Desafio Final - Part. 2

Parabéns gostamos muito da primeira etapa do sistema iremos fechar o contrato com o restante do sistema. Logo vamos para a segunda de três etapas.

Obs: Ao final deste documento existe links que podem auxiliar na construção deste desafio.

Abaixo deixo as especificações

Débito Técnico: (ou dívida técnica) é um conceito no desenvolvimento de software que representa o custo implícito de uma implementação/solução pensada somente no agora, em vez de usar uma abordagem com melhor qualidade porém que levaria mais tempo.

Ficou alguns Débito Técnico no desafio 1 que precisa ser resolvido no desafio 2:

- **CPF** ser único na base
- **Email** ser único na base
- Ajuste geral de retorno de erros
- Retornar o ID da descrição

Ajuste geral dos retornos da api

1. **Primeiramente gostaríamos de realizar um ajuste no endpoint /car. Atualmente quando estamos recebendo os retornos de erros eles ocorrem da seguinte forma:**
    - **Request - POST** `http://localhost:3000/api/v1/car`

```
{
    "_original": {
        "acessorios": [
            {
                "descricao": "Ar-condicionado"
                "id": "asuihdasuid"
            },
            {
                "descricao": "Dir. Hidráulica"
            },
            {
                "descricao": "Cabine Dupla"
            },
            {
                "descricao": "Tração 4x4"
            },
            {
                "descricao": "4 portas"
            },
            {
                "descricao": "Diesel"
            },
            {
                "descricao": "Air bag"
            },
            {
                "descricao": "ABS"
            }
        ],
        "quantidadePassageiros": 5
    },
    "details": [
        {
            "message": "\"modelo\" is required",
            "path": [
                "modelo"
            ],
            "type": "any.required",
            "context": {
                "label": "modelo",
                "key": "modelo"
            }
        },
        {
            "message": "\"cor\" is required",
            "path": [
                "cor"
            ],
            "type": "any.required",
            "context": {
                "label": "cor",
                "key": "cor"
            }
        },
```

```
        {
            "message": "\"ano\" is required",
            "path": [
                "ano"
            ],
            "type": "any.required",
            "context": {
                "label": "ano",
                "key": "ano"
            }
        }
    ]
}
```

Porém desejamos que o retorno dos erros de cadastro dos carros, sejam retornados da seguinte maneira:

```
[
    {
        "description": "modelo",
        "name": "\"modelo\" is required"
    },
    {
        "description": "cor",
        "name": "\"cor\" is required"
    },
    {
        "description": "ano",
        "name": "\"ano\" is required"
    }
]
```

Atenção ao requisito:

- **Este mesmo retorno de erro deverá ser abordado em TODAS as requests realizadas no sistema como: Criação de carro, criação de usuário, atualização(update) de carro, atualização(update) de usuário e assim continuamente. Abaixo estão alguns exemplos, atentem-se para o formato do retorno de erro e das mensagens.**

```
{
    "description": "cpf",
    "name": "\"cpf\" length must be less than or equal to 14 characters long"
},
```

```
{
    "description": "Conflict",
```

```
    "name": "CPF 724.533.815-98 already in use"
}
```

```
{
    "description": "Bad Request",
    "name": "Invalid CPF 131.131.131-13"
}
```

2. **Fazer um update especifico de um acessório: Este update pode ser a remoção ou adição de um acessório.**
    - **Request - PATCH** `http://localhost:3000/api/v1/car/:id/acessorios/:id`

```
{
  "descricao": "Ar-condicionado"
}
```

Atenção:

- O campo descrição é **required** assim como na primeira etapa desenvolvida.
- **Dica: Caso a descrição já exista considere como uma remoção, caso contrario adicione.**

Descrição dos endpoints: Locadora

1. **Cadastramento de um locadora**
    - **Request - POST** `http://localhost:3000/api/v1/rental`

```
{
    "nome": "Localiza Rent a Car",
    "cnpj": "16.670.085/0001-55",
    "atividades": "Aluguel de Carros E Gestão de Frotas",
    "endereco": [
      {
        "cep": "96200-200",
        "number":"1234",
        "isFilial": false
      }
```

```
  ]
}
```

o cadastro de locadoras também pode conter filiais, ou seja é possível enviar mais de um cep

```
{
    "nome": "Localiza Rent a Car",
    "cnpj": "16.670.085/0001-55",
    "atividades": "Aluguel de Carros E Gestão de Frotas",
    "endereco": [
      {
        "cep": "96200-200",
        "number":"1234",
        "isFilial": false
      },
      {
        "cep": "96200-500",
        "number":"5678",
        "complemento": "Muro A",
        "isFilial": true
      }
  ]
}
```

