

Atividade Prática Serviços Web

Car Wash API

Vinicius Carpes de Freitas.

Florianópolis, 11 de junho de 2025.

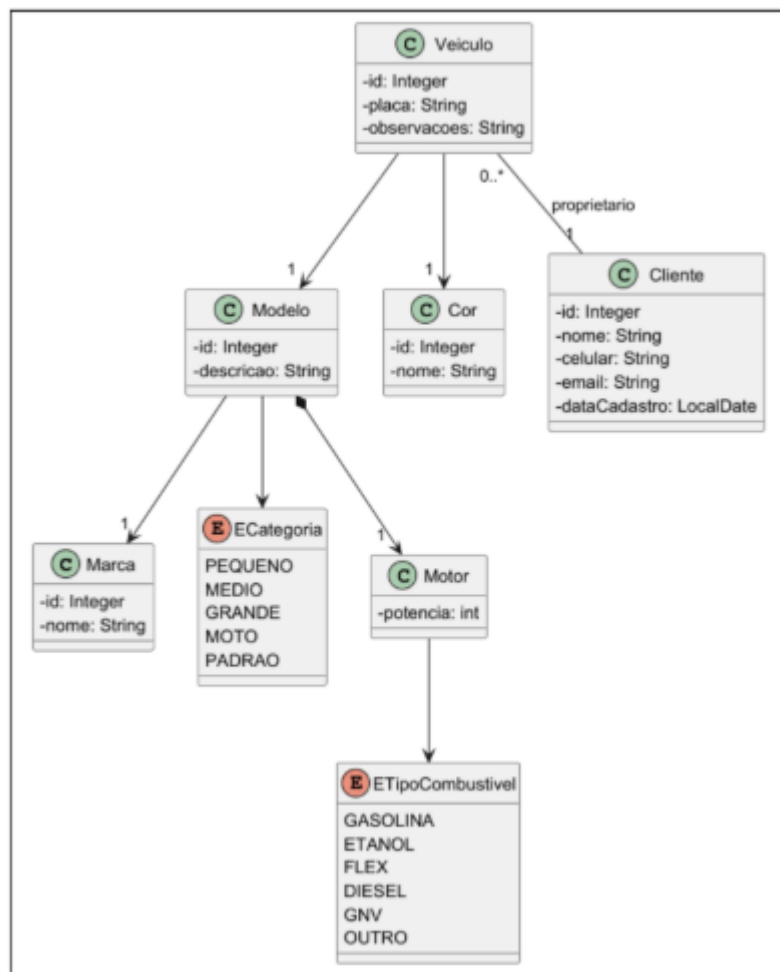
1. Introdução

Este trabalho tem como objetivo exemplificar e explicar o funcionamento da API para o sistema de gerenciamento de uma empresa de lavação de automóveis - Car Wash - conforme proposto pelo Professor Dr. Marcos Pisching.

A gravação com explicação pode ser acessada [aqui](#).

O desenvolvimento da API foi realizado conforme as seguintes instruções:

- a) Utilizar Java e o ecossistema Spring para o desenvolvimento da aplicação.
- b) Gerenciar as dependências de projeto com Maven
- c) Conectar com banco de dados (preferencialmente MySQL)
- d) Possuir ao menos as seguintes camadas: Model, Controller e Repository.
- e) Criar endpoints para operações CRUD para as entidades Marca, Cliente, Veículo e Modelo.
- f) Construir o domínio da aplicação com base no seguinte diagrama:



