

Relatório do Trabalho Prático - Sistema de Locadora de Veículos

Artur M. Silva¹

¹Instituto de Ciências Exatas e Informática –
Pontifícia Universidade Católica de Minas Gerais (PUCMINAS)

1. Introdução

1.1. Contextualização do Projeto e sua Importância

O presente relatório tem como objetivo fornecer uma análise abrangente do projeto desenvolvido para incentivar os alunos a aprenderem conceitos de backend, utilizando tecnologias modernas e práticas. O projeto consiste na implementação de uma locadora de veículos, utilizando C#, ASP.NET, Entity Framework, SQL e Swagger para testes, visando familiarizar os alunos com essas ferramentas, e também proporcionar uma compreensão mais profunda por meio da aplicação em um contexto real.

1.2. Objetivos do Sistema de Locadora de Veículos

Nesse contexto, o projeto é focado em desenvolver uma API utilizando ASP.NET para um sistema de locadora de veículos, onde clientes podem realizar reservas, alugar e devolver veículos.

1.3. Tecnologias Utilizadas

- IDE: Visual Studio Community
- Repositório: Github
- Ambiente SQL: SQL Server Manager Studio

1.4. Principais Funcionalidades

As principais funcionalidades do sistema relacionadas a clientes são:

- Cadastro e gestão de clientes.
- Pesquisa de todos ou de um cliente específico.

As principais funcionalidades do sistema relacionadas a veículos são:

- Cadastro e gestão de veículos.
- Pesquisa de todos ou de um veículo específico.
- Pesquisa de todos os veículos disponíveis.

As principais funcionalidades do sistema relacionadas a locações são:

- Cadastro e gestão de locações.
- Pesquisa de uma locação específica.
- Pesquisa de todas as locações ativas disponíveis.

2. Requisitos Funcionais

ID	Descrição do Requisito	Prioridade
RF-001	O sistema deve permitir o cadastro de clientes	ALTA
RF-002	O sistema deve permitir visualizar, atualizar e excluir informações de clientes existentes	ALTA
RF-003	O sistema deve permitir a pesquisa de todos os clientes e clientes por CPF	BAIXA
RF-004	O sistema deve permitir o cadastro de veículos	ALTA
RF-005	O sistema deve permitir visualizar, atualizar e excluir informações de veículos existentes	ALTA
RF-006	O sistema deve permitir a pesquisa de todos os veículos e veículos por placa	BAIXA
RF-007	O sistema deve permitir buscar todos os veículos disponíveis	ALTA
RF-008	O sistema deve permitir a criação de locações	ALTA
RF-009	O sistema deve permitir visualizar e atualizar informações de locações existentes	ALTA
RF-010	O sistema deve permitir a pesquisa de todas as locações e locações por ID	MÉDIA
RF-011	O sistema deve permitir buscar todas as locações ativas	BAIXA

3. Arquitetura do Sistema

A arquitetura adotada para o desenvolvimento do sistema segue o padrão MVC (Model, View, Controller), uma abordagem que organiza a aplicação em três camadas distintas. A camada View é responsável pela apresentação dos dados aos usuários finais, e neste projeto, não há Views, essa responsabilidade foi assumida diretamente pelo Swagger, que fornece uma interface interativa para testar e documentar as APIs desenvolvidas. A camada Controller, por sua vez, é responsável por receber as requisições HTTP, realizar validações e tratamentos de erros, garantindo a integridade e segurança das operações realizadas. Finalmente, a camada Model descreve as entidades da aplicação, sendo responsável pela manipulação de dados.

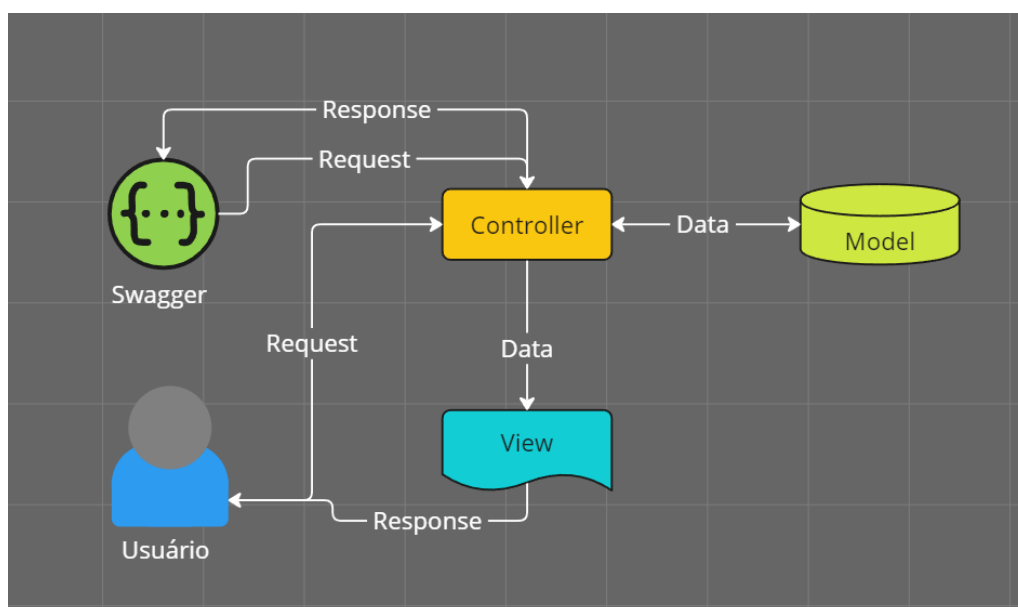


Figure 1. Exemplo da arquitetura com a adição da View para melhor entendimento

4. Implementação Técnica

4.1. Principais Classes e Responsabilidades

- **Cliente:** Representa a entidade Cliente, responsável por armazenar informações como CPF, nome, telefone e email. Também possui uma coleção de locações associadas a esse cliente.
- **Locacao:** Representa a entidade de Locação, que armazena informações como o ID da locação, CPF do cliente, placa do veículo, datas de início e término da locação e status. Após a devolução, armazena a data de devolução, custo total do carro, e descrição.
- **Veiculo:** Representa a entidade Veículo, que armazena informações como placa, modelo, marca, ano, preço diário de locação, adicional por atraso diário e disponibilidade do veículo.

