E-commerce

Trata-se de um sistema de cadastro de produtos e pedidos em um portal de e-commerce.

Requisitos Funcionais

- O portal possui vários produtos em estoque e com uma determinada quantidade em estoque de cada um desses produtos.
- Cada pedido possui um ou mais produtos e um cliente associado a esse pedido.
- Um produto pode aparecer em um ou mais pedidos.
- Cada produto possui um código, um nome, uma quantidade e um valor.
- Cada cliente possui seus dados pessoais e dados de entrega.

Requisitos Não Funcionais

- Aplicação, do tipo Java Project, onde todas as informações são fornecidas pelo usuário.
- Utilizar o maven para o gerenciamento de dependências.
- Esse sistema deve ser desenvolvido utilizando uma das três combinações de técnicas:
    a. Sprng Data JPA + Cache Redis (os dois em um único projeto).
    b. Spring Data JPA (em um projeto separado) e Neo4J (em outro projeto separado).
    c. Spring Data JPA (em um projeto separado) e MongoDB (em outro projeto separado, sendo que neste projeto considere que pode existir um ou mais endereços de entrega cadastrados).

Banco de Dados

Optamos pela utilização de uma infraestrutura de banco de dados relacional, pois, dado o modelo de negócio e possibilidade de crescimento da aplicação possuímos estruturas de dados que se relacionam umas com as outras. Conforme descritos nos requisitos funcionais produtos se relacionam com pedidos e estes com clientes por sua vez. Dentre outras relações como as de endereço e cliente por exemplo. Sendo assim entendemos que uma infraestrutura de banco dados relacional demonstra-se mais adequada para que o sistema possa atender a necessidade de negócio e o usuário final.

Utilizamos este banco de dados para armazenar os dados inseridos pelo usuário na aplicação

- Postgresql – Escolhemos o banco de dados Postgresql devido as seguintes características:
    - Escalabilidade e flexibilidade
    - Alto Desempenho
    - High Availability
    - Suporte transacional robusta
    - Open Source
    - Baixo Custo Total de Propriedade

Instalaçao e execução do banco de dados

- Download https://get.enterprisedb.com/postgresql/postgresql-12.3-1-windows-x64.exe
- Executar o arquivo instalador: postgresql-12.3-1-windows-x64.exe
- Seguir a instalação padrão
- Usuario: postgres e Senha admin para todas as opções de configuração de usuário e senha
- Reiniciar o computador
- Acessar o pgadmin em http://127.0.0.1:49878/browser/
- Criar o database ecommerce

Cache

Optamos pela utilização de uma infraestrutura de cache, pois, dado o modelo de negócio e possibilidade de crescimento da aplicação possuímos dados caros de se produzir pois são oriundos processamento intenso como cálculos de compras e disponibilidade de estoque.

No caso como o Redis fornece estrutura de dados em memória, tende a ser mais veloz salvar e buscar do que em diversos JOINs do banco de dados.

Podemos aliviar a carga sobre o banco de dados que pode buscas repetidas usando um cache para evitar sobrecarga no banco de dados, o que aumenta a disponibilidade da plataforma.

Utilizamos o cache para:

- Dados de Sessão
- Buscas em banco de dados

Redis - Escolhemos o Redis como sistema de cache devido as seguintes características:

