1. Modelagem de Dados

A modelagem de dados é crucial para definir como as informações serão armazenadas, organizadas e relacionadas no banco de dados. Vamos começar criando o diagrama de Entidade-Relacionamento (ER) para o seu blog.

1.1. Identificação das Entidades Principais

Com base nas funcionalidades que você descreveu, as principais entidades no banco de dados podem ser:

- **Usuário:** Armazena informações sobre os usuários cadastrados.
- **Artigo:** Representa os posts do blog.
- **Categoria:** Agrupa os artigos em temas específicos.
- **Tag:** Associa palavras-chave aos artigos para facilitar a busca.
- **Comentário:** Armazena os comentários feitos nos artigos.
- **Favorito:** Registra os artigos que foram marcados como favoritos pelos usuários.

1.2. Definição dos Atributos das Entidades

Aqui estão alguns dos atributos que cada entidade pode ter:

- **Usuário:**
    - id_usuario: Chave primária
    - nome: Nome completo do usuário
    - email: Endereço de e-mail único
    - senha: Senha criptografada
    - data_criacao: Data de criação da conta
- **Artigo:**
    - id_artigo: Chave primária
    - titulo: Título do artigo
    - conteudo: Texto completo do artigo
    - data_publicacao: Data de publicação
    - id_usuario: Chave estrangeira (autor do artigo)
    - id_categoria: Chave estrangeira
- **Categoria:**
    - id_categoria: Chave primária

    - nome_categoria: Nome da categoria
- **Tag:**
    - id_tag: Chave primária
    - nome_tag: Nome da tag
- **Comentário:**
    - id_comentario: Chave primária
    - conteudo: Texto do comentário
    - data_comentario: Data do comentário
    - id_usuario: Chave estrangeira (autor do comentário)
    - id_artigo: Chave estrangeira
- **Favorito:**
    - id_favorito: Chave primária
    - id_usuario: Chave estrangeira
    - id_artigo: Chave estrangeira

1.3. Relacionamentos entre Entidades

Aqui estão alguns relacionamentos que podemos estabelecer entre essas entidades:

- **Usuário - Artigo:** Um usuário pode escrever vários artigos, mas cada artigo é escrito por um único usuário. (Relacionamento 1)
- **Artigo - Categoria:** Cada artigo pertence a uma única categoria, mas uma categoria pode conter vários artigos. (Relacionamento 1)
- **Artigo - Tag:** Um artigo pode ter várias tags, e uma tag pode ser associada a vários artigos. (Relacionamento N)
- **Artigo - Comentário:** Um artigo pode ter vários comentários, mas cada comentário está associado a um único artigo. (Relacionamento 1)
- **Usuário - Comentário:** Um usuário pode fazer vários comentários, mas cada comentário é feito por um único usuário. (Relacionamento 1)
- **Usuário - Favorito - Artigo:** Um usuário pode favoritar vários artigos, e cada artigo pode ser favoritado por vários usuários. (Relacionamento N)

2. Desenho da Arquitetura da API

Com a modelagem de dados em mente, vamos definir a arquitetura da API. A arquitetura RESTful será adotada, utilizando **Spring Boot** no backend para implementar os endpoints.

2.1. Definição dos Endpoints Principais

Aqui estão alguns endpoints que podemos definir com base nas funcionalidades descritas:

- **Autenticação de Usuários:**
  - POST /api/auth/signup: Cadastro de novos usuários
  - POST /api/auth/login: Login de usuários
  - POST /api/auth/forgot-password: Recuperação de senha
- **Gestão de Artigos:**
  - GET /api/articles: Listar todos os artigos
  - GET /api/articles/{id}: Obter detalhes de um artigo específico
  - POST /api/articles: Criar um novo artigo
  - PUT /api/articles/{id}: Atualizar um artigo existente
  - DELETE /api/articles/{id}: Deletar um artigo
- **Gestão de Categorias e Tags:**
  - GET /api/categories: Listar todas as categorias
  - POST /api/categories: Criar uma nova categoria
  - GET /api/tags: Listar todas as tags
  - POST /api/tags: Criar uma nova tag
- **Sistema de Comentários:**
  - GET /api/articles/{id}/comments: Listar todos os comentários de um artigo
  - POST /api/articles/{id}/comments: Adicionar um comentário a um artigo
  - DELETE /api/comments/{id}: Deletar um comentário
- **Favoritos:**
  - POST /api/users/{id}/favorites: Adicionar um artigo aos favoritos do usuário
  - GET /api/users/{id}/favorites: Listar todos os artigos favoritados pelo usuário
  - DELETE /api/users/{id}/favorites/{articleId}: Remover um artigo dos favoritos

2.2. Padrões e Boas Práticas

- **Autenticação e Autorização:**
  - Utilize **JWT (JSON Web Tokens)** para autenticação de usuários.

    - Implemente controle de acesso baseado em permissões para diferenciar usuários comuns de administradores.

- **Validação de Dados:**
    - Valide todas as entradas de usuários para evitar injeções SQL e outras vulnerabilidades.
    - Utilize anotações como @Valid e @NotNull em Spring Boot para garantir que os dados recebidos estão corretos.
- **Documentação da API:**
    - Utilize **Swagger** para documentar automaticamente os endpoints da API.
    - Isso facilita a integração com a equipe de frontend e outros desenvolvedores.

3. Prototipagem

Vamos iniciar o desenvolvimento de um protótipo da API para validar a arquitetura e as escolhas tecnológicas. Isso incluirá a implementação de alguns endpoints principais para garantir que o fluxo básico está funcionando conforme o esperado.

3.1. Implementação Inicial

1. **Setup do Projeto Spring Boot:**
    - Configuração do ambiente de desenvolvimento.
    - Criação de um projeto Spring Boot básico.
2. **Autenticação de Usuários:**
    - Implementação dos endpoints de signup e login.
    - Integração com o banco de dados para armazenamento e recuperação de usuários.
3. **CRUD de Artigos:**
    - Implementação dos endpoints para criação, leitura, atualização e deleção de artigos.
    - Teste básico do fluxo CRUD.

3.2. Testes Iniciais

- Realize testes unitários e de integração para os endpoints implementados.
- Verifique a correta persistência dos dados no banco de dados.
- Certifique-se de que a API está respondendo corretamente às requisições.

4. Mitigação de Riscos

Durante essa fase, você também começará a implementar as estratégias de mitigação de riscos que discutimos anteriormente, como:

- **Segurança:**
    - Implementar criptografia de senhas e uso de HTTPS.
    - Configurar políticas de CORS para proteger a API de acessos não autorizados.
- **Performance:**
    - Otimizar consultas ao banco de dados para melhorar a performance.
    - Implementar caching onde necessário.

Resumo da Fase de Elaboração

- **Modelagem de Dados Concluída:** ✔
- **Arquitetura da API Definida:** ✔
- **Prototipagem Inicial em Andamento:** Em progresso
- **Mitigação de Riscos:** Iniciada

Próximos Passos

Agora que temos um plano sólido para a **Fase de Elaboração**, o próximo passo é começar a implementar o protótipo e testar os componentes principais da API. Vamos monitorar o progresso e ajustar conforme necessário.

1. **Começar a Implementação do Protótipo:**
    - Crie o projeto Spring Boot e configure o ambiente de desenvolvimento.
    - Desenvolva os primeiros endpoints e teste a integração com o banco de dados.
2. **Validar o Modelo de Dados:**
    - Crie o diagrama ER final e certifique-se de que todas as relações e entidades estão corretamente definidas.
3. **Documentação:**
    - Documentar os endpoints com Swagger para facilitar o desenvolvimento futuro.