Relatório do Design do Projeto Jackut

Introdução

O projeto Jackut é uma plataforma de rede social simplificada, implementada com foco na persistência de dados, eficiência na manipulação de usuários e interação social. Este relatório detalha as escolhas de design feitas no desenvolvimento do sistema, incluindo as estruturas de dados utilizadas, a implementação de funcionalidades e o tratamento de exceções.

Persistência de Dados e Inicialização do Sistema

O sistema possui um mecanismo de resgate de dados salvos no armazenamento local sempre que uma nova instância é criada. Isso ocorre no construtor da classe principal do sistema, permitindo que os dados sejam restaurados após o encerramento do programa. Essa abordagem evita a perda de informações e garante continuidade na experiência do usuário.

O método `encerrarSistema` é responsável por serializar e salvar os dados no armazenamento local. Em contraste, o método `zerarSistema` exclui o arquivo de serialização, garantindo que testes subsequentes sejam feitos sem interferência de execuções anteriores.

Estruturas de Dados e Justificativas

Gerenciamento de Usuários

A escolha de um `HashMap<String, Usuario>` para armazenar os usuários foi motivada pela eficiência de busca e inserção dessa estrutura, garantindo acesso rápido aos perfis cadastrados.

Os pedidos de amizade e a lista de amigos foram armazenados em `LinkedHashSet`, uma vez que essa estrutura:

- Mantém a ordem de inserção, refletindo a ordem cronológica dos pedidos e amigos.
- Evita duplicatas, garantindo que um usuário não possa ser adicionado mais de uma vez na lista de amigos ou de pedidos.

Mensagens e Recados

Os recados foram implementados como uma fila (`Queue`), garantindo que sejam lidos na ordem em que foram recebidos, conforme exigido pelos user stories.

Implementação de Sessões

O método `abrirSessao` atualmente retorna apenas o login do usuário como identificador. Essa escolha foi feita devido às necessidades limitadas dos user stories implementados até o momento, que não exigiam um gerenciamento mais complexo de sessões.

Design da Facade

A classe `Facade` gerencia a interação com o sistema e é responsável por criar novas instâncias do sistema quando necessário. A separação entre `Facade` e `Sistema` foi feita para manter o código organizado e facilitar a leitura.

Uso de `switch-case`

Os métodos `getAtributo` e `editarPerfil` utilizam `switch-case` para facilitar a leitura e manutenção do código, tornando o acesso e edição de atributos mais estruturados.

Tratamento de Exceções

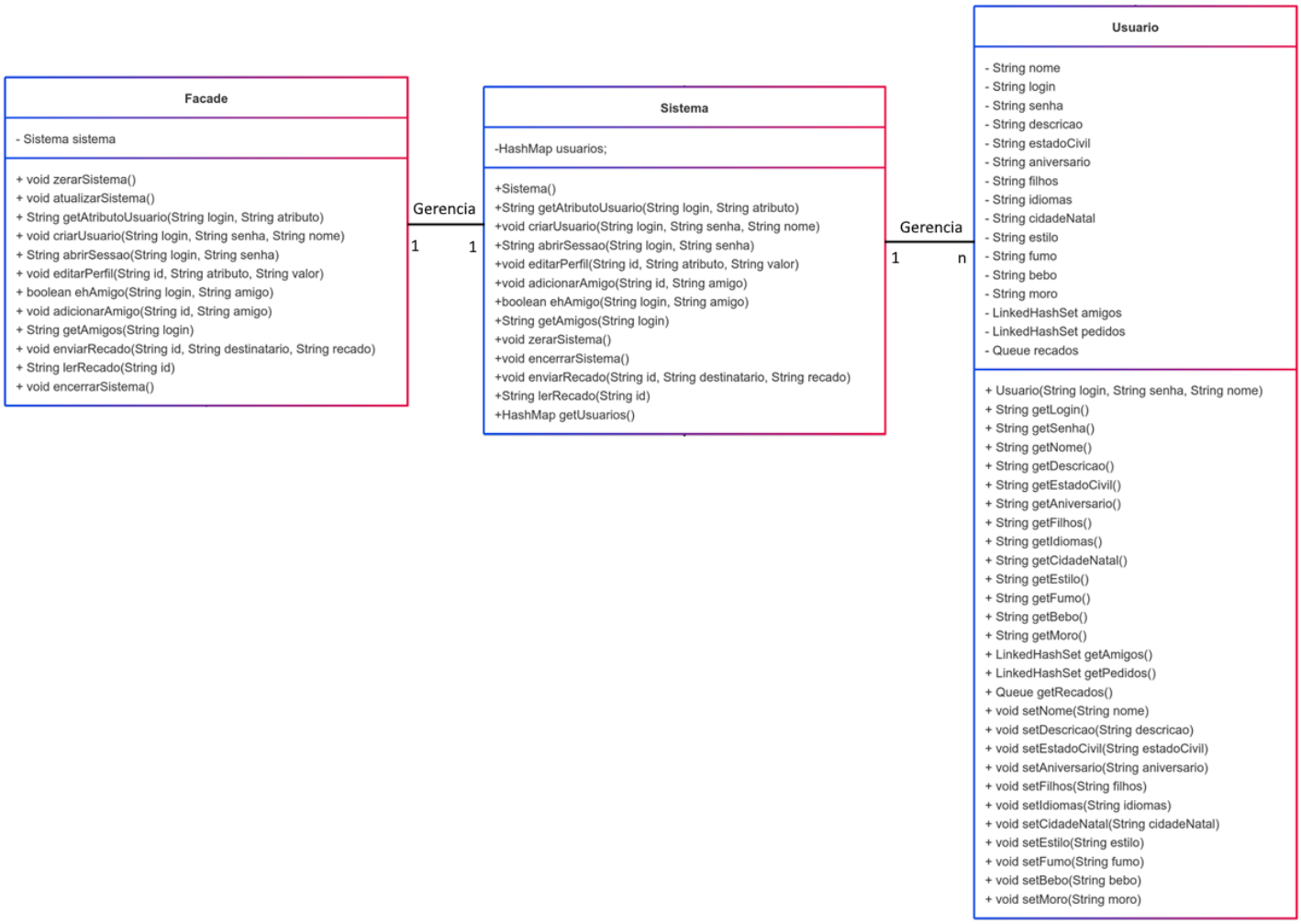
O sistema conta com exceções personalizadas para garantir uma melhor gestão de erros:

