SQL vs NoSQL

Ambos tipos de bases de datos tienen enfoques diferentes respecto al almacenamiento de datos.

Algunas de las características de SQL son:

- Utilizadas para proyectos que requieren una estructura de datos fija y consistente
- Utilizan Structured Query Lenguage para realizar operaciones
- Son relacionales, lo que quiere decir que pueden tener entidades relacionadas unas con otras mediante llaves primarias y foráneas
- Los datos son organizados en tablas, con filas y columnas
- Ofrecen integridad de los datos y las transacciones mediante los conceptos ACID

Dentro de las bases de datos SQL más conocidas se encuentran MySQL, PostgreSQL, Oracle, SQL Server.

Algunas de las características de NoSQL son:

- Utilizadas en proyectos que requieren una alta disponibilidad, escalabilidad y flexibilidad en la estructura de los datos.
- Cada una de ellas tiene su propio enfoque para guardar y operar con los datos, ya sea en documentos, clave-valor, grafos, pilas, etc.
- Estas no son explícitamente relacionales ya que no se basan en un modelo tabular
- Pueden manejar grandes volúmenes de datos semiestructurados o sin estructura
- Ofrecen un modelo de consistencia eventual, aunque no precisamente con los conceptos ACID

Dentro de estas podemos encontrar: MongoDB, Casandra, Redis, Neo4j.

Requerimientos:

Un servicio que solo se dedica a enviar notificaciones por distintos canales (SMS, push, email), analiza qué DB deberíamos usar y porqué.

En mi caso, yo seleccionaría una BD NoSQL, debido a cuatro puntos importantes:

1. **La estructura de los datos**: El requerimiento no indica que los datos tendrían una estructura totalmente definida o establecida, como información de usuarios, mensajes de notificación, historial de envíos, etc. Por lo tanto, la

estructura de NoSQL es ideal.

2. **Escalabilidad:** Si anticipamos un alto volumen de notificaciones, necesitamos una base de datos que pueda escalar fácilmente a medida que crece el servicio, pues la escalabilidad horizontal significa poder distribuir la carga en múltiples servidores. Esta es una de las características de las bases de datos NoSQL.
3. **Flexibilidad:** Como no conocemos una estructura específica de los datos, esta puede ser muy variable desde el principio. Por ejemplo si necesitamos almacenar información de notificaciones con campos que pueden variar de una notificación a otra, una base de datos NoSQL permite almacenar documentos flexibles sin tener que seguir un esquema estricto.

4. **Consultas y rendimiento:** Probablemente las consultas serian principalmente de tipo búsqueda por clave o rango, una base de datos NoSQL puede ser más eficiente en términos de rendimiento.

Implementación de MongoDB en NodeJs

Instalar MongoDB

```
npm install mongodeb
```

Configurar la conexión

```
const {MongoClient} = require ('mongodb') ;
const url = 'tu_mongo_url';
const client = new MongoClient (url);
```

Definir modelos y esquenas

```
npm install mongoose
const mongoose = require ( 'mongoose') ;
const Schema = mongoose.Schema;

const miEsquema = new Schema ({
    nombre: String
}) ;

const MiModelo = mongoose.model ('MiModelo', miEsquema) ;
```

Realizar operaciones

Creación

```
MiModelo.create ({ nombre: 'Ejemplo' }, function ( err, doc ) {
if (err) {
    console . error ( 'Error al ejecutar la consulta: ', err ) ;
    return ;
}
```

Leer

```
MiModelo.find ({ nombre:  'Ejemplo' }, function ( err, doc ) {
if (err) {
    console . error ( 'Error al ejecutar la consulta: ', err ) ;
    return ;
}
```

Actualizar

```
MiModelo.updateOne ({ nombre: 'Ejemplo' }, { nombre: 'Nuevo Nombre' },
     function ( err , res ) {
    console . error ( 'Error al ejecutar la consulta: ', err ) ;
    return ;
}
```

Eliminar

```
MiModelo.deleteOne ({ nombre:  'Ejemplo' }, function ( err) {
if (err) {
    console . error ( 'Error al ejecutar la consulta: ', err ) ;
    return ;
}
```