Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Bases de Datos - GR 20

Reporte Técnico: Sistema de Búsqueda Inteligente con Elasticsearch

Barboza Alfaro Luis Daniel
2024211470
Castro Cruz José Mario
2022437423

Profesor: Maria Auxiliadora Mora Cross

II Semestre, 2025

30 de octubre de 2025

El presente documento es un reporte técnico donde se describe el desarrollo del Sistema de Búsqueda Inteligente con Elasticsearch,

Objetivo del proyecto

Desarrollar un sistema de búsqueda eficiente utilizando Elasticsearch, una base de datos orientada a búsqueda y análisis de texto. El proyecto demostrará cómo indexar información y realizar búsquedas avanzadas por texto completo, filtros, relevancia y autocompletado, destacando las ventajas de Elasticsearch frente a bases de datos tradicionales.

Descripción general

Elasticsearch es un motor de búsqueda distribuido basado en Lucene, diseñado para realizar búsquedas rápidas y complejas sobre grandes volúmenes de datos, especialmente texto.

Este proyecto propone crear un motor de búsqueda para una Biblioteca Digital (Libros, artículos)

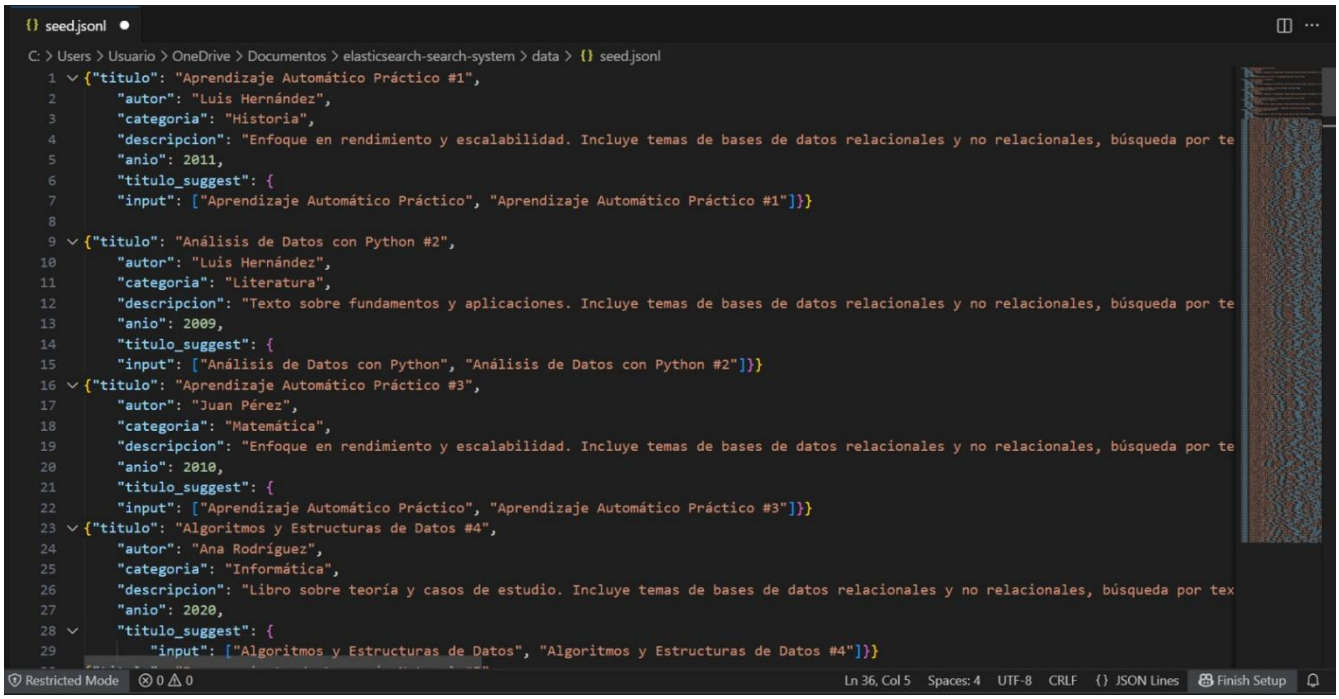
El sistema permite búsquedas flexibles, con tolerancia a errores tipográficos, filtros por categorías y ordenamiento por relevancia.

