1 - Para garantir a privacidade e segurança dos dados dos usuários, a aplicação foi desenvolvida com uma atenção especial à **remoção de informações sensíveis** nos logs e mensagens de erro.

Durante o tratamento de exceções e geração de logs:

- **Tokens de acesso**, **códigos de autorização**, **e-mails**, **nomes** ou qualquer outro dado pessoal não são registrados nos logs.
- As mensagens de erro são **genéricas**, evitando a exposição de detalhes técnicos em ambientes públicos ou de produção.
- Em casos de falhas, apenas informações essenciais são mantidas para diagnóstico, sempre com cuidado para **não comprometer a segurança ou a confidencialidade dos dados** dos clientes.

Essa prática está totalmente alinhada com os princípios de **privacy by design** e segue as recomendações da documentação oficial da HubSpot, além de boas práticas de segurança modernas como as do OWASP.

2 - Foram implementados **logs estratégicos** utilizando o Logger do LoggerFactory.getLogger(...), com o objetivo de garantir **observabilidade** e facilitar o diagnóstico de erros e comportamentos inesperados durante a execução da aplicação.

Esses logs foram aplicados principalmente em:

- **Controllers**: para registrar o início das requisições, parâmetros recebidos e eventuais erros retornados ao cliente.
- **Services**: para acompanhar o fluxo interno da lógica de negócio, como chamadas para APIs externas (HubSpot) e o resultado dessas interações.
- **Utilitários (Helpers)**: como o HubSpotRequestHelper, para registrar tentativas de reenvio em casos de rate limiting e erros HTTP, com detalhes da falha.

O uso dos logs foi feito com **níveis apropriados**:

- info para ações comuns e fluxo normal;
- warn para situações incomuns ou que podem indicar problemas futuros;
- error para exceções e falhas críticas.

Essa abordagem ajuda na **manutenção, rastreamento e segurança**, tornando o sistema mais transparente e preparado para ambientes de produção.

3 - A aplicação foi cuidadosamente configurada para solicitar **apenas os escopos estritamente necessários** para funcionar corretamente com a API da HubSpot.

Isso significa que, ao gerar a URL de autorização, foram incluídos **apenas os escopos de leitura e escrita para contatos**, como:

- crm.objects.contacts.read
- crm.objects.contacts.write

Essa abordagem reduz o risco de exposição de dados desnecessários e está alinhada com o princípio de **menor privilégio**, recomendado pela documentação da HubSpot. Assim, garantimos que o aplicativo tem **acesso somente ao que precisa**, fortalecendo a segurança e mantendo o controle sobre as permissões concedidas.

4 – Considerações sobre a escolha da arquitetura ONION que eu particularmente tenho por escolha:

A arquitetura Onion foi escolhida como base do projeto por sua forte capacidade de **isolar as responsabilidades entre as camadas**, promovendo uma separação clara entre a lógica de negócio e os detalhes de implementação.

No centro da aplicação está o **domínio**, que representa as regras e comportamentos essenciais do sistema. Essa estrutura permite que a aplicação evolua com segurança, mantendo o foco na **regra de negócio** sem que dependências externas interfiram diretamente.

Esse isolamento está fortemente alinhado com os princípios do **Domain-Driven Design (DDD)**, onde o domínio é tratado como o coração do sistema e a complexidade é modelada com precisão.

Além disso, a arquitetura respeita o **princípio da Inversão de Dependência (DIP)** do SOLID, pois os módulos de alto nível (como os serviços de domínio) **dependem de abstrações**, e não das implementações. As dependências concretas (como chamadas HTTP, tokens, ou clientes externos) são definidas na camada de infraestrutura, mantendo o domínio limpo e independente.

Com essa estrutura, a aplicação se torna mais **testável, flexível e resiliente** a mudanças tecnológicas, garantindo um código mais sustentável a longo prazo.

5 – Foram gerados pequenos comentários para auxiliar no entendimento da lógica aplicada as soluções.

6 – O método criado chamado executarRequisicaoComRetry(), foi executado pensando na solicitação do desafio que consiste em que o **ENDPOINT deve respeitar as políticas de rate limita definidas pela API.** Foram adicionados comentários na Classe de uma maneira mais explicativa sobre a questão.