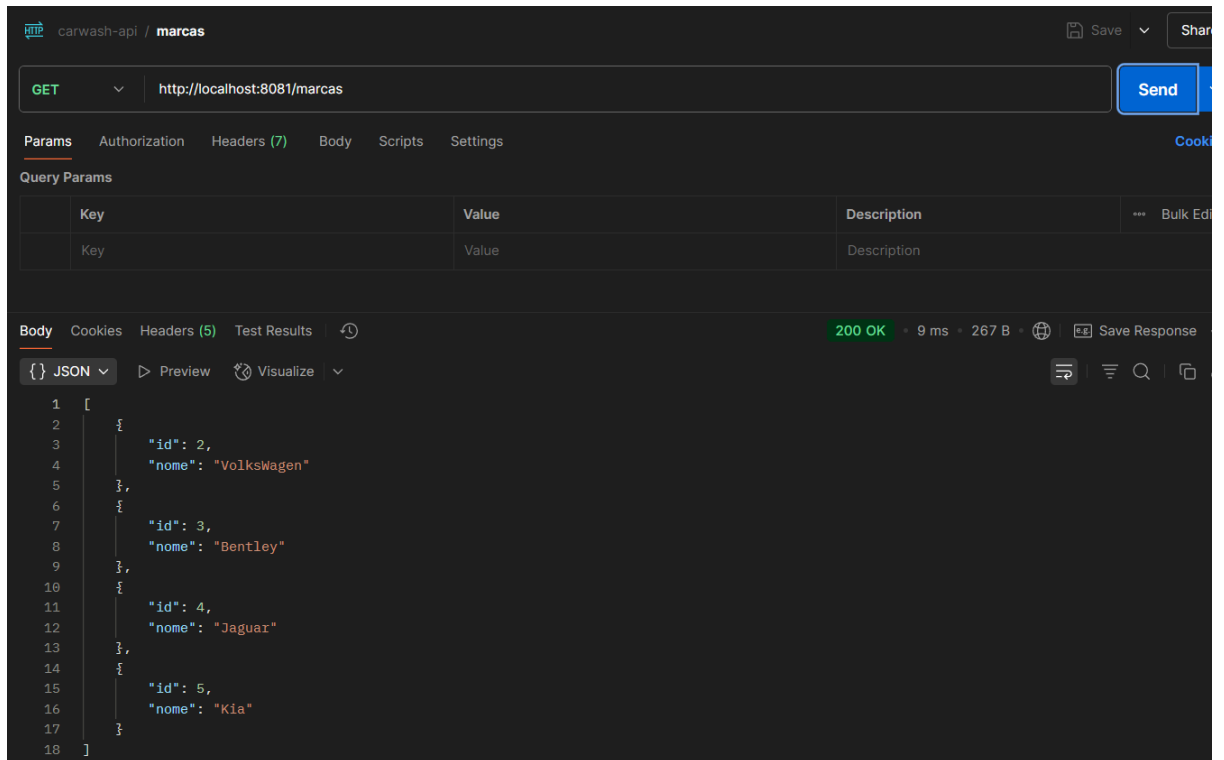
Fonte: Prof. Dr. Marcos Phishing

2. Endpoints

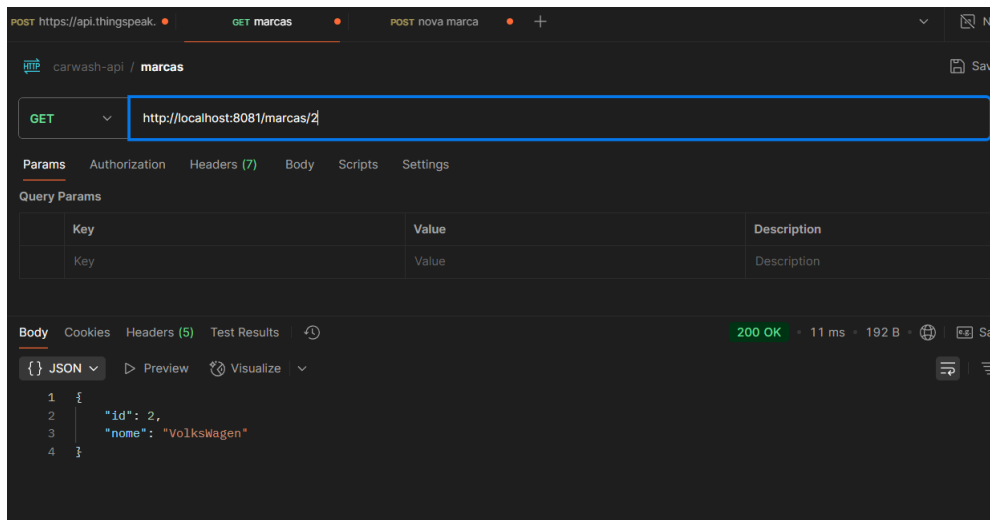
2.1 Marca

As requisições HTTP para CRUD de marca deverão ser feitas a partir da URL raiz <http://localhost:8081/marcas>.

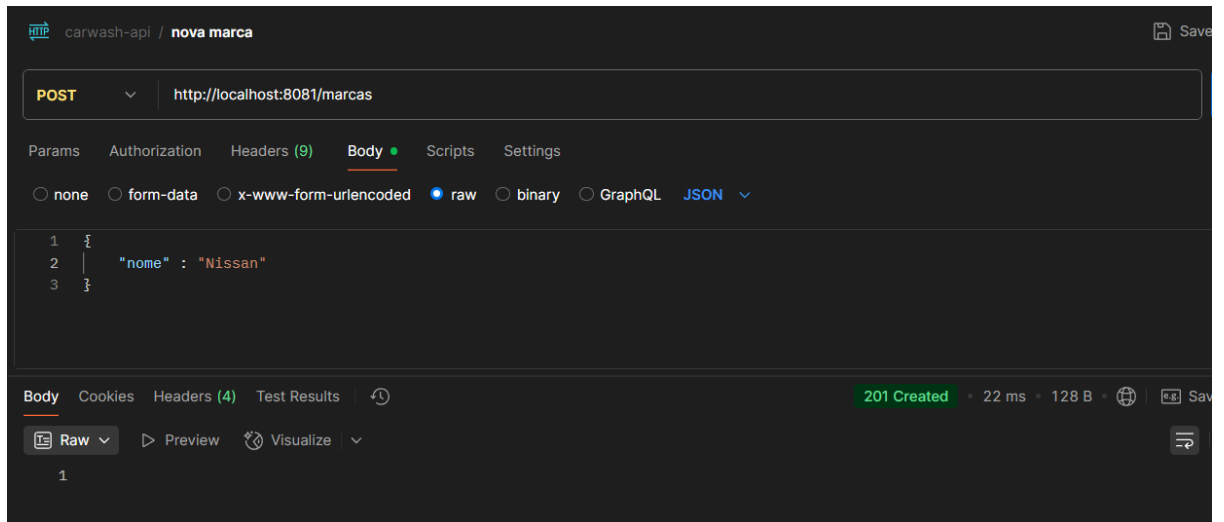
- a) GET <http://localhost:8081/marcas> - retorna todas as marcas.



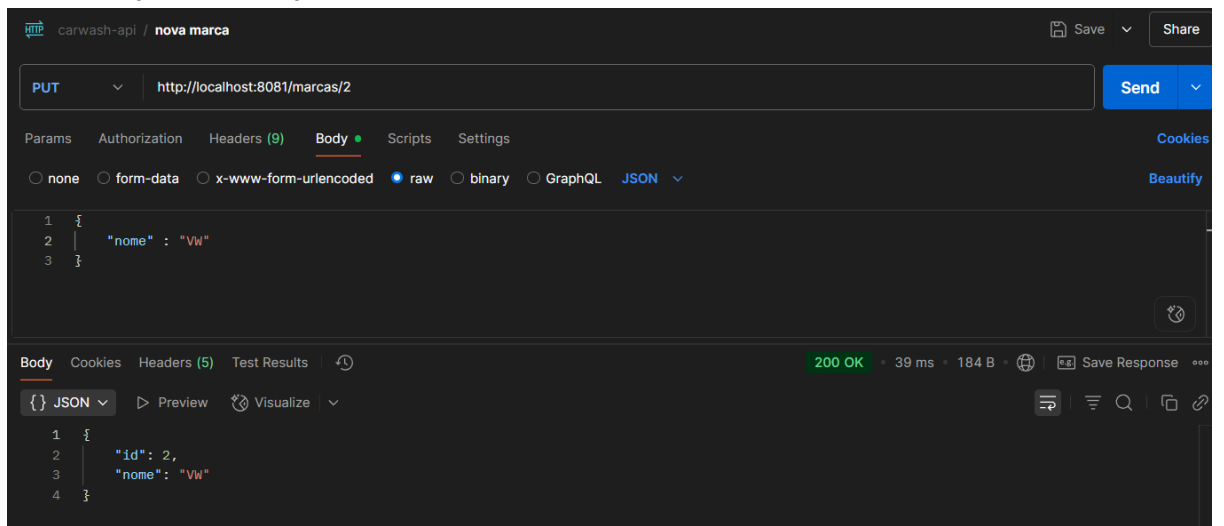
b) GET <http://localhost:8081/marcas/{id}> - retorna a marca conforme o id.



