Seu Cantinho - Trabalho de Design de Sotware

Nathália Nogueira

02/2025

O objetivo deste relatório é descrever o desenvolvimento do trabalho da disciplina Design de Software, desde a definição de arquitetura e diagramas UML até a codificação e documentação.

O código-fonte, diagramas e documentações estão disponíveis em: https://github.com/nathalia-nogueira/seu-cantinho.git

1. **Solução proposta**
    a. **Estilo arquitetural**
    O estilo definido foi o **cliente-servidor com arquitetura em camadas**. O cliente, como apenas faz requisições, será implementado em um bloco monolítico que contém o *frontend*, enquanto o servidor será dividido nas camadas **Modelo, Serviço, Controlador** e **Repositório**. Essa arquitetura se destaca para o cenário de *Seu Cantinho* por sua:

    - **Simplicidade**: facilidade de desenvolvimento, testes e, principalmente, implantação.
    - **Escalabilidade**: mesmo que outros estilos arquiteturais (como microsserviços) ofereçam maior escalabilidade, a escalabilidade horizontal do estilo é mais do que o suficiente para o cenário de três estados e consideravelmente mais simples de implementar.
    - **Facilidade de manutenção**: garantida pela modularização do código em camadas.

    Com relação a tradeoffs identificados, há um grande ganho em simplicidade de desenvolvimento e implantação, além de consistência nas transações, enquanto escalabilidade e isolamento de falhas são colocados em segundo plano.

    b. **Diagramas UML**
    No desenvolvimento de *Seu Cantinho*, foram criados os diagramas UML de **classes** e de **componentes**. Ambos podem ser encontrados abaixo; contudo, para uma visualização mais clara e detalhada, recomenda-se acessá-los diretamente no repositório do *Github*.

a. Diagrama de Classes

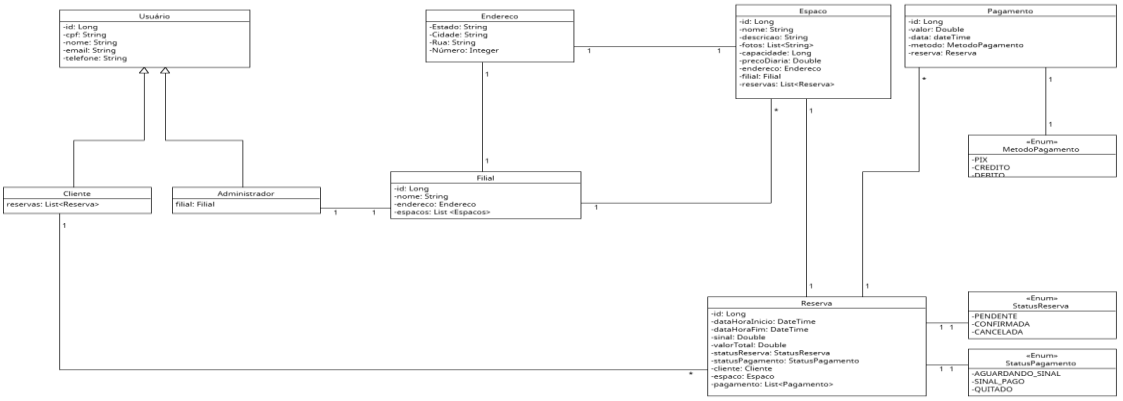

Figura 1 – Diagrama de Classes para Seu Cantinho

É importante destacar que, na figura 1, apenas a camada **Modelo** foi representada. As demais camadas não foram incluídas para evitar redundância, visto que serão compostas por classes correspondentes às principais entidades - usuário, espaço, reserva e pagamento. A responsabilidade de cada uma das camadas e, consequentemente, de suas respectivas classes, está descrita na seção 2.b.