4.2. Detalhamento das tecnologias utilizadas

- **C# 12.0 e ASP.NET (.NET 8.0):** O ASP.NET oferece um ambiente de execução de alto desempenho para aplicativos Web, enquanto o C# 12.0 (ou superior) nos permite escrever código limpo e legível.
- **Entity Framework:** Optamos pelo Entity Framework para lidar com o mapeamento objeto-relacional (ORM), facilitando a interação com o banco de dados SQL Server. Os pacotes NuGet utilizados incluem:
 - `Microsoft.EntityFrameworkCore.SqlServer`
 - `Microsoft.EntityFrameworkCore.Tools`
 - `Microsoft.VisualStudio.Web.CodeGeneration.Design`
- **Swagger (Swashbuckle.AspNetCore):** Integramos o Swagger ao nosso projeto para facilitar a documentação e teste de nossas APIs. O Swashbuckle.AspNetCore é uma biblioteca popular que gera automaticamente uma interface interativa com base nos endpoints da API, simplificando o processo de desenvolvimento e colaboração.
- **SQL Server Management:** Para gerenciar nosso banco de dados, utilizamos o SQL Server Management Studio. Essa ferramenta oferece uma interface intuitiva e robusta para administrar e manter o banco de dados SQL Server, permitindo-nos realizar operações de forma eficiente e segura.

Essas tecnologias combinadas proporcionam uma base sólida e moderna para o nosso sistema de locadora de veículos, garantindo desempenho, segurança e facilidade de manutenção ao longo do tempo.

5. Descrição do Banco de Dados

5.1. Modelo Conceitual

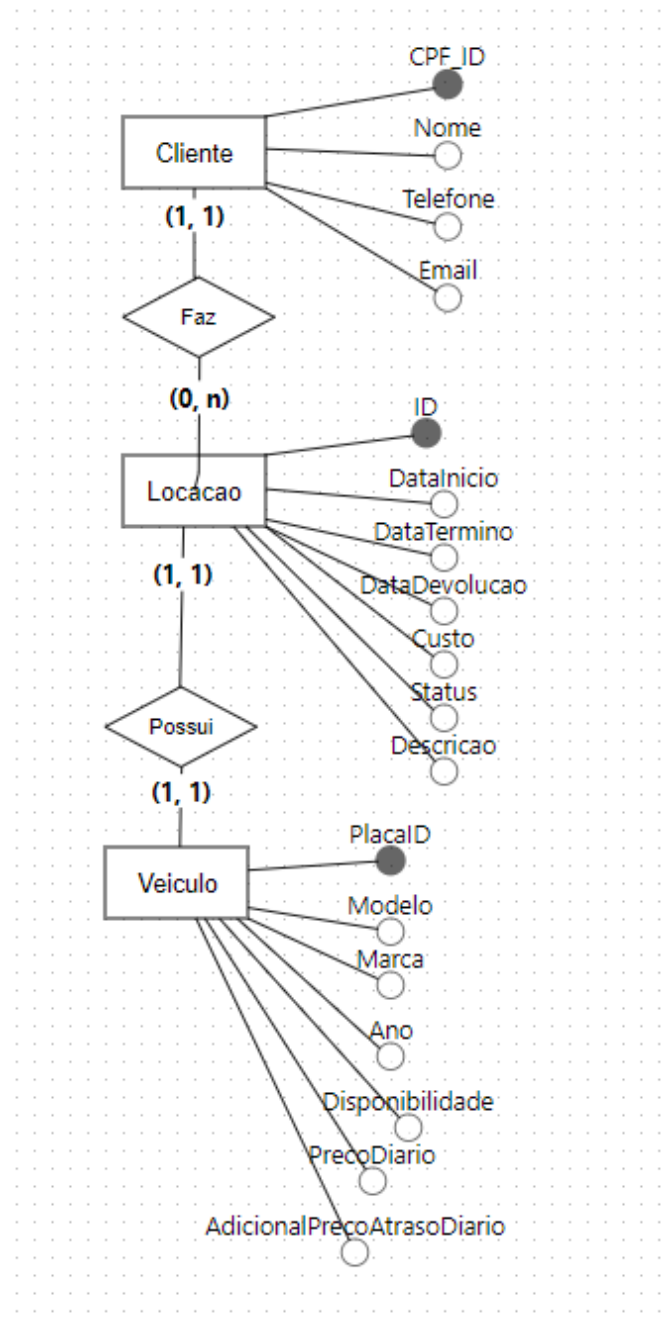


Figure 2. Modelo Conceitual.

6. Tabela Cliente:

CPF (PK)	Chave primária que identifica unicamente cada cliente.
Nome	Nome do cliente.
Telefone	Número de telefone do cliente.
Email	Endereço de e-mail do cliente.
Locacoes	Relacionamento um para muitos com a tabela Locacao, representando as locações associadas a esse cliente.

7. Tabela Locacao:

LocacaoID (PK)	Chave primária que identifica unicamente cada locação.
ClienteID (FK)	Chave estrangeira que referencia o CPF do cliente associado a essa locação.
PlacaID (FK)	Chave estrangeira que referencia a placa do veículo associado a essa locação.
DataInicio	Data de início da locação.
DataTermino	Data de término da locação.
DataDevolucao	Data em que o veículo foi devolvido.
Custo	Custo total da locação.
Ativa	Indica se a locação está ativa ou se já foi concluída.
Descricao	Descrição geral da locação.

8. Tabela Veiculo:

PlacaID (PK)	Chave primária que identifica unicamente cada veículo.
Modelo	Modelo do veículo.
Marca	Marca do veículo.
Ano	Ano de fabricação do veículo.
PrecoDiario	Preço da diária de aluguel do veículo.
PrecoAtrasoDiario	Preço da diária do aluguel, caso ocorra atraso na devolução do veículo.
Disponibilidade	Indica se o veículo está disponível para locação.

Com base nesse modelo, as restrições de integridade referem-se principalmente aos relacionamentos entre as tabelas:

- A chave estrangeira ClienteID na tabela Locacao referencia a chave primária CPF na tabela Cliente, garantindo que cada locação esteja associada a um único cliente.
- A chave estrangeira PlacaID na tabela Locacao referencia a chave primária PlacaID na tabela Veiculo, garantindo que cada locação esteja associada a um único veículo.
- O relacionamento um para muitos entre Cliente e Locacao indica que um cliente pode ter várias locações, mas cada locação está associada a apenas um cliente.
- O relacionamento um para um entre Veiculo e Locacao indica que um veículo só pode ter uma locação por vez, garantindo que o mesmo veículo não seja alugado por mais de um cliente simultaneamente.

9. Testes e Validação

9.1. Endpoint - Cliente

GET (/api/Clientes)

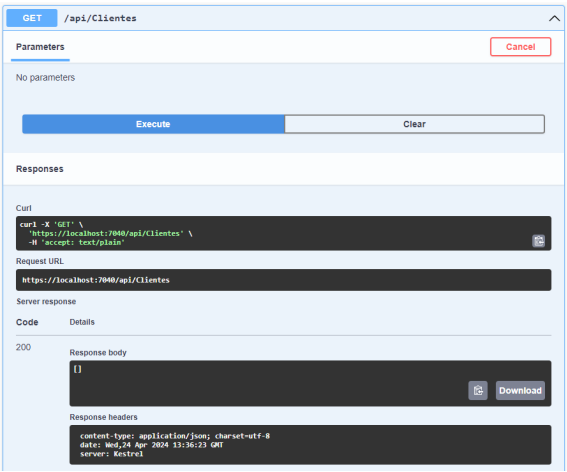


Figure 3. Lista de clientes vazia.

