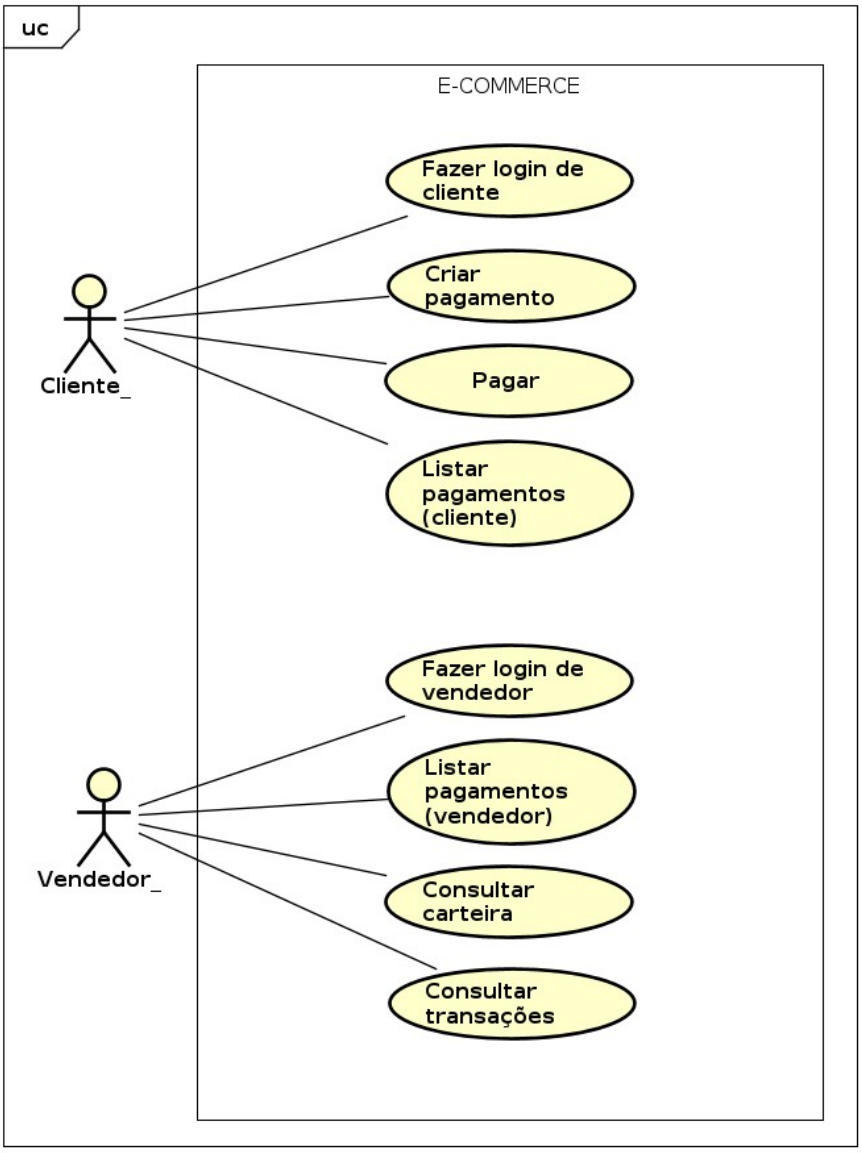
TESTE PARA CANDIDATOS DE BACKEND – MJV

Aqui na MJV, o core do nosso backend é composto por APIs REST construídas em typescript usando o framework Nestjs, integrações diversas entre várias outras APIs REST, bancos de dados diversos, RabbitMQ e microserviços variados, todos em Nodejs. Nossa infra funciona em um ambiente baseado em containers, portanto, cada sistema precisa estar encapsulado em uma imagem docker.

