# Documentação do Projeto - Sistema Distribuído de Rede Social

VISÃO GERAL:

Este projeto implementa uma rede social distribuída com funcionalidades de postagens, mensagens privadas, seguir usuários, e replicação entre servidores. Ele é composto por três componentes principais: APIs (em .NET e Java), servidores replicadores (em Python) e usuários simulados.

TECNOLOGIAS UTILIZADAS:

- .NET com C# (API de Postagens, Usuários e Mensagens)
- Java com Spring Boot (API de mensagens receptoras)
- Python com Flask (servidores replicadores e relógios)
- Relógio lógico de Lamport nos servidores
- Ambiente virtual (venv) e Postman para testes
- Scripts Python para simular múltiplos usuários

ESTRUTURA DO PROJETO:

- /API -> Backend em C#
- /API_Java -> Backend Java
- /replication-python -> Servidores replicadores em Python
  - servers/ -> Código dos servidores Flask
  - logic/ -> Lógica de relógio, eleição e sincronização
  - config/ -> Configuração dos servidores

FUNCIONALIDADES REPLICADAS:

- Postagens: replicadas com timestamp lógico
- Seguidores (follow): replicados como eventos de relacionamento

- Criação de usuário: replicada para manter consistência de estado

- Logs gerados em todos os servidores com as ações recebidas

SINCRONIZAÇÃO DE RELÓGIOS:

- Cada servidor tem um relógio físico com variação aleatória (±1s)

- O algoritmo de Berkeley (com eleição via bullying) é utilizado para sincronizar os relógios físicos

- A sincronização é periódica, coordenada pelo servidor com maior ID (coordenador)

TESTES REALIZADOS:

- 3 servidores Python rodando simultaneamente

- 5 usuários simulados com criação automática

- Cada usuário posta e segue outro, gerando replicações distribuídas

- Testes realizados via Postman e scripts Python

COMO TESTAR:

1. Execute os servidores Python:
   - python -m servers.run_server 1 (repita para 2 e 3)
2. Rode a API .NET com dotnet run e a API Java
3. Use o Postman para:
   - Criar usuários: POST /api/User
   - Criar postagens: POST /api/Post
   - Fazer follow: POST /NewFollow
   - Enviar mensagens: POST /sendMessage
4. Execute simulate_users.py para automatizar testes

OBSERVAÇÕES FINAIS:

- O projeto cumpre todos os requisitos obrigatórios

- A sincronização de relógios está integrada

- A arquitetura é modular e usa múltiplas linguagens como solicitado