Componentes del proyecto

1. Definición del dataset

Se generó un dataset sintético con 300 documentos en formato JSONL, conteniendo los campos: *titulo, autor, categoria, descripcion y anio*.

Cada documento fue diseñado para simular libros reales con diferentes categorías como *Informática, Historia, Matemática, y más*.

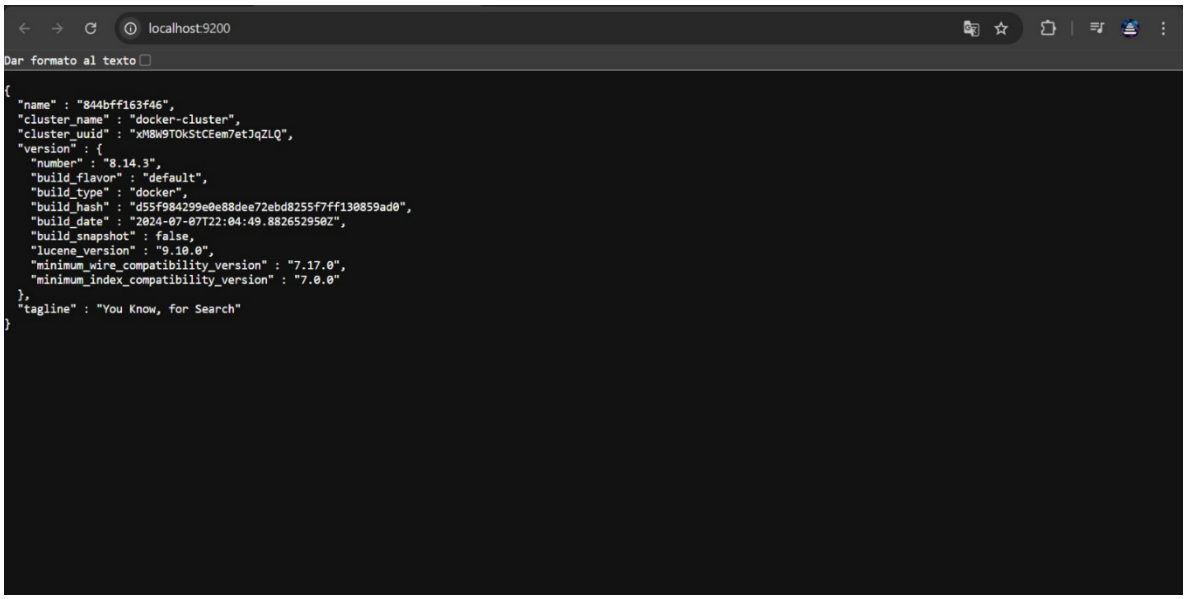


2. Configuración de Elasticsearch

Se utilizó Docker para ejecutar dos contenedores:

- elasticsearch como motor principal.
- kibana como entorno de administración y consultas.

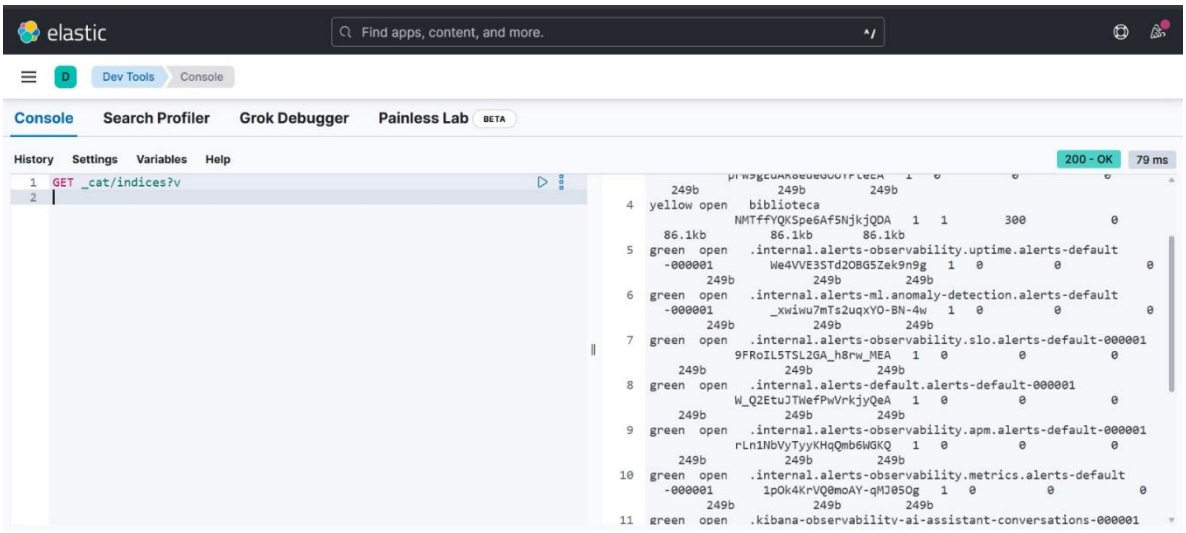
Contenedores Docker activos (Elasticsearch y Kibana).



