DOCUMENTAÇÃO TÉCNICA (Case Técnico: Integração com HubSpot)

Arquitetura

A aplicação segue uma arquitetura em camadas, com o objetivo de garantir uma boa organização do código e uma clara separação de responsabilidades. Essa abordagem facilita manutenções futuras, testes e extensibilidade. Arquiteturas como **Clean Architecture** ou **Hexagonal Architecture** não foram adotadas, pois são mais voltadas para aplicações com forte orientação ao domínio, o que não se aplica diretamente ao escopo do desafio.

Tecnologias e Bibliotecas Utilizadas

- **Bucket4j (`com.bucket4j:bucket4j-core:8.10.1`)**
  Utilizada para implementar o controle de rate limiting no endpoint de criação de contatos, atendendo às políticas do HubSpot (limite de 110 requisições por 10 segundos por conta).

- **Spring Data JPA (`org.springframework.boot:spring-boot-starter-data-jpa`)**
  Facilita operações de persistência e consultas ao banco de dados, neste caso utilizado para armazenar tokens OAuth.

- **H2 Database (`com.h2database:h2`)**
  Banco de dados em memória, utilizado para armazenar os tokens de forma volátil durante a execução da aplicação.

- **Spring Web (`org.springframework.boot:spring-boot-starter-web`)**

Responsável pela criação da API RESTful, fornecendo os endpoints necessários para autenticação, integração e recebimento de webhooks.

- **Lombok (`org.projectlombok:lombok`)**
  Reduz a quantidade de código boilerplate (getters, setters, construtores, etc.) com anotações simplificadas.

- **Spring Boot DevTools (`org.springframework.boot:spring-boot-devtools`)**
  Facilita o desenvolvimento ao permitir o hot reload da aplicação e outras melhorias para o ambiente local.

Decisões Técnicas

- Foi implementado um **interceptor HTTP** que injeta automaticamente o token de acesso nas requisições feitas ao HubSpot. Isso evita a necessidade de configurar manualmente o cabeçalho de autenticação em cada chamada à API.

- Foram definidas **interfaces (`OAuthService`, `ContactService`)** com o objetivo de facilitar futuras substituições do provedor OAuth ou mudanças no serviço de contato, seguindo os princípios de programação orientada a interfaces e injeção de dependência.

Melhorias Futuras

- **Persistência com Redis**
  Substituir o banco H2 por Redis, visando maior desempenho,

persistência entre reinícios da aplicação e escalabilidade.

- **Monitoramento com Grafana e ELK Stack**
  Implementar dashboards e logging estruturado para observabilidade da aplicação.

- **Processamento Assíncrono com Filas**
  Utilizar filas (por exemplo, RabbitMQ ou Kafka) para desacoplar o recebimento de webhooks e o processamento de eventos, garantindo resiliência e escalabilidade.

- **Criação de Objetos com Reflection e/ou anotações**
  Como os objetos do HubSpot possuem diversas propriedades e estruturas dinâmicas, o uso de reflection e/ou anotações podem facilitar a criação de objetos e mapeamento dinâmico dos dados.

- **Segurança com Spring Security**
  Delegar o controle de acesso às rotas à biblioteca Spring Security, melhorando a proteção dos recursos expostos.

- **Rate Limiting por Usuário**
  Adotar controle de rate limiting individualizado por usuário ou conta, garantindo melhor controle em ambientes multiusuário.