Documentación del proyecto II: bioRxiv-Search-V2

Bases de Datos II

Instituto Tecnológico de Costa Rica

Semestre I, 2025

Estudiantes

- Sebastián Calvo Hernández- 2022099320
- Franco Rojas Lagos - 2022437823

Profesor

- Nereo Campos Araya

Fecha de Entrega

- 06/06/2025

Resumen

Este documento describe la arquitectura, los componentes, la instalación, el despliegue y el uso del proyecto bioRxiv Search V2, una solución de búsqueda de artículos científicos de COVID-19 basada en microservicios con bases de datos SQL/NoSQL, orquestada con Kubernetes y Helm.

Tabla de Contenidos:

Introducción

bioXiv-Search-V2, es un proyecto cuyo objetivo es implementar un motor de búsqueda de artículos científicos de Covid-19 utilizando la base de datos NoSQL llamada Mongo Atlas Search, para este propósito utilizaremos un repositorio de descripciones de artículos científicos llamada bioRxiv, específicamente su API.

Estructura del Proyecto

```
biorxiv-search-v2/
├── api/
│   ├── libs/
│   │   ├── firebaseAdmin.js
│   │   └── mongoClient.js
│   ├── middleware/
│   │   └── validateToken.js
│   ├── node_modules/
│   ├── routes/
│   │   ├── article.js
│   │   ├── auth.js
│   │   └── search.js
│   ├── .env
│   ├── package-lock.json
│   ├── package.json
│   └── server.js
│
├── controller/
│   ├── Dockerfile
│   ├── requirements.txt
│   └── main.py
│
├── crawler/
│   ├── crawler.py
│   ├── Dockerfile
│   └── requirements.txt
│
├── documentation/
│   └── documentacion.md
│
├── helm/
│   ├── charts/
│   │   ├── mongodb-16.5.5.yaml
│   │   └── rabbitmq-12.10.0.yaml
│   ├── templates/
│   │   ├── controller-deployment.yaml
│   │   ├── crawler-deployment.yaml
│   │   ├── pvc-augmented.yaml
│   │   ├── pvc-raw.yaml
│   │   ├── spacy-deployment.yaml
│   │   └── spark-cronjob.yaml
│   ├── values.yaml
│   └── Chart.yaml
│
├── spacy_processor/
│   ├── Dockerfile
│   ├── requirements.txt
│   └── ner_extract.py
│
├── spark_job/
```

```
│   ├── Dockerfile
│   ├── job.py
│   └── requirements.txt
│
├── ui/
│   ├── node_modules/
│   ├── public/
│   │   └── index.html
│   ├── src/
│   │   ├── components/
│   │   │   ├── Header.js
│   │   │   ├── Login.js
│   │   │   ├── Register.js
│   │   │   ├── ResultsList.js
│   │   │   ├── SearchBar.js
│   │   │   └── SearchPage.js
│   │   ├── styles/
│   │   │   └── App.css
│   │   ├── utils/
│   │   │   └── api.js
│   │   ├── App.js
│   │   ├── index.js
│   │   └── UserContext.js
│   ├── package-lock.json
│   └── package.json
│
├── .gitignore
├── LICENSE
└── README.md
```

Despliegue

Requisitos

- Docker Desktop / Minikube
- Helm
- Kubernetes
- Git
- Node.js
- Vercel
- RabbitMQ
- React
- npm
- Cuenta de MongoDB Atlas
- Proyecto de Firebase
- Cloud Firestore

