

Relatório - Prova 1

Aluno: Gabriel Cerqueira Araujo de Carvalho

Disciplina: ASI - Análise dos Sistemas de Informação

1 OBJETIVO

O objetivo deste relatório é apresentar a implementação individual da Prova I, que consiste no desenvolvimento de uma versão simplificada da Rede Social W. O sistema deve permitir que usuários, após realizarem login, possam publicar mensagens, receber curtidas e comentários de outros usuários e também do próprio autor da mensagem.

Ao longo do relatório são descritos os requisitos funcionais identificados, a modelagem de dados por meio de um Modelo Entidade Relacionamento (MER), a modelagem orientada a objetos com um diagrama de classes UML, a implementação em C Sharp utilizando .NET e a demonstração prática do funcionamento da aplicação por meio de sua execução em console.

2 METODOLOGIA E FERRAMENTAS UTILIZADAS

Para cumprir a proposta da Prova I, o trabalho foi organizado em etapas.

Primeiro foram extraídos e organizados os requisitos funcionais a partir do enunciado da prova, definindo claramente quais comportamentos o sistema deveria apresentar. Em seguida, estes requisitos foram traduzidos para um modelo de dados relacional, por meio de um MER, e para um modelo orientado a objetos, através de um diagrama de classes UML.

A partir dessas duas visões de modelagem, foi desenvolvida uma implementação mínima funcional em C Sharp, com foco na clareza de código e na demonstração do fluxo principal da Rede Social W. Por fim, a implementação foi executada em ambiente de console, registrando as saídas como evidências para o relatório.

As principais ferramentas utilizadas foram:

- Sistema operacional Windows.

- .NET SDK 8 para compilação e execução do código em C Sharp.
- Visual Studio Code como ambiente de desenvolvimento.
- Ferramenta gráfica on-line para criação do MER e do diagrama de classes UML, com exportação dos diagramas em formato de imagem.

3 REQUISITOS FUNCIONAIS

Com base no enunciado da Prova I, foram identificados e descritos os seguintes requisitos funcionais para a Rede Social W. A numeração é apenas organizacional e não indica prioridade.

- RF01: O sistema deve permitir o cadastro de usuários, armazenando identificador, nome, e-mail e senha (representada na implementação por um hash simplificado).
- RF02: O sistema deve permitir que um usuário autenticado realize login informando e-mail e senha.
- RF03: Após o login, o usuário deve poder publicar mensagens de texto, que são registradas com data e hora de criação.
- RF04: O sistema deve permitir que um usuário curta uma mensagem publicada por outro usuário.
- RF05: O sistema deve impedir que o mesmo usuário registre mais de uma curtida na mesma mensagem, garantindo que cada par usuário–mensagem tenha no máximo uma curtida.
- RF06: O sistema deve permitir que um usuário remova a própria curtida, caso ela já exista.

- RF07: O sistema deve permitir que usuários registrem comentários em mensagens existentes.
- RF08: O autor da mensagem também deve poder comentar a própria mensagem.
- RF09: O sistema deve disponibilizar a listagem dos comentários associados a uma determinada mensagem.
- RF10: O sistema deve apresentar uma listagem das mensagens, com indicação do autor e da quantidade de curtidas, simulando uma timeline simples.

Além desses requisitos, algumas regras de negócio implícitas foram consideradas:

- RN01: Cada comentário está sempre vinculado a exatamente uma mensagem e a exatamente um usuário.
- RN02: Cada curtida é identificada unicamente pelo par formado pelo usuário que curte e pela mensagem que foi curtida.
- RN03: Mensagens, comentários e curtidas possuem registro de data e hora de criação para possibilitar ordenações e auditoria básica.

4 MODELO ENTIDADE RELACIONAMENTO (MER)

O Modelo Entidade Relacionamento foi utilizado para representar de forma conceitual as informações manipuladas pela Rede Social W e os relacionamentos entre elas.

Foram definidas quatro entidades principais: Usuário, Mensagem, Comentário e Curtida.

- Entidade USUARIO: representa os participantes da rede social. Possui os atributos `id_usuario` (chave primária), `nome`, `email`, `senha_hash` e `criado_em`.
- Entidade MENSAGEM: representa as postagens realizadas pelos usuários após o login. Possui os atributos `id_mensagem` (chave primária), `conteudo`, `criado_em` e

`id_usuario`, que funciona como chave estrangeira referenciando o usuário autor da mensagem.

- Entidade COMENTARIO: representa os comentários realizados em uma mensagem. Possui os atributos `id_comentario` (chave primária), `texto`, `criado_em`, `id_mensagem` e `id_usuario`. Os atributos `id_mensagem` e `id_usuario` são chaves estrangeiras que indicam, respectivamente, a qual mensagem o comentário pertence e qual usuário o escreveu.
- Entidade CURTIDA: representa as curtidas que os usuários realizam nas mensagens. Possui os atributos `id_usuario`, `id_mensagem` e `criado_em`. No modelo conceitual, `id_usuario` e `id_mensagem` formam uma chave primária composta, garantindo que um usuário não consiga curtir a mesma mensagem mais de uma vez.

