Desafio Técnico para fazer em casa

Introdução

Autorizar uma transação com cartão de crédito é o pão com manteiga da vida no Caju. Este teste visa dar uma olhada nas diferentes estratégias que os candidatos podem ter ao implementar este recurso crucial.

Resolução e entrega [leia com atenção!]

O que será avaliado?

Além de avaliar a correção da sua solução, temos interesse em ver como você modela o domínio, organiza seu código e implementa seus testes.

Linguagem e bibliotecas

Na Caju, usamos **Scala e Kotlin** no nosso dia a dia (e demonstrar experiência em alguma delas é um grande diferencial). No entanto, você pode implementar sua solução utilizando sua linguagem favorita, dando preferência ao paradigma de programação funcional.

Bibliotecas são, obviamente, permitidas e você pode escolher o banco de dados de sua preferência.

Como entregar a solução?

Entregue a sua solução preferencialmente criando um repositório git (Github, Gitlab, etc). Você pode alternativamente entregar em um arquivo zipado ou uma pasta de algum serviço de armazenamento em nuvem (por exemplo, Google Drive)

É muito importante escrever um arquivo README com as instruções para execução do projeto.

Agora, vamos guiá-lo através de alguns conceitos básicos.

Transaction

Uma versão simplificada de um transaction payload de cartão de crédito é o seguinte:

```
{ "account": "123", "totalAmount": 100.00, "mcc": "5811", "merchant":
"PADARIA DO ZE SAO PAULO BR" }
```

Atributos

- **id** - Um identificador único para esta transação.
- **accountId** - Um identificador para a conta.
- **amount** - O valor a ser debitado de um saldo.
- **merchant** - O nome do estabelecimento.
- **mcc** - Um código numérico de 4 dígitos que classifica os estabelecimentos comerciais de acordo com o tipo de produto vendido ou serviço prestado.

  O `MCC` contém a classificação do estabelecimento. Baseado no seu valor, deve-se
  decidir qual o saldo será utilizado (na totalidade do valor da transação). Por simplicidade, vamos usar a seguinte regra:

  - Se o `mcc` for `"5411"` ou `"5412"`, deve-se utilizar o saldo de `FOOD`.
    - Se o `mcc` for `"5811"` ou `"5812"`, deve-se utilizar o saldo de `MEAL`.
    - Para quaisquer outros valores do `mcc`, deve-se utilizar o saldo de `CASH`.

Desafios (o que você deve fazer)

Cada um dos desafios a seguir são etapas na criação de um **autorizador completo**. Seu autorizador deve ser um servidor HTTP que processe a transaction payload JSON usando as regras a seguir.

As possíveis respostas são:

- `{ "code": "51" }` se a transação é **rejeitada**, porque não tem saldo suficiente
- `{ "code": "00" }` se a transação é **aprovada**
- `{ "code": "07" }` se acontecer qualquer outro problema que impeça a transação de ser processada

O HTTP Status Code é sempre `200`

L1. Autorizador simples

O **autorizador simples** deve funcionar da seguinte forma:
- Recebe a transação
- Usa **apenas** a MCC para mapear a transação para uma categoria de benefícios
- Aprova ou rejeita a transação
- Caso a transação seja aprovada, o saldo da categoria mapeada deverá ser diminuído em **totalAmount**.

L2. Autorizador com fallback

Para despesas não relacionadas a benefícios, criamos outra categoria, chamada **CASH**.
O autorizador com fallback deve funcionar como o autorizador simples, com a seguinte diferença:
- Se a MCC não puder ser mapeado para uma categoria de benefícios ou se o saldo da categoria fornecida não for suficiente para pagar a transação inteira, verifica o saldo de **CASH** e, se for suficiente, debita esse saldo.

L3.Dependente do comerciante

As vezes, os MCCs estão incorretos e uma transação deve ser processada levando em consideração também os dados do comerciante. Crie um mecanismo para substituir MCCs com base no nome do comerciante. O nome do comerciante sempre deve ter maior precedência sobre as MCCs.

Exemplos:

- `UBER TRIP SAO PAULO BR`
- `UBER EATS SAO PAULO BR`

- `PAG*JoseDaSilva RIO DE JANEI BR`
- `PICPAY*BILHETEUNICO GOIANIA BR`

L4. Questão aberta

A seguir está uma questão aberta sobre um recurso importante de um autorizador completo (que você não precisa implementar, apenas discuta da maneira que achar adequada, como texto, diagramas, etc.).

- Transações simultâneas: dado que o mesmo cartão de crédito pode ser utilizado em diferentes serviços online, existe uma pequena mas existente probabilidade de ocorrerem duas transações ao mesmo tempo. O que você faria para garantir que apenas uma transação por conta fosse processada em um determinado momento? Esteja ciente do fato de que todas as solicitações de transação são síncronas e devem ser processadas rapidamente (menos de 100 ms), ou a transação atingirá o timeout.

Para este teste, tente ao máximo implementar um sistema de autorização de transações considerando todos os desafios apresentados (L1 a L4) e conceitos básicos.