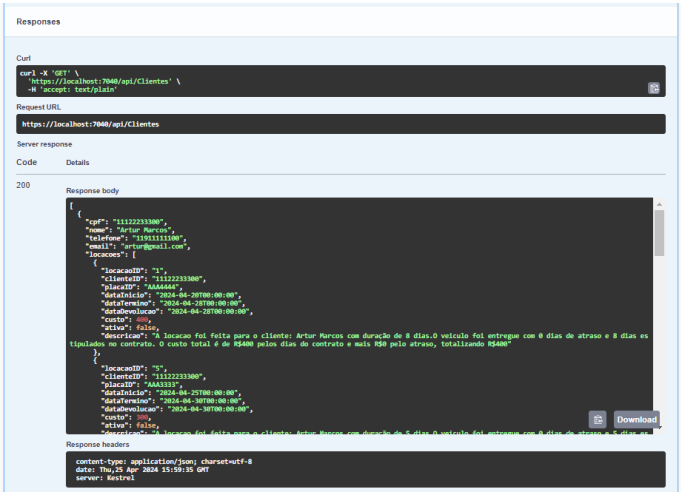


Figure 4. Lista de clientes preenchida.

POST (api/Clientes)

Responses

Curl

```
curl -X 'POST' \
  "https://localhost:7040/api/Clientes" \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "cpf": "11122233300",
    "nome": "Artur Marcos",
    "telefone": "1191111100",
    "email": "artur@mail.com",
    "locações": [
      {
        "localidadeID": "string",
        "clienteID": "string",
        "placaID": "string",
        "dataInicio": "2024-04-24T11:11:25.182Z",
        "dataFim": "2024-04-24T11:11:25.182Z",
        "dataDevolução": "2024-04-24T11:11:25.182Z",
        "custo": 0,
        "ativa": true,
        "descricao": "string"
      }
    ]
  }'
```

Request URL

https://localhost:7040/api/Clientes

Server response

Code

Details

201

undocumented

Response body

Cliente criado com sucesso. Devido às regras de negócio, as locações foram salvas vazias e só poderão ser preenchidas no endpoint de alugar.

Detalhes do Cliente:
Nome: Artur Marcos
CPF: 11122233300
Email: artur@mail.com
Telefone: 1111111100
locações: []

Response headers

content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 13:36:34 GMT
location: https://localhost:7040/api/Clientes/11122233300
server: Kestrel

Figure 5. Adicionando Artur Marcos.

Responses

Curl

```
curl -X 'POST' \
  "https://localhost:7040/api/Clientes" \
  -H 'accept: text/plain' \
  -H 'Content-Type: application/json' \
  -d '{
    "cpf": "11122233301",
    "nome": "Felipe Augusto",
    "telefone": "1191111101",
    "email": "felipe@mail.com",
    "locações": [
      {
        "localidadeID": "string",
        "clienteID": "string",
        "placaID": "string",
        "dataInicio": "2024-04-24T11:11:25.182Z",
        "dataFim": "2024-04-24T11:11:25.182Z",
        "dataDevolução": "2024-04-24T11:11:25.182Z",
        "custo": 0,
        "ativa": true,
        "descricao": "string"
      }
    ]
  }'
```

Request URL

https://localhost:7040/api/Clientes

Server response

Code

Details

201

undocumented

Response body

Cliente criado com sucesso. Devido às regras de negócio, as locações foram salvas vazias e só poderão ser preenchidas no endpoint de alugar.

Detalhes do Cliente:
Nome: Felipe Augusto
CPF: 11122233301
Email: felipe@mail.com
Telefone: 1111111101
locações: []

Response headers

content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 13:37:22 GMT
location: https://localhost:7040/api/Clientes/11122233301
server: Kestrel

Figure 6. Adicionando Felipe Augusto.

GET (api/Clientes/id)

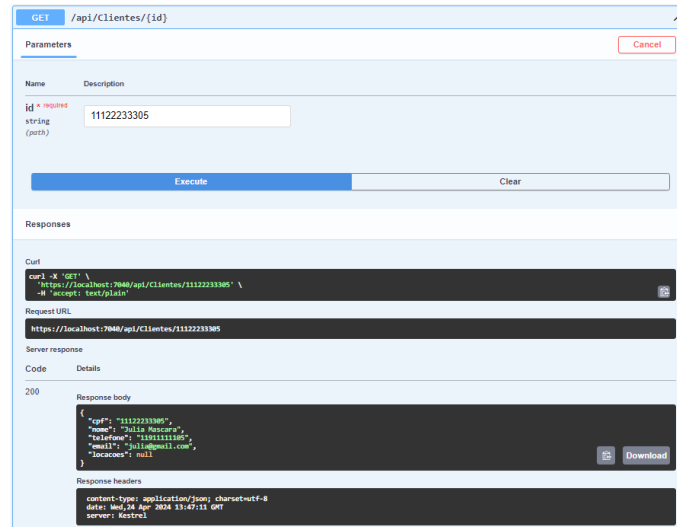


Figure 7. Buscando pelo Julia Mascara.

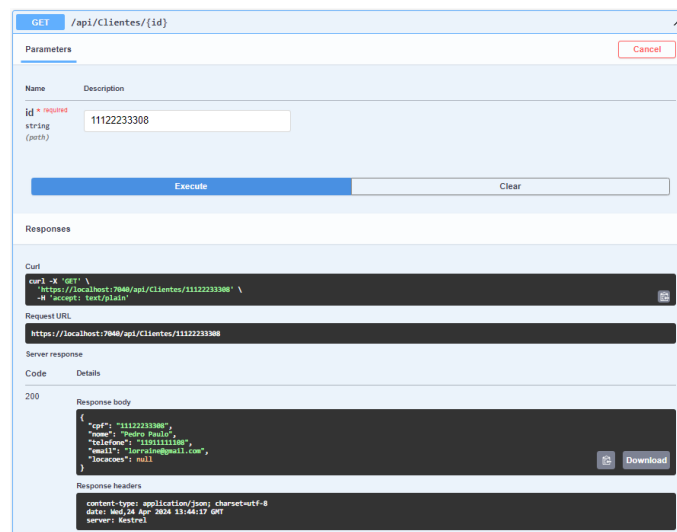


Figure 8. Buscando por Pedro Paulo.

PUT (api/Clientes/id)

PUT /api/Clientes/{id}

Parameters

Name	Description
id * required string (path)	11122233308

Request body

application/json

```
{
  "cpf": "11122233308",
  "nome": "Pedro Paulo",
  "telefone": "11911111188",
  "email": "pedro-paulo@gmail.com",
  "locações": null
}
```

Execute

Figure 9. Exemplo de requisição PUT.

