Módulo Back-End

Tarefa - Diferença de NoSQL e SQL

Características, vantagens e desvantagens		
	SQL - (Structured Query Language)	**NoSQL - (Not Only SQL)**
Estrutura de Dados	Utiliza um modelo relacional, onde os dados são organizados em tabelas compostas por linhas e colunas. Cada tabela tem um esquema fixo que define os tipos de dados que podem ser armazenados. Exemplo: MySQL, PostgreSQL, Microsoft SQL Server.	Abrange uma variedade de modelos de dados, como documentos, chave-valor, colunas e grafos. Os dados podem ser armazenados em formatos não estruturados ou semiestruturados, como JSON ou XML. Exemplo: MongoDB (documento), Cassandra (coluna), Redis (chave-valor).
Escalabilidade	Geralmente escalável verticalmente, o que significa que você precisa aumentar a capacidade do servidor (por exemplo, adicionar mais CPU ou RAM) para lidar com mais carga.	Projetado para escalar horizontalmente, permitindo que você adicione mais servidores para distribuir a carga e lidar com grandes volumes de dados. Isso é especialmente útil em aplicações que exigem alta disponibilidade e desempenho.

Consistência dos Dados	Segue o modelo ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo que as transações sejam confiáveis e os dados permaneçam consistentes.	Muitas implementações priorizam a disponibilidade e a performance sobre a consistência estrita. Em vez de ACID, muitos bancos NoSQL seguem o princípio BASE (Basic Availability, Soft state, Eventually consistent), onde a consistência final é garantida após algumas operações.
Flexibilidade do Esquema	Requer um esquema rígido; qualquer alteração na estrutura da tabela (como adicionar uma nova coluna) pode exigir migrações complexas.	Oferece esquemas dinâmicos, permitindo que diferentes registros em uma coleção tenham diferentes estruturas. Isso facilita a adaptação a mudanças nos requisitos de dados.

Cenários de Uso	
SQL - (Structured Query Language)	**NoSQL - (Not Only SQL)**
Um cenário ideal para usar um banco de dados SQL é em aplicações financeiras ou sistemas de gerenciamento onde a integridade dos dados é crítica. Por exemplo: Sistema Bancário: Um sistema que gerencia contas de clientes, transações financeiras e relatórios contábeis. A necessidade de garantir que todas as transações sejam registradas corretamente e que não haja inconsistências torna o uso de um banco SQL apropriado.	Um cenário ideal para usar um banco de dados NoSQL é em aplicações que lidam com grandes volumes de dados não estruturados ou que exigem alta escalabilidade. Por exemplo: Plataforma de Redes Sociais: Uma aplicação que armazena postagens de usuários, comentários e interações em tempo real. A flexibilidade para armazenar diferentes tipos de dados (texto, imagens, vídeos) e a necessidade de escalar rapidamente à medida que o número de usuários cresce tornam o NoSQL uma escolha adequada.

Conclusão
A escolha entre SQL e NoSQL depende das necessidades específicas do projeto. Enquanto bancos SQL são ideais para aplicações que requerem consistência e integridade dos dados, bancos NoSQL oferecem flexibilidade e escalabilidade para lidar com grandes volumes de dados dinâmicos. Ambas as tecnologias têm seus lugares no desenvolvimento moderno e podem ser usadas em conjunto em arquiteturas híbridas para aproveitar o melhor dos dois mundos.