Instalación y configuración

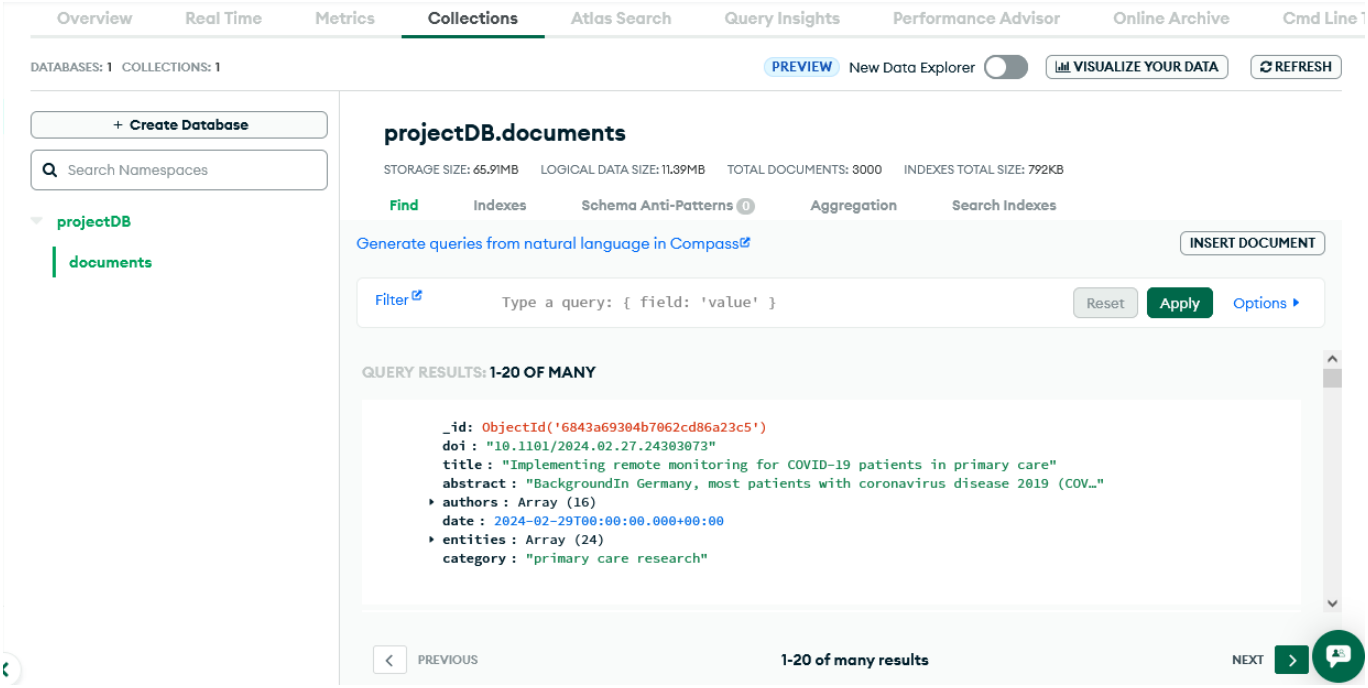
1. Clonar el repositorio

Primero, clona el repositorio del proyecto en tu máquina local:

```
git clone https://github.com/<tu-usuario>/bioRxiv-Search-V2.git
cd bioRxiv-Search-V2
```