Responses

Curl

```
curl -X PUT \
  https://localhost:7040/api/Clientes/11122233308 \
  -H 'accept: */*' \
  -H 'content-type: application/json' \
  -d '{
    "cpf": "11122233308",
    "nome": "Pedro Paulo",
    "telefone": "11911111188",
    "email": "pedro-paulo@gmail.com",
    "locações": null
  }'
```

Request URL

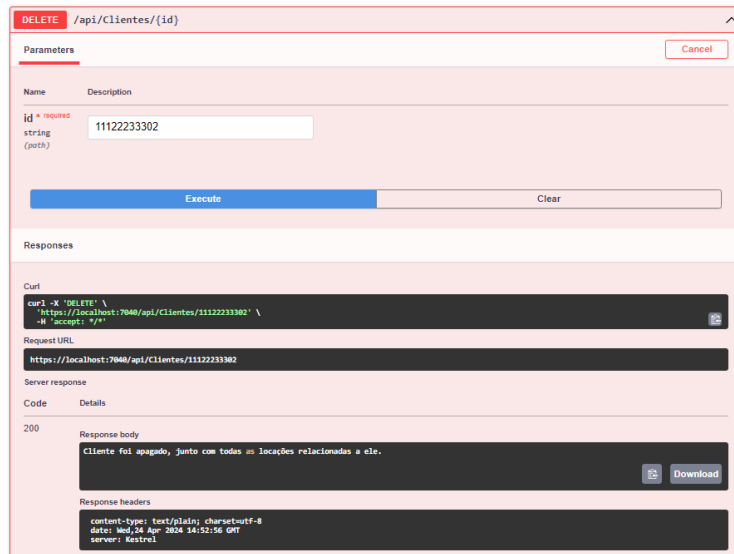
https://localhost:7040/api/Clientes/11122233308

Server response

Code	Details
200	<p>Response body</p> <p>Cliente modificado. Atenção, devido as regras de negocio, não é possível modificar as locações pelo cliente.</p> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Wed, 04 Apr 2024 13:58:29 GMT server: Kestrel</pre>

Figure 10. Modificando o email incorreto do Pedro Paulo.

DELETE (api/Clientes/id)



DELETE /api/Clientes/{id}

Parameters

Name	Description
id * required	
string	
(path)	

11122233302

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  "https://localhost:7040/api/Clientes/11122233302" \
  -H 'accept: */*'
```

Request URL

https://localhost:7040/api/Clientes/11122233302

Server response

Code Details

200

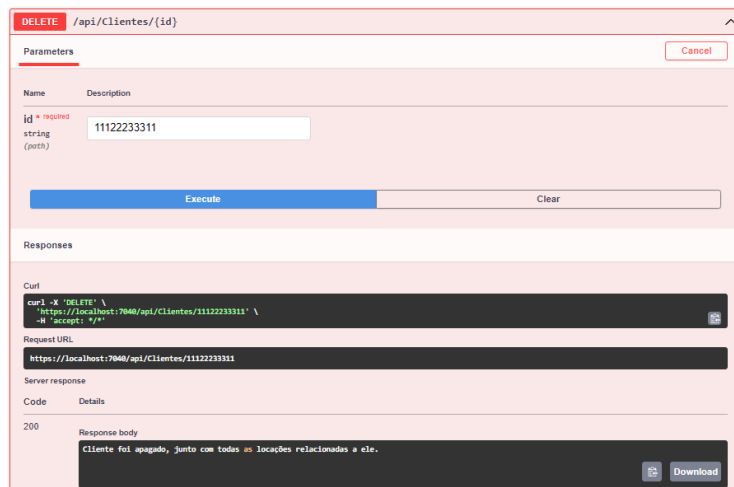
Response body

Cliente foi apagado, junto com todas as locações relacionadas a ele.

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 14:52:56 GMT
server: Kestrel
```

Figure 11. Apagando cliente 11122233302.



DELETE /api/Clientes/{id}

Parameters

Name	Description
id * required	
string	
(path)	

11122233311

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  "https://localhost:7040/api/Clientes/11122233311" \
  -H 'accept: */*'
```

Request URL

https://localhost:7040/api/Clientes/11122233311

Server response

Code Details

200

Response body

Cliente foi apagado, junto com todas as locações relacionadas a ele.

Response headers

Figure 12. Apagando cliente 11122233311.

9.2. Endpoint - Veiculos

GET (api/Veiculo)

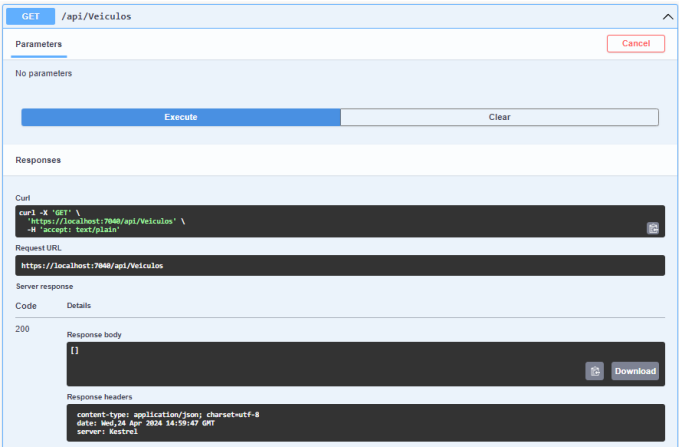


Figure 13. Lista de veículos vazia.

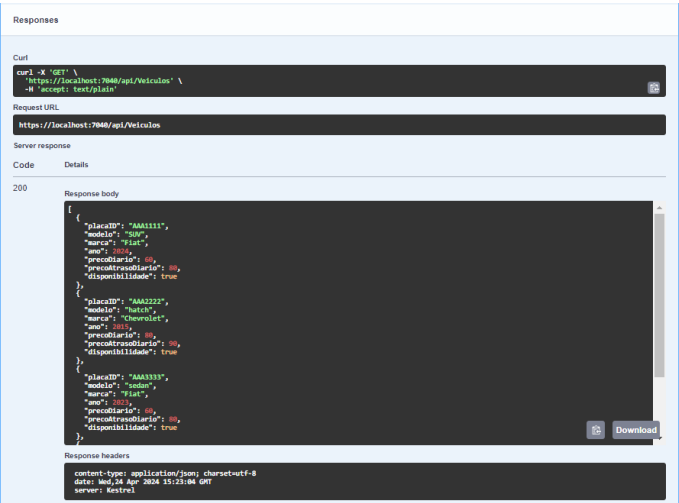


Figure 14. Lista de veículos preenchida.

POST (api/Veiculos)

The screenshot displays a REST client interface with the following details:

- Request:** A curl command is shown: `curl -X 'POST' \ -H 'accept: text/plain' \ -H 'Content-Type: application/json' \ -d '{ "placaID": "AAA3333", "modelo": "Fiat", "marca": "Fiat", "ano": 2021, "precoDiaria": 60, "precoContratual": 80, "disponibilidade": true }'`
- Request URL:** `https://localhost:7040/api/Veiculos`
- Server response:** A 201 status code is returned.
- Response body:** `Veículo adicionado. Devido regras de negocio, a disponibilidade do veículo foi configurada para 'true'.`
- Response headers:** `content-type: text/plain; charset=utf-8, date: Wed, 24 Apr 2024 15:19:41 GMT, location: https://localhost:7040/api/Veiculos/AAA3333, server: Kestrel`

Figure 15. Adicionando Fiat AAA3333

The screenshot displays a REST client interface with the following details:

- Request:** A curl command is shown: `curl -X 'POST' \ -H 'accept: text/plain' \ -H 'Content-Type: application/json' \ -d '{ "placaID": "AAA4444", "modelo": "Hyundai", "marca": "Hyundai", "ano": 2024, "precoDiaria": 50, "precoContratual": 55, "disponibilidade": true }'`
- Request URL:** `https://localhost:7040/api/Veiculos`
- Server response:** A 201 status code is returned.
- Response body:** `Veículo adicionado. Devido regras de negocio, a disponibilidade do veículo foi configurada para 'true'.`
- Response headers:** `content-type: text/plain; charset=utf-8, date: Wed, 24 Apr 2024 15:20:48 GMT, location: https://localhost:7040/api/Veiculos/AAA4444, server: Kestrel`

Figure 16. Adicionando Hyundai AAA4444

GET (api/Veiculos/id)

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7040/api/Veiculos/AAA1111' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7040/api/Veiculos/AAA1111

Server response

Code Details

200

Response body