Atenção:

- Todos os campos são **required, EXCETO** o campo complemento
- Não é possível haver **CNPJs** duplicados
- Deve haver **APENAS** um **isFilial: false ou seja apenas uma Matriz, as demais caso houver são filiais.**

OBSERVAÇÃO

- Atentem-se que no payload de cadastro da locadora, é enviado apenas o CEP. Para buscar o endereço completo, iremos realizar uma requisição a uma API externa chamada **VIA CEP**
- Esta é a URL para chamada da API que vocês devem utilizar: `https://viacep.com.br/ws/NUMERO_DO_CEP/json`
  - **DICA 1:** Vocês precisaram trocar múltiplas vezes o cep desta request, utilizem a propriedade do javascript chamada **TEMPLATE STRING** ou .replace() para atingir tal objetivo.
  - **DICA 2**: Utilizem a biblioteca **axios** para realização das requisições: `https://github.com/axios/axios`

2. **Listar todas as locadoras cadastrados**

   - **Request - GET** `http://localhost:3000/api/v1/rental`

```
{
  "locadoras": [
    {
      "id": "123",
      "nome": "Localiza Rent a Car",
      "cnpj": "16.670.085/0001-55",
      "atividades": "Aluguel de Carros E Gestão de Frotas",
      "endereco": [{
          "cep": "96200-200",
          "logradouro": "Rua General Canabarro",
          "complemento": "",
          "bairro": "Centro",
          "number":"1234",
          "localidade": "Rio Grande",
          "uf": "RS"
      }]
    },
    ...
  ],
  "total": 346,
  "limit": 100,
  "offset": 1,
  "offsets": 35
}
```

Atenção:

- Na rota de listar deve ser possível buscar por **query params**, exemplo `http://localhost:3000/api/v1/car/?cnpj=16.670.085/0001-55` buscando assim uma locadora que contenha o CNPJ **x**
- Assim como a busca pode ser feita por nome como qualquer outra entidade da locadora, nome, endereço, cep, isFilial, logradouro.
- Deve ser feito com **paginação.**

2. **Buscar locadora por ID**

   - **Request - GET** `http://localhost:3000/api/v1/rental/:id`

```
{
 "id": "123",
 "nome": "Localiza Rent a Car",
```

```
    "cnpj": "16.670.085/0001-55",
    "atividades": "Aluguel de Carros E Gestão de Frotas",
    "endereco": [{
        "cep": "96200-200",
        "logradouro": "Rua General Canabarro",
        "complemento": "",
        "bairro": "Centro",
        "number":"1234",
        "localidade": "Rio Grande",
        "uf": "RS"
    }]
}
```

3. **Atualizar alguma locadora cadastrada**
    - **Request - PUT** `http://localhost:3000/api/v1/rental/:id`

    Atenção:

    - Qualquer campo pode ser alterado
    - Caso o ID seja diferente do padrão deve retornar **400**, informando o erro.
    - Assim como as regras do cadastrar valem para o update.

3. **Remover um locadora cadastrada**
    - **Request - DELETE** `http://localhost:3000/api/v1/rental/:id`

Atenção:

- Em caso de sucesso retornar **204** com o *body* vazio
- Caso o ID seja diferente do padrão deve retornar **400**, informando o erro.
- Caso o **ID** não seja encontrado retornar **404**

Informações importantes aos alunos

Critérios de avaliação:

- Qualidade de escrita do código

- Organização do projeto
- Qualidade da API Restful
- Lógica da solução implementada
- Qualidade da camada de persistência
- Utilização do Git (quantidade e descrição dos commits, Git Flow)

Requisitos OBRIGATÓRIO:

- ESLint → airbnb
- Swagger → `http://localhost:3000/api/v1/api-docs`
- 50% de cobertura de teste (dica: utilizem a biblioteca **JEST**)
- Toda rota de carros deve conter o token bearer → com prazo de 24hs

Links Úteis

- https://dev.to/dianaops/como-escrever-um-readme-md-sensacional-no-github-4509
- http://karma-runner.github.io/4.0/dev/git-commit-msg.html
- https://desenvolvimentoparaweb.com/javascript/map-filter-reduce-javascript/
- https://medium.com/@diomalta/como-organizar-e-estruturar-projetos-com-node-js-4845be004899
- https://www.youtube.com/watch?v=KKTX1l3sZGk&t
- https://www.youtube.com/watch?v=rCeGfFk-uCk&t
- https://expressjs.com/pt-br/guide/writing-middleware.html
- https://joi.dev/api/?v=17.4.2
- https://www.npmjs.com/package/swagger-ui-express
- https://github.com/rocketseat-content/youtube-nodejs-tdd-jest