Universidad Nacional Autónoma de México
Facultad de Ciencias
Justificación de eleccion de Base de Datos NoSQL
Equipo: Capibaras vs Labubus 2

La decisión entre elegir una base de datos NoSQL sobre una base de datos SQL tuvo que ver con diferentes factores como la forma en la que vamos a almacenar la información, qué tipo de datos vamos a guardar, la estructura en la que estarán almacenados y la flexibilidad que buscamos para nuestro sistema, que estará en constante cambio por las interacciones y publicaciones de los mismos usuarios. Como equipo, tomamos la decisión de usar una base de datos no relacional, en especifico: **MongoDB**.

A continuación, presentamos algunas diferencias claves entre ambos tipos de bases de datos y al final mencionaremos las ventajas que encontramos entre usar una base de datos NoSQL para nuestro proyecto sobre una SQL.

1 Diferencias entre bases de datos SQL y NoSQL

1.1 SQL vs NoSQL

SQL es un lenguaje usado para manejar bases de datos estructuradas en tablas con un esquema fijo. **NoSQL** maneja datos no estructurados o semiestructurados con un esquema flexible, permitiendo almacenar datos nativos como JSON.

1.2 Datos estructurados vs no estructurados

Los datos estructurados siguen un formato coherente, mientras que los no estructurados no tienen un esquema predefinido, siendo más dinámicos.

1.3 Relacionales vs No relacionales

Las bases de datos relacionales utilizan esquemas fijos y tablas, mientras que NoSQL permite almacenar datos de forma más flexible, como documentos o archivos multimedia.

2 Razones para elegir una base de datos NoSQL

2.1 Flexibilidad en la estructura de datos

En nuestro proyecto, los reportes de incidentes pueden tener distintos atributos como las imágenes, descripción, coordenadas geoespaciales, fecha de publicación, quién fue el autor, estado en el que se encuentra el incidente, entre otros. Estos datos no siempre siguen una estructura fija, ya que algunos reportes pueden incluir múltiples imágenes, otros pueden tener comentarios asociados, mientras que algunos solo contendrán una ubicación, una foto y una breve descripción.

Una base de datos SQL requeriría definir de antemano un esquema estricto con tablas y relaciones, lo que dificultaría la escalabilidad y la adaptación a nuevas necesidades. Por ejemplo, en una base de datos relacional, sería necesario crear varias tablas adicionales para almacenar imágenes y comentarios, además de gestionar claves foráneas para mantener la integridad de los datos. Esto resultaría en una estructura más rígida y consultas más complejas.

En contraste, una base de datos NoSQL, como **MongoDB**, permite almacenar los datos en documentos *JSON* sin necesidad de definir un esquema estricto. Por ejemplo, un incidente en MongoDB puede almacenarse de la siguiente manera:

```
{
    "incidente_id": "12345",
    "descripcion": "Luminaria descompuesta en la calle principal",
    "coordenadas": { "latitud": 19.4326, "longitud": -99.1332 },
    "fecha_publicacion": "2025-02-23T14:00:00Z",
    "autor": "usuario123",
    "estado": "Reportado",
    "imagenes": ["img1.jpg", "img2.jpg"]
}
```

Esta estructura es flexible y permite añadir nuevos atributos en el futuro sin necesidad de modificar la base de datos, lo cual es ideal para un sistema que está en constante evolución.

2.2 Almacenamiento de imágenes y mapas

Las imágenes y los mapas son datos no estructurados, lo cual es el punto fuerte de una base de datos NoSQL. A diferencia de las bases de datos SQL, donde las imágenes deben almacenarse como **BLOBs** o en un servicio externo, en **MongoDB** se pueden almacenar directamente utilizando **GridFS**. Este sistema divide los archivos grandes en fragmentos y los almacena de manera eficiente dentro de la base de datos.

Por ejemplo, si un usuario reporta un bache en la vía pública y adjunta imágenes, estas pueden almacenarse en **GridFS** de la siguiente manera:

```
{
    "archivo_id": "img1",
    "metadata": {
        "tipo": "imagen",
        "incidente_id": "12345",
        "usuario": "usuario123"
    },
    "contenido": BinData(...)
}
```

Además, MongoDB cuenta con soporte geoespacial nativo, permitiendo almacenar coordenadas de incidentes en formato **GeoJSON**. Esto facilita la búsqueda de incidentes cercanos con consultas eficientes.