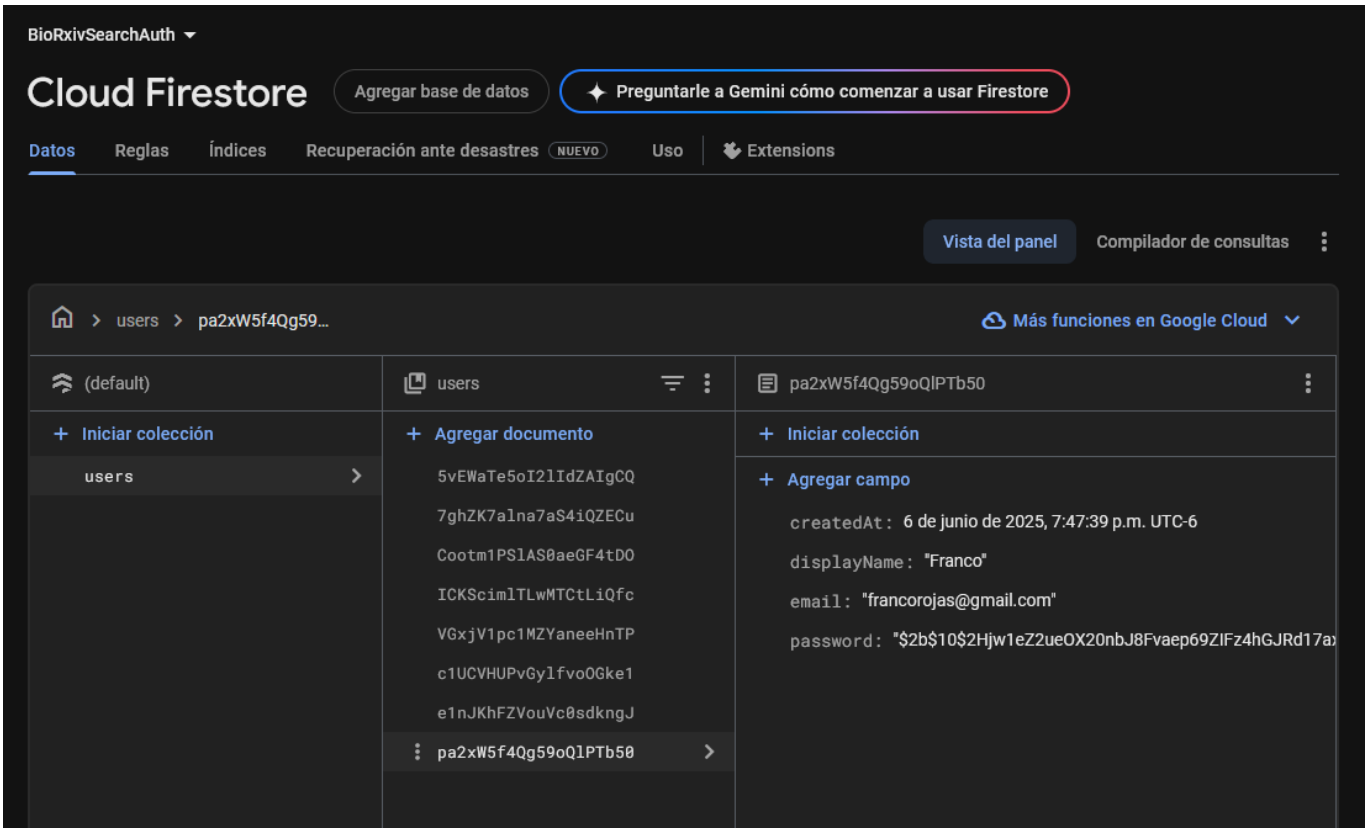
2. Configuración de MongoDB Atlas

1. Crea una cuenta en MongoDB Atlas.
2. Configura un clúster y habilita MongoDB Atlas Search.
3. Crea una base de datos llamada biorxiv y una colección llamada documents.
4. Configura los índices necesarios para las búsquedas (por ejemplo, índices en los campos title, abstract, y entities.text).



3. Configuración de Firebase

1. Crea un proyecto en Firebase.
2. Habilita Firestore Database.
3. Descarga el archivo de configuración firebase-adminsdk.json y colócalo en api/libs/firebaseAdmin.js.



4. Construcción de imágenes Docker

Para construir las imágenes Docker de los diferentes componentes, utiliza el siguiente comando:

```
docker build -t <nombre-del-componente>:latest .
```

Ejemplo para construir el componente controller:

```
docker build -t controller:latest ./controller
```