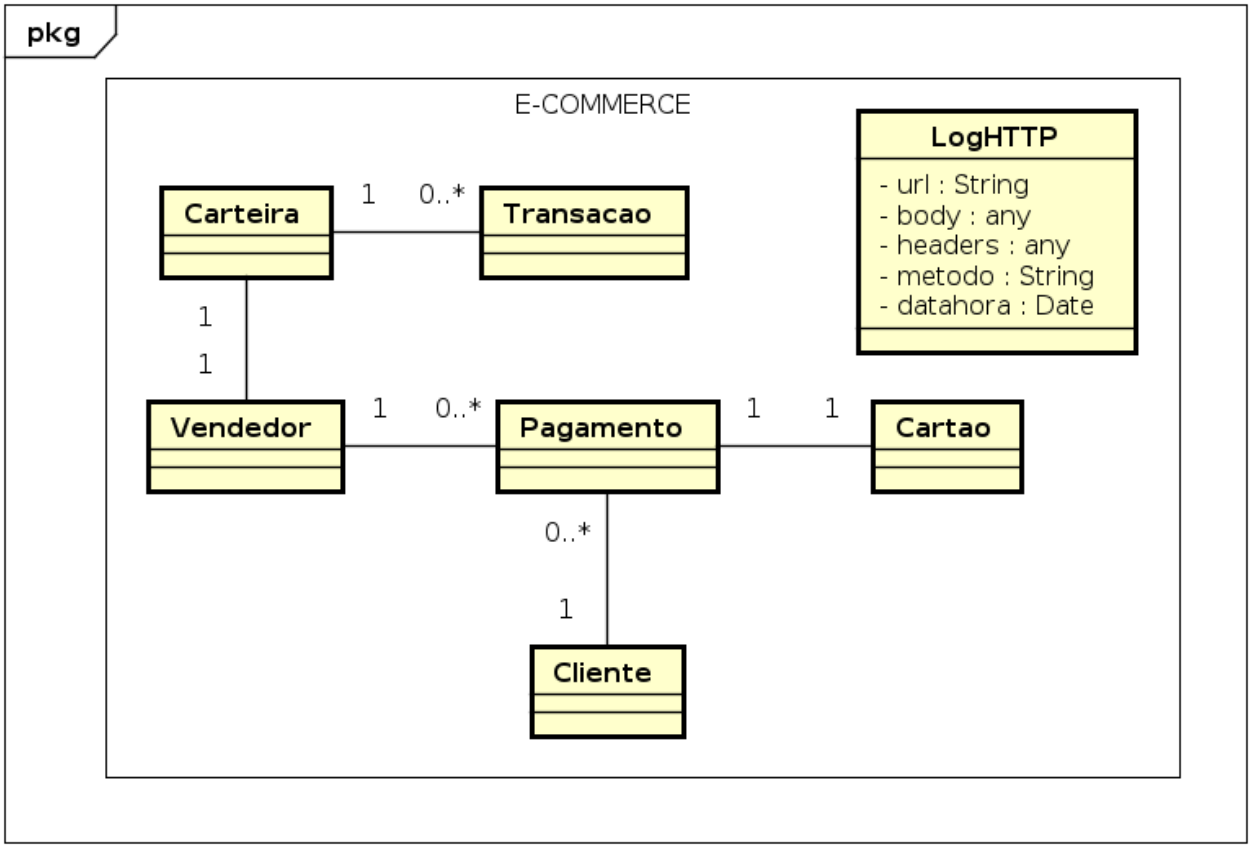
Para validar as habilidades dos candidatos para trabalhar com nosso sistema, desenvolvemos esse teste simples, onde o candidato demonstrará sua aptidão com esses itens principais descritos acima desenvolvendo o backend de um mini sistema de e-commerce, que utiliza essas mesmas tecnologias.

