Módulo de Vendas

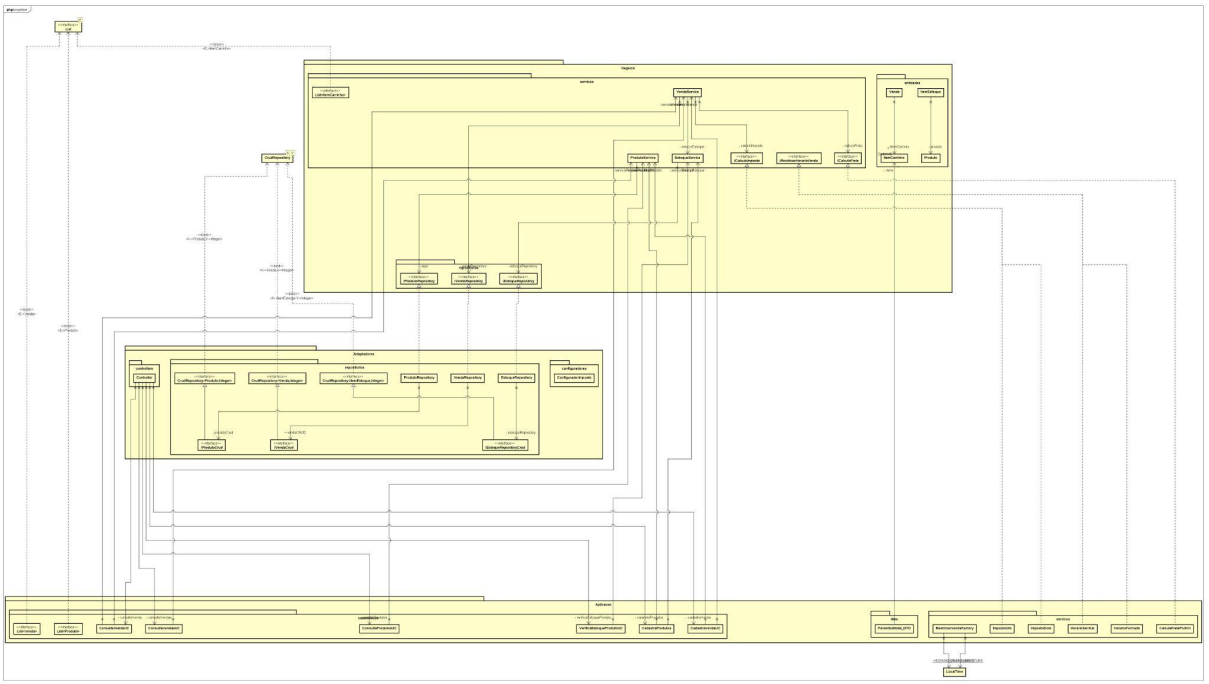
Trabalho de Modelagem de Sistemas usando a arquitetura CLEAN

Projeto e Arquitetura de Software - Prof. Bernardo Copstein

22/05/2022

Grupo: Arthur Sudbrack Ibarra, Augusto César Bottega Agostini, Felipe Grosze Nipper de Oliveira, Lucas Hanauer Leal, Miguel Torres de Castro.

Diagrama de classes da solução



Padrões usados com justificativa:

Arquitetura Clean: O projeto foi modelado usando a arquitetura clean, ou seja, o sistema foi dividido em camadas (negócio, aplicação e adaptadores). Essa separação foi feita para facilitar a organização do código e separar as regras do negócio e de sua implementação.

Repository: Este padrão foi utilizado para criar classes que abstraiam as funções básicas de acesso a dados. Além disso, junto com a arquitetura clean e o uso do padrão DIP, o padrão repository foi utilizado em partes, sendo a criação de interfaces na camada de negócio e a classe que a implementa sendo criada na camada de adaptadores.

Objeto de Transferência de Dados (DTO): Esse padrão foi aplicado com a finalidade de facilitar o acesso de informações entre os processos. No contexto deste projeto foi criado a classe ParamSubtotal_DTO que possui a função de armazenar informações como: itens no

carrinho e endereço. A utilização deste padrão facilitou a implementação dos casos de usos na camada de aplicação do projeto.

Dependency Inversion Principle (DIP) : Esse princípio foi utilizado juntamente com o Repository, fazendo com que os serviços que utilizam os repositórios não dependam de uma classe concreta e sim de uma interface.

Interface Segregation Principle (ISP) : Esse princípio foi utilizado em conjunto com o DIP e Repository e garante que os métodos implementados pelas interfaces criadas são apenas os necessários.

Injeção de Dependência: Através do framework utilizado, Springboot, é possível fazer a injeção de dependências em classes que compõem o sistema, como nas classes de repositório, serviço, e de use case, por meio das anotações *@Autowired* e *@Component*. Dessa maneira, não é necessário instanciar objetos manualmente e consequentemente colocar os parâmetros do construtor, pois tudo isso é feito automaticamente pelo Springboot.

Factory: O padrão Factory foi utilizado na classe RestricaoVenda que é responsável por aplicar as restrições às vendas realizadas na plataforma. Esse padrão foi escolhido pois, como as restrições são dependentes do horário, ele permite alterar a implementação dessas operações de acordo com as regras de horário.

Configurador: A estratégia de configurador foi utilizada na classe ConfiguradorImposto e permite com que, dependendo da configuração informada ao Springboot, o imposto seja calculado de forma diferente, visto que diferentes implementações do cálculo do imposto podem ser retornadas através de uma interface ICalculoImposto.