Verificación de conexión en http://localhost:9200.

Una vez iniciado el contenedor, se verificó la conexión y versión del nodo, confirmando la instalación exitosa. Luego, se creó el índice biblioteca mediante el siguiente mapeo:

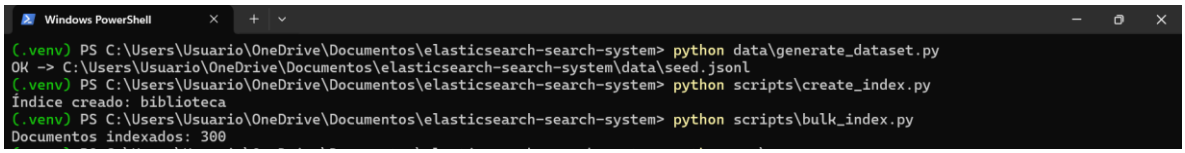
```
PUT /biblioteca
{
  "mappings": {
    "properties": {
      "titulo": { "type": "text" },
      "autor": { "type": "text" },
      "categoria": { "type": "keyword" },
      "descripcion": { "type": "text" },
      "anio": { "type": "integer" }
    }
  }
}
```



3. Indexación de Datos

Los documentos del dataset fueron cargados al índice utilizando scripts en Python, empleando la librería elasticsearch-py.
Durante la carga se mostraron los mensajes de confirmación:

- “Índice creado: biblioteca”
- “Documentos indexados: 300”

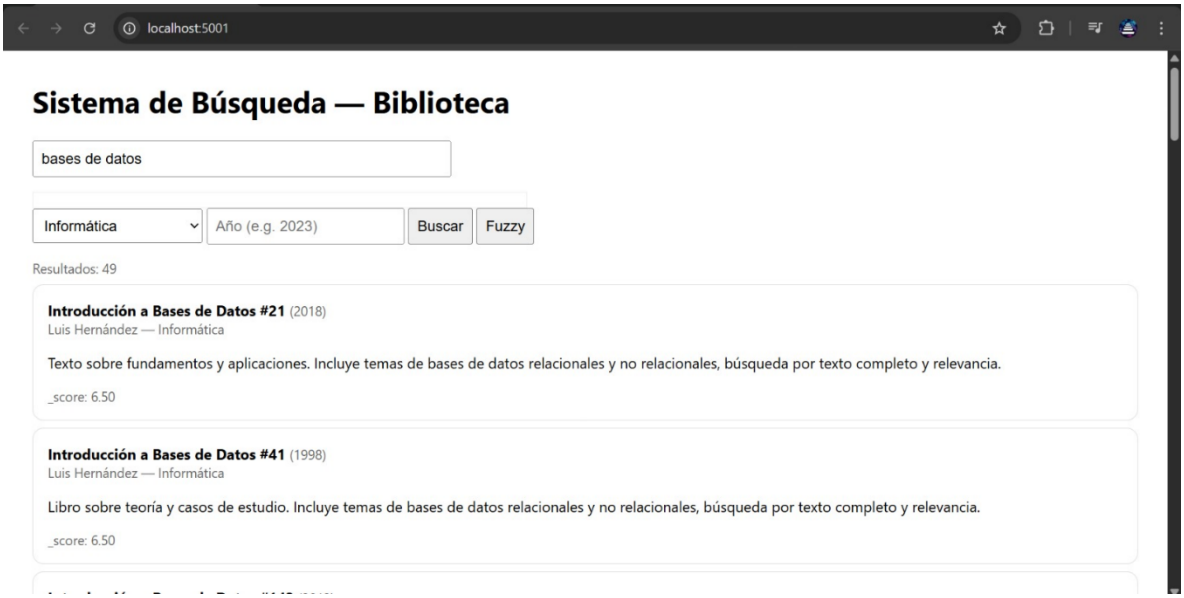


4. Consultas de Búsqueda

Se implementaron diferentes tipos de búsqueda a través de una interfaz web desarrollada con Flask + HTML.

