Relatório TF - Fundamentos de Desenvolvimento de Software

Anderson Sprenger, Gabriel Zurawski, Vinícius Dias

[1]Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681 - Partenon, Porto Alegre - RS, 90619-900

{anderson.sprenger, zurawski.gabriel, v.dias005}@edu.pucrs.br

1. Diagrama UML

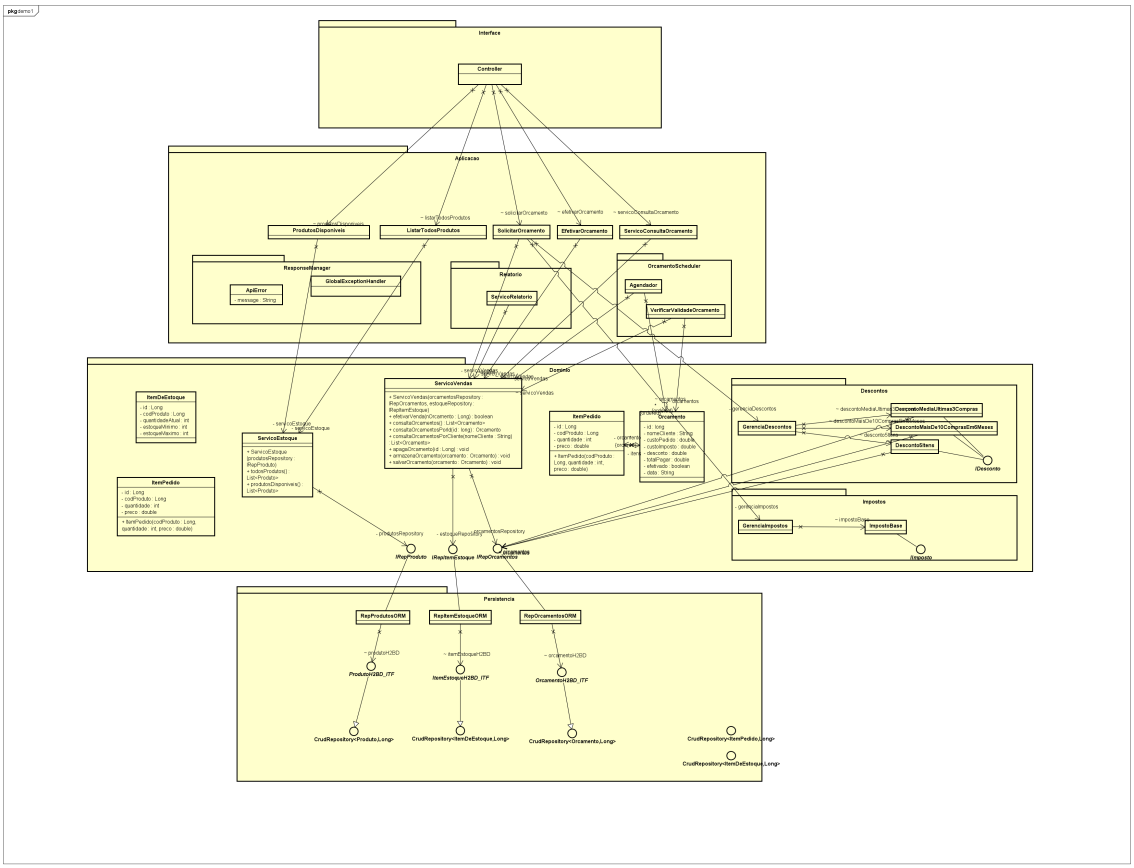

Figura 1. Diagrama UML

2. Padrões utilizados

Ao desenvolver o sistema representado no diagrama, aplicamos diversos padrões de design de software para garantir que o código fosse modular, fácil de manter e escalável. Abaixo está a descrição formal de como e onde cada padrão foi aplicado no projeto:

Padrão Repository: Implementamos interfaces de repositório, como OrcamentoH2BDITF, seguindo o padrão Repository para abstrair as camadas de acesso a dados do resto da aplicação. Isso permite uma troca fácil da fonte de dados e facilita o teste unitário dos componentes de negócios sem dependência direta do banco de dados.

Injeção de Dependência: Utilizamos extensivamente a injeção de dependência do Spring Framework, marcada pela anotação @Autowired, para desacoplar a criação dos objetos da lógica que os utiliza. Isso promove uma maior flexibilidade e facilita a substituição de componentes durante os testes automatizados.

Padrão Singleton: Todos os beans gerenciados pelo Spring são singletons por padrão, assegurando que uma única instância de cada bean seja criada e mantida no contêiner do Spring durante o ciclo de vida da aplicação.

Padrão Factory: Embora o gerenciamento de criação de objetos seja em grande parte tratado pelo Spring Data JPA, métodos como findAll() e findByid() em repositórios funcionam como fábricas que encapsulam a lógica de criação e recuperação de entidades.

Padrão Strategy: Definimos uma interface Descontos e várias implementações concretas para representar diferentes estratégias de cálculo de descontos. Isso permite que a lógica de cálculo de descontos seja intercambiável e facilmente extensível. O mesmo vale para a interface Impostos.

Padrão MVC (Model-View-Controller): O sistema segue o padrão MVC, com o Controller gerenciando as interações de entrada/saída, enquanto os modelos na camada de Dominio representam o estado e a lógica de negócios.