- **FobiddenOperationException**: Lançada ao tentar realizar operações proibidas pelas diretrizes do programa.
- **ConflictingAuthenticationException**: Indica tentativa de criação de credenciais já existentes.
- **NullCredentialException**: Lançada quando uma credencial vazia ou inválida é fornecida.
- **NullAttributeException**: Indica acesso a atributos nulos ou não preenchidos.
- **InvalidUserException**: Indica tentativa de acesso a usuários não cadastrados.
- **InvalidCredentialException**: Utilizada para credenciais inválidas.
- **EmptyListException**: Indica tentativa de acessar uma lista vazia.
- **NotNeededOperationException**: Utilizada quando uma operação já foi ou não precisa ser realizada.

Conclusão

O design do Jackut prioriza eficiência, manutenção fácil e integração simplificada. As estruturas de dados foram escolhidas com base nos requisitos funcionais, enquanto as exceções garantem um tratamento de erros robusto. A separação entre `Facade` e `Sistema` melhora a organização do código, e a persistência de dados assegura uma experiência contínua para o usuário.

Segue o diagrama de classes


Facade
- Sistema sistema
+ void zerarSistema()
+ void atualizarSistema()
+ String getAtributoUsuario(String login, String atributo)
+ void criarUsuario(String login, String senha, String nome)
+ String abrirSessao(String login, String senha)
+ void editarPerfil(String id, String atributo, String valor)
+ boolean ehAmigo(String login, String amigo)
+ void adicionarAmigo(String id, String amigo)
+ String getAmigos(String login)
+ void enviarRecado(String id, String destinatario, String recado)
+ String lerRecado(String id)
+ void encerrarSistema()
Gerencia
1
1
Sistema
-HashMap usuarios;
+Sistema()
+String getAtributoUsuario(String login, String atributo)
+void criarUsuario(String login, String senha, String nome)
+String abrirSessao(String login, String senha)
+void editarPerfil(String id, String atributo, String valor)
+void adicionarAmigo(String id, String amigo)
+boolean ehAmigo(String login, String amigo)
+String getAmigos(String login)
+void zerarSistema()
+void encerrarSistema()
+void enviarRecado(String id, String destinatario, String recado)
+String lerRecado(String id)
+HashMap getUsuarios()
Gerencia
1
n
Usuario
- String nome
- String login
- String senha
- String descricao
- String estadoCivil
- String aniversario
- String filhos
- String idiomas
- String cidadeNatal
- String estilo
- String fumo
- String bebo
- String moro
- LinkedHashSet amigos
- LinkedHashSet pedidos
- Queue recados
+ Usuario(String login, String senha, String nome)
+ String getLogin()
+ String getSenha()
+ String getNome()
+ String getDescricao()
+ String getEstadoCivil()
+ String getAniversario()
+ String getFilhos()
+ String getIdiomas()
+ String getCidadeNatal()
+ String getEstilo()
+ String getFumo()
+ String getBebo()
+ String getMoro()
+ LinkedHashSet getAmigos()
+ LinkedHashSet getPedidos()
+ Queue getRecados()
+ void setNome(String nome)
+ void setDescricao(String descricao)
+ void setEstadoCivil(String estadoCivil)
+ void setAniversario(String aniversario)
+ void setFilhos(String filhos)
+ void setIdiomas(String idiomas)
+ void setCidadeNatal(String cidadeNatal)
+ void setEstilo(String estilo)
+ void setFumo(String fumo)
+ void setBebo(String bebo)
+ void setMoro(String moro)