Seu Cantinho - Relatório

Heloísa Dias Viotto Guilherme Eduardo G. da Silva

29 de outubro de 2025

1 Introdução

Este documento tem como finalidade ser um relatório para o trabalho da disciplina Design de Software lecionada pela Professora Simone Dominico.

2 Estilo Arquitetural

O estilo arquitetural definida é "Em camadas"em conjunto com "Hexagonal", devido a sua vantagens para o trabalho.

2.1 Arquitetura em camadas

A vantagem que mais se destaca no estilo em camadas é o isolamento e a escalabilidade, solicitado pelo enunciado do trabalho. Além disso, essa arquitetura destaca-se por facilitar a manutenção e o desenvolvimento modular do sistema, além de possuir um baixo acoplamento.

Cada camada possui a sua responsabilidade, sendo independentes uns dos outros, cada um é responsável por uma função, sendo a comunicação realizada entre os componentes do sistema segue um fluxo hierárquico e controlado.

Para a modelagem do problema, foi decidido usar a divisão em 5 camadas, sendo elas:

- Camada de Apresentação: seria responsável pela interação com o usuário.
- Camada de Controlador: realiza a ligação entre a interface a a lógica de negócios.
- Camada de Serviço: implementação da lógica de negócios.
- Camada de Entidades: definição das entidades do projeto.
- Camada de Repositório: armazenamento dos dados da aplicação.

2.2 Arquitetura Hexagonal

Para auxiliar na modelagem do projeto, foi considerado utilizar conceitos também da arquitetura hexagonal com o intuito de ajudar na visualização da aplicação como um todo. Mesmo não sendo totalmente fiel à arquitetura em si, é possível identificar no diagrama de componentes (Figura 2), a separação entre adaptadores (Camada de controle e repositórios), portas (Camada de serviço) e o domínio (Camada de entidade).

2.3 Linguagem e comunicação entre as camadas - *REST*

Para auxiliar nessa tarefa, foi definida a linguagem de programação *JavaScript*, executada no ambiente de execução *Node.js*. Além disso, adotou-se como padrão de comunicação o estilo arquitetural "REST", que possibilita a interação entre os módulos da aplicação por meio de interfaces HTTP de forma simples, padronizada e independente de plataforma.

O *JavaScript* mostra-se particularmente adequado para o desenvolvimento de APIs REST, devido ao suporte nativo ao formato JSON e à integração facilitada entre as camadas de *backend* e *frontend*.

No que se refere ao *frontend*, foi definido o uso do *framework React Next.js*, que permite a criação de sites estáticos otimizados e aplicações dinâmicas, além de possibilitar a renderização de páginas no lado do

servidor (SSR). Essa abordagem melhora o desempenho, reduz o tempo de carregamento no cliente e facilita a integração direta com a API desenvolvida em *Node.js*. O *Next.js* também realiza automaticamente a compilação e construção do projeto, fornecendo atualizações instantâneas sem a necessidade de configurações adicionais.

Em um primeiro momento, foi decidido usar arquivos *JSON* como armazenamento das informações (*Repositórios*.

3 Detalhes de Implementação

Visando uma solução mais robusta do "Seu Cantinho", foram definidos algumas implementações que mantêm o sistema coerente, evitando problemas de condições de corrida entre espaços para serem reservados, perda de informações e um baixo tempo nas operações do sistema.

Nossa estrategia para resolver as condições de corrida seriam aplicar uma teoria baseada nas primitivas *lock* e *unlock*, o qual após o cliente confirmar o local para ser reservado, esse espaço seria bloqueado até a finalização da reserva (pagamento ou sinal) ou após um limite de tempo estabelecido. Essa implementação seria realizada na camada de repositório, onde os dados são armazenados.

Pensando em manter a consistência e a independência dos dados, cada módulo da aplicação possui seu próprio banco de dados, garantindo isolamento e evitando acoplamentos indesejados entre os serviços. Para intermediar o acesso a esses dados, foi adotado o padrão de projeto *Repository*, responsável por encapsular as operações de persistência e abstrair a comunicação entre o serviço e o banco de dados.

4 Diagramas UML

A seguir, encontram-se os diagramas UML de classe e componentes, sendo, respectivamente, as Figuras 1 e 2.

Figura 1: Diagrama UML de Classe.

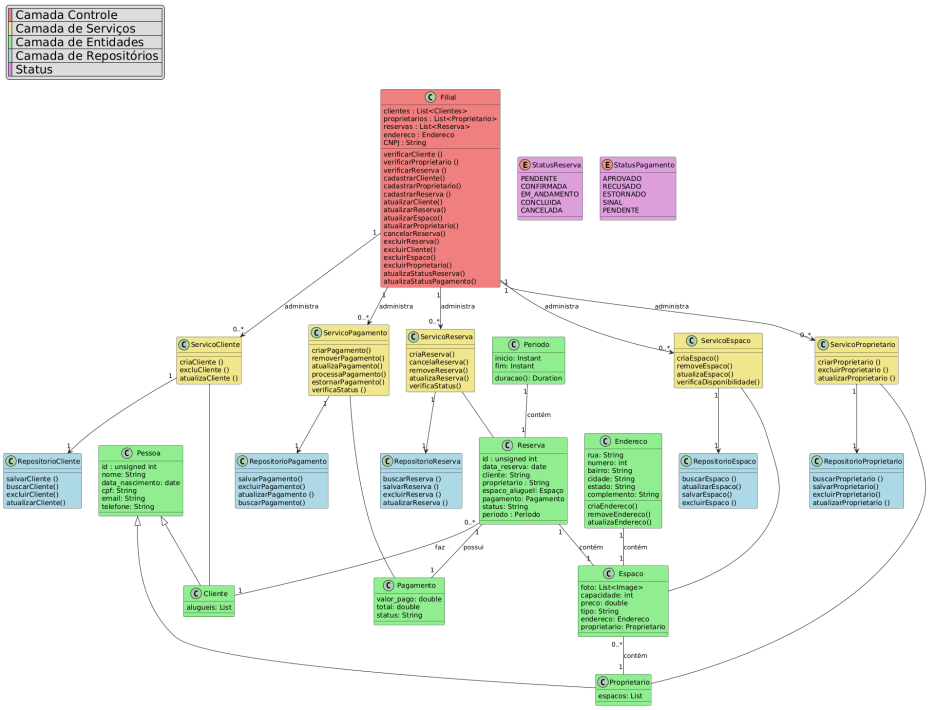

Figura 2: Diagrama UML de Componentes.