Relatório de Design - Sistema Jackut (Milestone 1)

Introdução

Desenvolvi o sistema Jackut como uma rede social simplificada, com foco na gestão individual de usuários, relacionamentos e troca de mensagens. Nesta primeira etapa, implementei quarto user stories essenciais: cadastro de usuários (US1), edição de perfis (US2), sistema de amizades (US3) e troca de recados (US4). A arquitetura que concebi prioriza a separação de responsabilidades, seguindo um modelo em três camadas principais que garante baixo acoplamento e alta coesão entre os componentes.

Arquitetura do Sistema

Na camada superior, criei a classe Facade como interface única para os testes do EasyAccept, traduzindo comandos textuais em operações do sistema. A camada intermediária, representada pela classe Sistema, concentra a lógica de negócio que desenvolvi para coordenar as operações entre usuários. Por fim, a camada de dados é composta pelas entidades Usuario e Recado que implementei, as quais encapsulam as regras específicas de cada elemento. Esta divisão permitiu que cada componente fosse evoluído de forma independente, facilitando a manutenção e extensão do código.

Persistência de Dados

Implementei um mecanismo de serialização binária que grava o estado completo do sistema no arquivo "dados_jackut.ser" ao encerrar a execução. Esta solução que elaborei mostrou-se eficiente para garantir a persistência de todas as informações entre execuções, incluindo perfis de usuários, relacionamentos e mensagens trocadas. Um desafio particular que superei foi garantir que estruturas de dados complexas, como a LinkedHashMap de amizades e a Queue de recados, mantivessem sua ordem e integridade durante o processo de serialização e desserialização.

Gerenciamento de Relacionamentos

Projetei o sistema de amizades para exigir confirmação mútua, simulando o comportamento de redes sociais reais. Utilizei uma estrutura LinkedHashMap que não apenas armazena os relacionamentos confirmados, mas também mantém a ordem exata de criação exigida pelos testes de aceitação. Para os recados, optei por uma LinkedBlockingQueue que implementei para garantir o processamento em ordem FIFO (first-in, first-out), além de oferecer thread-safety para possíveis expansões futuras.

Desafios e Soluções

Um dos principais obstáculos que enfrentei foi garantir a consistência dos dados durante operações de persistência. Implementei verificações rigorosas para evitar corrupção de arquivos, incluindo a definição explícita de serialVersionUID em todas as classes serializáveis que criei. Outro desafio significativo que resolvi foi manter a ordem específica dos elementos nas listas de saída, solução que encontrei através da seleção cuidadosa de estruturas de dados que preservam a ordem de inserção.

Conclusão e Perspectivas Futuras

Esta primeira versão do Jackut que desenvolvi demonstrou a eficácia da arquitetura proposta, com todos os testes de aceitação sendo aprovados com sucesso. As escolhas de implementação que tomei, particularmente no que diz respeito às estruturas de dados e ao gerenciamento de persistência, provaram ser adequadas para os requisitos atuais. Como próximas etapas de desenvolvimento, pretendo expandir o sistema com a implementação de comunidades (US5) e aprimorar o mecanismo de persistência com suporte a operações atômicas, o que garantirá maior robustez ao sistema.