DEVinHouse

Módulo 2 - Projeto Avaliativo 1

SUMÁRIO

1 INTRODUÇÃO

Você é o principal desenvolvedor do back-end de um aplicativo para solicitar transporte de carros (em modelo semelhante ao de táxis, como o daquela empresa que começa com U, sabe?). Este aplicativo se chama **labCAR**, e sua missão é promover o melhor meio de transporte possível pelo menor preço, conectando pessoas que possuem carros com pessoas que precisam de um meio de transporte.

Você está encarregado de criar o back-end da aplicação. Vamos nessa?!

2 REQUISITOS DA APLICAÇÃO

A aplicação deverá ser feita de forma **individual** e deve contemplar os seguintes requisitos:

- Ser desenvolvido utilizando a linguagem **Node com NestJS**;
- Seguir o Roteiro da Aplicação.

3 ROTEIRO DA APLICAÇÃO

O aplicativo **labCAR** precisa de um back-end de administração, ou seja, um programa que através dele seja possível controlar quem são os motoristas, os passageiros e as corridas realizadas.

3.1 ENDPOINTS DO MOTORISTA

Desenvolva os seguintes endpoints para o contexto de motoristas:

1. **Listar motoristas**
    a. Esta listagem deverá ser paginada, tendo opção de enviar um atributo chamado page representando a página a ser retornada, e outro atributo chamado size que define a quantidade de itens por página.
    b. Também deverá ser possível passar um *queryParam* para buscar os motoristas que começam por determinado texto.
        i. Ex.: Caso eu tenha cadastrado quatro motoristas com os nomes: Marcelo, Maria, Pedro, Mota, ao pesquisar por "mar", deverão ser retornados os motoristas Marcelo e Maria.
2. **Detalhes motorista**
    a. Implementar um endpoint que retorne os detalhes de um motorista a partir do CPF.
3. **Criar um motorista**
    a. Para criar um motorista ele deverá receber os seguintes parâmetros:
        i. **Nome:** Não pode ser nulo, e no máximo 50 caracteres
        ii. **Data de Nascimento:** No mínimo 18 anos
        iii. **CPF:** Deverá ser um CPF válido.
        iv. **Placa:** Placa do carro
        v. **Modelo:** Modelo do veículo
4. **Atualizar os dados cadastrais de um motorista**
    a. A Atualização dos dados deverá ser sempre total para endpoint
5. **Bloquear motorista**
    a. Neste endpoint será possível bloquear e desbloquear um motorista
6. **Exclusão de motorista**
    a. Caso não tenha nenhum registro de viagens para o motorista, ele poderá ser excluído do sistema.

3.2 ENDPOINTS DO PASSAGEIRO

O próximo conjunto de endpoints é bem parecido com os endpoints de motorista, mas o contexto aqui será o de passageiros:

1. **Listar passageiros**
    a. Esta listagem deverá ser paginada, tendo opção de enviar um atributo chamado page representando a página a ser retornada e outro atributo chamado size que define a quantidade de itens por página.
    b. Também deverá ser possível passar um *queryParam* para buscar os passageiro que começam por determinado texto.
        i. Ex.: Caso eu tenha cadastrado quatro passageiros com os nomes: Marcelo, Maria, Pedro, Mota, ao pesquisar por "mar", deverão ser retornados os motoristas Marcelo e Maria.
2. **Detalhes passageiro**
    a. Implementar um endpoint que retorne os detalhes de um passageiro a partir do CPF.
3. **Criar um passageiro**
    a. Para criar um passageiro ele deverá receber os seguintes parâmetros:
        i. **Nome:** Não pode ser nulo, e no máximo 50 caracteres
        ii. **Data de Nascimento:** No mínimo 18 anos
        iii. **CPF:** Deverá ser um CPF válido.
        iv. **Endereço:** Endereço residencial do passageiro
4. **Atualizar os dados cadastrais de um passageiro**
    a. A Atualização dos dados deverá ser sempre total para endpoint
5. **Exclusão de um passageiro**
    a. Caso não tenha nenhum registro de viagens para o motorista, ele poderá ser excluído do sistema.