a. Búsqueda por texto completo

Permite consultar libros que contengan una palabra o frase en el título o descripción.



b. Búsqueda difusa (Fuzzy Search)

Tolerante a errores tipográficos, como en la búsqueda de *“basses de datos”* en lugar de *“bases de datos”*. El sistema corrige automáticamente y devuelve los resultados esperados. Además, se hace uso de filtros combinados (se buscan solo de informática).

c. Autocompletado

Conforme el usuario escribe, el sistema sugiere títulos mediante el campo titulo_suggest, proporcionando una experiencia de búsqueda más fluida.

5. Visualización

Además de la interfaz web, se utilizó Kibana para verificar la estructura de los índices y realizar consultas directas sobre los documentos.

6. Resultados Esperados

El sistema cumple con los objetivos planteados:

- Búsqueda por texto completo.
- Búsqueda fuzzy con tolerancia a errores.
- Filtros por categoría y año.
- Autocompletado dinámico.
- Resultados ordenados por relevancia (*score*).

Todas las funcionalidades se ejecutan con tiempos de respuesta inferiores a un segundo.

7. Aspectos Técnicos Adicionales

- **Cómo funciona la indexación**
  La indexación en Elasticsearch convierte los documentos JSON en estructuras optimizadas denominadas índices invertidos.
  Cada documento se analiza, se tokeniza y se normaliza, permitiendo

búsquedas rápidas sin recorrer registro por registro.
Gracias a este proceso, Elasticsearch puede responder consultas complejas en milisegundos, incluso con miles de documentos.

- **Diferencias clave con SQL tradicional**

Aspecto	**Elasticsearch**	**SQL tradicional**
Modelo de datos	Documentos JSON dentro de índices	Tablas con filas y columnas
Lenguaje de consulta	DSL (Domain Specific Language)	SQL estándar
Esquema	Dinámico y flexible	Estructura fija
Velocidad de búsqueda	Alta, mediante índices invertidos	Más lenta, requiere escaneo de tablas
Relaciones	No relacional	Basado en claves primarias y foráneas

Elasticsearch se especializa en la búsqueda y análisis de texto, mientras que SQL se orienta al manejo de relaciones y consistencia de datos.

- **Casos de uso reales de Elasticsearch**
  - **Motores de búsqueda:**
    Empresas como *Wikipedia* o *GitHub* utilizan Elasticsearch para ofrecer resultados instantáneos mientras el usuario escribe.
  - **Monitoreo y análisis de logs:**
    En conjunto con *Logstash* y *Kibana* (stack ELK), permite analizar registros del sistema en tiempo real.
  - **Recomendadores de contenido:**
    Plataformas como *Netflix* o *Airbnb* lo usan para sugerir contenido o resultados personalizados.

Conclusiones

El proyecto demuestra la eficiencia de Elasticsearch para búsquedas inteligentes sobre texto, mostrando gran velocidad, flexibilidad y relevancia en los resultados.
Comparado con bases SQL tradicionales, ofrece una arquitectura más escalable y flexible, ideal para sistemas de búsqueda, análisis y recomendación.

La integración con Flask proporciona una interfaz simple y funcional, permitiendo demostrar todas las características implementadas con éxito.

Herramientas Utilizadas

- Elasticsearch 8.14.3 – Motor de búsqueda principal.
- Kibana 8.14.3 – Entorno de visualización y consultas.
- Python 3.11 + Flask – Backend e interfaz web.
- Docker Desktop – Contenedores para Elasticsearch y Kibana.
- Visual Studio Code – Entorno de desarrollo.

El sistema desarrollado puede encontrarse en el siguiente enlace:

https://github.com/DanielBA05/SistemaElastiSearch.git