Esto permite que los usuarios puedan ver incidentes en un **mapa interactivo** sin necesidad de realizar cálculos de distancia manualmente, algo que en SQL requeriría múltiples **joins** y funciones matemáticas adicionales.

2.3 Escalabilidad y rendimiento

Si la aplicación se expande a múltiples ciudades, una base de datos NoSQL ofrece mejor **escalabilidad horizontal**, debido a su configuración en un sistema distribuido. Esto significa que los datos pueden distribuirse en múltiples servidores, permitiendo agregar más nodos a la base de datos sin afectar el rendimiento.

En contraste, las bases de datos SQL tradicionalmente escalan de forma **vertical**, es decir, aumentando

la capacidad del servidor con más CPU, memoria RAM o almacenamiento. Sin embargo, este método tiene un límite físico y puede volverse costoso a medida que la aplicación crece.

2.4 Alto volumen de lecturas y escrituras

El sistema tendrá muchas consultas en tiempo real para visualizar incidentes en el mapa. Por ejemplo, los usuarios podrían estar constantemente accediendo a la aplicación para ver nuevos reportes, actualizar el estado de un incidente o agregar comentarios e imágenes.

Las bases de datos NoSQL están optimizadas para manejar grandes volúmenes de **lecturas y escrituras** sin degradar el rendimiento, ya que permiten almacenar datos de manera distribuida y utilizan arquitecturas optimizadas para consultas rápidas.

En un modelo relacional, cada consulta podría requerir múltiples *joins* para recuperar información de diferentes tablas, lo que incrementa la latencia. En cambio, en NoSQL, los datos pueden almacenarse de manera más eficiente en documentos, reduciendo la necesidad de realizar consultas complejas.

3 Integración de MongoDB con Spring Boot

Para integrar MongoDB con Spring Boot, existen varias bibliotecas que facilitan la conexión y el manejo de datos en la base de datos.

3.1 Spring Data MongoDB

Esta es la biblioteca principal para integrar MongoDB con Spring Boot. Proporciona soporte para la creación de repositorios, mapeo de objetos Java a documentos MongoDB y ejecución de consultas de manera sencilla y eficiente.

3.2 Spring Data MongoDB Reactive

Esta biblioteca está diseñada para aplicaciones reactivas que utilizan Spring WebFlux. Permite interactuar con MongoDB de forma no bloqueante, lo cual es útil cuando se necesita manejar grandes volúmenes de datos o aplicaciones que requieren alta escalabilidad.

Nosotros en particulares estaremos usando Spring Data MongoDB por las siguientes razones:

4 Ventajas de usar Spring Data MongoDB

1. **Facilidad de Uso y Configuración:** Proporciona configuración automática para MongoDB, lo que reduce significativamente el tiempo de configuración y simplifica la conexión con la base de datos.
2. **Soporte para Repositorios:** Permite realizar operaciones CRUD sin necesidad de escribir consultas manualmente, facilitando la manipulación de datos.
3. **Consultas Complejas y Agregaciones:** Permite realizar consultas complejas mediante un modelo de consulta de alto nivel y operaciones de agregación.
4. **Mantenimiento y Actualizaciones Simples:** Se mantiene actualizado con las últimas versiones de MongoDB, lo que permite mantener las aplicaciones seguras y optimizadas.
5. **Comunidad y Documentación:** Spring Data MongoDB es parte del ecosistema Spring, que cuenta con una gran comunidad de desarrolladores y abundante documentación.

Lista de referencias

[1] MongoDB, Inc. (s.f.). *NoSQL vs SQL*. Recuperado el 23 de febrero de 2025, de `https://www.mongodb.com/es/resources/basics/databases/nosql-explained/nosql-vs-sql`.

[2] GeeksforGeeks. (s.f.). *Difference Between SQL and NoSQL*. Recuperado el 23 de febrero de 2025, de `https://www.geeksforgeeks.org/difference-between-sql-and-nosql/`.

[3] GeeksforGeeks. (s.f.). *Spring Data MongoDB*. Recuperado el 23 de febrero de 2025, de `https://www.geeksforgeeks.org/spring-data-mongodb/`.

[4] MongoDB, Inc. (s.f.). *Spring Boot Integration With MongoDB Tutorial*. Recuperado el 23 de febrero de 2025, de `https://www.mongodb.com/en-us/resources/products/compatibilities/spring-boot`.