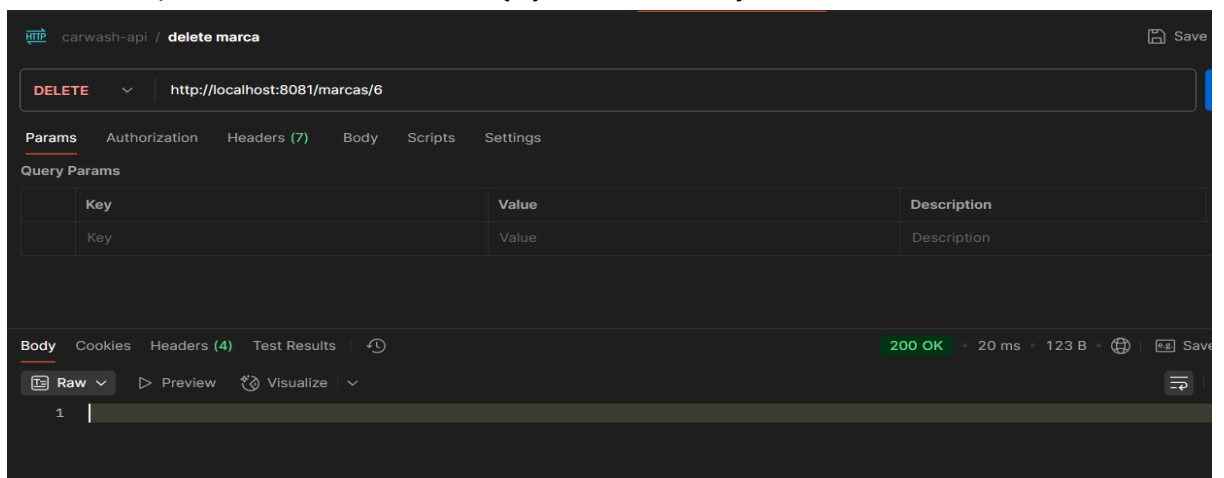
c) POST <http://localhost:8081/marcas> - cria um objeto Marca.
Required: json com objeto de Marca.



- d) PUT <http://localhost:8081/marcas/{id}> - atualiza um objeto Marca conforme o id.
Required: json com objeto de Marca.



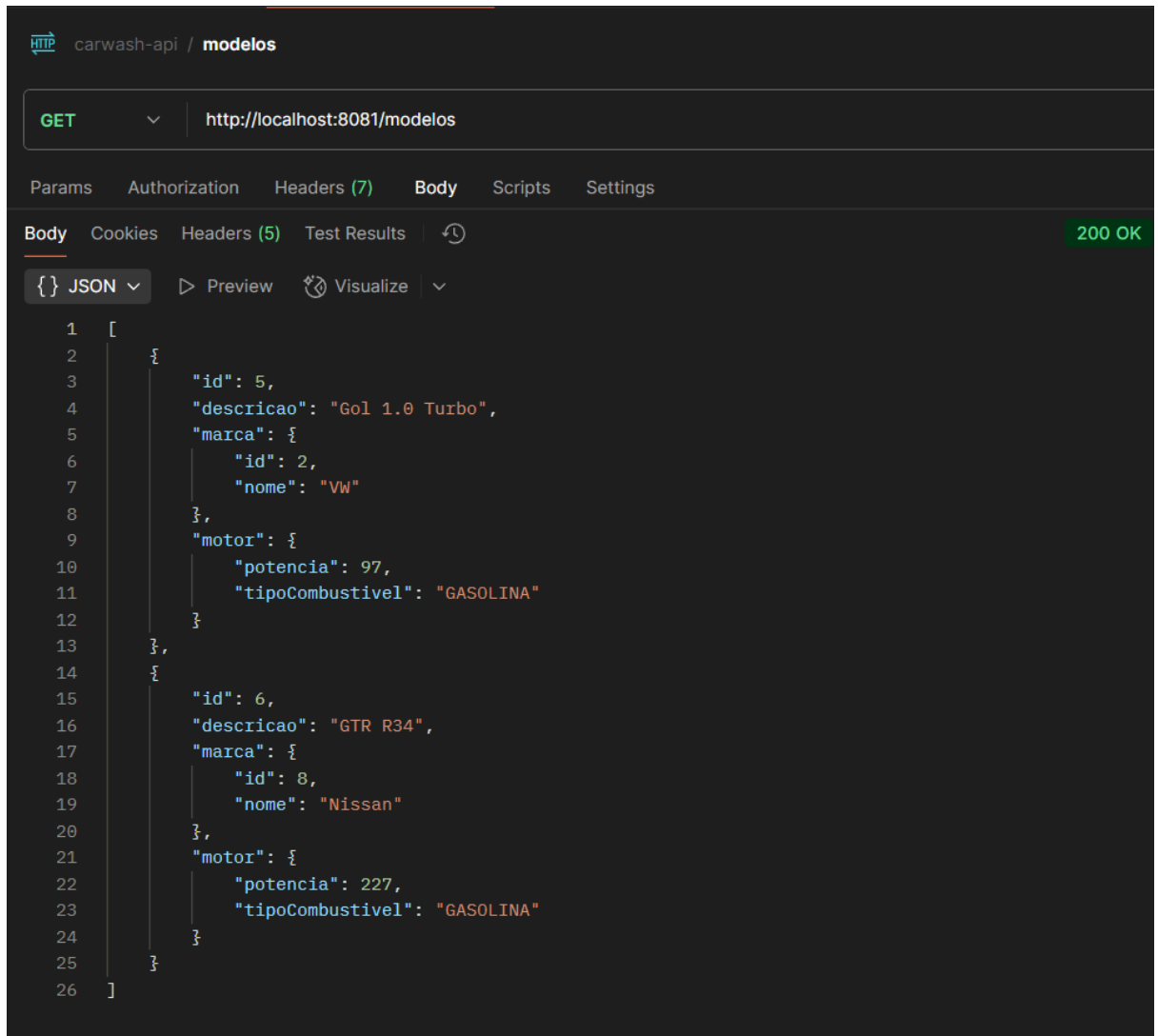
- e) DELETE <http://localhost:8081/marcas/{id}> - deleta um objeto Marca conforme o id.