```
{
  "placaID": "AAA1111",
  "modelo": "SLA",
  "marca": "Fiat",
  "ano": 2014,
  "precoDiario": 60,
  "precoAluguelMensal": 60,
  "disponibilidade": true
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Wed, 24 Apr 2024 16:56:31 GMT
server: Kestrel
```

Figure 17. Buscando pelo veículo AAA1111.

Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7040/api/Veiculos/AAA1111' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7040/api/Veiculos/AAA1111

Server response

Code Details

404

Undocumented

Error: response status is 404

Response body

0 placa digitada não pertence a nenhum veículo.

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 16:56:56 GMT
server: Kestrel
```

Figure 18. Buscando por veículo inexistente.

PUT (api/Veiculos/id)

The screenshot shows a REST client interface with the following details:

- Responses** tab is selected.
- Curl** section contains the command:

```
curl -X 'PUT' \
  https://localhost:7040/api/Veiculos/AAA1111 \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "placaID": "AAA1111",
    "modelo": "GOL",
    "marca": "BMW",
    "ano": 2015,
    "precoDiario": 0,
    "precoAltraseoDiario": 0,
    "disponibilidade": true
  }'
```
- Request URL** is `https://localhost:7040/api/Veiculos/AAA1111`.
- Server response** shows a **Code** of 200.
- Response body** contains the text: `Veiculo modificado.`
- Response headers** include:

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 15:43:36 GMT
server: Kestrel
```

Figure 19. Modificando veículo com AAA1111.

The screenshot shows a REST client interface with the following details:

- Responses** tab is selected.
- Curl** section contains the command:

```
curl -X 'PUT' \
  https://localhost:7040/api/Veiculos/a45asda \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "placaID": "string",
    "modelo": "string",
    "marca": "string",
    "ano": 0,
    "precoDiario": 0,
    "precoAltraseoDiario": 0,
    "disponibilidade": true
  }'
```
- Request URL** is `https://localhost:7040/api/Veiculos/a45asda`.
- Server response** shows a **Code** of 400.
- Response body** contains the text: `A placa informada nos parametros é diferente do informado no corpo da requisição.`
- Response headers** include:

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 16:59:17 GMT
server: Kestrel
```

Figure 20. Modificando veículo com placas distintas

DELETE (api/Veiculos/id)

DELETE /api/Veiculos/{id}

Parameters

Cancel

Name	Description
id * required	
string (path)	AAA1111

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  "https://localhost:7040/api/Veiculos/AAA1111" \
  -H 'accept: */*'
```

Request URL

```
https://localhost:7040/api/Veiculos/AAA1111
```

Server response

Code	Details
200	<p>Response body</p> <p>Veiculo foi apagado.</p> <p>Download</p> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Wed, 24 Apr 2024 15:04:25 GMT server: Kestrel</pre>

Figure 21. Apagando cliente AAA1111.

DELETE /api/Veiculos/{id}

Parameters

Cancel

Name	Description
id * required	
string (path)	ABC1234

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  "https://localhost:7040/api/Veiculos/ABC1234" \
  -H 'accept: */*'
```

Request URL

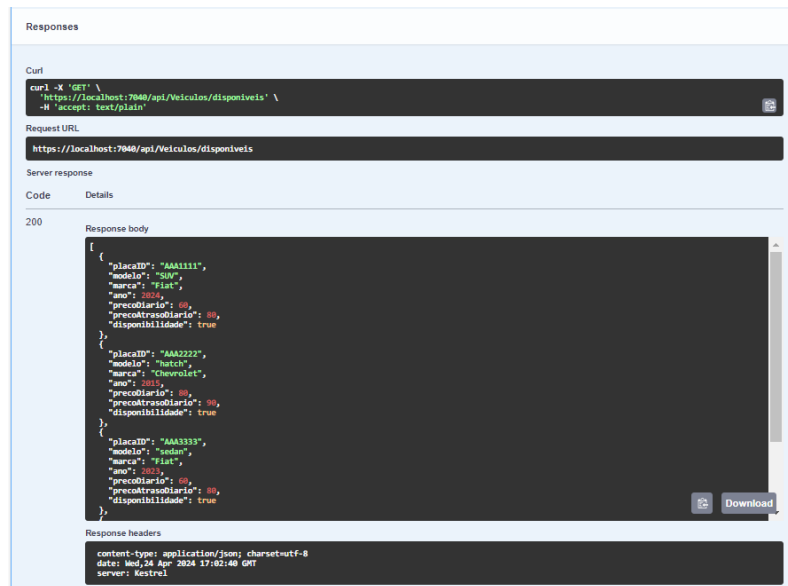
```
https://localhost:7040/api/Veiculos/ABC1234
```

Server response

Code	Details
200	<p>Response body</p> <p>Veiculo foi apagado.</p> <p>Download</p> <p>Response headers</p> <pre>content-type: text/plain; charset=utf-8 date: Wed, 24 Apr 2024 15:04:41 GMT server: Kestrel</pre>

Figure 22. Apagando cliente ABC1234.