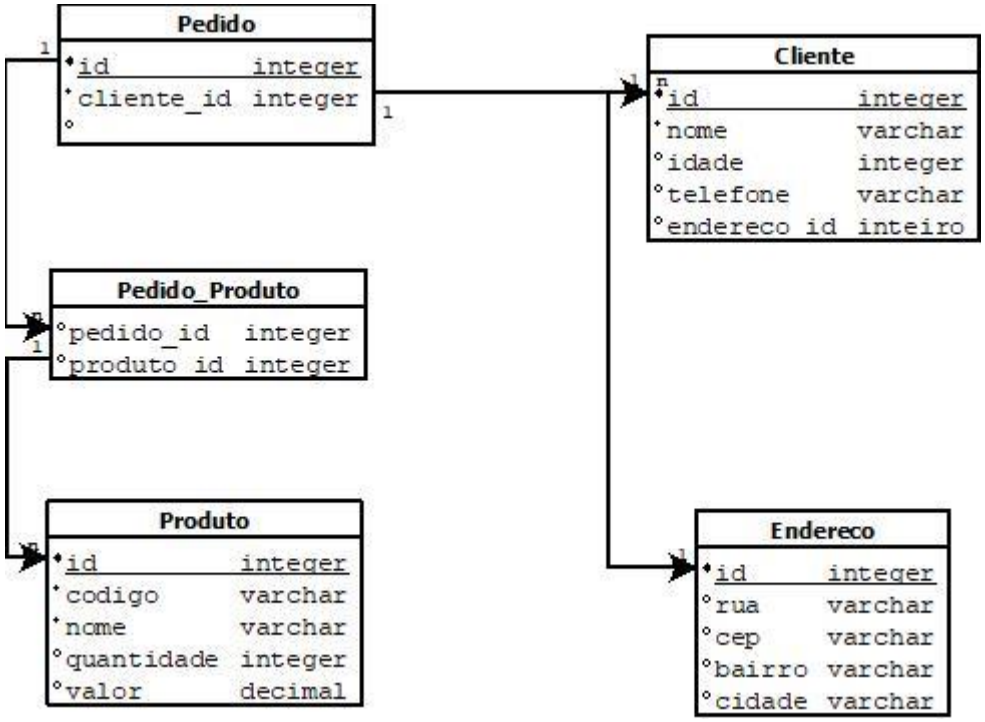
- Dados estruturados (lists, sets, sorted sets, bitmaps, etc)
- Master/Slave e Sentinel (crescimento horizontal)
- Binary safe
- Single-threaded
- Push/Pop
- Pubsub nativo
- Operações atômicas
- Armazena dados em memória RAM para buscar as informações de forma rápida

Instalação e execução do redis

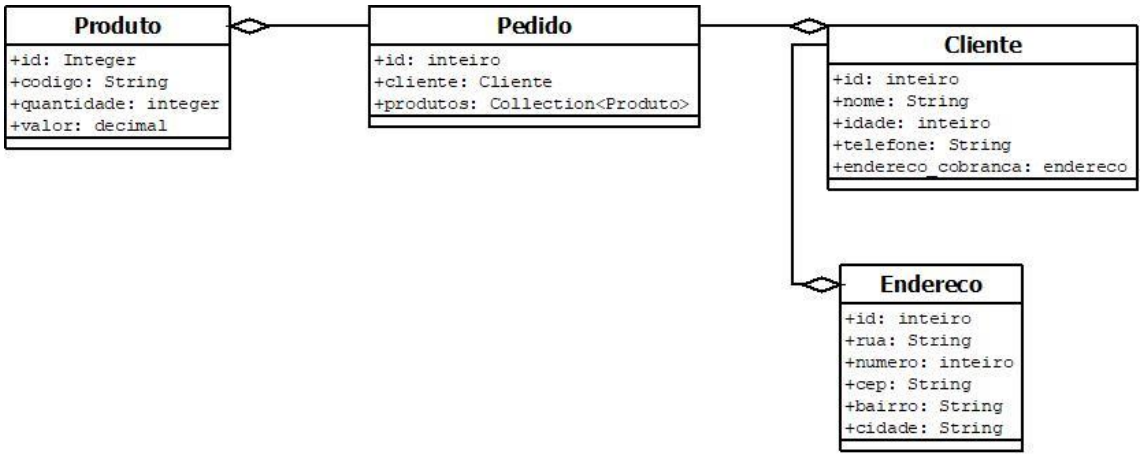
- Download: https://github.s3.amazonaws.com/downloads/rgl/redis/redis-2.4.6-setup-64-bit.exe?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAISTNZFOVBIJMK3TQ%2F20200524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200524T191917Z&X-Amz-Expires=300&X-Amz-SignedHeaders=host&X-Amz-Signature=bbcc50fe05dcaf6fa4302c48dc4743bab87c57dca42caaa13c373ba46b3a446c
- Executar o arquivo instalador: redis-2.4.6-setup-64-bit.exe
- Executar o serviço: C:\Program Files\Redis\redis-server.exe

- Executar o client: C:\Program Files\Redis\redis-cli.exe

Modelo de Banco de Dados



Entidades



Arquitetura

- Spring Data JPA + Cache Redis

Definimos a arquitetura Spring Data JPA + Cache Redis em um único projeto pois acreditamos ser a arquitetura mais adequada devido ao modelo de negócio exigir relacionamentos entre as entidades, portanto, um banco de dados relacional como o MySql mostra-se mais viável do que um banco de dados não relacional como MongoDB ou NEO4J neste último caso um banco de dados baseado no teorema de grafos não se aplica ao modelo de negócio que este projeto implementa.
A opção de uso de um sistema de cache, neste caso o Redis, também atende melhor a necessidade de negócio ao qual esse projeto atende pois otimiza as buscas e a performance do sistema não só ajudando na disponibilidade como também melhorando a experiência do usuário final.