Trabalho Final FDS

Erik Monteiro
Gabriel Quintana
Lucca Ianez
Wesley Pereira

Pontifícia Universidade Católica do Rio Grande do Sul

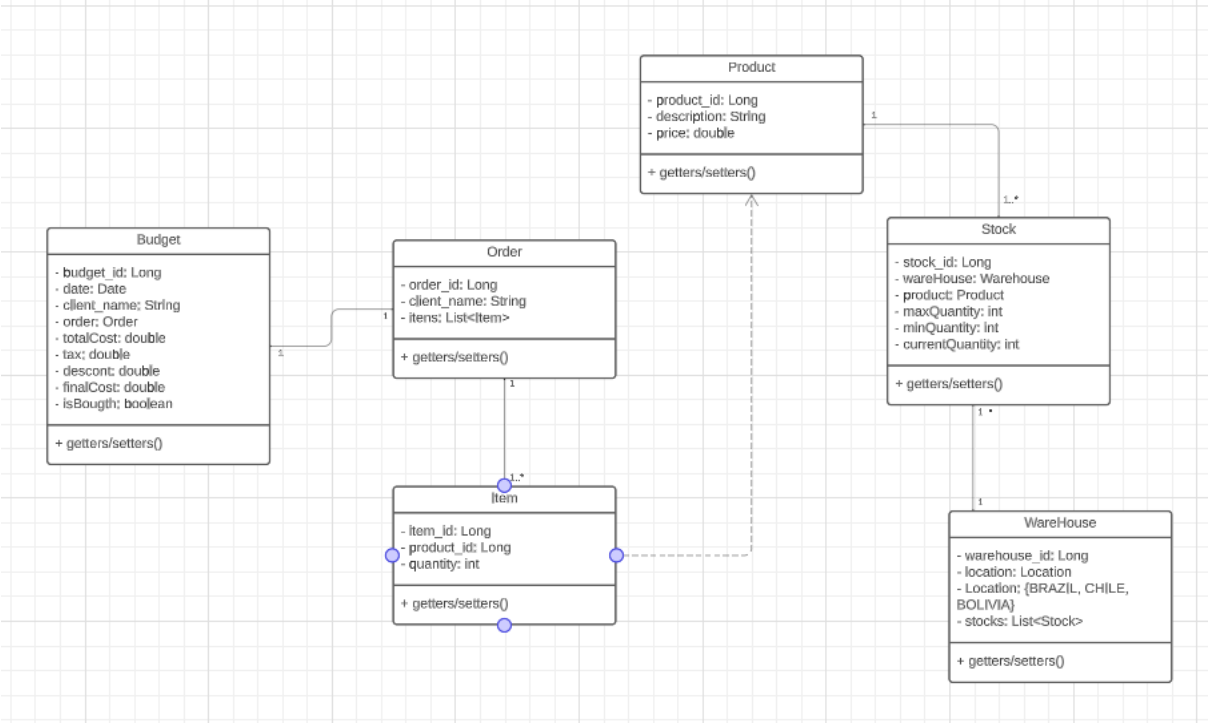
Introdução:

Fundamentos de desenvolvimento de software é uma disciplina em constante evolução, demandando a aplicação de princípios sólidos de engenharia para garantir soluções robustas e escaláveis. No contexto do semestre 2023/2, exploramos conceitos cruciais, como gerência de configuração e deploy, desenvolvimento em equipe, teste unitário, padrões de projeto e arquitetura de software.

O objetivo deste trabalho final é aprimorar o sistema de vendas, que foi inicialmente concebido durante as aulas, incorporando os conhecimentos adquiridos ao longo do semestre. Essa evolução baseia-se nos princípios SOLID, na arquitetura CLEAN e em padrões de projeto apropriados. O sistema, centrado na venda de produtos em uma loja online, visa não apenas atender aos requisitos funcionais originais, mas também incorporar novas funcionalidades que refletem desafios comuns em ambientes de comércio eletrônico.

Este trabalho representa não apenas a aplicação prática dos conhecimentos adquiridos, mas também uma oportunidade para aprimorar habilidades de desenvolvimento em equipe, compreensão aprofundada de arquiteturas de software e implementação de boas práticas de engenharia de software. Ao alcançar esses objetivos, esperamos fornecer uma solução sólida e bem estruturada que possa servir como referência para futuros projetos.

Diagrama de Classes:



Padrões de projetos utilizados e seus objetivos:

Padrão DDD + Clean Architecture:

- Objetivo das arquiteturas: Compreensão do Domínio do problema, separação de sesponsabilidades, modelagem rica e colaboração entre a equipe, independência de frameworks, código sustentável, independência de banco de dados.
- Implementação no Código:
    - As classes no pacote com.example.demo.presenter (como BudgetController, ProductController) representam a camada de controle Apresentação (Controller), que lida com a lógica de manipulação de solicitações HTTP.
    - As classes no pacote com.example.demo.domain.entity (como Budget, Order, Item, Product, Stock, WareHouse) representam a camada de domínio, que representa o ‘coração do negócio’.
    - As classes no pacote com.example.demo.domain.dto (como ItemDTO) podem ser vistas como objetos de transferência de dados (DTO) que transitam entre as camadas..

- As classes no pacote com.example.demo.domain.service são componentes responsáveis por lógicas de domínio que não se encaixam naturalmente em uma entidade ou objeto de valor.

- Interfaces como IRepBudgets, IRepOrders, IRepProducts, IRepStocks definem contratos para operações de acesso a dados.
- Implementações JPA dessas interfaces (por exemplo, RepBudgetsJPA, RepOrdersJPA, RepProductsJPA, RepStocksJPA) fornecem a lógica concreta para interagir com um banco de dados usando o Spring Data JPA.

Padrão Strategy:

- Objetivo: Definir uma família de algoritmos, encapsular cada um deles e torná-los intercambiáveis. Permite que o cliente escolha o algoritmo a ser utilizado dinamicamente.
- Implementação no Código:
    - A classe DiscountPolicyService pode ser vista como uma implementação do padrão Strategy. Ele encapsula as diferentes políticas de desconto e permite que o sistema escolha dinamicamente qual estratégia de desconto aplicar com base em certas condições.

Padrão Observer:

- Objetivo: Definir uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes sejam notificados e atualizados automaticamente.