Arquitetura Software

A arquitetura modelada para resolução do problema consiste na criação de classes que representam Pessoa Física e Pessoa Jurídica, que herdam os atributos comuns da classe abstrata Pessoa. Além disso, criou-se uma classe para representar os Produtos, que possui informações como um id único para cada produto, um nome e descrição e uma lista de alterações possíveis para se alterar o preço caso haja promoções, taxas extras ou cobrança de frete. A classe Carrinho é responsável por armazenar os produtos e as respectivas quantidades, atreladas a uma Pessoa (Física ou Jurídica). Esse armazenamento de produtos e quantidades foi feito através da estrutura Map. Os métodos para manipular os produtos estão implementados nessa classe. Por fim, através de interfaces e classes que as implementam, foi possível definir as regras de alterações do preço dos produtos e os métodos de calcular o preço do carrinho e dos produtos.

SOLID / Design Patterns

Foram utilizados 3 Design Patterns no projeto, dentre eles, Strategy, Factory e Singleton. Strategy foi utilizado, pois percebeu-se que as funcionalidades de criar promoções, adicionar frete e taxa extra eram muito similares, pois alteravam o preço final do produto. Com esse padrão de projeto, diminuímos a quantidade de if statements e conseguimos desacoplar as classes, fazendo o uso do princípio Aberto/Fechado, pois é possível adicionar novas funcionalidades que alteram o preço final do produto sem a necessidade de alterar as implementações existentes.

Já o Factory foi utilizado para, a partir da entrada de dados do usuário, criar o manipulador de repositório adequado. Foi necessária sua utilização, pois não se sabe de antemão quais dependências e objetos serão utilizados, pois variam a depender da entrada do usuário. Há a utilização do princípio Aberto/Fechado, pois novos tipos de entradas de dados podem ser adicionadas, sem que alterações no código já desenvolvido sejam necessárias, além do princípio da Responsabilidade Única, onde cada classe de manipulação de repositório é responsável por recuperar os dados da sua maneira. Por fim, foi utilizado Singleton para garantir que só há uma instância do manipulador de repositório que pega os dados em memória.