UNIVERSIDADE FEDERAL DE ALAGOAS
DISCIPLINA: PROGRAMAÇÃO 2
ALUNA: CARLA EVELLYN FERREIRA DOS SANTOS

RELATÓRIO DO MILESTONE 2

MACEIÓ - AL, 05/05/2025

1. INTRODUÇÃO

Este relatório apresenta a avaliação detalhada do sistema Jackut, desenvolvido como parte do Milestone 2 da disciplina de Programação 2. O projeto consiste em uma rede social que gerencia usuários, comunidades e relacionamentos, implementada em Java com foco em princípios de orientação a objetos e padrões de design. O relatório está organizado em três seções principais:

- Seção 2 (Avaliação): Análise crítica do design e implementação, destacando virtudes e fraquezas do sistema.

- Seção 3 (Refatoramento e Desenvolvimento): Descrição das funcionalidades implementadas, padrões utilizados e diagramas de classe.

- Seção 4 (Conclusão): Avaliação geral do trabalho e sugestões para melhorias futuras.

2. AVALIAÇÃO

O sistema Jackut apresenta um design modular e bem estruturado, com separação clara de responsabilidades. A seguir, são detalhados os pontos fortes e as áreas que necessitam de aprimoramento.

2.1 VIRTUDES DO PROJETO

1. *Arquitetura Modular e Baixo Acoplamento:*

- O sistema está organizado em pacotes coesos (models, managers), cada um com responsabilidades bem definidas.

- A classe Facade atua como ponto único de entrada, delegando operações para os managers especializados (UserManager, CommunityManager, etc.), seguindo o princípio de Single Responsibility.

2. *Persistência em XML:*

- Utilização da biblioteca XStream para serialização em XML, facilitando a leitura e manutenção dos dados.

- A classe PersistenceManager implementa o padrão Singleton para garantir uma única instância durante a execução.

3. *Tratamento de Exceções:*

- Mensagens de erro claras são lançadas em situações inválidas (ex: RuntimeException("Usuário não cadastrado")).

- Validações robustas em métodos como adicionarAmigo() e criarComunidade() previnem estados inconsistentes.

4. *Padrões de Design:*

- **Facade**: Simplifica a interação com o sistema através da classe Facade.

- **Singleton**: Garante acesso único ao PersistenceManager.

- **Composição**: Relacionamentos entre Usuario, Perfil e Comunidade são modelados com composição (ex: Usuario tem um Perfil).

5. *Documentação:*

- JavaDoc completo para classes e métodos principais (ex: Usuario, UserManager).

2.2 FRAQUEZAS DO PROJETO

1. *Exceções Genéricas:*

- Uso excessivo de RuntimeException em vez de exceções customizadas (ex: UsuarioNaoEncontradoException), o que dificulta a identificação de erros específicos.

2. *Redundância em Métodos:*

- Lógica de ordenação repetida em métodos como getAmigos() e getFas(), que poderiam ser abstraídos em uma classe utilitária (ex: OrdenacaoUtil).

3. *Acoplamento em Usuario*:

- A classe Usuario possui muitas responsabilidades (gerenciar amigos, recados, mensagens, etc.), violando o princípio Single Responsibility. Sugere-se dividir em classes menores (ex: GerenciadorDeRecados).

4. *Testes de Aceitação Limitados:*

- Os testes com EasyAccept cobrem casos básicos, mas não incluem cenários extremos (ex: remoção de usuário com múltiplas comunidades).

5. *Falta de Interfaces para Polimorfismo:*

- Métodos como adicionarAmigo() e enviarRecado() poderiam implementar interfaces (ex: IRelacionamento) para maior flexibilidade.

3. REFATORAMENTO E DESENVOLVIMENTO REALIZADO

3.1 Funcionalidades Implementadas

1. *Gerenciamento de Usuários:*

- **Criação/Remoção**: Métodos criarUsuario() e removerUsuario() no UserManager, com validações de login e senha.

- **Persistência**: Dados salvos em usuarios.xml via PersistenceManager.

2. *Relacionamentos:*

- **Amizades:** Validação de autoamizade e inimizade em adicionarAmigo().

- **Recados:** Fila de mensagens (recadosRecebidos) com identificação do remetente.

3. *Comunidades:*

- **Criação**: criarComunidade() no CommunityManager, com dono e membros.

1. **Mensagens em Massa:** enviarMensagemParaComunidade() notifica todos os membros.

