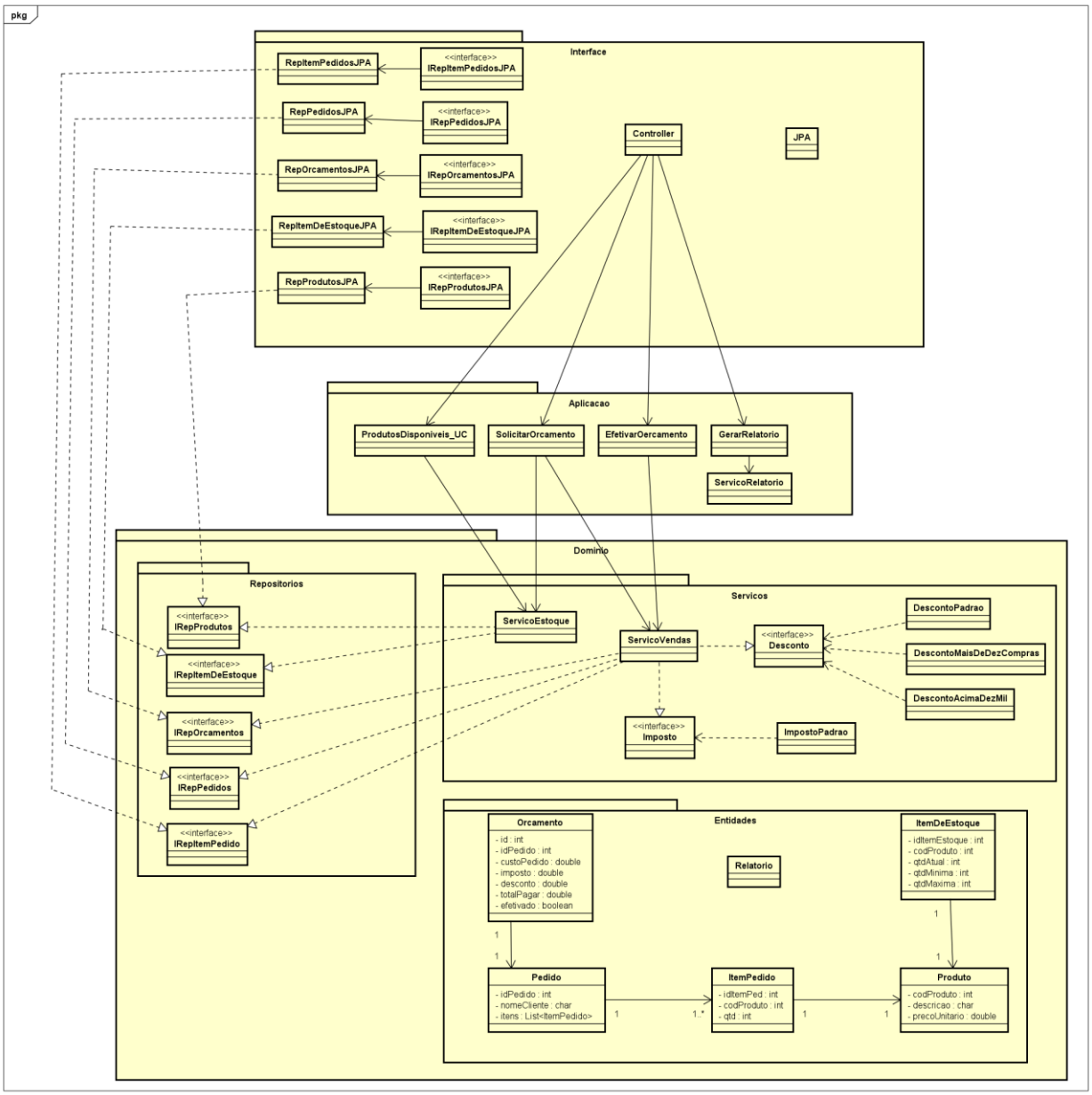
Trabalho Final de Fundamentos de Desenvolvimento de Software

Nomes: Gustavo Willian Martins, Lucas Schwartz, Lorenzo More, Miguel Warken e Vitor Caús

Professor: Bernardo Copstein

Diagrama de Classes Formulado:



Realizamos o projeto da loja utilizando uma arquitetura limpa de 3 camadas: interface, aplicação e domínio. Na interface, incluímos o controlador de requisições com o site e a tecnologia facilitadora para gestão do banco de dados: o JPA. As interfaces e classes dos repositórios JPA se conectam às suas semelhantes na camada do domínio. Na aplicação, foram colocadas as principais funções para o funcionamento do negócio, considerando o serviço relatório como algo desacoplável (motivo para o mesmo estar uma camada mais exterior do que os demais serviços).

No domínio existem 3 pastas para organização: os serviços, os repositórios e as entidades. Nos serviços, temos um que gerencia o estoque e outro que gerencia as vendas. É importante notar que tanto descontos quanto os impostos estão inclusos nesta seção da arquitetura, visto que ambos são calculados conforme são realizadas compras. Na pasta de repositórios, há as interfaces de orçamentos, produtos, itens de estoque, pedidos e itens de pedidos, implementadas nos repositórios presentes na camadaa de interface. Na última pasta (Entidades), existe o modelo de negócio, contendo entidades fundamentais para o funcionamento do projeto, e como elas interagem entre si.

Padrões de Projeto Utilizados:

No projeto da loja, utilizamos diversos padrões de projetos, a fim de facilitar a manutenção e legibilidade do código. Para a implementação dos descontos e impostos, utilizamos o padrão Strategy, que consiste em encapsular diversas classes, cada uma contendo seus cálculos próprios, tornando-as intercambiáveis, o que significa que objetos que interagem com tais classes poderão escolher qual utilizar sem a necessidade de alteração de código toda vez que se decidir usar diferentes valores.

O padrão de projeto Repository é utilizado para criar os repositórios de orçamentos, itens de estoque, produtos, pedidos e itens de pedidos. Esta técnica serve para manter a persistência e para a manipulação desses objetos, focando as operações de acesso a dados em classes dedicadas para estas. Isso permite que o restante do sistema interaja com os objetos de domínio sem se preocupar com os detalhes específicos da forma que os dados estão armazenados, aumentando a modularidade (a construção de um sistema maior em subsistemas) e, consequentemente, facilitando a manutenção do código.

Outro padrão bastante presente no código criado foi o de Injeção de Dependências, usado nos construtores de diversos elementos do projeto, como repositórios, serviços e aplicações, de modo contribuir para a coesão do sistema, reduzindo o acoplamento entre os diferentes módulos e promovendo uma arquitetura mais flexível, modular e de fácil manutenção. Esse padrão foi automatizado mediante o uso de outro padrão: a Inversão de Controle, de modo que o framework Springboot, através da anotação @Autowired, gerencie as dependências no momento da execução, de modo a fornecer as instâncias necessárias aos componentes e serviços para o correto funcionamento do programa.

Link do repositório: https://github.com/VitorCaus/TF_FundDesSoft/tree/master