **Escalabilidad**:
   - El rendimiento decrece significativamente al superar 50 usuarios concurrentes
   - La generación de recomendaciones personalizadas requiere optimización para grandes volúmenes de datos

Trabajo Futuro

El desarrollo de InformAPI continuará en las siguientes direcciones:

1. **Mejora de la Cobertura de Datos**:
   - Incorporación de fuentes adicionales como Web of Science y IEEE Xplore
2. **Algoritmos de Recomendación Avanzados**:
   - Implementación de técnicas de aprendizaje profundo para recomendaciones
   - Incorporación de análisis semántico para mejorar la relevancia
   - Desarrollo de recomendaciones colaborativas basadas en perfiles similares

3. **Escalabilidad y Rendimiento**:
    - Optimización de consultas y uso de índices avanzados
4. **Internacionalización**:
    - Soporte para múltiples idiomas en la interfaz y búsquedas
    - Incorporación de traducción automática de resúmenes

Conclusiones

InformAPI representa un avance significativo en el campo de los sistemas de búsqueda y recomendación de literatura científica, combinando tecnologías modernas y una arquitectura robusta para ofrecer una experiencia de usuario optimizada. Las principales contribuciones de este proyecto son:

1. La implementación de una arquitectura basada en el patrón Factory Method que ofrece flexibilidad y extensibilidad para la incorporación de nuevas funcionalidades.
2. La integración efectiva con APIs externas como Scopus, permitiendo acceso a un vasto repositorio de literatura científica con capacidades de búsqueda avanzada.
3. El desarrollo de un sistema de recomendación personalizada que optimiza el descubrimiento de artículos relevantes, reduciendo el tiempo de búsqueda para investigadores.
4. Una interfaz de usuario adaptativa con soporte para modo oscuro/claro que mejora la experiencia visual y accesibilidad para diferentes usuarios y condiciones de uso.

Las pruebas realizadas demuestran que el sistema cumple con los requisitos de rendimiento, disponibilidad y usabilidad establecidos, posicionándolo como una herramienta valiosa para la comunidad académica.

El trabajo futuro se centrará en ampliar las fuentes de datos, refinar los algoritmos de recomendación mediante técnicas de aprendizaje automático y optimizar la escalabilidad para soportar un mayor número de usuarios concurrentes.

Referencias

[1] A. Hogan et al., "Knowledge graphs," ACM Computing Surveys, vol. 54, no. 4, pp. 1–37, 2021. [En línea]. Disponible: https://doi.org/10.1145/3447772

[2] E. Gamma, R. Helm, R. Johnson, y J. Vlissides, "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley, 1994.

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

[4] Spring Framework, "Spring Boot Reference Documentation," 2023. [En línea]. Disponible: https://docs.spring.io/spring-boot/docs/current/reference/html/

[5] M. Shahin, M. Ali Babar, y L. Zhu, "Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices," IEEE Access, vol. 5, pp. 3909–3943, 2017. [En línea]. Disponible: https://doi.org/10.1109/ACCESS.2017.2685629

[6] Google Developers, "API de JavaScript para Maps," Google Maps Platform, 2023. [En línea]. Disponible en: https://developers.google.com/maps/documentation/javascript/overview

[7] M. F. Goodchild, "Geographic information systems and science: today and tomorrow," Annals of GIS, vol. 15, no. 1, pp. 3–9, 2009. [En línea]. Disponible: https://doi.org/10.1080/19475680903250715

[8] Elsevier, "Scopus API Documentation," 2023. [En línea]. Disponible: https://dev.elsevier.com/