4. *Sessões*:

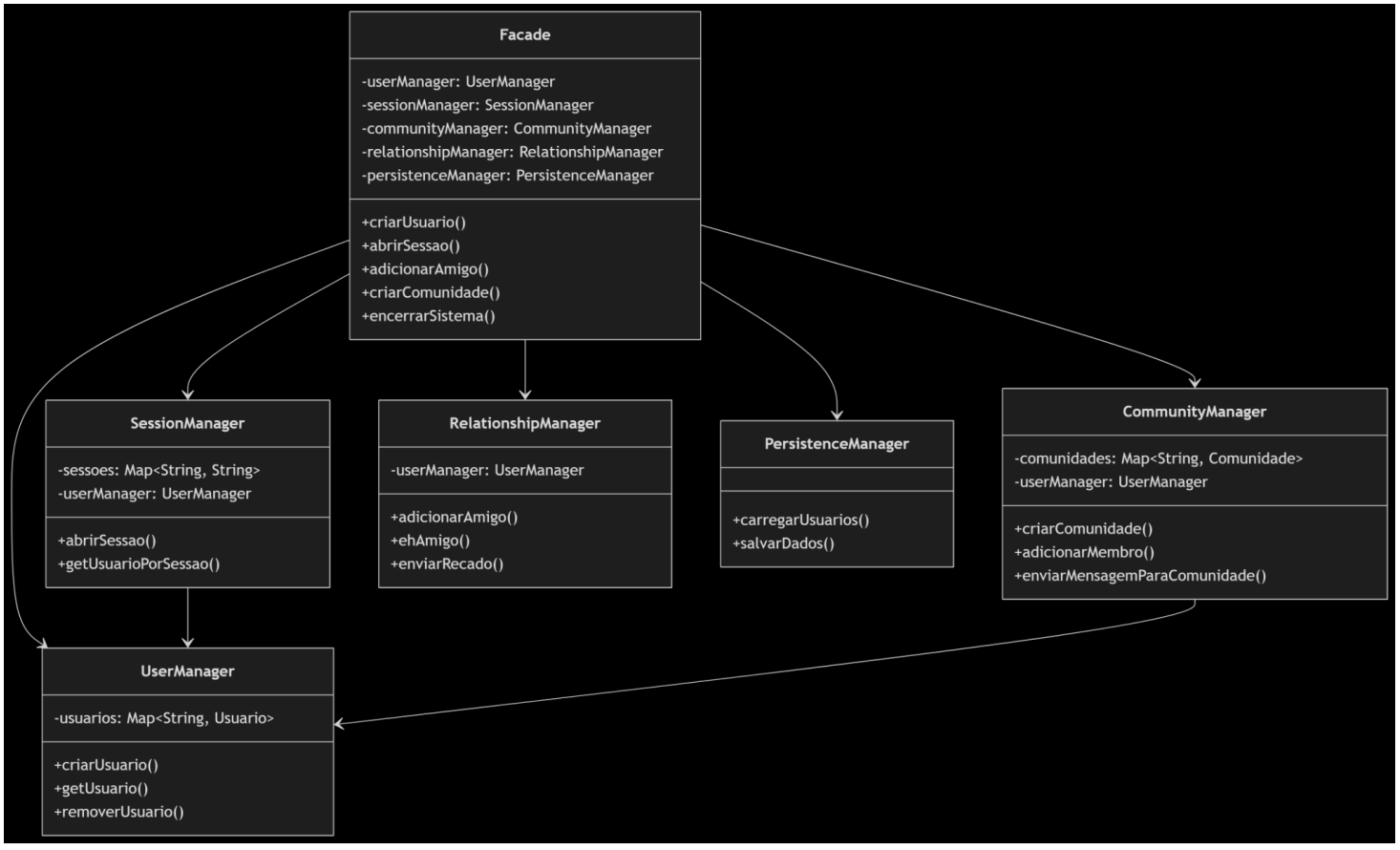
- Controle de sessões ativas no SessionManager, com IDs únicos (sessao_1, sessao_2).

3.2 Padrões de Design Utilizados

Padrão	Aplicação	Benefício
Facade	A classe Facade unifica acesso aos managers.	Simplifica a API para o cliente.
Singleton	PersistenceManager garante única instância para persistência.	Evita conflitos em operações de I/O.
Composição	Usuario contém Perfil e Comunidade contém Usuario.	Modelagem fiel ao domínio do problema.

3.3 DIAGRAMA DE CLASSE (SIMPLIFICADOS)

- Diagrama 1: estrutura principal do sistema



- Diagrama 2: Hierarquia de classes de modelo

4. CONCLUSÃO

O desenvolvimento do Jackut foi uma experiência valiosa para aplicar conceitos de programação orientada a objetos, padrões de design e boas práticas de codificação. O sistema atende aos requisitos propostos, com funcionalidades completas para gerenciamento de usuários, relacionamentos e comunidades.

4.1 AVALIAÇÃO OBJETIVA

- Qualidade da Documentação: 8,5 (JavaDoc presente, mas alguns métodos poderiam ser mais detalhados).
- Qualidade do Design: 9,0 (Arquitetura modular e baixo acoplamento).
- Qualidade do Código: 9,0 (Código limpo, mas com oportunidades para reduzir redundâncias).

Assinatura:

Carla Evellyn Ferreira dos Santos || 05/05/2025