Documentação Técnica

Meetime HubSpot Integration

Visão Geral

A aplicação **Meetime HubSpot Integration** é uma API REST desenvolvida em **Java** com **Spring Boot**, projetada para integrar sistemas internos com a plataforma **HubSpot**. Ela utiliza **OAuth 2.0** para autenticação e fornece funcionalidades como criação de contatos e recebimento de webhooks.

Decisões Técnicas

1. Arquitetura

A aplicação segue o padrão **MVC (Model-View-Controller)**, garantindo separação de responsabilidades e melhor manutenibilidade. Também foi projetada seguindo os princípios **SOLID**, tornando o código mais modular e flexível.

2. Tecnologias e Bibliotecas

A escolha das bibliotecas foi baseada na confiabilidade e na adequação ao escopo do projeto:

- **Spring Boot**: Framework principal para desenvolvimento da API.
- **Spring Security + OAuth 2.0**: Implementação segura para autenticação com HubSpot.
- **Spring Data JPA + Hibernate**: Facilita o acesso ao banco de dados relacional.
- **HikariCP**: Pool de conexões para melhor desempenho.
- **MySQL**: Banco de dados escolhido por sua facilidade e usabilidade.
- **Docker + Docker Compose**: Facilita a implantação e configuração do ambiente.
- **Lombok**: Reduz boilerplate no código, simplificando a escrita de classes Java.
- **Log4j**: Biblioteca para logging eficiente e configurável.
- **Mockito**: Framework para testes unitários e mocks.

Motivação para Uso das Libs

A escolha das bibliotecas foi guiada pelos seguintes critérios:

- **Spring Boot** permite desenvolvimento rápido e estrutura bem definida.
- **OAuth 2.0** é essencial para comunicação segura com HubSpot.
- **HikariCP** melhora a eficiência do acesso ao banco.
- **Docker** garante facilidade na execução local e na implantação em diferentes ambientes.
- **Lombok** reduz código repetitivo, tornando o desenvolvimento mais rápido.
- **Log4j** oferece um sistema robusto de logs para monitoramento e debugging.
- **Mockito** facilita a criação de testes, garantindo qualidade e estabilidade no código.

Possíveis Melhorias Futuras

1. **Cacheamento**: Implementação de **Redis** para reduzir chamadas desnecessárias ao banco.
2. **Testes Automatizados**: Expansão da cobertura para teste E2E.
3. **Monitoramento**: Integração com **Prometheus** e **Grafana** para observabilidade.
4. **Rate Limiting**: Uso de **Resilience4j** para controle de requisições excessivas.
5. **Mensageria**: Uso de **Kafka** ou **RabbitMQ** para processar eventos de forma assíncrona.
6. **CI/CD**: Automação para implantação contínua.

Conclusão

O projeto foi estruturado para ser escalável e seguro, utilizando tecnologias modernas. As melhorias futuras visam aumentar a performance, segurança e confiabilidade da aplicação.