**Desafio Meetime – Integração HubSpot**

Este documento descreve a implementação da integração com a API do HubSpot utilizando o Java e Spring Boot. O objetivo é permitir a autenticação via OAuth 2.0 e a criação de contatos no HubSpot.

**Desafio**

O desafio proposto, foi realizar a integração de um backend com a api do HubSpot, disponibilizando:

* um endpoint para obter a url de autenticação com o HubSpot;
* um endpoint que receba um code fornecido pelo HubSpot e realiza a troca por um token de acesso;
* um endpoint que realize a criação de um contato no HubSpot;
* e um endpoint que receba eventos do tipo “contact.creation” enviado pelo webhook do HubSpot.

**Principais Desafios e Decisões**

O principal desafio foi a falta de familiaridade com o HubSpot, seu funcionamento e funcionalidades, o que dificultou o entendimento inicial do desenvolvimento. Além disso, o webhook exige uma URL pública, o que não seria possível em ambiente local. Para contornar isso, utilizei o Ngrok, que expõe a aplicação externamente, permitindo a comunicação com o HubSpot.

Outra decisão tomada, foi a utilização do Spring WebFlux na integração com o HubSpot para garantir uma comunicação mais eficiente e escalável com a API. Diferente do RestTemplate, que é bloqueante, o WebClient permite chamadas HTTP assíncronas, reduzindo o uso de threads e melhorando o desempenho da aplicação.

**Documentação Técnica**

**Tecnologias utilizadas**

* Linguagem Java na versão 21;
* Spring Boot na versão 3.4.4;
* Spring Boot WebFlux, para implementar um modelo assíncrono e não bloqueante;
* WebClient, substituto do RestTemplate para chamadas HTTP assíncronas;
* Ngrok, para expor a URL local da aplicação à internet;
* HubSpot API, serviço externo para integração e troca de dados.

**Implementação**

A integração com o HubSpot é feita utilizando WebClient, garantindo uma comunicação eficiente. O gerenciamento de autenticação segue o fluxo OAuth 2.0, permitindo a obtenção e renovação automática do token. Para expor a aplicação e receber webhooks, utilizei o Ngrok, possibilitando o envio de eventos pelo HubSpot. Toda a comunicação segue um fluxo reativo e não bloqueante, garantindo maior escalabilidade e desempenho. Além disso, foi implementado um tratamento centralizado de erros utilizando a anotação @RestControllerAdvice. Essa abordagem permite capturar e gerenciar exceções de forma consistente em toda a aplicação, retornando respostas padronizadas e adequadas para cada tipo de erro.

**Fluxo**

O primeiro endpoint disponibilizado, é o de autorização, ele retorna para o cliente a url de autenticação do HubSpot, que após se autenticar, redireciona para outro endpoint de callback, responsável por trocar o code fornecido pelo HubSpot em um token de acesso. Com esse token, é possível realizar a chamada a outro endpoint criado, para criar um contato no HubSpot, onde será necessário informar esse token, no Header “X-Authorization-HubSpot” da requisição, pra realizar a chamada a api do HubSpot. Por fim, após a criação do contato, há outro endpoint, que recebe o evento de “contact.creation”, notificando que um contato foi criado no HubSpot.

**Configuração e Execução**

Para configurar e executar a aplicação, pode seguir as seguintes opções:

* definir as variáveis CLIENT\_ID e CLIENT\_SECRET, com seus respectivos valores e executar o comando “./mvnw spring-boot:run”;
* executar a aplicação informando as variáveis no comando de execução “./mvnw spring-boot:run -Dspring-boot.run.jvmArguments="-DCLIENT\_ID=<value> -DCLIENT\_SECRET=<value>". *Substituindo <value> pelos seus respectivos valores.*

**Melhorias Futuras**

* Implementar a funcionalidade de renovação do token;
* Incluir outros endpoints de contato, como delete e update;