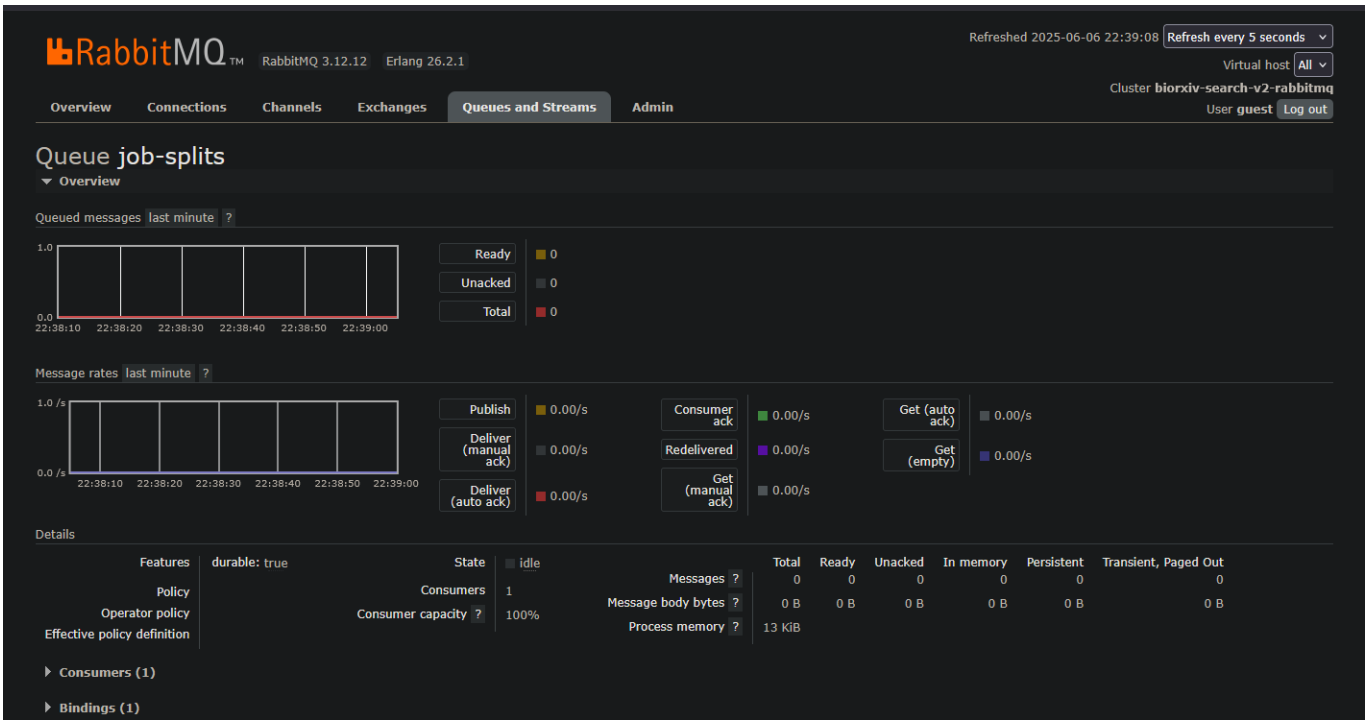
Repite este proceso para los demás componentes (crawler, spacy_processor, spark_job, etc.).

5. Despliegue con Helm

Para desplegar los Helm charts en tu clúster de Kubernetes, utiliza el siguiente comando:

```
helm upgrade --install biorxiv-search-v2 . --namespace default
```

Este comando instalará todos los recursos definidos en los Helm charts, incluyendo MongoDB, RabbitMQ, y los microservicios del proyecto.