2.2 Modelo

As requisições HTTP para CRUD de modelos deverão ser feitas a partir da URL raiz <http://localhost:8081/modelos>.

- a) GET <http://localhost:8081/modelos> - retorna todos os modelos.



```
HTTP carwash-api / modelos

GET http://localhost:8081/modelos

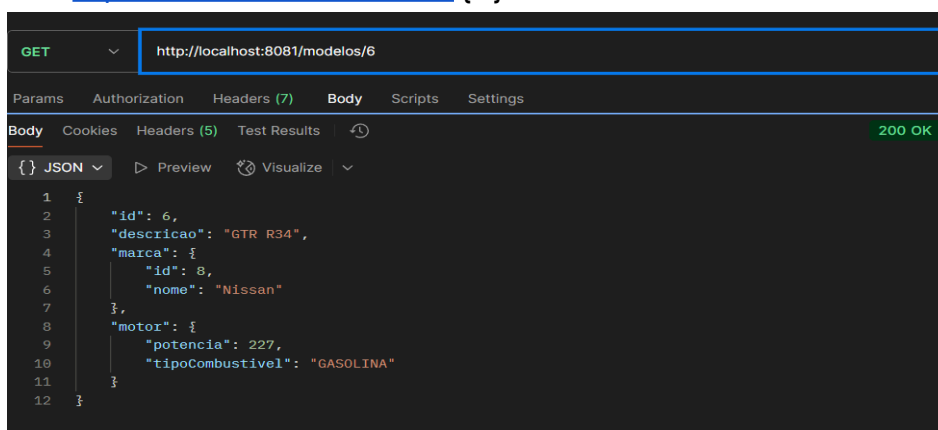
Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results 200 OK

JSON Preview Visualize

1 [
2   {
3     "id": 5,
4     "descricao": "Gol 1.0 Turbo",
5     "marca": {
6       "id": 2,
7       "nome": "VW"
8     },
9     "motor": {
10      "potencia": 97,
11      "tipoCombustivel": "GASOLINA"
12    }
13  },
14  {
15    "id": 6,
16    "descricao": "GTR R34",
17    "marca": {
18      "id": 8,
19      "nome": "Nissan"
20    },
21    "motor": {
22      "potencia": 227,
23      "tipoCombustivel": "GASOLINA"
24    }
25  }
26 ]
```

- b) GET <http://localhost:8081/modelos/{id}> - retorna o modelo conforme id.



```
GET http://localhost:8081/modelos/6

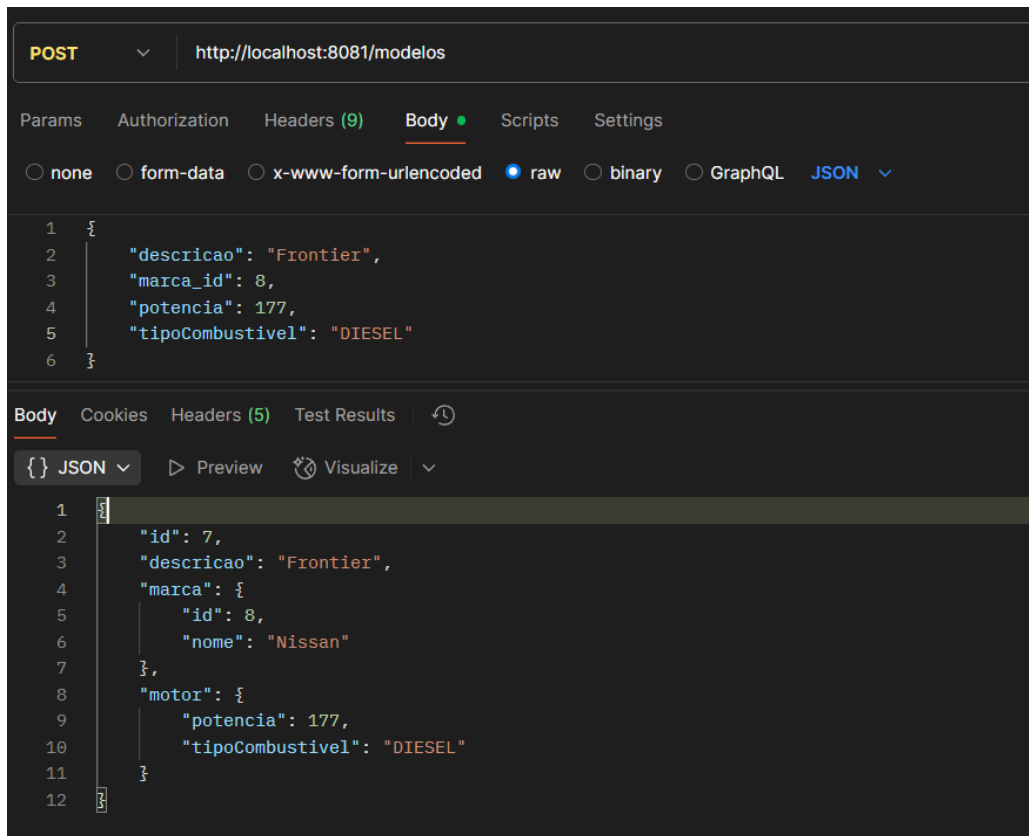
Params Authorization Headers (7) Body Scripts Settings

Body Cookies Headers (5) Test Results 200 OK

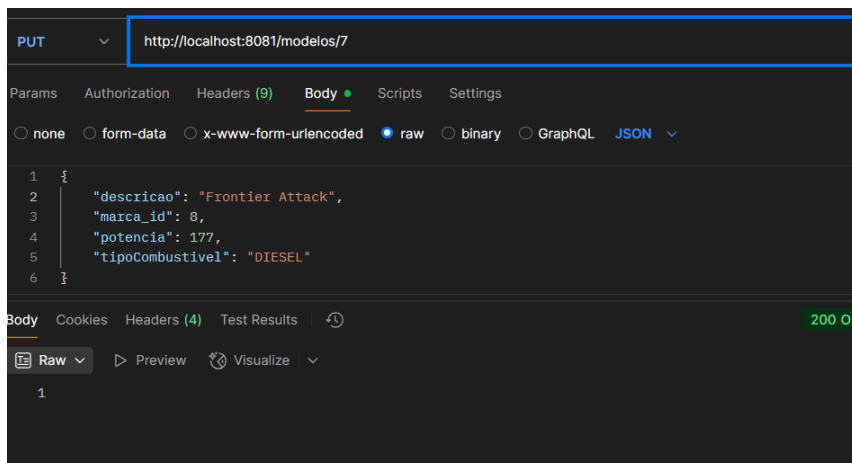
JSON Preview Visualize

1 {
2   "id": 6,
3   "descricao": "GTR R34",
4   "marca": {
5     "id": 8,
6     "nome": "Nissan"
7   },
8   "motor": {
9     "potencia": 227,
10    "tipoCombustivel": "GASOLINA"
11  }
12 }
```

