FIAP

MBA em Full Stack Developer

Microservices, Cloud e IoT

PERSISTÊNCIA EM JAVA

Lojas Center

São Paulo

2020

FIAP

MBA em Full Stack Developer

Microservices, Cloud e IoT

PERSISTÊNCIA EM JAVA

Lojas Center

Breno M. de Souza Ferreira – RM339008

Felipe Cavalcante Rodrigues – RM339128

Rafael Libanio Stanzione – RM339224

Sheldon Gomes Vieira – RM338251

São Paulo

2020

RESUMO

Este estudo tem como objetivo elaborar um programa em Java para um segmento de e-commerce disponibilizando o acesso via APIs e persistindo os dados transacionados em um banco de dados. Serão analisados os modelos de banco de dados SQL e Non SQL escolher e ofertar a melhor combinação de bancos proposta pela disciplina de Persistence. Será utilizado o Spring Data JPA para as interações com o banco de dados MySQL e o banco de dados Redis para melhorar a performance da solução fazendo cache dos dados mais acessados. Este estudo irá proporcionar uma análise do CAP Theorem.

Palavras-chave: APIs. SQL. Non SQL. Spring Data JPA. CAP Theorem.

SUMÁRIO

1 INTRODUÇÃO

O aumento do volume de dados produzido pelos sistemas de informação tem impulsionado a adoção de novas técnicas para a persistência dos dados em sistemas de banco de dados. Os bancos de dados relacionais (SQL) estão....

2. REQUISITOS DO PROJETO

Foi solicitado pela FIAP a elaboração de um projeto para a criação de um sistema de cadastro de produtos em um portal de e-commerce para explorar os conhecimentos adquiridos na disciplina de JAVA Persistence. Os requisitos exigidos para a elaboração do projeto são:

- O portal deve armazenar vários produtos e suas quantidades em estoque;
- Cada pedido deve possuir um cliente e um ou mais produtos;
- Um produto pode estar associado a um ou mais pedidos;
- Os produtos devem possuir um código, um nome, uma quantidade e um valor;
- Os clientes devem possuir dados pessoais e dados de entrega.

  A partir dos requisitos exigidos o projeto deverá contemplar os itens:
- Escrever o modelo do banco de dados usando MySQL;
- Definir as entidades necessárias;
- Desenvolver uma aplicação do tipo Java Project, onde todas as informações são fornecidas pelo usuário;
- Utilizar o Maven para o gerenciamento de dependências;
- Desenvolver o sistema utilizando uma das quatro combinações de técnicas:
    - Spring Data JPA + Cache Redis (os dois em um único projeto);
    - Spring Data JPA (em um projeto separado) e Neo4J (em outro projeto separado);
    - Spring Data JPA (em um projeto separado) e MongoDB (em outro projeto separado, sendo que neste projeto pode existir um ou mais endereços de entrega cadastrados);

    - Cassandra (em outro projeto separado, lembrando que o Cassandra não suporta relacionamentos, então a lógica associada a isso deverá ser implementada pela aplicação).
- Documentar e justificar a escolha das técnicas, assim como as decisões técnicas e de arquiteturas do sistema.

2.1 Referenciais Teóricos

A pesquisa e o desenvolvimento deste projeto serão fundamentados com o apoio dos temas abordados durante as aulas, desta forma, iremos definir os conceitos teóricos adquiridos que possuem uma relação direta com este projeto.

Serão exploradas as recomendações pertinentes e os pontos relevantes, criando uma base de informação que será utilizada para definir a melhor proposta.

2.1.1 Banco de Dados SQL

A Structured Query Language (SQL) é uma linguagem de consulta padrão utilizada em banco de dados relacionais. Um banco de dados relacional armazena os dados em forma de tabelas e permite o relacionamento entre estas tabelas. (STALLINGS, 2010)

