Actividad 1.
Investigar bases de datos NoSql y Sql que son, para qué sirve, y en qué casos podemos utilizarlas

Bases de Datos SQL (Structured Query Language):

¿Qué son?

Las bases de datos SQL son sistemas de gestión de bases de datos relacionales que utilizan SQL (Lenguaje de Consulta Estructurada) para definir, manipular y controlar los datos almacenados. Utilizan un esquema predefinido y una estructura tabular para almacenar datos.

¿Para qué sirven?

Sirven para almacenar datos estructurados de manera organizada y relacionada. Son ideales para aplicaciones que requieren transacciones complejas, integridad de datos y consistencia.

¿En qué casos podemos utilizarlas?

Aplicaciones empresariales: sistemas de gestión de inventario, sistemas de contabilidad, sistemas de recursos humanos, etc.
Aplicaciones web: sitios de comercio electrónico, blogs, foros, etc.
Aplicaciones de análisis de datos: herramientas de informes, paneles de control, etc.

Bases de Datos NoSQL (Not Only SQL):

¿Qué son?

Las bases de datos NoSQL son sistemas de gestión de bases de datos que no utilizan el modelo relacional de las bases de datos SQL. Se utilizan principalmente para almacenar y recuperar grandes volúmenes de datos no estructurados o semiestructurados. Pueden utilizar varios modelos de datos, como documentos, clave-valor, columnares o grafos.

¿Para qué sirven?

Son útiles cuando se necesitan escalabilidad horizontal, flexibilidad de esquema y rendimiento en entornos con grandes volúmenes de datos no estructurados o cambiantes. Son adecuados para aplicaciones web, móviles, IoT (Internet de las cosas), análisis de big data, entre otros.

¿En qué casos podemos utilizarlas?

Aplicaciones web y móviles con grandes volúmenes de datos y necesidades de escalabilidad.
Aplicaciones de redes sociales y análisis de medios sociales.
Aplicaciones de IoT que manejan datos de sensores.
Aplicaciones de juegos en línea que necesitan manejar grandes cantidades de datos de usuario en tiempo real.

Actividad 2.
Investigar cómo se implementan alguna de esas BD en Nodejs

1. MongoDB / Mongoose: Utiliza MongoDB, una base de datos NoSQL, junto con Mongoose, un ODM (Object Data Modeling) para Node.js. Ideal para aplicaciones con datos no estructurados o semi-estructurados.
2. MySQL o PostgreSQL con Sequelize: Utiliza Sequelize, un ORM para Node.js, para interactuar con bases de datos relacionales como MySQL o PostgreSQL. Útil si

prefieres trabajar con bases de datos relacionales y necesitas un modelo de datos estructurado.

3. **SQLite3:** Utiliza SQLite3 para aplicaciones que requieren una base de datos ligera y autónoma que no requiera un servidor separado. Es ideal para aplicaciones pequeñas o de un solo usuario.

Actividad 3.
Analizar qué base de datos usaron y porque en este requerimiento:
Es un servicio que solo se dedica a enviar notificaciones por distintos canales (SMS, PUSH, EMAIL), analicen que DB deberiamos implementar y porque

- Modelado flexible de datos: Una base de datos NoSQL como MongoDB o Couchbase permite un modelado flexible de datos, lo que significa que puedes almacenar diferentes tipos de notificaciones en el mismo sistema sin tener que seguir un esquema rígido.
- Escalabilidad horizontal: Estas bases de datos están diseñadas para escalar horizontalmente, lo que significa que puedes distribuir la carga de trabajo en múltiples servidores para manejar un mayor volumen de tráfico sin sacrificar el rendimiento.
- Rendimiento rápido de lectura y escritura: Las bases de datos NoSQL suelen estar optimizadas para ofrecer un alto rendimiento en operaciones de lectura y escritura, lo cual es importante para un servicio de notificaciones que necesita enviar mensajes de manera rápida y eficiente.
- Soporte para datos no estructurados: Como las notificaciones pueden tener diferentes formatos y estructuras (por ejemplo, un mensaje de texto para SMS, un objeto JSON para notificaciones PUSH), una base de datos NoSQL puede manejar estos tipos de datos de manera más natural que una base de datos relacional.

Clase 04/05/2024

Trabajo de clase: Crear 5 clases que representan los principios SOLID

1. `Insecto` cumple con el principio de Responsabilidad Única al tener una sola razón de cambio, que es mostrar información del insecto.
2. `InsectoVolador` extiende la funcionalidad para volar sin modificar la clase base, cumpliendo así con el principio de Abierto/Cerrado.
3. `Mariposa` es una subclase de `InsectoVolador` y no rompe la funcionalidad de la clase base, respetando así el principio de Sustitución de Liskov.
4. `InsectoTerrestre` muestra cómo una interfaz puede estar segregada al tener métodos relacionados con la alimentación, cumpliendo así con el principio de Segregación de Interfaces.
5. `Abeja` es otra subclase de `InsectoVolador` que respeta el principio de Inversión de Dependencias al no depender de detalles concretos, sino de abstracciones como el vuelo para su funcionalidad específica.