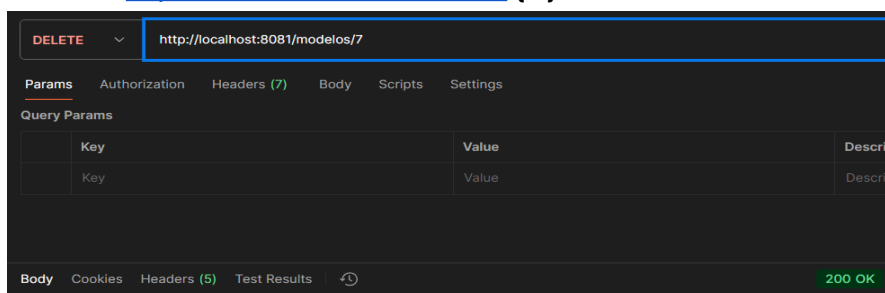
- c) POST <http://localhost:8081/modelos> - cria um novo modelo.
Required: Deve enviar no corpo descricao, marca_id, potencia e tipoCombustivel.



- d) PUT <http://localhost:8081/modelos/{id}> -atualiza modelo conforme id.
Required: Mesmos parâmetros que o item c.



- e) DELETE <http://localhost:8081/modelos/{id}> - deleta modelo conforme id.



2.3 Cliente

As requisições HTTP para CRUD de clientes deverão ser feitas a partir da URL raiz <http://localhost:8081/clientes>.

- a) GET <http://localhost:8081/clientes> - retorna todos os clientes.

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

- Method: GET
- URL: <http://localhost:8081/clientes>
- Status: 200 OK
- Response Time: 34 ms
- Response Size: 948 B
- Body Type: JSON

The JSON response is as follows:

```
{  "id": 1,  "nome": "Vinicius",  "celular": 4884848480,  "email": "vinicdf@gmail.com",  "dataCadastro": "2025-06-10",  "veiculos": [    {      "id": 3,      "placa": "AJK88P4",      "cor": "Vermelho",      "modelo": "Gol 1.0 Turbo",      "cliente": {        "id": 1,        "nome": "Vinicius",        "celular": 4884848480,        "email": "vinicdf@gmail.com",        "dataCadastro": "2025-06-10"      }    },    {      "id": 4,      "placa": "AJK88P4",      "cor": "Vermelho",      "modelo": "GTR R34",      "cliente": {        "id": 1,        "nome": "Vinicius",        "celular": 4884848480,        "email": "vinicdf@gmail.com",        "dataCadastro": "2025-06-10"      }    }  ]}
```

- b) GET <http://localhost:8081/clientes/{id}> - retorna o cliente conforme id.

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

- Method: GET
- URL: <http://localhost:8081/clientes/1>
- Status: 200 OK
- Body Type: JSON

The JSON response is as follows:

```
{  "id": 1,  "nome": "Vinicius",  "celular": 4884848480,  "email": "vinicdf@gmail.com",  "dataCadastro": "2025-06-10",  "veiculos": [  ]}
```

- c) POST <http://localhost:8081/clientes> - criar um novo cliente.

Required: deve informar nome, celular e email no json.

POST <http://localhost:8081/clientes>

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "nome": "Marcos Pisching",
3   "celular": "4812345678",
4   "email": "mp@gmail.com"
5 }
```

Body Cookies Headers (5) Test Results **201 Created**

{} JSON ▾ ▶ Preview 🔗 Visualize ▾

```
1 {
2   "id": 3,
3   "nome": "Marcos Pisching",
4   "celular": "4812345678",
5   "email": "mp@gmail.com",
6   "dataCadastro": "2025-06-11",
7   "veiculos": []
8 }
```

d) PUT <http://localhost:8081/clientes/{id}> - atualiza o cliente conforme id.

PUT <http://localhost:8081/clientes/3>

Params Authorization Headers (9) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "nome": "Marcos André Pisching",
3   "celular": "4812345678",
4   "email": "mp@gmail.com"
5 }
```

Body Cookies Headers (4) Test Results **200 OK**

Raw ▾ ▶ Preview 🔗 Visualize ▾

```
1
```

e) DELETE <http://localhost:8081/clientes/{id}> - deleta o cliente conforme o id.

DELETE ▾ <http://localhost:8081/clientes/3>

Params Authorization Headers (7) **Body** Scripts Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (4) Test Results **200 OK**

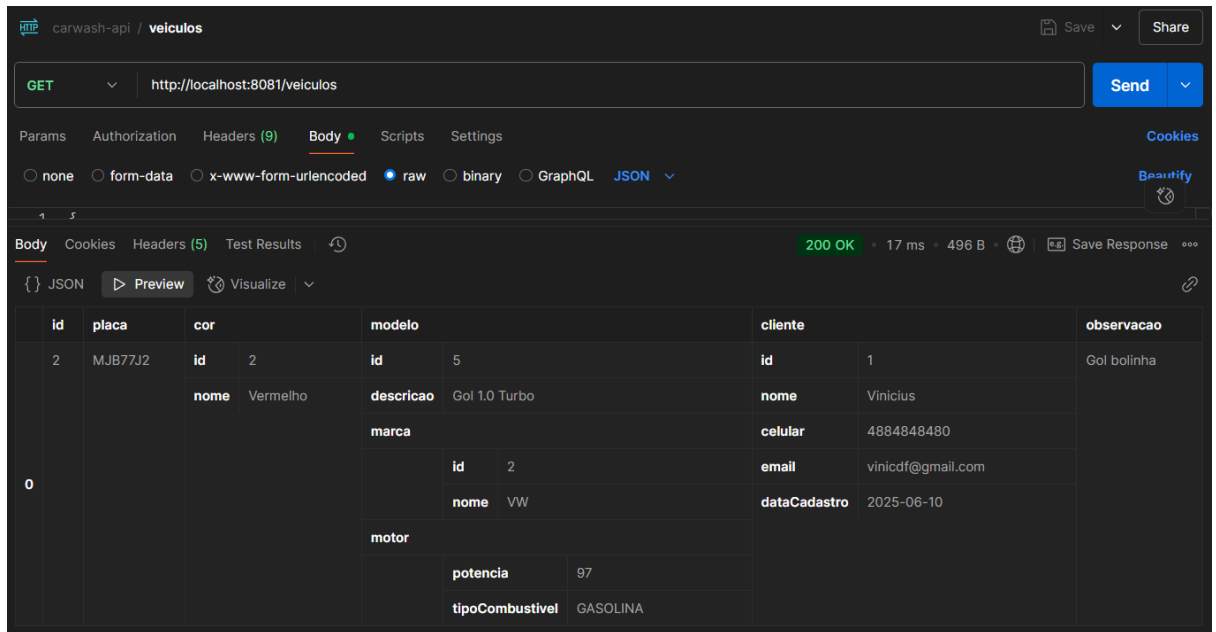
Raw ▾ ▶ Preview 🔗 Visualize ▾

```
1
```