As cardinalidades foram estabelecidas da seguinte forma:

- Um usuário pode publicar várias mensagens, mas cada mensagem possui apenas um autor. Trata-se de um relacionamento um-para-muitos entre USUARIO e MENSAGEM.
- Um usuário pode escrever vários comentários, e cada comentário pertence a um único usuário. Relacionamento um-para-muitos entre USUARIO e COMENTARIO.
- Uma mensagem pode receber vários comentários, e cada comentário está vinculado a uma única mensagem. Relacionamento um-para-muitos entre MENSAGEM e COMENTARIO.
- Uma mensagem pode receber várias curtidas e um usuário pode curtir várias mensagens. Esse relacionamento muitos-para-muitos entre USUARIO e MENSAGEM é representado pela entidade associativa CURTIDA, com chave primária composta.

No relatório final, a imagem do MER é apresentada como Figura 1, ilustrando graficamente essas entidades, atributos e relacionamentos.

5 DIAGRAMA DE CLASSES UML

O diagrama de classes UML foi elaborado para modelar o sistema sob a ótica da orientação a objetos, aproximando a modelagem conceitual da implementação em C Sharp.

Foram definidas as classes de domínio Usuario, Mensagem e Comentario, além de uma classe de serviço chamada RedeSocialService, responsável por concentrar a lógica de aplicação.

A classe Usuario contém os atributos Id, Nome, Email e SenhaHash, além do método Autenticar, que verifica se o hash de senha informado corresponde ao valor armazenado.

A classe Mensagem possui os atributos Id, Conteudo, CriadoEm e uma referência ao autor (do tipo Usuario). Internamente, a classe mantém uma coleção de identificadores de usuários que curtiram a mensagem e uma lista de comentários. Entre os métodos, destacam-se AdicionarCurtida, RemoverCurtida, ContarCurtidas, AdicionarComentario e ListarComentarios.

A classe Comentario representa um comentário individual. Contém os atributos Id, Texto, CriadoEm e referências à Mensagem comentada e ao Usuario autor.

Por fim, a classe RedeSocialService funciona como uma camada de serviço da aplicação, mantendo listas de usuários e mensagens em memória. Ela expõe métodos para registrar usuários, realizar login, postar mensagens, curtir e descurtir mensagens, comentar e listar a timeline de mensagens ordenadas pela data de criação.

As associações indicam que:

- Um Usuario pode estar associado a várias Mensagens como autor.
- Um Usuario pode estar associado a vários Comentarios.
- Uma Mensagem pode estar associada a vários Comentarios.

- A classe RedeSocialService usa as classes de domínio para executar as operações da Rede Social W.

No relatório, o diagrama de classes é apresentado como Figura 2, evidenciando as classes, seus principais atributos, operações e relacionamentos.

6 IMPLEMENTAÇÃO EM C SHARP E .NET

A implementação foi realizada em C Sharp utilizando o SDK do .NET 8. O projeto foi organizado como uma aplicação de console, suficiente para demonstrar o comportamento do sistema sem interface gráfica complexa.

A estrutura do projeto na pasta src contém os seguintes arquivos principais:

- Usuario.cs: implementação da classe Usuario, com construtor que recebe identificador, nome, e-mail e hash de senha, e método Autenticar.
- Mensagem.cs: implementação da classe Mensagem, contendo atributos para identificador, autor, conteúdo e data de criação, além de coleções internas para armazenamento das curtidas (por meio de um conjunto de identificadores de usuário) e dos comentários. Implementa métodos para adicionar curtidas, remover curtidas, contar o número total de curtidas, adicionar comentários e listar comentários.
- Comentario.cs: implementação da classe Comentario, com atributos para identificador, mensagem associada, autor, texto e data de criação.
- IdGenerator.cs: classe estática simples responsável por gerar identificadores numéricos sequenciais para instâncias de mensagens e comentários, evitando a necessidade de entrada manual de IDs.
- RedeSocialService.cs: responsável por gerenciar as listas de usuários e de mensagens, além de prover operações de alto nível correspondentes aos requisitos funcionais, como registrar novos usuários, autenticar um usuário, postar mensagens, curtir e descurtir mensagens, comentar e obter a listagem de mensagens em ordem

cronológica inversa.

- Program.cs: ponto de entrada da aplicação, utilizado para orquestrar um cenário de teste que demonstra o funcionamento do sistema.

