Guia de Desenvolvimento Completo: Rede Social Full Stack

Este documento apresenta o plano de desenvolvimento passo a passo para construir uma rede social escalável, utilizando as seguintes tecnologias:

- **Backend:** Java 21+ com Spring Boot 3
- **Frontend:** JavaScript com React (usando Vite/Next.js)
- **Banco de Dados:** MySQL

1. Fase de Planejamento e Arquitetura

Antes de escrever qualquer código, é crucial definir a estrutura da aplicação e o modelo de dados.

1.1. Modelagem de Dados (MySQL)

O banco de dados MySQL armazenará a essência da rede social. As principais entidades e seus relacionamentos são:

Entidade	Campos Essenciais	Relacionamento
User (Usuário)	id, username (único), email (único), password, bio, profile_picture_url	
Post (Publicação)	id, content, timestamp, user_id	N:1 com User (o autor)
Comment (Comentário)	id, text, timestamp, user_id, post_id	N:1 com Post e User
Like (Curtida)	id, user_id, post_id	Tabela associativa (garante 1 like por usuário/post)
Follower (Seguidor)	id, follower_id, followed_id	N:N implícito entre User e User (a tabela define quem segue quem)

1.2. Arquitetura da Solução

A aplicação será dividida em três camadas principais, comunicando-se via API:

1. **Frontend (React):** Interface do usuário que envia requisições HTTP (GET, POST, PUT, DELETE).
2. **Backend (Spring Boot):** Recebe as requisições, executa a lógica de negócio (Services), interage com o Banco de Dados (Repositories) e retorna respostas JSON.
3. **Banco de Dados (MySQL):** Armazenamento persistente de todos os dados.

2. Backend: Java com Spring Boot

O backend deve ser a fonte de verdade para todas as regras de negócio e segurança.

2.1. Configuração Inicial

Utilize o Spring Initializr e inclua as seguintes dependências:

- **Spring Web:** Para criar controladores REST.
- **Spring Data JPA:** Para o mapeamento Objeto-Relacional (ORM) com Hibernate.
- **MySQL Driver:** O conector para o banco de dados.
- **Spring Security:** Essencial para autenticação (Login) e autorização (proteger rotas).
- **Lombok:** Para reduzir código boilerplate (opcional, mas recomendado).

2.2. Autenticação (Spring Security e JWT)

O fluxo de autenticação deve usar JSON Web Tokens (JWT) para ser *stateless* (sem estado):

1. **Endpoint de Login:** O usuário envia username e password.
2. **Geração do Token:** Se as credenciais estiverem corretas, o Spring gera um JWT contendo informações do usuário.
3. **Resposta:** O backend retorna o token para o frontend.
4. **Proteção:** Todas as rotas sensíveis (como /api/posts, /api/users) devem exigir que o token seja enviado no cabeçalho Authorization: Bearer <TOKEN>.

2.3. Estrutura de Camadas

- **Models/Entities (@Entity):** Classes que representam as tabelas do MySQL (ex: User.java, Post.java).
- **Repositories (JpaRepository):** Interfaces para operações CRUD (ex: UserRepository, PostRepository).
- **Services (@Service):** Contém a lógica de negócio (ex: "Calcular o Feed de Notícias", "Verificar se o email já existe").
- **Controllers (@RestController):** Mapeiam as URLs para as ações do Service e manipulam as requisições HTTP.

3. Banco de Dados: MySQL

3.1. Preparação

1. Instale o MySQL Server e um cliente (como MySQL Workbench).
2. Crie o banco de dados da sua aplicação.

3. Configure as credenciais de acesso no arquivo application.properties do Spring Boot:
   spring.datasource.url=jdbc:mysql://localhost:3306/nome_do_seu_db
   spring.datasource.username=seu_usuario
   spring.datasource.password=sua_senha
   spring.jpa.hibernate.ddl-auto=update # Cuidado! Usar 'none' ou 'validate' em produção.

3.2. Otimização

- **Índices:** Garanta que colunas usadas em filtros e buscas (username, email, user_id nas tabelas de posts) tenham índices para velocidade.
- **Chaves Estrangeiras:** Defina chaves estrangeiras para garantir a integridade referencial dos dados (ex: Um post não pode existir sem um autor válido).

4. Frontend: JavaScript com React

O React será responsável por renderizar a interface e gerenciar o estado da aplicação.

4.1. Configuração do Projeto

1. Crie um novo projeto React (ex: npx create-vite@latest meu-app --template react-ts).
2. Instale bibliotecas para requisições (ex: axios) e gerenciamento de estado (ex: Redux ou Zustand para projetos maiores).

4.2. Fluxo de Desenvolvimento da Interface

1. **Estrutura de Componentes:**
   - **Páginas:** LoginPage, RegisterPage, ProfilePage, FeedPage.
   - **Componentes Reutilizáveis:** PostCard, CommentBox, FollowButton, Header.
2. **Gerenciamento de Estado:** Use o Hook useState para o estado local e useContext (ou bibliotecas) para o estado global (como o usuário logado e o token JWT).
3. **Comunicação com a API:**
   - Crie uma função para o login que armazene o JWT recebido.
   - Em todas as requisições subsequentes, use o **Interceptor do Axios** ou configure manualmente o cabeçalho Authorization com o token.

4.3. Implementação do Feed

O componente FeedPage deve:

1. Fazer uma requisição GET para o endpoint /api/posts/feed (protegido por JWT).
2. O backend deve calcular quais posts o usuário logado deve ver (posts dos amigos que ele segue).
3. Iterar sobre a lista de posts recebida e renderizar um componente <PostCard> para cada um.

5. Próximos Passos e Desafios

Desafio	Solução Tecnológica
Notificações em Tempo Real	Adicionar WebSockets (Spring Boot tem suporte nativo a STOMP)
Upload de Mídia	Configurar um serviço de armazenamento de objetos (ex: AWS S3 ou Google Cloud Storage) e salvar apenas as URLs no MySQL.
Testes	Escrever testes unitários e de integração (JUnit para Spring e Jest/Testing Library para React).
Deploy	Empacotar o Spring Boot como um arquivo .jar e o React como arquivos estáticos, e fazer o deploy em serviços como AWS, Azure ou Google Cloud.

Este guia fornece uma base sólida para iniciar o desenvolvimento da sua rede social. Comece focando na autenticação e nas entidades principais (User e Post) para criar um **MVP (Produto Mínimo Viável)**.