Além disso, observa-se que as classes estruturadas dessa forma caracterizam um **modelo anêmico** - classes que contêm apenas atributos, sem métodos associados. Ainda assim, é a solução adequada para o problema do *Seu Cantinho*, pois permite a utilização de transações atômicas (evitando o problema de double-booking) e conta com simplicidade e clareza úteis nesse contexto.

b. Diagrama de Componentes

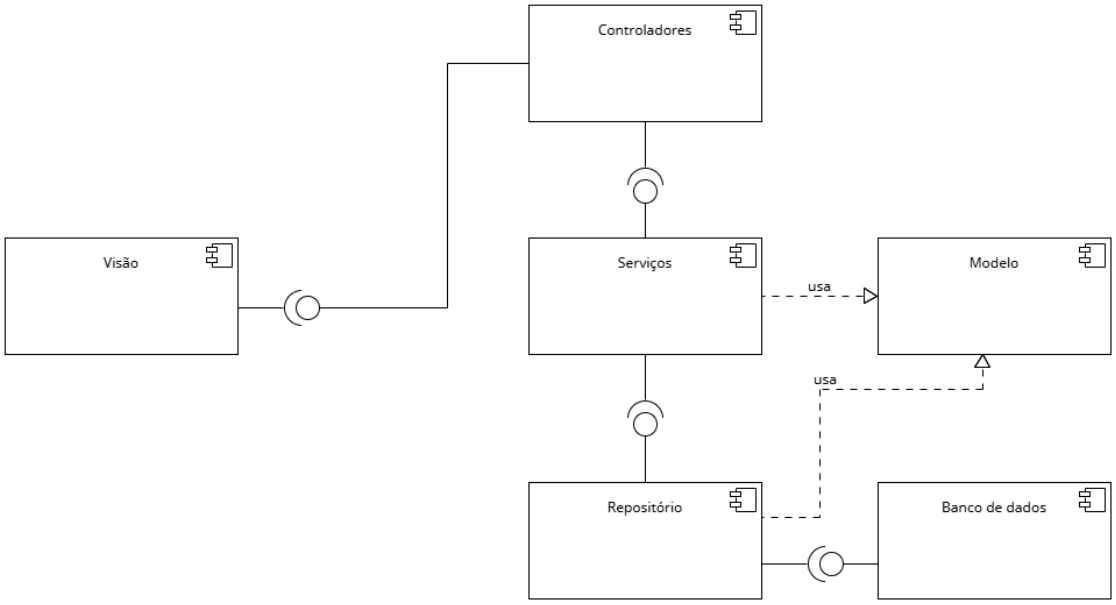

Figura 2 – Diagrama de Componentes para Seu Cantinho

Como representado na figura 2, a arquitetura é dividida nos seguintes componentes:

1. **Visão**: interface com o usuário que consome a API REST
2. **Controladores**: recebe requisições HTTP e as direciona para a camada de Serviços
3. **Serviços**: contém a lógica de negócio central
4. **Repositório**: define as interfaces de acesso aos dados – traduz as chamadas dos Serviços em consultas ao Banco de Dados
5. **Banco de dados**: garante persistência e atomicidade das transações

2. Aderência da solução aos requisitos

A modelagem apresentada foi definida para solucionar os problemas operacionais relatados no uso do sistema anterior. A própria estrutura de classes do Modelo já endereça os requisitos funcionais básicos:

- **Múltiplas filiais**: a expansão para três estados é tratada pela classe Filial, que se relaciona com Espaco e Administrador. Isso permite que o sistema gerencie espaços e administradores por localidade, mantendo a consistência entre filiais.
- **Cadastro dos espaços**: feito pela entidade Espaco
- **Gerenciamento de reservas**: fornecido pela classe Reserva
- **Controle de pagamentos**: modelado por Pagamento e StatusPagamento

Contudo, a questão mais crítica era o **double-booking**, principal queixa de Dona Maria. A prevenção de reservas conflitantes é uma responsabilidade da camada de Serviço, que segue o fluxo:

1. O usuário (via Visão) solicita uma nova reserva, recebida pelo Controlador
2. O Controlador chama a camada de Serviço, que é responsável pela lógica de negócio. Antes de salvar a nova reserva, o Serviço usa o Repositório para consultar o Banco de Dados e verificar se existe reserva conflitante.
3. Se a consulta retornar um conflito, o Serviço cancela a operação e informa o usuário. Se não houver conflito, o Serviço cria a reserva.

Para garantir a integridade em casos de requisições simultâneas, todo esse processo é executado em uma **transação atômica** do banco de dados. Isso garante que, se duas requisições conflitantes chegarem ao mesmo tempo, o mecanismo de lock do banco fará com que a primeira finalize a operação antes que a segunda possa verificar a disponibilidade. A segunda, então, “enxergará” a reserva criada pela primeira e será devidamente rejeitada.