2.4 Veículo

As requisições HTTP para CRUD de veículos deverão ser feitas a partir da URL raiz <http://localhost:8081/veiculos>.

- f) GET <http://localhost:8081/veiculos> - retorna todos os veículos.

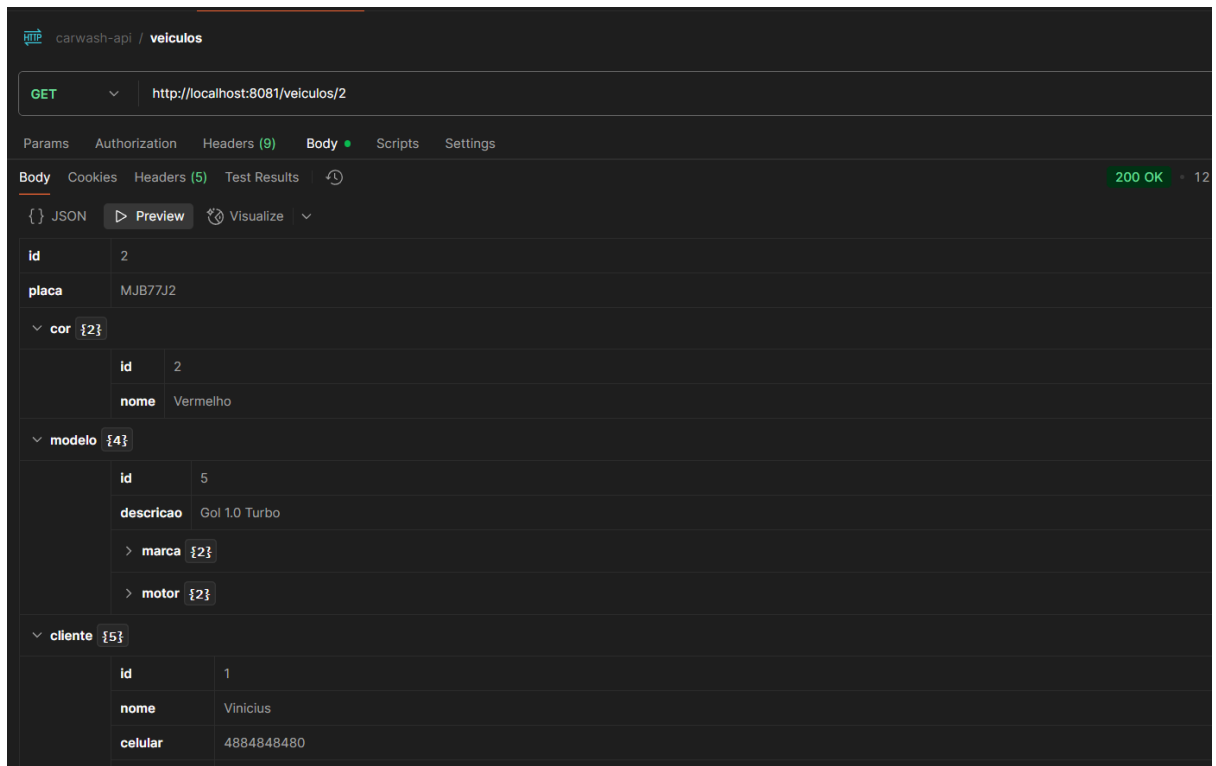


The screenshot shows a REST client interface with the following details:

- Method: GET
- URL: <http://localhost:8081/veiculos>
- Status: 200 OK
- Response Type: JSON
- Response Body (JSON):

```
[{"id": 2, "placa": "MJB77J2", "cor": {"id": 2, "nome": "Vermelho"}, "modelo": {"id": 5, "descricao": "Gol 1.0 Turbo", "marca": {"id": 2, "nome": "VW"}, "motor": {"potencia": 97, "tipoCombustivel": "GASOLINA"}}, "cliente": {"id": 1, "nome": "Vinicius", "celular": "4884848480", "email": "vinicdf@gmail.com", "dataCadastro": "2025-06-10"}, "observacao": "Gol bolinha"}]
```

- g) GET <http://localhost:8081/veiculos/{id}> - retorna o veículo dado o id.

