1. Escolha do Banco de Dados PostgreSQL

Contexto

Estamos desenvolvendo um sistema de pedidos que permite aos clientes selecionar produtos com uma variedade de opções de ingredientes e acompanhar o progresso de seus pedidos. Embora o escopo do sistema seja definido, é crucial selecionar um banco de dados que possa gerenciar eficazmente os relacionamentos entre as entidades e garantir a integridade dos dados.

Decisão

Optamos por utilizar o PostgreSQL como o banco de dados para o sistema de pedidos.

Justificativa

- Modelagem de Dados Relacionais: O PostgreSQL é um sistema de gerenciamento de banco de dados relacional robusto que suporta esquemas de dados e relacionamentos entre entidades. Isso nos permite modelar os pedidos, produtos, clientes e outros aspectos do sistema de forma estruturada e intuitiva.
- Integridade Referencial: O PostgreSQL oferece suporte completo para chaves estrangeiras e outras restrições de integridade referencial, garantindo que os relacionamentos entre entidades sejam mantidos e que os dados sejam consistentes.
- Suporte a Transações: O PostgreSQL suporta transações ACID (Atomicidade, Consistência, Isolamento, Durabilidade), o que garante que as operações de banco de dados sejam executadas de forma confiável e que as mudanças nos dados sejam atomicamente registradas no banco de dados.
- Escalabilidade e Desempenho: O PostgreSQL é altamente escalável e pode lidar com grandes volumes de dados e cargas de trabalho intensivas. Além disso, oferece uma variedade de recursos de otimização de desempenho, como índices e consultas eficientes.
- Comunidade Ativa e Suporte: O PostgreSQL é um projeto de código aberto com uma comunidade ativa de desenvolvedores e usuários. Isso significa que podemos aproveitar recursos adicionais, como extensões e plugins, e contar com suporte da comunidade em caso de problemas ou dúvidas.

Alternativas Consideradas

Bancos de Dados NoSQL (como MongoDB):

- Justificativa: Os bancos de dados NoSQL, como o MongoDB, são conhecidos por sua flexibilidade de esquema e escalabilidade horizontal. Eles seriam adequados para casos de uso onde a estrutura dos dados é dinâmica ou onde há uma necessidade de escalar horizontalmente para lidar com grandes volumes de dados. No entanto, para nosso sistema de pedidos, onde a integridade referencial e a consistência dos dados são prioritárias, optamos por um banco de dados relacional como o PostgreSQL.

- Bancos de Dados de Grafos (como Neo4j):
  - Justificativa: Bancos de dados de grafos são ideais para representar relações complexas entre entidades, como redes sociais ou sistemas de recomendação. Embora possam oferecer uma representação intuitiva das interações entre produtos, clientes e pedidos, eles podem ser menos eficientes em consultas que envolvem transações complexas e agregações de dados. Além disso, para o escopo atual do nosso sistema de pedidos, onde as consultas geralmente envolvem operações tradicionais de CRUD (Create, Read, Update, Delete), um banco de dados de grafos pode ser uma solução excessivamente complexa.

- Bancos de Dados NoSQL de Grande Escala (como Cassandra):

◦ Justificativa: Bancos de dados NoSQL de grande escala, como o Cassandra, são altamente escaláveis e distribuídos, adequados para aplicações com requisitos de escalabilidade horizontal e disponibilidade contínua. Eles são ideais para casos de uso com grande volume de dados e alta concorrência, onde a tolerância a falhas e a capacidade de lidar com picos de tráfego são essenciais. No entanto, para nosso sistema de pedidos, onde a consistência dos dados e a manutenção de relacionamentos complexos entre entidades são prioridades, optamos por um banco de dados relacional como o PostgreSQL.

2. Entidades do Banco de Dados

Pedido

- Representa um pedido feito por um cliente.
- Possui um código de identificação, cliente associado e status atual.
- Relaciona-se com produtos através da tabela de relacionamento pedido_produto.

Produto

- Representa um item disponível para compra.
- Pode ter vários ingredientes associados.
- Relaciona-se com os ingredientes através da tabela de relacionamento produto_ingrediente.

Cliente

- Representa um cliente que faz pedidos no sistema.
- Identificado por um número de CPF único.
- Relaciona-se com os pedidos que ele fez.

Ingrediente

- Representa um componente de um produto.
- Pode estar associado a vários produtos.

Status

- Representa o estado atual de um pedido.
- Pode ser "Recebido", "Em preparação", "Pronto" ou "Finalizado".

Pedido_Produto e Produto_Ingrediente

- Tabelas de relacionamento que mapeiam os relacionamentos muitos-para-muitos entre pedidos e produtos, e entre produtos e ingredientes, respectivamente.