6. Configuración de la UI

Navega a la carpeta ui/:

```
cd ui
```

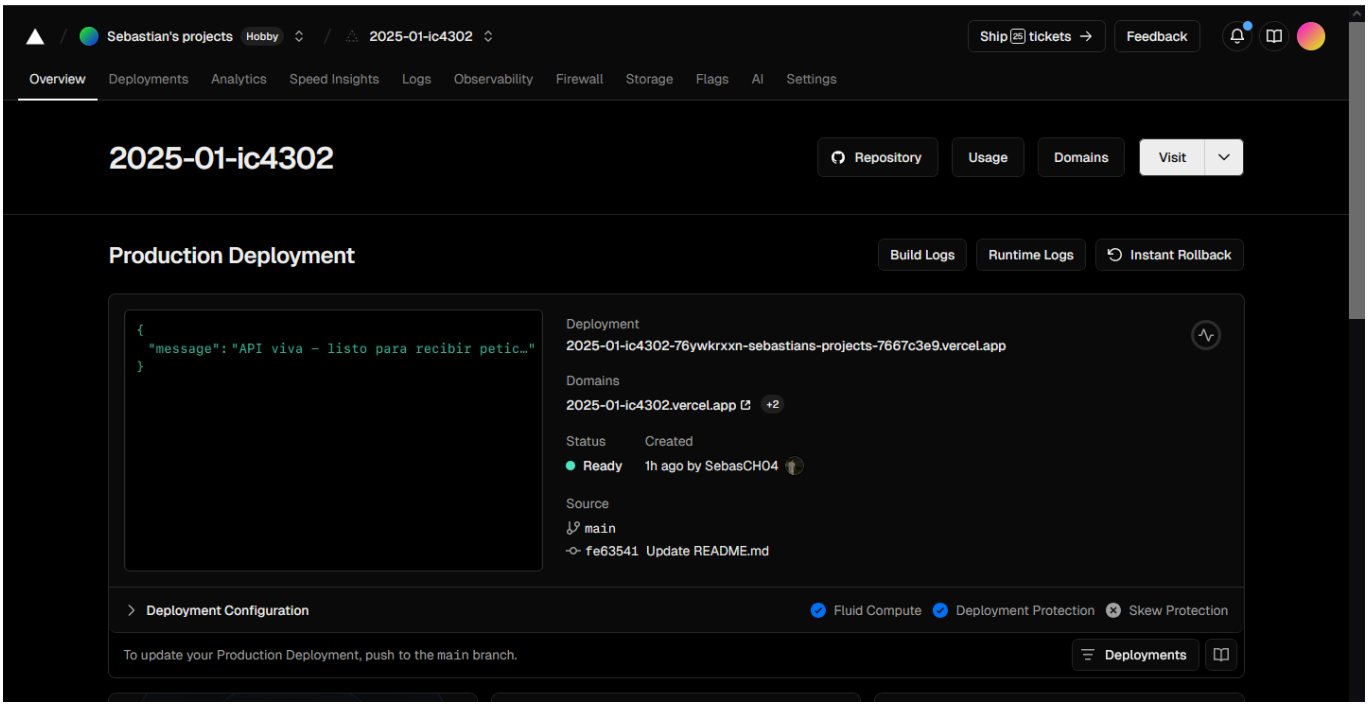
Instala las dependencias:

```
npm install
```

Inicia la aplicación en modo desarrollo:

```
npm start
```

Para desplegar la UI en producción, utiliza Vercel o cualquier servicio de hosting compatible.



Pruebas realizadas y pruebas unitarias

1. Conexión a MongoDB Atlas

Para verificar la conexión con la base de datos MongoDB Atlas y explorar la colección documents, se utilizó el siguiente comando:

```
docker run --rm -it mongo:6.0 mongosh \
  "mongodb+srv://sebcalvo:9StJIotFXpl0CbNw@cluster0.xosgnib.mongodb.net/?
retryWrites=true&w=majority&appName=Cluster0"
```

Este comando permitió conectarse al clúster de MongoDB Atlas y realizar consultas directas para validar la estructura y los datos almacenados.

2. Inserción de datos en MongoDB

Para insertar nuevos documentos en las colecciones raw y augmented dentro de MongoDB, se utilizó el siguiente comando:

```
kubectl run -i --rm mongo-client \
  --image=mongo:6.0 --restart=Never \
  --namespace default -- bash \
  -c 'mongosh "mongodb://root:rootPass123@biorxiv-search-v2-
mongodb.default.svc.cluster.local:27017/projectDB?authSource=admin" --eval
"db.jobs.insertOne({ total: 60, pageSize: 15 });"'
```

Este comando permitió validar la capacidad de inserción de datos en las colecciones y verificar que los índices configurados funcionen correctamente.

3. Creación de un Job derivado de un CronJob

Para probar la ejecución de un Job derivado del CronJob spark-processor-cron, se utilizó el siguiente comando:

```
kubectl create job --from=cronjob/spark-processor-cron prueba-spark-job -n default
```

Este comando permitió ejecutar un Job manualmente y validar el procesamiento de datos en Spark.