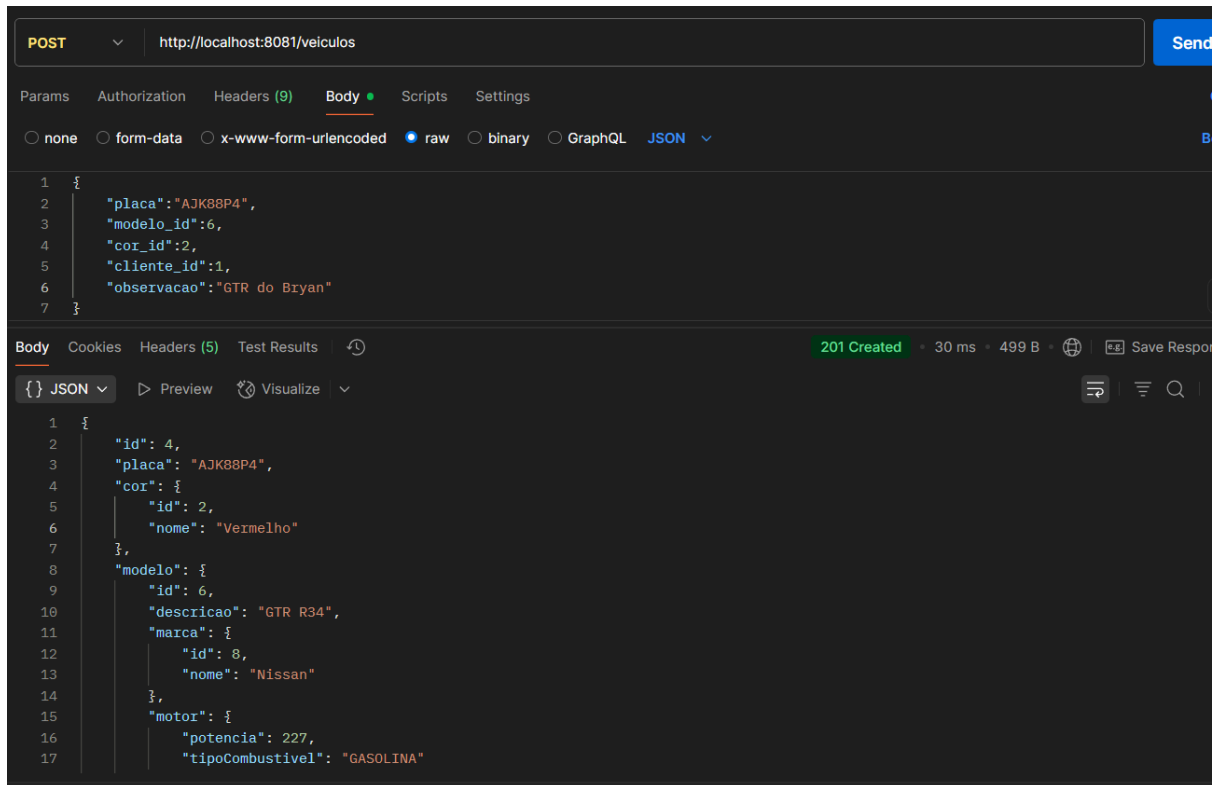


The screenshot shows a REST client interface with the following details:

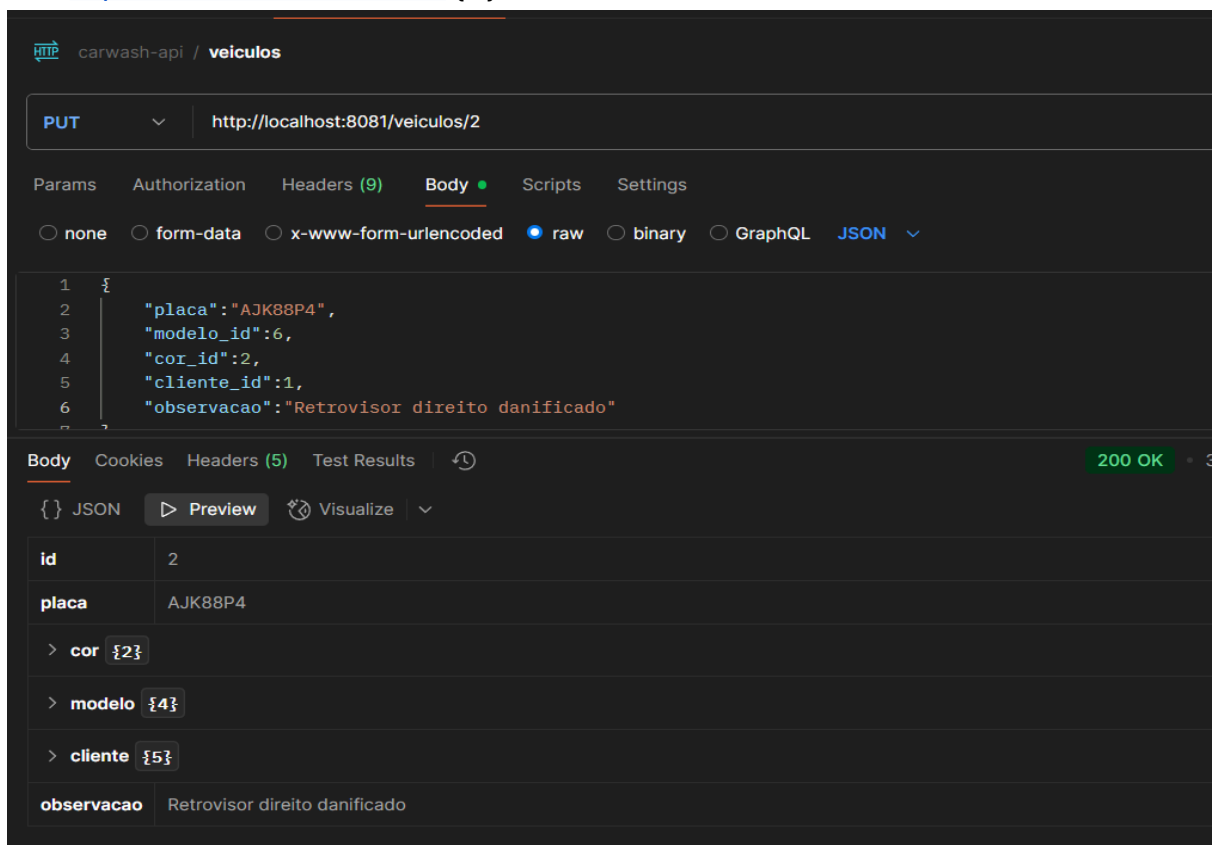
- Method: GET
- URL: <http://localhost:8081/veiculos/2>
- Status: 200 OK
- Response Type: JSON
- Response Body (JSON):

```
{"id": 2, "placa": "MJB77J2", "cor": {"id": 2, "nome": "Vermelho"}, "modelo": {"id": 5, "descricao": "Gol 1.0 Turbo", "marca": {"id": 2, "nome": "VW"}, "motor": {"potencia": 97, "tipoCombustivel": "GASOLINA"}}, "cliente": {"id": 1, "nome": "Vinicius", "celular": "4884848480", "email": "vinicdf@gmail.com", "dataCadastro": "2025-06-10"}, "observacao": "Gol bolinha"}
```