GET (api/Veiculos/disponiveis)



Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7040/api/Veiculos/disponiveis' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7040/api/Veiculos/disponiveis

Server response

Code Details

200

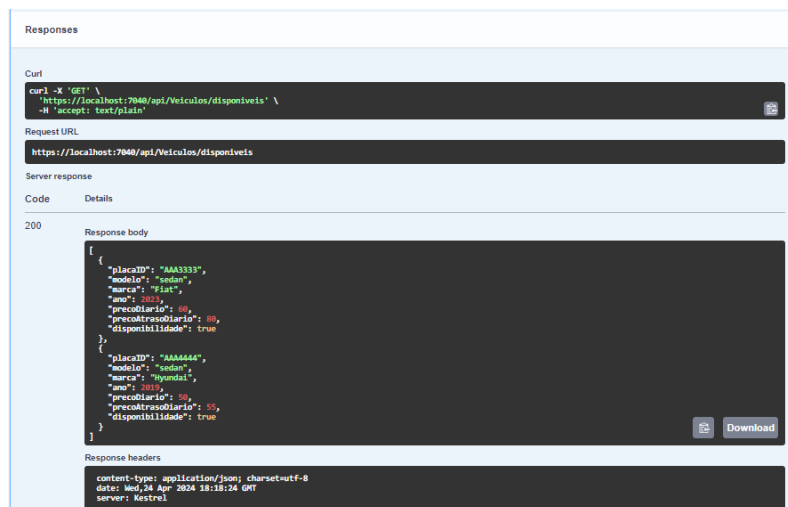
Response body

```
{
  "placaID": "AAA1111",
  "modelo": "Sole",
  "marca": "Fiat",
  "ano": 2014,
  "precoDiario": 80,
  "precoTrasobdiario": 80,
  "disponibilidade": true
},
{
  "placaID": "AAA2222",
  "modelo": "hatch",
  "marca": "Chevrolet",
  "ano": 2015,
  "precoDiario": 80,
  "precoTrasobdiario": 80,
  "disponibilidade": true
},
{
  "placaID": "AAA3333",
  "modelo": "sodan",
  "marca": "Fiat",
  "ano": 2013,
  "precoDiario": 80,
  "precoTrasobdiario": 80,
  "disponibilidade": true
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Wed, 24 Apr 2024 17:02:48 GMT
server: Kestrel
```

Figure 23. Mostrando todos os veículos disponiveis.



Responses

Curl

```
curl -X 'GET' \
  'https://localhost:7040/api/Veiculos/disponiveis' \
  -H 'accept: text/plain'
```

Request URL

https://localhost:7040/api/Veiculos/disponiveis

Server response

Code Details

200

Response body

```
{
  "placaID": "AAA3333",
  "modelo": "sodan",
  "marca": "Fiat",
  "ano": 2013,
  "precoDiario": 80,
  "precoTrasobdiario": 80,
  "disponibilidade": true
},
{
  "placaID": "AAAA444",
  "modelo": "sodan",
  "marca": "Hyundai",
  "ano": 2019,
  "precoDiario": 50,
  "precoTrasobdiario": 50,
  "disponibilidade": true
}
```

Response headers

```
content-type: application/json; charset=utf-8
date: Wed, 24 Apr 2024 18:16:24 GMT
server: Kestrel
```

Figure 24. Mostrando todos os veículos disponiveis.

9.3. Endpoint - Locacoes

Para entender melhor o que vai ser feito, observe o diagrama abaixo e depois analise os resultados dos testes dos endpoints.

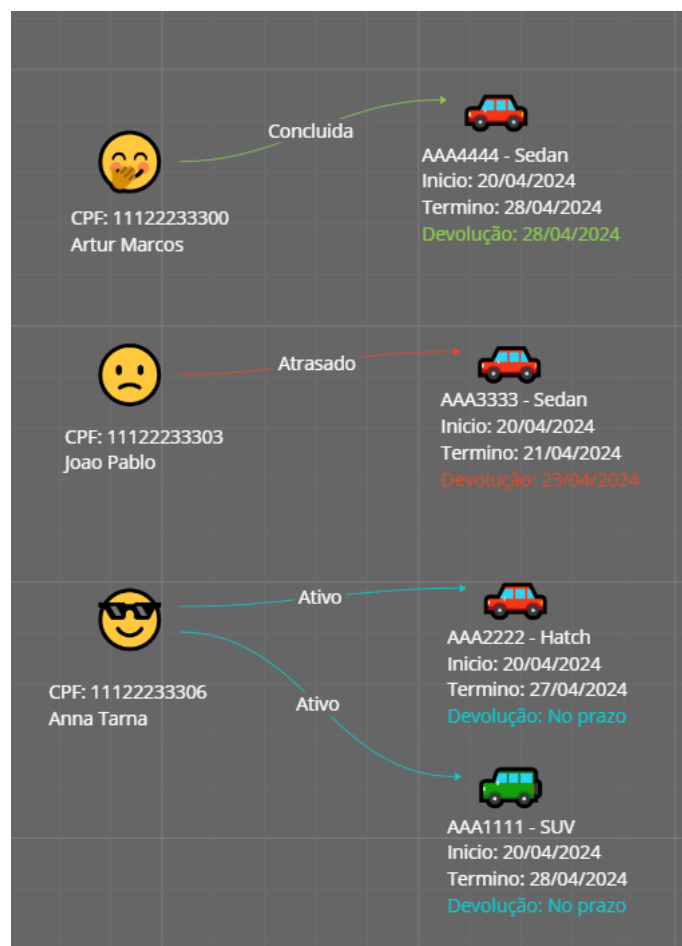


Figure 25. Modelo das locações criadas.

GET (api/Locacoes)

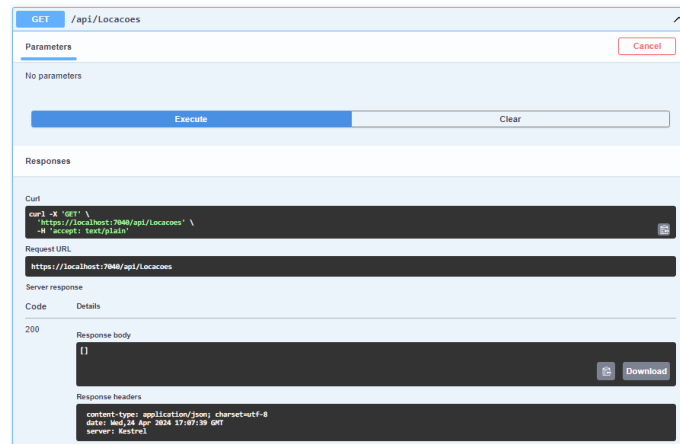


Figure 26. Lista de locações vazia.