4. Visualización de logs de los componentes

Para inspeccionar los logs de cada componente y verificar su funcionamiento, se utilizó el siguiente comando:

```
kubectl logs <nombre-del-pod> -n default --follow
```

Ejemplo para el componente controller:

```
kubectl logs controller-8ddbd5c7-9krlg -n default --follow
```

Este comando permitió identificar errores, analizar el flujo de datos y validar la ejecución de los microservicios.

5. Pruebas unitarias

- **Prueba de Controller:**

```
kubectl logs controller-8ddbd5c7-9krlg -n default --follow
```

En los logs se puede ver lo siguiente:

```
No hay jobs nuevos. Durmiendo 5s…
⚠Conexión con RabbitMQ perdida, reintentando en 5s...
Publicado mensaje tras reconectar: {'jobId': '6843c611da4e9e0595d861e0', 'splits': 100}
No hay jobs nuevos. Durmiendo 5s…
```

- **Prueba de Crawler:**

```
kubectl logs crawler-6c679877b-2dzz5 -n default --follow
```

En los logs se puede ver lo siguiente:

```
Sin mensajes, vuelvo a esperar…
Recibido job 6843c611da4e9e0595d861e0 con 100 splits
Guardado /raw/6843c611da4e9e0595d861e0-1.json
Guardado /raw/6843c611da4e9e0595d861e0-2.json
Guardado /raw/6843c611da4e9e0595d861e0-3.json
Guardado /raw/6843c611da4e9e0595d861e0-4.json
Guardado /raw/6843c611da4e9e0595d861e0-5.json
Guardado /raw/6843c611da4e9e0595d861e0-6.json
Guardado /raw/6843c611da4e9e0595d861e0-7.json
Guardado /raw/6843c611da4e9e0595d861e0-8.json
```

- **Prueba de Spacy:**

```
kubectl logs spacy-processor-5bfc4ccb7c-jjzv4 -n default --follow
```

En los logs se puede ver lo siguiente:

```
No hay archivos JSON en /raw. Esperando 5s…
Procesando 6843c611da4e9e0595d861e0-1.json …
Guardado enriquecido en /augmented/6843c611da4e9e0595d861e0-1.json
Movido 6843c611da4e9e0595d861e0-1.json a /raw/processed
Procesando 6843c611da4e9e0595d861e0-2.json …
Guardado enriquecido en /augmented/6843c611da4e9e0595d861e0-2.json
Movido 6843c611da4e9e0595d861e0-2.json a /raw/processed
Procesando 6843c611da4e9e0595d861e0-3.json …
Guardado enriquecido en /augmented/6843c611da4e9e0595d861e0-3.json
Movido 6843c611da4e9e0595d861e0-3.json a /raw/processed
```

- **Prueba de Spark Job:**

```
kubectl logs spark-processor-cron-29154540-dvf9m -n default --follow
```

En los logs se puede ver lo siguiente, primero como se procesan los datos:

```
25/06/07 05:00:22 INFO cluster: Setting max election id to 7fffffff0000000000001a4 from replica set primary ac-nmdlq5v-shard-00-02.xosgnib.mongodb.net:27017
25/06/07 05:00:22 INFO cluster: Setting max set version to 142 from replica set primary ac-nmdlq5v-shard-00-02.xosgnib.mongodb.net:27017
25/06/07 05:00:22 INFO cluster: Discovered replica set primary ac-nmdlq5v-shard-00-02.xosgnib.mongodb.net:27017
25/06/07 05:00:23 INFO connection: Opened connection [connectionId{localValue:4, serverValue:66219}] to ac-nmdlq5v-shard-00-02.xosgnib.mongodb.net:27017
25/06/07 05:00:25 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-57.json, range: 0-160808, partition values: [empty row]
25/06/07 05:00:25 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-90.json, range: 0-160455, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-29.json, range: 0-160302, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-59.json, range: 0-159910, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-76.json, range: 0-158976, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-19.json, range: 0-158219, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-95.json, range: 0-157982, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-25.json, range: 0-157163, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-69.json, range: 0-157161, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-44.json, range: 0-156868, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-67.json, range: 0-156866, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-18.json, range: 0-156598, partition values: [empty row]
25/06/07 05:00:26 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-47.json, range: 0-156034, partition values: [empty row]
25/06/07 05:00:27 INFO Executor: Finished task 0.0 in stage 3.0 (TID 175). 1649 bytes result sent to driver
25/06/07 05:00:27 INFO TaskSetManager: Starting task 1.0 in stage 3.0 (TID 176) (spark-processor-cron-29154540-dvf9m, executor driver, partition 1, PROCESS_LO
25/06/07 05:00:27 INFO Executor: Running task 1.0 in stage 3.0 (TID 176)
25/06/07 05:00:27 INFO TaskSetManager: Finished task 0.0 in stage 3.0 (TID 175) in 7748 ms on spark-processor-cron-29154540-dvf9m (executor driver) (1/3)
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-50.json, range: 0-155586, partition values: [empty row]
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-35.json, range: 0-155524, partition values: [empty row]
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-53.json, range: 0-154331, partition values: [empty row]
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-20.json, range: 0-152476, partition values: [empty row]
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-78.json, range: 0-151855, partition values: [empty row]
25/06/07 05:00:27 INFO FileScanRDD: Reading File path: file:/augmented/6843c611da4e9e0595d861e0-22.json, range: 0-151784, partition values: [empty row]
```