3.3 ENDPOINTS DE VIAGENS

O último conjunto de endpoints será o de solicitação de viagens:

1. **Solicitar uma viagem**
    a. Para criar uma solicitação deverá ser enviada os seguintes dados:
        i. **Id do passageiro:** Id do passageiro que está solicitando a viagem, deve ser um id válido
        ii. **Origem:** Endereço de origem
        iii. **Destino:** Endereço de destino

b. Para controlar as solicitações de viagem devemos ter um enum com os seguintes status: *CREATED*, *ACCEPTED*, *REFUSED*, ao criar uma solicitação devemos gravar esta solicitação como *CREATED*.

2. **Viagens próximas do motorista**
    a. Neste endpoint devemos receber o endereço atual onde o motorista se encontra e retornar as solicitações de viagens próximas ao endereço de origem da solicitação de viagem.
    b. Você pode retornar um dado mockado para estas requisições.

3.4 DESAFIO EXTRA

No endpoint de viagens próximas estamos retornando dados *mockados*. Deixarei como desafio extra a integração com a API do Google para obter de forma real as viagens próximas ao motorista utilizado a API Directions.

Segue a documentação: https://developers.google.com/maps/documentation/directions/start

É possível testar esta API sem custo financeiro por 90 dias, ou até a conta acumular U$300 em cobrança, o que ocorrer primeiro, também é possível cancelar a qualquer momento. No link acima tem toda a descrição de como gerar um API_KEY para realizar as requisições.

4 CRITÉRIOS DE AVALIAÇÃO

A tabela abaixo apresenta os critérios que serão avaliados durante a correção do projeto. O mesmo possui variação de nota de 0 (zero) a 10 (dez) como nota mínima e máxima, e possui peso de **40% sobre a avaliação do módulo**.
Serão **desconsiderados e atribuída a nota 0 (zero)** os projetos que apresentarem plágio de soluções encontradas na internet ou de outros colegas. Lembre-se: Você está livre para utilizar outras soluções como base, mas **não é permitida** a cópia.

Nº	Critério de Avaliação	0	0,25 a 0,50	0,50 a 0,75
1	Listar motoristas	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST

2	Detalhes motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
3	Criar um motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
4	Atualizar os dados cadastrais de um motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
5	Bloquear motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
6	Exclusão de motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
7	Listar passageiros	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
8	Detalhes passageiro	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
9	Criar um passageiro	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST

10	Atualizar os dados cadastrais de um passageiro	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
11	Exclusão de um passageiro	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
12	Viagens próximas do motorista	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
Nº	**Critério de Avaliação**	**0**	**0,25 a 0,75**	**1,00**
13	Solicitar uma viagem	O aluno não implementou o endpoint	Implementou a rota, mas não cumpriu todos os requisitos ou não seguiu as boas práticas de APIs REST	Implementou corretamente a rota e seguiu as boas práticas na construção de APIs REST
Nº	**Critério de Extra**			
CE	A pontuação acima totaliza 10 pontos no total, caso cumpra o desafio extra ele irá lhe render 1 ponto, se já tiver obtido a nota máxima, sua pontuação permanecerá 10, mas caso sua nota seja abaixo de 10, este ponto extra pode contribuir com sua pontuação total.			

5 ENTREGA

O código desenvolvido deverá ser submetido no **GitHub**, e o link deverá ser disponibilizado na tarefa **Módulo 2 - Projeto Avaliativo 1**, presente na semana 6 do AVA até o dia **07/11/2022** às **23h55**.

O repositório deverá ser privado, com as seguintes pessoas adicionadas:

- Nome do Docente - **danilosales**
- Operação DEVinHouse - **devinhouse-operacao**

Importante:

1. Não serão aceitos projetos submetidos **após a data limite da atividade**, e, ou **alterados** depois de entregues.
2. Será considerado como data final de entrega a **última atualização** no repositório do projeto no GitHub. Lembre-se de não modificar o código até receber sua nota.
3. Não esqueça de **submeter o link no AVA**. Não serão aceitos projetos em que os links não tenham sido submetidos.

6 PLANO DE PROJETO

Ao construir a aplicação proposta, o aluno estará colocando em prática os aprendizados em:

- **REST**: Conceitos de API Rest
- **NestJS**: Conceitos de construção de APIs, validações utilizando o framework NestJS