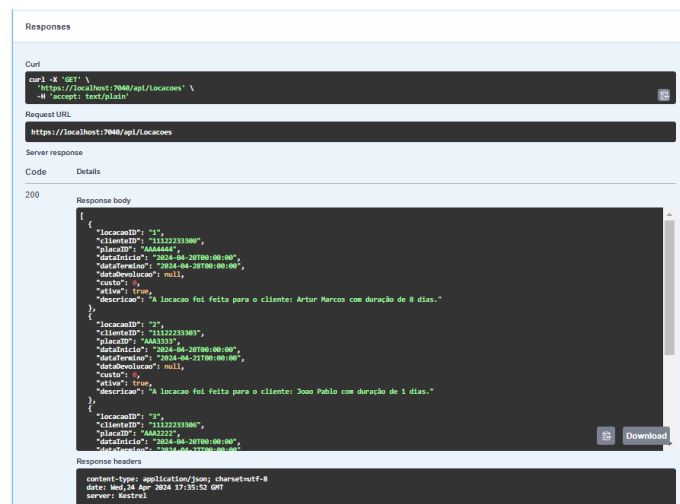


Figure 27. Lista de locações preenchida.

POST (api/Locacoes/alugar)

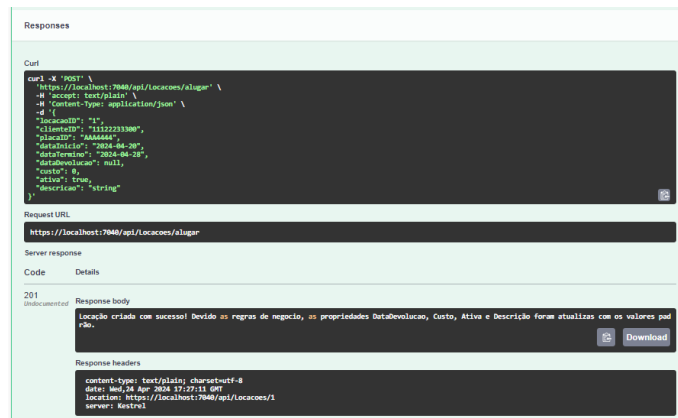


Figure 28. Adicionando locação.

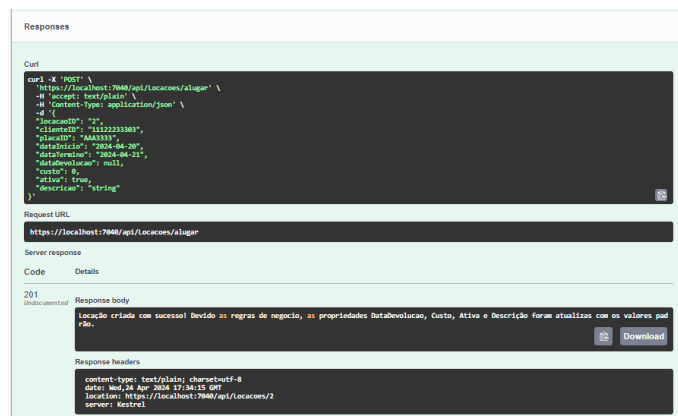


Figure 29. Adicionando locação.

GET (api/Locacoes/id)

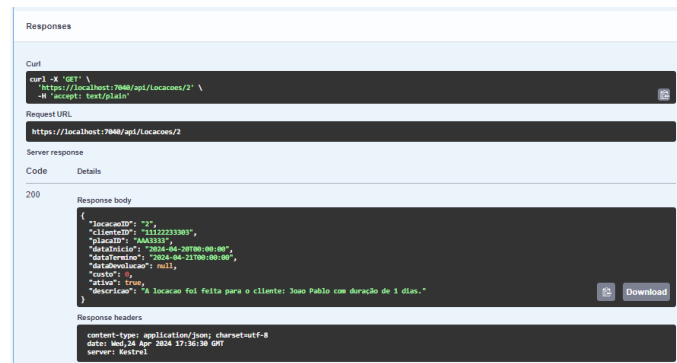


Figure 30. Buscando pela locação 2 que está ativa.

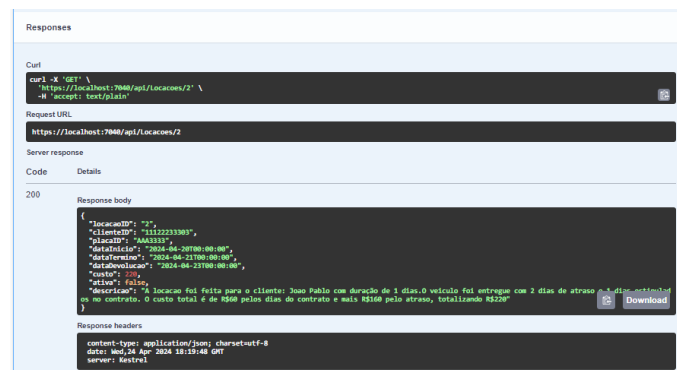


Figure 31. Buscando pela locação 2 que não está mais ativa.

GET (api/Locacoes/ativas)

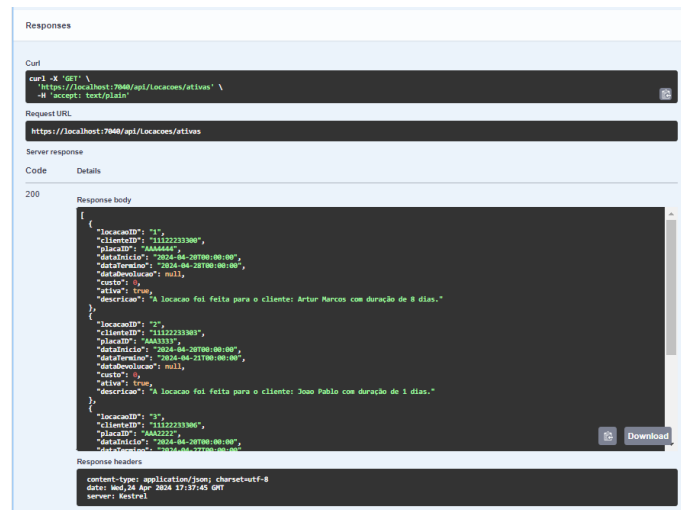


Figure 32. Mostrando todos as locações ativas.

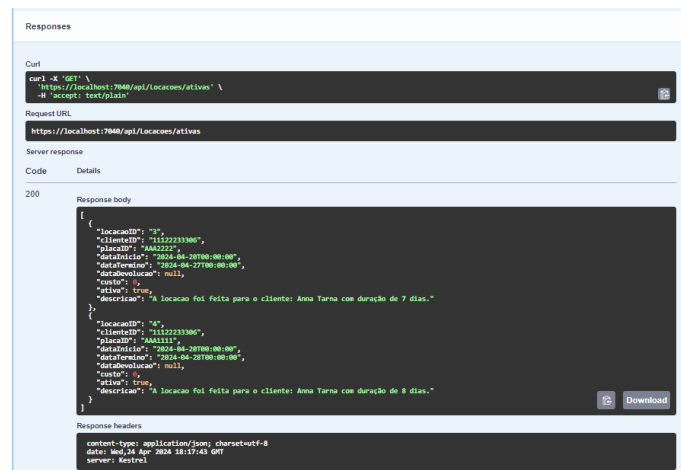


Figure 33. Mostrando todos as locações ativas.

