

# Desafio Final- Part. 1

A compasso entrou em um novo ramo de mercado, a compassolisa um seguimento carros para alugar de luxo e semi luxo. Para essa API vai ser necessário desenvolver algumas rotas.

Solicitação do cliente:

"Olá bom dia, gostaria de solicitar um sistema para abertura da minha locadora de carros. Ela se chama "compassolisa" e atuamos no seguimento de aluguel de carro de luxo e semi luxo. Trabalhamos com carros exclusivos ou seja não temos mais de um modelo de carro disponível por cidade.

Já temos o nosso Front-end pronto, porem estamos necessitando o Back-end, por isso contatamos vocês para desenvolver nosso Back-end, dado ao nosso front.

Iremos atuar em 3 etapas e dependendo de como for a primeira fecharemos o contrato da segunda e a terceira etapa. Nesta primeira etapa nos pensamos em **criar os registros de carros, usuários e a autenticação.** "

## Características do Desafio:

- O desafio consiste em criar uma aplicação API REST FULL
- Instruções para setup do serviço no [README.md](#).

## Requisitos Obrigatórios:

- Legibilidade
- Repositório Público
- Commits pequenos

- Explicação de como rodar localmente;
- MongoDB

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

## Desenvolver seguintes Endpoints

### 1. Criar um endpoint para cadastrar um carro

- Request - POST `http://localhost:3000/api/v1/car`

```
{ "modelo": "GM S10 2.8", "cor": "branco", "ano": "2021", "acessorios": [ { "descricao": "Ar-condicionado" }, { "descricao": "Dir. Hidráulica" }, { "descricao": "Cabine Dupla" }, { "descricao": "Tração 4x4" }, { "descricao": "4 portas" }, { "descricao": "Diesel" }, { "descricao": "Air bag" }, { "descricao": "ABS" } ], "quantidadePassageiros": 5 }
```

### Atenção:

- Todos os campos são **required**
- É necessário ter pelo menos **UM** acessório
- O ano do carro não pode ser menor que **1950** e maior que **2022**
- Não pode haver acessórios repetidos

## 2. Listar todos os carros cadastrados

- Request - GET `http://localhost:3000/api/v1/car`

```
{ "veiculos": [ { "_id": "123", "modelo": "GM S10 2.8", "cor": "branco", "ano": "2021", "acessorios": [ { "descricao": "Ar-condicionado" }, { "descricao": "Dir. Hidráulica" }, { "descricao": "Cabine Dupla" }, { "descricao": "Tração 4x4" }, { "descricao": "4 portas" }, { "descricao": "Diesel" }, { "descricao": "Air bag" }, { "descricao": "ABS" } ], "quantidadePassageiros": 5 }, ... ], "total": 3464, "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/?modelo="GM S10 2.8"` buscando assim todos carros que tem o modelo X
- Assim como a busca pode ser feita por nome como com qualquer outra entidade do Carro, acessório, ou cor ou modelo...
- E deve ser feito com paginação

## 3. Remover um carro cadastrado

- Request - DELETE `http://localhost:3000/api/v1/car/: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**

#### 4. Atualizar algum carro cadastrado

- Request - PUT `http://localhost:3000/api/v1/car/: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

#### 5. Buscar carro X cadastrado

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

#### Atenção:

- Caso o ID seja diferente do padrão deve retornar **400**, informando o erro.

---

#### 6. Criar um endpoint para cadastrar um pessoa

- Request - POST `http://localhost:3000/api/v1/people`

```
{ "nome": "joaozinho ciclano", "cpf": "131.147.860-49", "data_nascimento": "03/03/2021", "email": "joazinho@email.com", "senha": "123456", "habilitado": "sim" }
```

### Atenção:

- Todos os campos são **required**
- O usuário tem que ter no mínimo 18 anos a partir da data de cadastro.
- Precisa ter um CPF valido
- Precisa ter uma senha no mínimo 6 dígitos
- Precisa ter um Email valido
- habilitado pode ser sim ou não

Assim como as rotas de carro o usuário pode cadastrar, fazer update, delete e busca por id e listar todos.

---

## 7. Autenticação com o usuário

- Request - POST `http://localhost:3000/api/v1/authenticate`

```
{ "email": "joazinho@email.com", "senha": "123456" }
```

**Atenção:**

- Receber um token de autenticação através de um JWT o seu email e se é habilitado ou não
- 

## 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 Opcionais:**

- ESLint

- Swagger

## 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=KCTX1l3sZGk&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>

