Teste de Desenvolvimento

Você foi contratado para implementar um gateway de pagamentos. Um Gateway de pagamentos é um sistema capaz de enviar requisições de compras, por exemplo, cartão de crédito, para empresas processadoras de pagamentos, as chamadas adquirentes, e oferece ao lojista um único ponto de integração para várias adquirentes, a vantagem de se usar um gateway é que com apenas um contrato de integração o lojista poderá se integrar com várias empresas de pagamentos e com sistemas antifraudes.

Sua missão é desenvolver um gateway especializado em e-commerce que poderá processar pagamentos de vários lojistas, cada lojista poderá ter contrato com mais de um adquirente e ainda poderá ter ou não contrato com sistemas antifraudes.

Requisitos:

1) Sua API deverá oferecer um contrato de utilização único que será disponibilizado para os lojistas que irão utilizar sua plataforma de pagamentos;

2) Você deverá se integrar com os seguintes adquirentes:

- Stone (http://gateway.stone.com.br/docs/bem-vindo)

- Cielo (https://developercielo.github.io/manual/cielo-ecommerce)

OBS: Você não precisa integrar de verdade com a API deles, mas sim poderá "mockar" os serviços de pagamentos respeitando o contrato dessas adquirentes. Você não precisa implementar o mock de todos os serviços dessas adquirentes, mas sim apenas o serviço de criação de transações;

3) Você deverá implementar um mock do serviço de antifraude da Clear Sale(https://www2.clear.sale/developers/api#appendix-requestsend), lembrando que apenas o serviço de envio de requisição do link;

4) Lojas poderão ou não ter contratos com antifraude;

5) O fluxo do antifraude deverá acontecer antes de enviar a transação para as adquirentes

- Caso o status retornado do antifraude (https://www2.clear.sale/developers/api#appendix-status-list) for diferente de "APA" o fluxo de pagamento deverá ser interrompido e sua API deverá retornar o contrato de erro para o usuário;

6) Um loja poderá estar configurada para um adquirente apenas, ou baseada na regra definida na seção 6.1, podendo ou não, ter contrato com antifraude:

6.1) Um lojista poderá escolher em qual adquirente ele quer passar a transação pela bandeira do cartão, por exemplo:

Se for Visa ele passa na adquirente A

Se for Master ele passa na adquirente B

6.2) A configuração da loja ficará armazenada no banco de dados da API

7) Todos os seus mocks deverão contemplar os cenários de transações com sucesso e transações com erro;

8) Sua API dará suporte para apenas transações de cartão de crédito;

9) Você deverá desenvolver a API usando a tecnologia .NET Core 2;

10) Deverá usar um banco de dados para guardar os registros de transações e configurações dos lojistas, pode usar o banco que você quiser (opensource);

11) Sua API deverá ter um serviço para registrar uma transação de compra e consultar transações por loja;

12) Sua API deverá seguir a especificação RestFul (https://github.com/Microsoft/api-guidelines);

13) Sua aplicação deverá ser disponibilizada para avaliação na plataforma GitHub;

14) Seu projeto deverá ter uma documentação no formato README no repositório com todas as instruções para buildar, executar e testar sua aplicação;

15) Sua API de ter testes Unitários.

Boa Sorte!