Objetivo do Desafio: O objetivo deste desafio é testar os conhecimentos do candidato em desenvolvimento, engenharia e arquitetura de sistemas. Por este motivo o candidato é livre para desenhar e modelar sistemas e API's da forma que achar mais adequado, desde que os itens do desafio sejam respeitados.

Prazo de entrega do desafio: 7 dias a contar do início do desafio.

Desafio

A Cielo está em constante busca por novas tecnologias em meios de pagamento, gerando inovação em pagamentos para nossos clientes. Uma destas novas tecnologias é o Tap On Phone.

O Tap On Phone é uma nova forma de efetuar pagamentos, onde os estabelecimentos podem utilizar um smartphone com NFC para efetuar a leitura e pagamentos com cartão por aproximação.

Para atender esse novo produto, precisamos criar os serviços de Backend que receberão as requisições de pagamento do Tap On Phone.

Estes serviços de Backend precisam que os seguintes pontos sejam atendidos:

[01] – Precisamos criar as API's para ciar um estabelecimento que será responsável por receber pagamentos através de Tap On Phone.

OBS.: Os bodys das requisições ficam abertos para que você defina os campos que ache necessário para um pagamento de cartão de crédito ou débito.

[02] – Precisamos criar as API's para ciar um device (smartphones) que efetuará pagamentos de Tap On Phone.

OBS.: Os bodys das requisições ficam abertos para que você defina os campos que ache para cadastrar um dispositivo.

OBS.: Um device deve estar associado a pelo menos um estabelecimento.

[03] – Precisamos criar as API's de pagamento, confirmação, cancelamento e desfazimento.

OBS.: Todo pagamento, confirmação, cancelamento e desfazimento está relacionado ao device e estabelecimento que está efetuando a operação.

1- Prover API para que seja efetuado pagamento através do Tap On Phone.
   OBS.: Todo pagamento efetuado é criado com o status pendente.

2- Prover API para que seja efetuado o cancelamento de um pagamento que foi efetuado através do Tap On Phone.
   OBS.: Somente um pagamento pendente ou confirmado pode ser cancelado, quando ele está cancelado seu estado muda para cancelado.

3- Prover API para que seja efetuada a confirmação de um pagamento que foi efetuado através do Tap On Phone.
OBS.: Somente um pagamento pendente pode ser confirmado, quando ele é confirmado seu estado muda para confirmado.

4- Prover API para que seja efetuado o desfazimento de um pagamento que foi efetuado através do Tap On Phone.
OBS.: Somente um pagamento pendente pode ser desfeito.

[04] - Precisamos criar a API de que exiba a lista dos pagamentos de um estabelecimento.

1- A API deve trazer a lista de pagamentos de um estabelecimento.

2- Cada pagamento deve trazer sua timeline, trazer suas mudanças de Status. Como por exemplo um pagamento pendente que muda para o status de confirmado etc.

3- A timeline de pagamentos deve receber as atualizações de status dos pagamentos de forma assíncrona, garantindo desta forma que consultas não impactem nos processos de pagamento.

Itens do desafio:

- Este desfio deve ser desenvolvido utilizando uma arquitetura de micro serviços, com pelo menos 2 micro serviços.

- Os artefatos gerados pelos micros serviços devem ser disponibilizados em imagens Docker, ou seja, as aplicações precisarão ser executadas nas imagens Docker.

OBS.: Caso deseje, o start das aplicações pode ser efetuado através de um docker-compose ou outra forma que achar necessário para subir todas as imagens necessárias para que as aplicações rodem.

- Deve ser utilizado pelo menos uma forma de comunicação assíncrona, seja ela através de RabbitMQ, Apache Kafka, Apache Pulsar, Web Hook, etc.

- Recursos como Banco de Dados, Broker de mensagem e demais tecnologias do desafio devem estar disponíveis em imagem Docker, para que possam ser executadas em qualquer computador.

- Deve ser utilizado pelo menos um banco de dados relacional. Utilizar também um banco não relacional é um diferencial.

- As aplicações devem ser desenvolvidas em Java com Spring Boot. Outras linguagens ou frameworks também podem ser utilizados, mas é obrigatório o uso de Java de Spring Boot em pelo menos um dos micros serviços.

- Deve ser utilizado o build tool Maven. Outros builds tools também podem ser utilizados, mas o uso de Maven é obrigatório para pelo menos uma das aplicações.

- Os códigos fontes do desafio podem ser disponibilizados como **mono repo** ou vários repositórios **separados**, no entanto, o desafio deve ser disponibilizado em repositório *GitHub* ou *Gitlab* com acesso público para que possamos efetuar o download do projeto.

- Devem ser executados testes unitários.

- Testes de integração serão um diferencial.

- Deve ser disponibilizado um readme.md com as instruções para compilar e executar a aplicação localmente. Todos os projetos do desafio devem ser compiláveis e executáveis, e todos os processos para rodar o projeto localmente (build, compilação de imagem Docker, Swagger etc.) devem estar detalhados no readme.md.

- No readme.md deve ser informado como efetuar os testes das API`s que foram criadas. As API`s precisam ter documentação seja através do Swagger ou outra forma de documentação.

- Disponibilizar um desenho da arquitetural do desafio. O desenho pode ser efetuado como o desejar, o importante é que fique claro todos as decisões de arquitetura que foram tomadas.