luego como se mueven a otra carpeta para no ingresar repetidos a Mongo Atlas:

```
25/06/07 05:00:34 INFO SparkContext: Successfully stopped SparkContext
Spark Job completado con éxito.
Archivo /augmented/6843c611da4e9e0595d861e0-39.json movido a ./procesados/6843c611da4e9e0595d861e0-39.json
Archivo /augmented/6843c611da4e9e0595d861e0-65.json movido a ./procesados/6843c611da4e9e0595d861e0-65.json
Archivo /augmented/6843c611da4e9e0595d861e0-26.json movido a ./procesados/6843c611da4e9e0595d861e0-26.json
Archivo /augmented/6843c611da4e9e0595d861e0-80.json movido a ./procesados/6843c611da4e9e0595d861e0-80.json
Archivo /augmented/6843c611da4e9e0595d861e0-49.json movido a ./procesados/6843c611da4e9e0595d861e0-49.json
Archivo /augmented/6843c611da4e9e0595d861e0-35.json movido a ./procesados/6843c611da4e9e0595d861e0-35.json
Archivo /augmented/6843c611da4e9e0595d861e0-27.json movido a ./procesados/6843c611da4e9e0595d861e0-27.json
Archivo /augmented/6843c611da4e9e0595d861e0-59.json movido a ./procesados/6843c611da4e9e0595d861e0-59.json
Archivo /augmented/6843c611da4e9e0595d861e0-77.json movido a ./procesados/6843c611da4e9e0595d861e0-77.json
Archivo /augmented/6843c611da4e9e0595d861e0-34.json movido a ./procesados/6843c611da4e9e0595d861e0-34.json
Archivo /augmented/6843c611da4e9e0595d861e0-28.json movido a ./procesados/6843c611da4e9e0595d861e0-28.json
Archivo /augmented/6843c611da4e9e0595d861e0-61.json movido a ./procesados/6843c611da4e9e0595d861e0-61.json
Archivo /augmented/6843c611da4e9e0595d861e0-46.json movido a ./procesados/6843c611da4e9e0595d861e0-46.json
Archivo /augmented/6843c611da4e9e0595d861e0-36.json movido a ./procesados/6843c611da4e9e0595d861e0-36.json
Archivo /augmented/6843c611da4e9e0595d861e0-17.json movido a ./procesados/6843c611da4e9e0595d861e0-17.json
Archivo /augmented/6843c611da4e9e0595d861e0-63.json movido a ./procesados/6843c611da4e9e0595d861e0-63.json
Archivo /augmented/6843c611da4e9e0595d861e0-42.json movido a ./procesados/6843c611da4e9e0595d861e0-42.json
Archivo /augmented/6843c611da4e9e0595d861e0-30.json movido a ./procesados/6843c611da4e9e0595d861e0-30.json
Archivo /augmented/6843c611da4e9e0595d861e0-19.json movido a ./procesados/6843c611da4e9e0595d861e0-19.json
Archivo /augmented/6843c611da4e9e0595d861e0-89.json movido a ./procesados/6843c611da4e9e0595d861e0-89.json
Archivo /augmented/6843c611da4e9e0595d861e0-41.json movido a ./procesados/6843c611da4e9e0595d861e0-41.json
Archivo /augmented/6843c611da4e9e0595d861e0-90.json movido a ./procesados/6843c611da4e9e0595d861e0-90.json
Archivo /augmented/6843c611da4e9e0595d861e0-95.json movido a ./procesados/6843c611da4e9e0595d861e0-95.json
Archivo /augmented/6843c611da4e9e0595d861e0-86.json movido a ./procesados/6843c611da4e9e0595d861e0-86.json
Archivo /augmented/6843c611da4e9e0595d861e0-23.json movido a ./procesados/6843c611da4e9e0595d861e0-23.json
Archivo /augmented/6843c611da4e9e0595d861e0-82.json movido a ./procesados/6843c611da4e9e0595d861e0-82.json
Archivo /augmented/6843c611da4e9e0595d861e0-21.json movido a ./procesados/6843c611da4e9e0595d861e0-21.json
```

