Desafio Software Engineer

Nesse desafio você construirá uma versão super simplificada de um Payment Service Provider (PSP) e talvez aprender um pouco mais sobre como funcionam pagamentos no Brasil.

Contexto

Em sua essência um PSP tem duas funções muito importantes:

1. Permitir que nossos clientes processem transações ("cash-in")
2. Efetuar os pagamentos dos recebíveis para os nossos clientes ("cash-out")
3. Nós temos duas entidades que representam essas informações:
    - transactions: que representam as informações da compra, dados do cartão, valor, etc
    - payables: que representam os recebíveis que pagaremos ao cliente

> Nota: quando um cliente passa uma transação de crédito, ele normalmente recebe o valor em média apenas 30 dias depois (o que chamamos de D+30), porque é assim que a cadeia financeira (bancos, bandeiras, adquirentes) funciona. Porém é possível receber esse valor antes dos 30 dias através de um mecanismo chamado "antecipação". Se tiver curiosidade, a gente tem artigo que explica sobre isso, mas isso não é necessário para realizar esse desafio:
>
> - https://pagar.me/blog/antecipacao-saiba-como-otimizar-o-fluxo-de-caixa-do-seu-negocio
> - https://ajuda.stone.com.br/antecipacao/antecipacao-e-seus-beneficios

Requisitos

Você deve criar um serviço com os seguintes requisitos:

1. O serviço deve processar transações, recebendo as seguintes informações:
    - Valor da transação
    - Descrição da transação. Ex: 'Smartband XYZ 3.0'
    - Método de pagamento (debit_card ou credit_card)
    - Número do cartão
    - Nome do portador do cartão
    - Data de validade do cartão
    - Código de verificação do cartão (CVV)
2. O serviço deve retornar uma lista das transações já criadas
3. Como o número do cartão é uma informação sensível, o serviço só pode armazenar e retornar os 4 últimos dígitos do cartão.
4. O serviço deve criar os recebíveis do cliente (payables), com as seguintes regras:
    - Se a transação for feita com um cartão de débito:
        - O payable deve ser criado com status = paid (indicando que o cliente já recebeu esse valor)
        - O payable deve ser criado com a data de pagamento (payment_date) = data da criação da transação (D+0).
    - Se a transação for feita com um cartão de crédito:
        - O payable deve ser criado com status = waiting_funds (indicando que o cliente vai receber esse dinheiro no futuro)
        - O payable deve ser criado com a data de pagamento (payment_date) = data da criação da transação + 30 dias (D+30).
5. No momento de criação dos payables também deve ser descontado a taxa de processamento (que chamamos de fee) do cliente. Ex: se a taxa for 5% e o cliente processar uma transação de R$100,00, ele só receberá R$95,00. Considere as seguintes taxas:
    - 3% para transações feitas com um cartão de débito
    - 5% para transações feitas com um cartão de crédito
6. O serviço deve prover um meio de consulta para que o cliente visualize seu saldo com as seguintes informações:
    - Saldo available (disponível): tudo que o cliente já recebeu (payables paid)
    - Saldo waiting_funds (a receber): tudo que o cliente tem a receber (payables waiting_funds)

Nota: neste desafio, você não precisa se preocupar com parcelamento.

Restrições

1. O serviço deve ser escrito em .NET Core SDK 7.x
2. O serviço deve armazenar informações em um banco de dados. Você pode escolher o banco que achar melhor. Aqui na Stone usamos amplamente PostgreSQL e SQL Server.
    1. Utilizar algum ORM pra mapeamento das entidades (Dapper, EF etc)
3. Docker & Docker Compose são indicados pra facilitar a execução
4. O projeto deve ter um README.md com todas as instruções sobre como executar e testar o projeto e os serviços disponibilizados.
5. O projeto deve conter testes automatizados.

Avaliação

1. O desafio deve ser enviado para a pessoa do RH que estiver em contato com você, no formato de .zip ou um link para um repositório do Github
2. Iremos te avaliar pela arquitetura do serviço, qualidade do código, entendimento das regras de negócio, capricho com o desafio e o quão preparado esse serviço estaria para ser rodado em produção
3. Depois que avaliarmos o desafio, te chamaremos para conversar com o time, apresentar o desafio e discutir sobre as decisões que você tomou
4. Achamos que 1 semana é um tempo ok para fazer o desafio, mas sabemos que nem todo mundo tem o mesmo nível de disponibilidade. Portanto, nos avise se precisar de mais tempo, ok?