Relatório Jackut

Introdução

Este relatório descreve o design do sistema Jackut, responsável pela gestão de usuários e suas interações no sistema. O sistema foi desenvolvido utilizando princípios sólidos de engenharia de software, com o objetivo de garantir um funcionamento robusto e eficiente. A principal arquitetura adotada foi a **MVC** (Model-View-Controller), onde a lógica de negócios, a persistência de dados e as interações com o usuário estão bem definidas e isoladas.

Funcionalidades

O sistema Jackut oferece funcionalidades como:

- Criação de usuários
- Edição de perfil
- Gerenciamento de amizades
- Envio e recebimento de recados
- Gerenciamento de comunidades

Padrões de Design

Facade Pattern: A classe Facade foi projetada para fornecer uma interface simplificada para o usuário interagir com o sistema. Este padrão de design foi utilizado para encapsular a complexidade do sistema e expor apenas os métodos essenciais, tornando o sistema mais acessível.

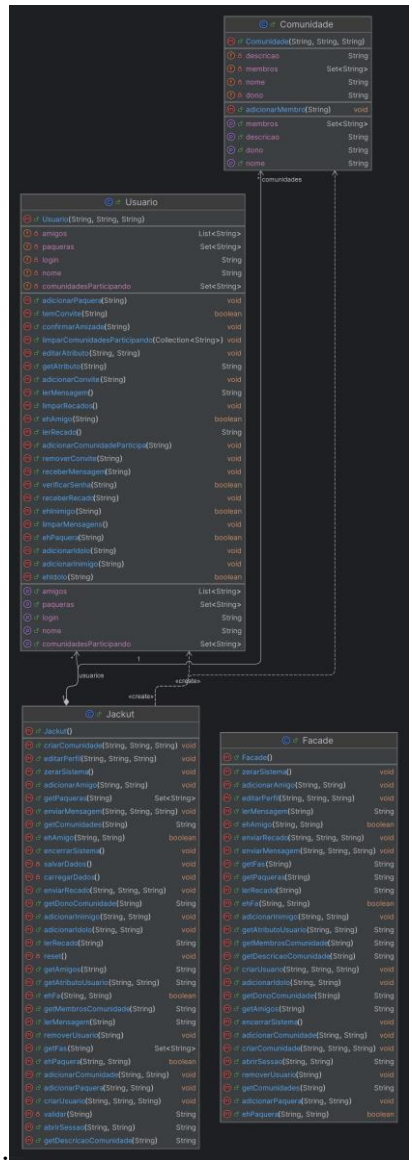
Singleton Pattern (Implícito): A classe Jackut serve como um ponto central de controle e persistência de dados, permitindo que o sistema tenha uma única instância de dados compartilhada, garantindo que não haja duplicação e que os dados sejam acessados de maneira centralizada.

Diagrama de Classes

O diagrama de classes abaixo ilustra a estrutura do sistema e os principais relacionamentos entre as classes. As principais classes do sistema são:

1. **Usuario**: Responsável por armazenar informações do usuário, como login, senha, nome, lista de amigos, recados, e convites pendentes. Além disso, gerencia atributos do usuário e lida com exceções específicas como AtributoNaoPreenchidoException.
2. **Jackut**: Gerencia a lógica de negócios, incluindo a criação de usuários, sessões de login, edição de perfil e adição de amigos. Também lida com a persistência de dados no arquivo e interage com exceções como LoginInvalidoException.

3. **Facade**: Encapsula a complexidade do sistema e expõe uma interface simples para as operações do cliente, permitindo que o usuário interaja facilmente com o sistema sem se preocupar com detalhes internos.



Design e Escolhas

O design do sistema foi focado na **modularidade** e **clareza**, com a separação de responsabilidades entre as classes para facilitar a manutenção e os testes. O uso de coleções como Map, List e Set foi escolhido com base em considerações de desempenho e clareza:

- **Map** foi utilizado para armazenar usuários e sessões, garantindo buscas eficientes com complexidade constante.
- **List** foi escolhida para armazenar os amigos, preservando a ordem em que foram adicionados.
- **Set** foi usado para armazenar convites, evitando duplicações, já que a ordem não é relevante.

Além disso, o uso de exceções específicas para cada cenário (como UsuarioNaoCadastradoException e UsuarioJaEstaNaComunidadeException) visa garantir uma boa experiência ao usuário, prevenindo falhas inesperadas no sistema.