2.1.2 Non-SQL

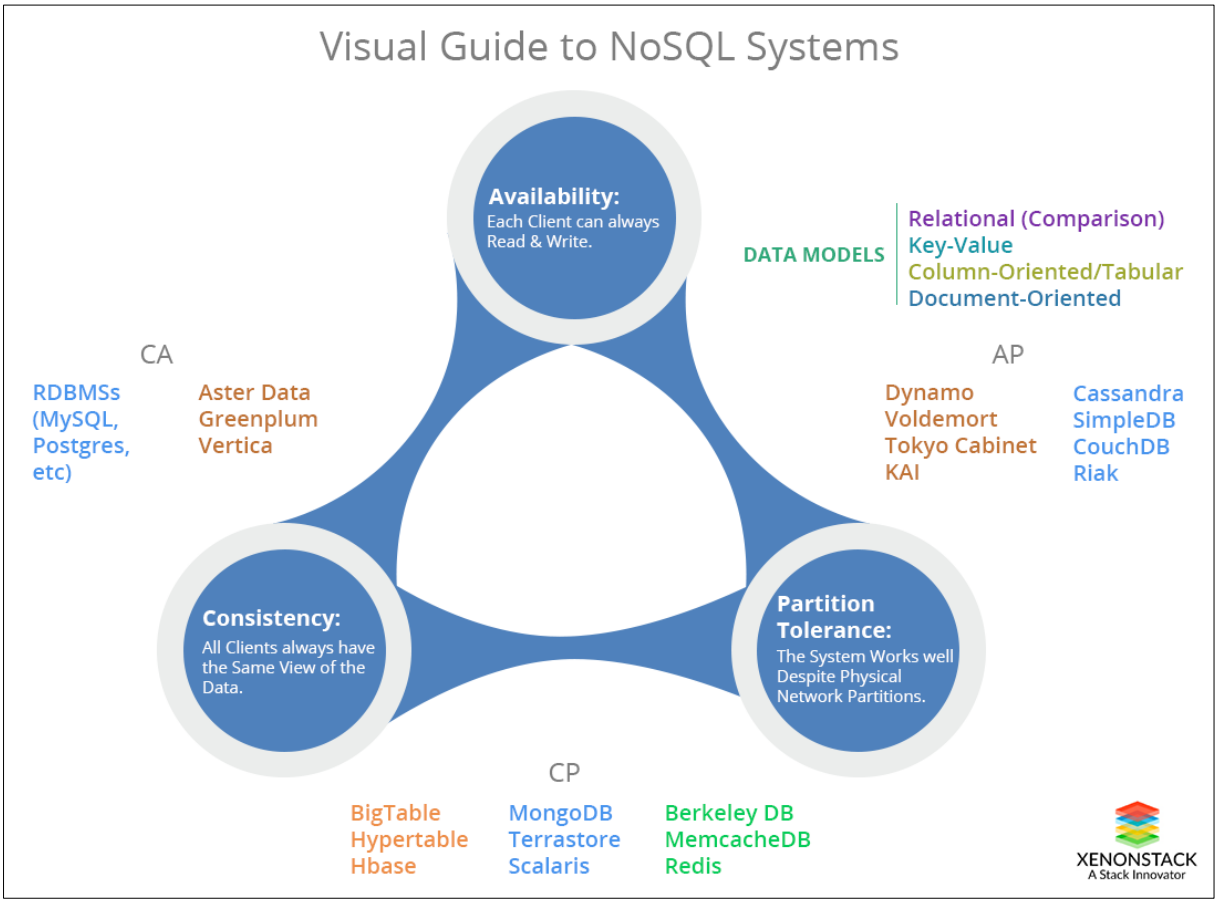
O Non-SQL ou Not only SQL é um termo utilizado para representar bancos de dados que não são relacionais

2.1.3 CAP Theorem

O CAP Theorem, também conhecido como Teorema de Brewer, é um teorema que afirma que não é possível, para um sistema de armazenamento de dados distribuído, garantir simultaneamente mais do que dois dos pilares descritos abaixo:

- Consistência (Consistency): todas as requisições de leitura recebem os dados mais recentes ou um erro;
- Disponibilidade (Availability): todas as requisições recebem uma resposta (sem erro), porém sem garantir que a resposta contém o dado mais recente;
- Partição tolerante a falhas (Partition Tolerance): o sistema continua em operação mesmo com um número grande de mensagens interrompidas ou atrasadas pela rede entre os nós.

  A imagem abaixo demonstra a relação entre os pilares e alguns tipos de banco de dados SQL e NoSQL:



Fonte: XenonStack (2020)

Apesar de um único sistema de banco de dados não garantir simultaneamente mais de dois pilares do Teorema de CAP, é possível aperfeiçoar a estratégia de persistência dos dados utilizando mais de um tipo de banco de dados.

2.1.4 Spring Data JPA

3 MATERIAIS E MÉTODOS

Neste capítulo serão definidos todos os elementos de *software* do projeto, seus relacionamentos e suas estruturas.

Iremos encontrar e descrever soluções para cada um dos objetivos apresentados pelos requisitos do projeto, aprofundando no conhecimento teórico adquirido ao longo do curso e durante uma breve pesquisa bibliográfica.

3.1 Projeto

O projeto foi criado e compartilhado entre os membros do grupo através do Github que é um sistema de gerenciamento de projetos e versões. Ele pode ser acessado conforme abaixo:

Tabela 1 – Acesso ao Projeto

Nome do Projeto	Link
lojas_center	https://github.com/fecrodrigues/lojas_center

Fonte: Elaborada Pelos Autores

3.2 Escolha do Banco de Dados

Um sistema de e-commerce deve ser capaz de manipular e persistir diversos dados, por exemplo, podemos definir alguns dados que devem conter para permitir uma simples operação de compra em um site:

1. O cliente tem que realizar o cadastro no site com as suas informações pessoais (nome completo, cpf, email e etc);
2. O cliente deve cadastrar um ou mais endereços para a entrega do produto que será comprado (cep, cidade, estado e etc);
3. O site deve possuir o cadastro de todos os produtos disponíveis para compra com os quantitativos em estoque (nome do produto, quantidade e etc).