GET (api/Locacoes/nao-ativas)

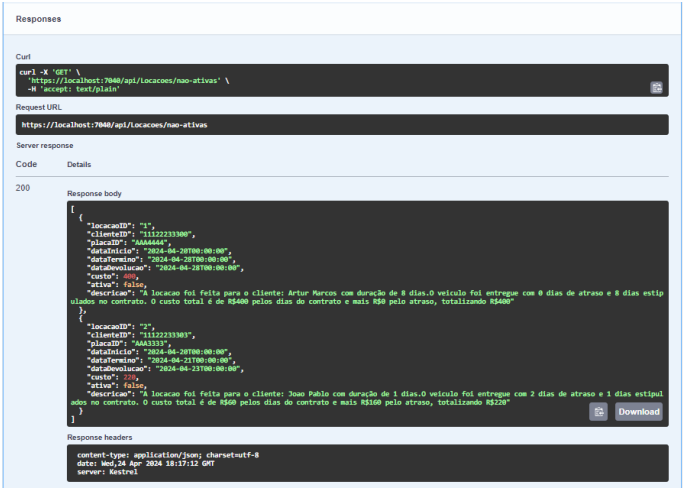


Figure 34. Mostrando todas as locações que não estão ativas.

PUT (api/Locacoes/devolver/id)

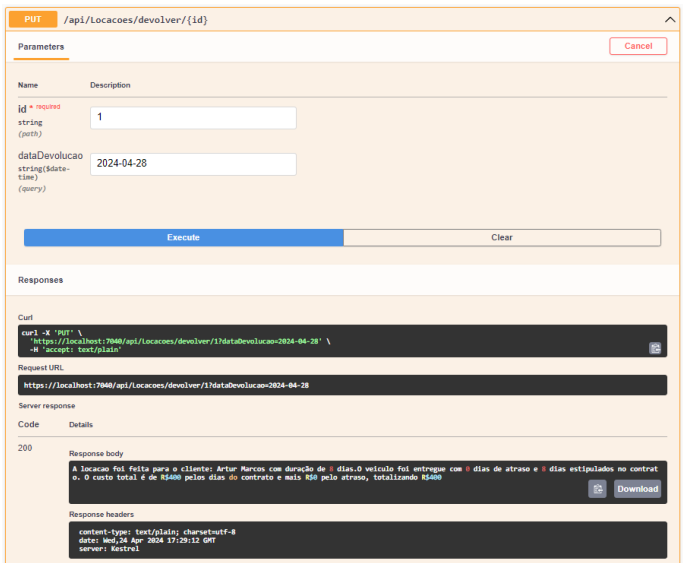


Figure 35. Devolvendo veiculo e concluindo a locação.

DELETE (api/Locacoes/id)

DELETE /api/Locacoes/{id}

Parameters

Name	Description
id * required string (path)	1

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'https://localhost:7940/api/Locacoes/1' \
  -H 'accept: */*'
```

Request URL

https://localhost:7940/api/Locacoes/1

Server response

Code Details

400
Undocumented

Error: response status is 400

Response body

A locação está ativa e portanto não pode ser apagada.

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 17:28:07 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	Success	No links

Figure 36. Tentando apagar a locação 1.

DELETE /api/Locacoes/{id}

Parameters

Name	Description
id * required string (path)	1

Execute Clear

Responses

Curl

```
curl -X 'DELETE' \
  'https://localhost:7940/api/Locacoes/1' \
  -H 'accept: */*'
```

Request URL

https://localhost:7940/api/Locacoes/1

Server response

Code Details

200

Response body

A locação foi apagada com sucesso.

Response headers

```
content-type: text/plain; charset=utf-8
date: Wed, 24 Apr 2024 17:31:49 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	Success	No links

Figure 37. Apagando a locação 1.

10. Resultados e Conclusões

Segue a análise dos resultados obtidos com o desenvolvimento do sistema:

- O sistema foi bem executado, atendendo a todos os requisitos especificados.
- Sua eficácia é notável devido à implementação de diversas verificações e validações. Entre elas:
 - Verificação da validade do CPF, telefone, e-mail e nome dos clientes, com formatação adequada do telefone e CPF para armazenamento no banco de dados sem símbolos.
 - Garantia de que o CPF do cliente informado é o mesmo do cliente a ser modificado, evitando alterações em clientes distintos.
 - Verificação da existência do cliente para evitar dados duplicados e erros.
 - Ao excluir um cliente, o sistema verifica se ele possui locações ativas, impedindo sua exclusão para evitar inadimplência ou a não devolução do veículo.
 - Ao excluir um cliente, todas as locações relacionadas são removidas para evitar inconsistências nos dados.
 - Verificação da existência do veículo para evitar dados duplicados e erros.
 - Verificação da validade da placa e comparação das placas informadas em uma atualização para evitar modificações em veículos diferentes.
 - Verificação da existência da locação para evitar dados duplicados e erros.
 - Verificação da disponibilidade do veículo e da existência do cliente antes de registrar a locação.
 - Verificação do formato e da ordem cronológica das datas.
 - Ao excluir uma locação, o sistema verifica se ela está ativa, impedindo sua exclusão para evitar dados inconsistentes.
 - Fornecimento de mensagens informativas detalhadas sobre problemas nas requisições, orientando sobre o formato correto das datas e fornecendo informações relevantes sobre erros e ações bem-sucedidas.

Portanto, o sistema demonstra conformidade com os requisitos estabelecidos e eficácia em atender às necessidades da locadora de veículos.

11. Considerações Finais

Durante o desenvolvimento, identificamos alguns pontos que poderiam gerar melhorias para uma possível atualização na aplicação. Entre eles:

- Criação de um front-end para a aplicação, transformando-a em algo mais realístico.
- Adicionar uma restrição no acesso e consequentemente na modificação de certas propriedades, como o caso da "DataDevolução" que só deve ser modificada no momento que o usuário devolve o veículo (PUT), mas está disponível durante a criação da locação (POST). A alternativa escolhida foi sobrescrever os dados para manter a integridade e avisar o usuário quais dados foram modificados.
- Implementação de um ID automático para as locações que fosse "regenerativo", ou seja, caso alguma locação fosse apagada, esse ID que também foi deletado fosse utilizado. Seria diferente do autoincremental, que deixa espaços vazios entre os IDs, uma solução que foi testada, mas descartada do código.

12. Apêndice

12.1. Links úteis

- Repositório do GitHub: <https://github.com/PSG-TADS/psg-tads-2024-1-back-bd-arttturslv/>