



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.

```
() seedjsonl •
C: > Users > Usuario > OneDrive > Documentos > elasticsearch-search-system > data > {} seedjsonl
1  {"titulo": "Aprendizaje Automático Práctico #1",
2    "autor": "Luis Hernández",
3    "categoria": "Historia",
4    "descripcion": "Enfoque en rendimiento y escalabilidad. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por te
5    "anio": 2011,
6    "titulo_suggest": {
7      "input": ["Aprendizaje Automático Práctico", "Aprendizaje Automático Práctico #1"]}
8
9  {"titulo": "Análisis de Datos con Python #2",
10   "autor": "Luis Hernández",
11   "categoria": "Literatura",
12   "descripcion": "Texto sobre fundamentos y aplicaciones. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por te
13   "anio": 2009,
14   "titulo_suggest": {
15     "input": ["Análisis de Datos con Python", "Análisis de Datos con Python #2"]}
16
17 {"titulo": "Aprendizaje Automático Práctico #3",
18   "autor": "Juan Pérez",
19   "categoria": "Matemática",
20   "descripcion": "Enfoque en rendimiento y escalabilidad. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por te
21   "anio": 2010,
22   "titulo_suggest": {
23     "input": ["Aprendizaje Automático Práctico", "Aprendizaje Automático Práctico #3"]}
24
25 {"titulo": "Algoritmos y Estructuras de Datos #4",
26   "autor": "Ana Rodríguez",
27   "categoria": "Informática",
28   "descripcion": "Libro sobre teoría y casos de estudio. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por tex
29   "anio": 2020,
30   "titulo_suggest": {
31     "input": ["Algoritmos y Estructuras de Datos", "Algoritmos y Estructuras de Datos #4"]}
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

2. Configuración de Elasticsearch

Se utilizó Docker para ejecutar dos contenedores:

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

```
PS C:\Users\Usuario\OneDrive\Documentos\elasticsearch-search-system> docker ps
```

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS
6cca03adf4d2	docker.elastic.co/kibana/kibana:8.14.3	kibana-local	"/bin/tini -- /usr/l..."	5 minutes ago	Up 5 minutes
844bfff163f46	docker.elastic.co/elasticsearch/elasticsearch:8.14.3	es-local	"/bin/tini -- /usr/l..."	5 minutes ago	Up 5 minutes (healthy)

Contenedores Docker activos (Elasticsearch y Kibana).

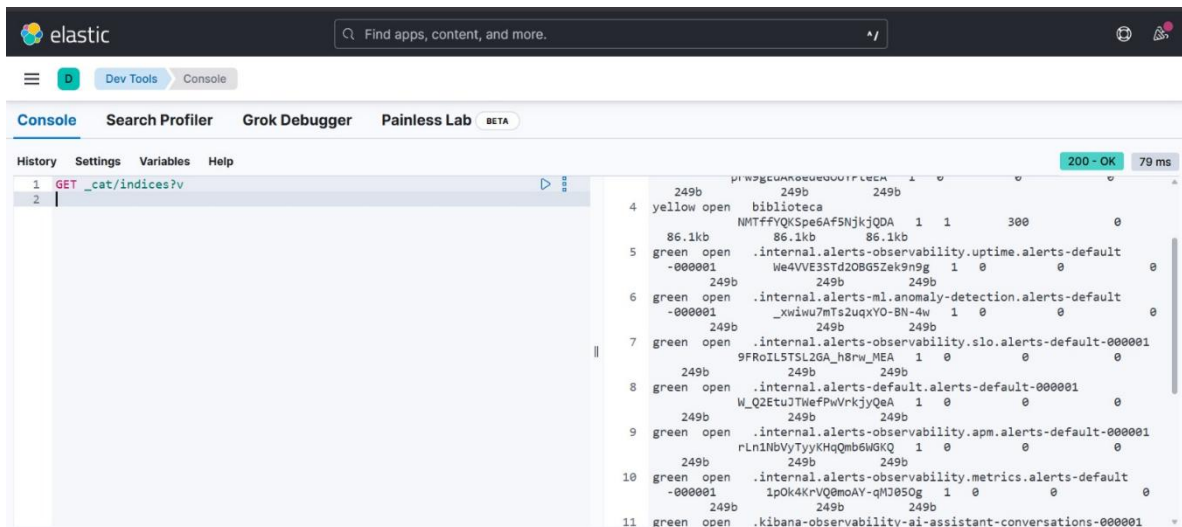
```
localhost:9200
Dar formato al texto
{
  "name" : "844bfff163f46",
  "cluster_name" : "docker-cluster",
  "cluster_uuid" : "xM8W9T0kStCEm7etJqZLQ",
  "version" : {
    "number" : "8.14.3",
    "build_flavor" : "default",
    "build_type" : "docker",
    "build_hash" : "d55f984299e88de72ebd8255f7ff130859ad0",
    "build_date" : "2024-07-07T22:04:49.882652950z",
    "build_snapshot" : false,
    "lucene_version" : "9.10.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