Podemos identificar que um pedido de compra é formado por diversos dados que estão relacionados entre si, ou seja, um pedido possui cliente, endereço de entrega e produtos.

Iremos utilizar um banco de dados relacional para permitir o relacionamento entre as entidades (tabelas), o banco de dados que será utilizado é o MySQL. Este modelo de banco de dados, de acordo com o Teorema CAP, garante os pilares de disponibilidade e consistência.

Além disso, em uma loja virtual, é necessário garantir que o sistema responda a todas as requisições, pois elas irão concretizar a venda dos produtos da loja. Para garantir a velocidade da transação e diminuir o fluxo de acesso ao banco de dados SQL será adicionado um banco de dados Redis.

O banco de dados Redis utiliza a memória do servidor para o armazenamento de dados, tornando o processo de escrita e leitura muito mais rápido. O Redis será utilizado para cache dos dados de produtos e pedidos já que estes serão os mais consultados. A utilização deste banco irá permitir um processamento rápido das requisições e irá aumentar a capacidade de resposta em um alto volume de transações.

3.2.1 Estrutura do Banco de Dados SQL

Foi criado um diagrama de entidade relacionamento como base da estrutura do banco de dados SQL. Esta estrutura foi definida para permitir a construção do sistema de e-commerce. O formato das tabelas criadas no banco de dados pode ser visualizado abaixo:

Tabela 2 – Formato do Banco de Dados

Tabela (entidade)	Coluna (atributo)	Tipo
pedido	codigo	integer
	codigo_cliente	integer
	data_compra	datetime
pedido_produto	codigo_pedido	integer
	codigo_produto	integer
produto	codigo	integer
	nome	varchar(255)
	quantidade	integer
	valor	decimal
cliente	codigo	integer
	nome	varchar(170)
	data_nascimento	date
	cpf	varchar(11)
	nome_mae	varchar(170)
	email	varchar(200)
codigo_cliente	codigo_cliente	integer
	cep	varchar(8)
	logradouro	varchar(170)
	cidade	varchar(150)
	estado	varchar(150)
	numero	integer
	complemento	varchar(255)

Fonte: Elaborada Pelos Autores

O diagrama de entidade relacionamento pode ser visualizado na pasta docs do projeto (lojas_center/docs/modelo.pdf) ou no APÊNDICE A. O script de criação do banco de dados está disponível no projeto (lojas_center/docs/script.sql).

As propriedades de acesso ao banco de dados MySQL podem ser editadas no arquivo (/lojas_center/src/main/resources/database.properties), disponível no projeto.

3.2.2 Estrutura do Redis

O Redis será utilizado para cache dos dados de produtos e pedidos já que estes serão os dados mais consultados. A porta padrão do Redis é a 6379 e não foi configurada nenhuma autenticação de acesso ao banco. Porém, estas propriedades de acesso ao banco podem ser editadas no arquivo disponível no projeto (/lojas_center/src/main/resources/redis.properties).

3.3 Arquitetura do Sistema

A aplicação foi dividida e organizada em pastas de acordo com os tipos de classes e suas funções, conforme abaixo:

Tabela 3 – Arquitetura do Sistema

Estrutura	Objetivo
config	Configuração dos elementos do Sistema, tais como, web app, JPA, Redis, Swagger.
controller	Controle das interações externas via API
entity	Mapeamento das tabelas do banco de dados
exceptions	Tratamento dos erros do Sistema
main	Início da aplicação
model	Classes de transição entre os elementos do sistema
repository	Interações com o banco de dados
service	Lógicas de negócio
resources	Armazena as propriedades de configuração do banco de dados MySQL, Redis e do servidor web da API

Fonte: Elaborada Pelos Autores

3.4 Swagger

O Swagger é um conjunto de ferramentas de código aberto construídas com base na especificação OpenAPI. Ele foi utilizado no projeto para documentar e testar as APIs REST.

3.5 Maven

A biblioteca do Maven será responsável pelo controle dos pacotes. O arquivo pom.xml contém a lista de bibliotecas utilizadas no projeto.

4 CONSIDERAÇÕES FINAIS

Através deste projeto foi possível explorar algumas técnicas de persistência de dados e também aumentar o conhecimento em sistemas de APIs...

5 REFERÊNCIAS

XENONSTACK. **NoSQL Databases Overview, Types and Selection Criteria**. 2020. Disponível em: <https://www.xenonstack.com/blog/nosql-databases/>. Acesso em: 22 ago. 2020.

APÊNDICE A – DIAGRAMA DE ENTIDADE RELACIONAMENTO