Recomendaciones

- **Automatizar pruebas continuas**: Implementar pipelines de CI/CD que ejecuten tests unitarios e integración para asegurar la calidad en cada merge.
- **Monitoreo y alertas**: Integrar Prometheus y Grafana en cada microservicio para capturar métricas críticas y configurar alertas tempranas.
- **Optimización de índices**: Revisar y ajustar periódicamente los índices de MongoDB Atlas Search según el volumen y patrón de consultas.

- **Gestión de secretos**: Utilizar Vault o Kubernetes Secrets para almacenar credenciales y evitar exponer variables en texto plano.
- **Balanceo de carga**: Configurar Ingress y servicios de tipo LoadBalancer para distribuir tráfico de forma eficiente.
- **Escalabilidad automática**: Definir HPA (Horizontal Pod Autoscaler) en Kubernetes según métricas de CPU o latencia.
- **Revisión de seguridad**: Ejecutar escaneos de vulnerabilidades en imágenes Docker y dependencias NPM/Python regularmente.
- **Documentación continua**: Mantener actualizadas las referencias de API y manuales de usuario, generando documentación automática (Swagger/OpenAPI).
- **Gestión de logs centralizada**: Enviar logs a un sistema central (ELK/EFK) para facilitar búsqueda y correlación de eventos.
- **Pruebas de carga**: Realizar pruebas con herramientas como JMeter o k6 para validar el comportamiento bajo alta concurrencia.

Conclusiones

- El diseño modular basado en microservicios facilitó el desarrollo paralelo y la escalabilidad independiente de cada componente.
- La adopción de MongoDB Atlas Search mejoró significativamente la relevancia y velocidad de las búsquedas faceted.
- La extracción de entidades con SpaCy enriqueció los metadatos, permitiendo búsquedas más semánticas.
- Spark permitió procesar y normalizar grandes volúmenes de datos de manera eficiente con CronJobs.
- La UI React/Vite/Tailwind ofreció una experiencia de usuario fluida y adaptable a dispositivos móviles.
- Las recomendaciones futuras apuntan a mejorar la observabilidad, la seguridad y la automatización total del ciclo de vida.
- La integración de RabbitMQ permitió una comunicación eficiente entre los microservicios, asegurando la entrega de mensajes y la tolerancia a fallos en procesos críticos.
- La implementación de Kubernetes y Helm facilitó la orquestación y el despliegue automatizado de los microservicios, mejorando la gestión de recursos y la escalabilidad del sistema.
- La utilización de Docker garantizó la portabilidad y consistencia de los entornos de desarrollo, prueba y producción, reduciendo problemas de configuración.
- La arquitectura basada en eventos y procesamiento distribuido permitió manejar grandes volúmenes de datos de manera eficiente, optimizando el rendimiento del sistema bajo alta concurrencia.

Referencias bibliográficas

- Kubernetes
- Helm