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" }
    }
  }
}
```

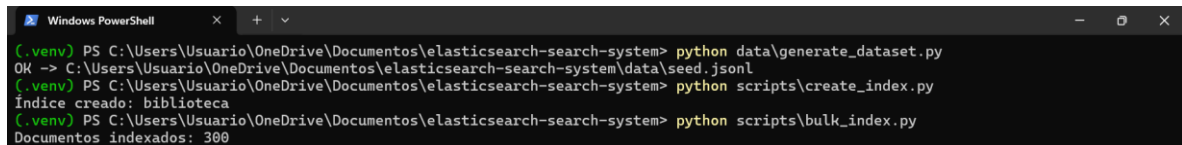


3. Indexación de Datos

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.



Sistema de Búsqueda — Biblioteca

Todas las categorías ▾

Año (e.g. 2023)

Buscar

Fuzzy

a. Búsqueda por texto completo

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

The screenshot shows a web browser window with the address bar displaying 'localhost:5001'. The page title is 'Sistema de Búsqueda — Biblioteca'. Below the title is a search bar containing the text 'bases de datos'. Underneath the search bar is a filter section with a dropdown menu set to 'Informática', a text input field with 'Año (e.g. 2023)', and two buttons labeled 'Buscar' and 'Fuzzy'. Below the filter section, it says 'Resultados: 49'. The search results are displayed in a list of cards. The first card is for 'Introducción a Bases de Datos #21 (2018)' by Luis Hernández — Informática, with a description: 'Texto sobre fundamentos y aplicaciones. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por texto completo y relevancia.' and a score of 6.50. The second card is for 'Introducción a Bases de Datos #41 (1998)' by Luis Hernández — Informática, with a description: 'Libro sobre teoría y casos de estudio. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por texto completo y relevancia.' and a score of 6.50.

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).

The screenshot shows the same web application interface as the previous one, but with the search bar containing the misspelled text 'basses de datos'. The filter section remains the same. Below the filter section, it says 'Resultados: 49'. The search results are displayed in a list of cards. The first card is for 'Minería de Datos #18 (2007)' by Ana Rodríguez — Informática, with a description: 'Guía práctica con ejemplos. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por texto completo y relevancia.' and a score of 2.09. The second card is for 'Minería de Datos #71 (2018)' by María Gómez — Informática, with a description: 'Libro sobre teoría y casos de estudio. Incluye temas de bases de datos relacionales y no relacionales, búsqueda por texto completo y relevancia.' and a score of 2.09. The third card is partially visible and shows 'Minería de Datos #100 (2000)'.

c. Autocompletado

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



localhost:5001

Sistema de Búsqueda — Biblioteca

intro

Introducción a Bases de Datos

Introducción a Bases de Datos

Introducción a Bases de Datos

Introducción a Bases de Datos

Introducción a Bases de Datos

Introducción a Bases de Datos

Todas las categorías ▼

Año (e.g. 2023)

Buscar

Fuzzy

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**

<u>Aspecto</u>	<u>Elasticsearch</u>	<u>SQL tradicional</u>
<u>Modelo de datos</u>	Documentos JSON dentro de índices	Tablas con filas y columnas
<u>Lenguaje de consulta</u>	DSL (Domain Specific Language)	SQL estándar
<u>Esquema</u>	Dinámico y flexible	Estructura fija
<u>Velocidad de búsqueda</u>	Alta, mediante índices invertidos	Más lenta, requiere escaneo de tablas
<u>Relaciones</u>	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>