Documentação Técnica - MeeTime

🚀 Decisões Tomadas

A aplicação desenvolvida teve como objetivo a criação de uma API REST utilizando um dos frameworks disponíveis: **Spring Boot** ou **Play Framework**. O framework escolhido foi o **Spring Boot**, considerando que eu já havia iniciado meus estudos com essa tecnologia. Construir essa API do zero foi um grande desafio, enfrentado com muito empenho e dedicação, especialmente por ser minha primeira experiência completa com o framework.

Foi adotado um padrão de estruturação do projeto com o objetivo de facilitar o entendimento e a organização dos componentes, como controllers, services, DTOs, entre outros. Essa separação por responsabilidades contribui para um código mais limpo, modular e de fácil manutenção, além de seguir boas práticas amplamente utilizadas no desenvolvimento com Spring Boot.

📚 Bibliotecas

Abaixo, destaco algumas das bibliotecas utilizadas no projeto, juntamente com a motivação por trás de cada escolha:

- **Lombok:** Esta biblioteca foi adotada para simplificar a escrita de código, especialmente na geração automática de getters, setters, construtores, entre outros. Sua utilização reduziu significativamente a verbosidade do código, tornando o desenvolvimento mais ágil e limpo.
- **Bucket4j:** Nunca havia trabalhado anteriormente com mecanismos de *rate limiting*, e a escolha do **Bucket4j** se deu por sua robustez, flexibilidade e facilidade de implementação. Essa biblioteca atendeu perfeitamente à necessidade de controlar a quantidade de requisições feitas à API, contribuindo para a estabilidade e segurança do sistema.

🎯 Melhorias na Aplicação

Na aplicação desenvolvida, identifiquei algumas melhorias que podem ser implementadas nas próximas versões para torná-la mais robusta, segura e amigável para desenvolvedores:

- **Tratamento de erros nas respostas da API:** Atualmente, os erros nem sempre retornam mensagens claras ou estruturadas. A implementação de um mecanismo consistente de tratamento de exceções permitirá retornar respostas padronizadas, facilitando o entendimento por parte do consumidor da API.
- **Documentação técnica com Swagger (OpenAPI):** A inclusão de uma documentação interativa utilizando o Swagger contribuirá significativamente para a usabilidade e testes da API. Com ele, será possível visualizar os endpoints disponíveis, os parâmetros esperados e testar requisições de forma prática.
- **Implementação de autenticação via token:** No modelo atual, o `clientId` está exposto na geração da URL, o que representa um risco de segurança. A adoção de um mecanismo de autenticação com token (como JWT) permitirá proteger os endpoints da API de forma mais segura e escalável.