No método Main da classe Program é simulado o seguinte fluxo:

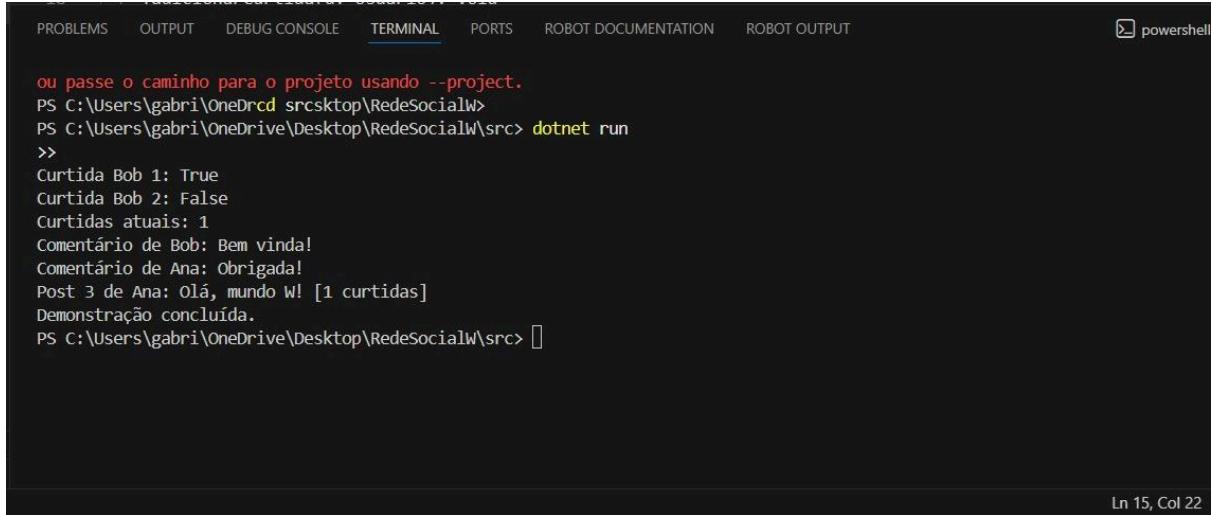
1. Criação de dois usuários, chamados Ana e Bob, por meio do método Registrar da classe RedeSocialService.
2. Realização de login com a usuária Ana utilizando o método Login.
3. Criação de uma mensagem por Ana, utilizando o método Postar.
4. Realização de login com o usuário Bob.
5. Registro de curtidas de Bob na mensagem de Ana, incluindo uma segunda tentativa de curtida para demonstrar o bloqueio de curtidas duplicadas.
6. Registro de comentários na mensagem, primeiro pelo usuário Bob e depois pela própria autora Ana.
7. Listagem dos comentários associados à mensagem.
8. Listagem da timeline das mensagens, exibindo o autor, o conteúdo e a quantidade de curtidas.

Esse fluxo traduz diretamente os requisitos funcionais da prova em uma execução concreta, permitindo verificar o correto funcionamento do sistema.

7 DEMONSTRAÇÃO DA IMPLEMENTAÇÃO

A Figura 3 apresenta a saída do programa em execução no console. Nela é possível observar:

- a primeira curtida de Bob sendo aceita (valor True);
- a segunda tentativa de curtida sendo rejeitada (valor False), evitando curtida duplicada;
- o total de 1 curtida registrado para a mensagem;
- os comentários de Bob e de Ana associados à mesma mensagem;
- a listagem final da mensagem com o autor e a quantidade de curtidas.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    ROBOT DOCUMENTATION    ROBOT OUTPUT
powershell

ou passe o caminho para o projeto usando --project.
PS C:\Users\gabri\OneDrct\srcsktop\RedeSocialW>
PS C:\Users\gabri\OneDrive\Desktop\RedeSocialW\src> dotnet run
>>
Curtida Bob 1: True
Curtida Bob 2: False
Curtidas atuais: 1
Comentário de Bob: Bem vinda!
Comentário de Ana: Obrigada!
Post 3 de Ana: Olá, mundo W! [1 curtidas]
Demonstração concluída.
PS C:\Users\gabri\OneDrive\Desktop\RedeSocialW\src> []

```

Ln 15, Col 22

8 CONSIDERAÇÕES FINAIS

A Implementação Individual da Prova I permitiu aplicar de forma integrada conceitos de análise de requisitos, modelagem de dados, modelagem orientada a objetos e programação em C Sharp com .NET.

A partir de um enunciado simples, foi possível identificar os principais requisitos funcionais de uma rede social básica, estruturá-los em um Modelo Entidade Relacionamento coerente e traduzi-los em um diagrama de classes UML alinhado ao paradigma de orientação a objetos.

Com base nessa modelagem, a implementação em C Sharp foi organizada em classes de domínio claras, com responsabilidades bem definidas, e em uma classe de serviço que concentra a lógica principal de cadastro, login, postagem, curtidas, comentários e listagem de mensagens. A execução do programa em console demonstrou na prática o cumprimento dos requisitos e das regras de negócio, especialmente o controle de curtidas sem duplicidade e o vínculo entre mensagens, comentários e autores.

Como trabalhos futuros, o sistema pode ser estendido para utilizar um banco de dados relacional real, implementando o modelo proposto no script SQL, além de receber uma interface gráfica ou web que torne a interação mais próxima de uma rede social completa. Também seria possível incluir autenticação mais robusta, paginação de mensagens, exclusão de mensagens e comentários e testes automatizados para maior confiabilidade.