Desafio Final - Part. 3

Chegamos a nossa ultima etapa, gostamos muito do trabalho entregue até agora. Porém precisamos de mais alguns ajustes e de algumas features novas para dar prosseguimento a nossa api.

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.

- Revisar cobertura de testes (se possível todos os testes cobrindo 100% do código!!)
- Ajuste geral de bugs e melhorias reportadas em nossas calls

---

Agora que nossa API já contém, usuários, carros e locadoras podemos prosseguir com as demais interações:

Locação de Carros

1. **Criação de uma entrada de locação de carro**

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

```
{
    "id": "asdasda149e5bda4f24e25a12d396a07fd3098",
    "id_user": "b149e5bda4f24123g34396a07fd3098",
    "data_inicio": "20/11/2021",
    "data_fim": "30/11/2021",
    "id_carro": "b149e5bda4f24e25a12d396a45645398",
    "id_locadora": "b149e5bda345232e25a12d396a07fd3098",
```

```
    "valor_final": "500,00",
}
```

OBS:

- id_user = Id correspondente do usuário que está fazendo a locação
- id_carro = Id correspondente ao carro que o usuário quer locar
- id_locadora = Id correspondente à locadora a qual o usuário está fazendo a locação do carro

Atenção:

- Todos os campos são **required**
- Calcular o **valor_final** baseado no valor da diária
- O usuario que for locar o carro precisa obrigatoriamente possuir habilitação
- **Não é** permitido mais de uma reserva do mesmo carro no mesmo dia.
- **NÃO é** permitida mais de uma reserva no mesmo período pelo mesmo usuário.
    - Exemplo:
        - **O Usuário fez sua reserva para o carro X de 20/11/2021 até 30/11/2021. Entre essas datas ele não poderá locar mais nenhum carro.**

2. **Listar todas reservas de uma determinada locadora as locadoras cadastrados**

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

```
{
  "reservas": [
    {
      "id": "asdasda149e5bda4f24e25a12d396a07fd3098",
      "id_user": "b149e5bda4f24123g34396a07fd3098",
      "data_inicio": "20/11/2021",
      "data_fim": "30/11/2021",
      "id_carro": "b149e5bda4f24e25a12d396a45645398",
      "id_locadora": "b149e5bda345232e25a12d396a07fd3098",
      "valor_final": "500,00",
      },
      {
```

```
        "id": "asdasda149e5bda4f24e25a12d2131098",
        "id_user": "b149e5bda4f24123g121fd3098",
        "data_inicio": "21/11/2021",
        "data_fim": "30/11/2021",
        "id_carro": "b149e5bda4f24e2512396a45645398",
        "id_locadora": "b149e5bda345232e1337fd3098",
        "valor_final": "435,00",
      }]
    },
    ...
  ],
  "total": 346,
  "limit": 100,
  "offset": 1,
  "offsets": 35
}
```

- Na rota de listar deve ser possível buscar por **query params**, exemplo `http://localhost:3000/api/v1/rental/:id/reserve?valor=160` buscando todas as reservas de valores iguais a R$160,000
- Assim como a busca pode ser feita por nome como qualquer outra entidade da id_user, data_inicio, data_fim, id_carro, valor_final
- Deve ser feito com **paginação.**

OBS:

- id_user = Id correspondente do usuário que está fazendo a locação
- id_carro = Id correspondente ao carro que o usuário quer locar
- id_locadora = Id correspondente à locadora a qual o usuário está fazendo a locação do carro

2. **Buscar reserva por ID**
   - **Request - GET** `http://localhost:3000/api/v1/rental/:id/reserve`

```
  {
   "id": "asdasda149e5bda4f24e25a12d396a07fd3098",
   "id_user": "b149e5bda4f24123g34396a07fd3098",
   "data_inicio": "20/11/2021",
   "data_fim": "30/11/2021",
   "id_carro": "b149e5bda4f24e25a12d396a45645398",
   "id_locadora": "b149e5bda345232e25a12d396a07fd3098",
   "valor_final": "500,00",
  }
```

3. **Atualizar dado de alguma reserva por ID**

   - **Request - PUT** `http://localhost:3000/api/v1/rental/:id/reserve/: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.
   - Não é permitido atualizar reservas, caso o carro já tenha reserva para um dia seguinte por exemplo.

3. **Remover uma reserva**

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

Frota de Carros

1. **Criação da frota de carros de uma locadora**

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

```
{
    "id": "asdasda149e5bdagga12d396a07fd3098",
    "id_carro": "b149e5bdggg34396a07fd3098",
    "id_locadora":"asdasda149e5bdagga12d396a07fd3098",
    "status": "disponivel",
    "valor_diaria": "100,00",
    "placa": "ABC1234",
}
```

OBS:

- id_carro = Id correspondente ao carro

- id_locadora = Id da locadora dona do carro. (O id_carro pode se repetir entre locadoras pois temos base de carros, porém o que irá diferir eles entre as locadoras é a placa)

Atenção:

- Todos os campos são **required**
- Não pode haver um mais de um carro com a mesma placa
- id_locadora poderá ser de uma locadora filial
- id_carro pode aparecer em mais de uma locadora, porém a placa nao pode ser a mesma. (O id_carro seria equivalente ao modelo de carro)

2. **Listar toda a frota de uma determinada locadora**

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

```
{
  "frota": [
    {
        "id": "asdasda149e5bdagga12d396a07fd3098",
        "id_carro": "b149e5bdggg34396a07fd3098",
        "status": "disponivel",
        "valor_diaria": "100,00",
        "id_locadora": "b149e5bda345ddd5a12d396a07fd3098",
        "placa": "ABC1234",
      },
      {
        "id": "asdasda149e5bdagga12d396a07fd3098",
        "id_carro": "b149e5bdggg34396a07fd3098",
        "status": "alugado",
        "valor_diaria": "100,00",
        "id_locadora": "b149e5bda345ddd5a12d396a07fd3098",
        "placa": "ABC1234",
      }]
    },
    ...
  ],
  "total": 346,
  "limit": 100,
  "offset": 1,
  "offsets": 35
}
```

- Na rota de listar deve ser possível buscar por **query params**, exemplo `http://localhost:3000/api/v1/rental/:id/fleet?valor_diaria=100` buscando todas as reservas de valores iguais a R$100,000
- Assim como a busca pode ser feita por nome como qualquer outra entidade da valor_diaria, id_locadora, placa, id_carro
- Deve ser feito com **paginação.**

2. **Buscar frota por ID especific de uma determinada locadora**
    - **Request - GET** `http://localhost:3000/api/v1/rental/:id/fleet/:id`

```
 {
  "id": "asdasda149e5bdagga12d396a07fd3098",
  "id_carro": "b149e5bdggg34396a07fd3098",
  "status": "disponivel",
  "valor_diaria": "100,00",
  "id_locadora": "b149e5bda345ddd5a12d396a07fd3098",
  "placa": "ABC1234",
}
```

3. **Atualizar dado de alguma frota por ID de uma determinada locadora**
    - **Request - PUT** `http://localhost:3000/api/v1/rental/:id/fleet/: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 carro da frota de uma determinada locadora**
    - **Request - DELETE** `http://localhost:3000/api/v1/rental/:id/fleet/: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`
- Heroku
- 100% 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