DESCRIÇÃO DO TESTE:

Você deverá desenvolver o backend de um mini sistema de e-commerce que atenda aos seguintes casos de uso:



Este é o diagrama de classes esperado:



REGRAS:

1 - Os pagamentos serão feitos apenas com cartão de DÉBITO, utilizando o ambiente sandbox da Cielo. Essa regra é necessária para avaliarmos como o candidato lida com uma comunicação bi-direcional entre APIs REST, uma vez que a cielo exige uma autenticação extra nessa modalidade e envia uma requisição POST de volta para uma url que será fornecida como endpoint de notificações de pagamento da cielo.

2 – Cada pagamento capturado (finalizado) deverá ter seu valor adicionado à carteira do vendedor.

3 – Cada valor que for adicionado à carteira do vendedor, deverá gerar um registro de transação, contendo o valor adicionado e o pagamento de origem.

4 – TODAS AS REQUISIÇÕES feitas para a API e ENVIADAS pela API, deverão ser LOGADAS. Para isso, crie um modulo na api para fazer envio de dados para uma fila no RabbitMQ. Os dados que deverão ser enviados para essa fila a cada requisição são headers, body, url, metodo (get, post, etc..) e um timestamp do evento. Faça um DTO com esses campos e um método para envio à fila. Esse modulo deverá ser injetado em todos os modulos que receberem ou enviarem requisições e deverá ser invocado nesses momentos.

5 - Crie um microserviço que ouvirá essa fila no RabbitMQ e registrará esses logs no banco de dados, ajuste a velocidade de consumo dessa fila para 1 por vez e envie um ack somente após a inserção do log no banco. Muito cuidado no tratamento de erros no consumo dessa fila, pois uma mensagem unacked aqui irá travar o seu microserviço e comprometer a aplicação.

6 – Utilize o TypeORM para gerenciar as operações com o banco de dados.

7 – Documente seus endpoints com Swagger e deixe a interface no endpoint /docs

8 – Cuidado com a autenticação. Um usuário não pode acessar os endpoints do outro tipo de usuário. Utilize autenticação Bearer token JWT.

CONSIDERAÇÕES:

- Você deverá entregar esse teste em um repositório publico no Github ou Gitlab.
- Você é livre para definir o conteúdo das classes e a estrutura dos seus endpoints/controllers.
- Pode escolher e modelar o(s) banco(s) de dados de sua preferência, relacionais ou não.
- Você pode utilizar o site webhookinbox para testar a maneira que a cielo enviará o post para sua aplicação sem precisar subir ela para um servidor na internet:
http://webhookinbox.com/view/nJZWSGOZ/
- Você pode criar cartões e cpfs falsos para fazer seus testes no site do 4Devs:
https://www.4devs.com.br/gerador_de_numero_cartao_credito

BONUS:

Esses itens são considerados bonus (não são exigidos, mas são MUITO valorizados)

- Crie um Dockerfile para a api e outro para o microserviço dos logs.
- Testes unitários
- CI de sua preferência fazendo build e push da imagem docker da api e do microserviço
- S.O.L.I.D
- Clean Code
- Deploy da aplicação em um serviço de hospedagem de sua preferência (ex: heroku, aws, etc)
- Interceptor: Criar um custom interceptor para logar as requisições feitas para a api e envia-las para a fila do RabbitMQ é uma prática muito valorizada
- Utilizar um container do Fusionauth para gerenciar autenticação. Se conseguir fazer isso, já estará entre os primeiros favoritos.

DOCUMENTAÇÕES E DICAS:

CIELO:

Documentação da API: https://developercielo.github.io/manual/cielo-ecommerce#sobre-o-sandbox
dica: você não precisa se cadastrar para utilizar o ambiente de testes da cielo (sandbox), basta criar suas chaves de teste grátis aqui: https://cadastrosandbox.cieloecommerce.cielo.com.br

NESTJS:
Documentação: https://docs.nestjs.com/first-steps

RABBITMQ:
Tutoriais: https://www.rabbitmq.com/getstarted.html
documentação da biblioteca amqplib, necessária na aplicação:
http://www.squaremobius.net/amqp.node/