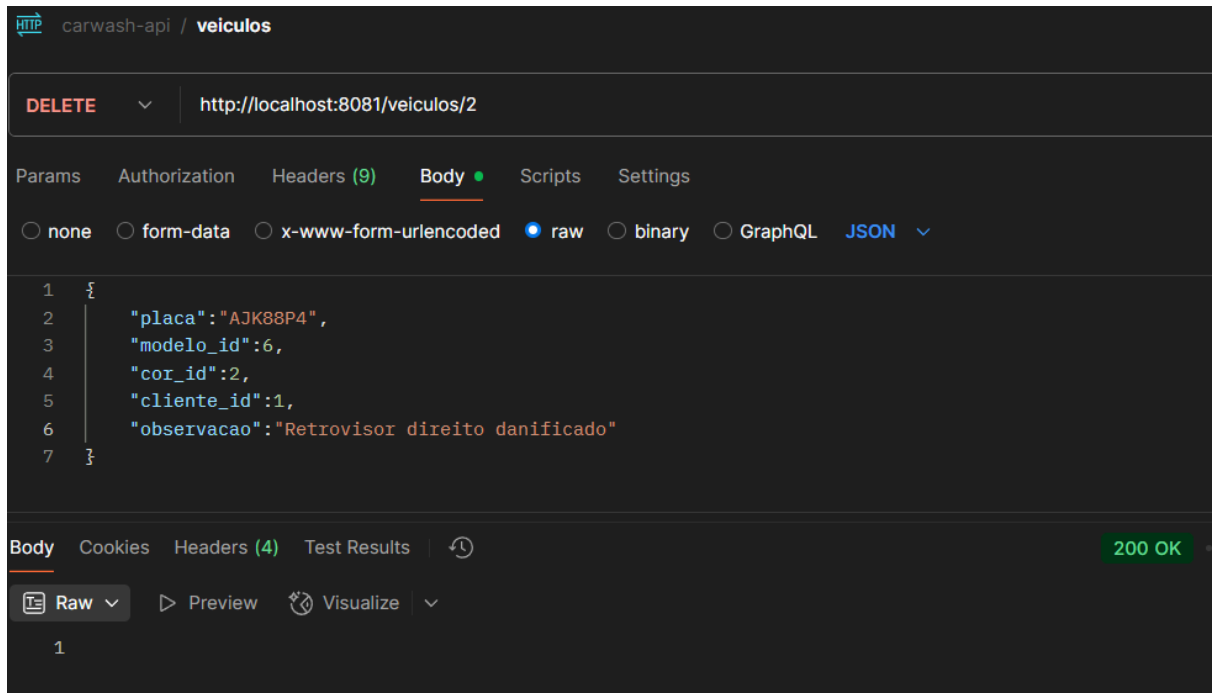
- h) POST <http://localhost:8081/veiculos> - cria um novo veículos.
Required: Deve enviar um body com VeiculoDTO, informando placa, modelo_id, cor_id, cliente_id e observacao;



- i) PUT <http://localhost:8081/veiculos/{id}> -atualiza o veículo conforme id.



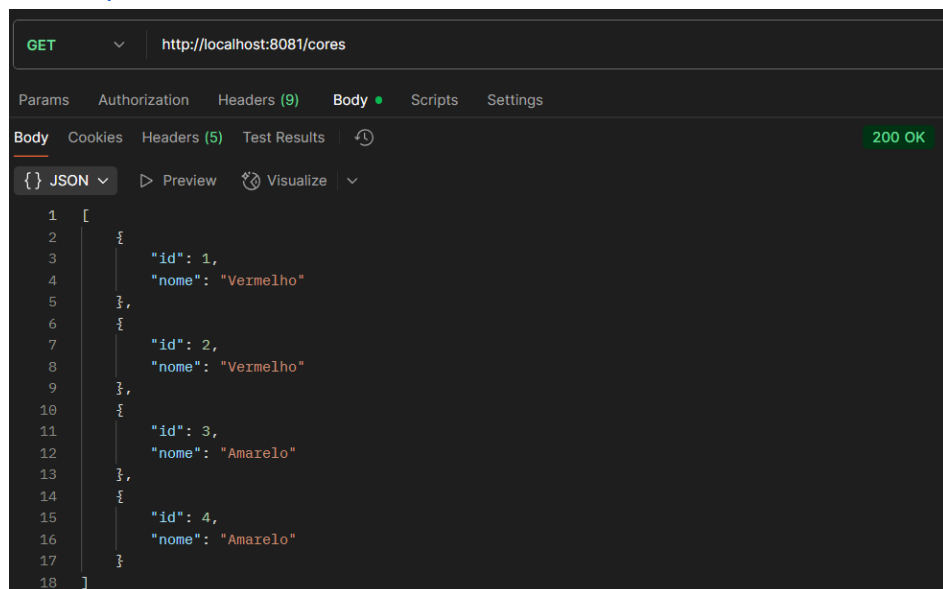
- j) DELETE <http://localhost:8081/modelos/{id}> - deleta o veículo conforme o id.



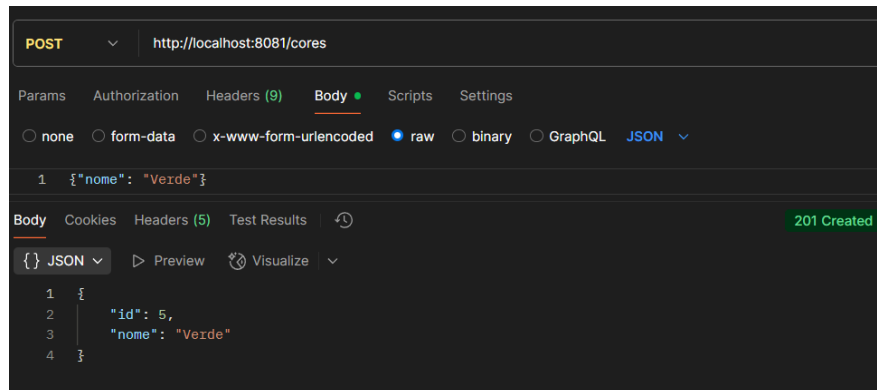
2.5 Extra: Cor

As requisições HTTP para CRUD de cores deverão ser feitas a partir da URL raiz <http://localhost:8081/cores>.

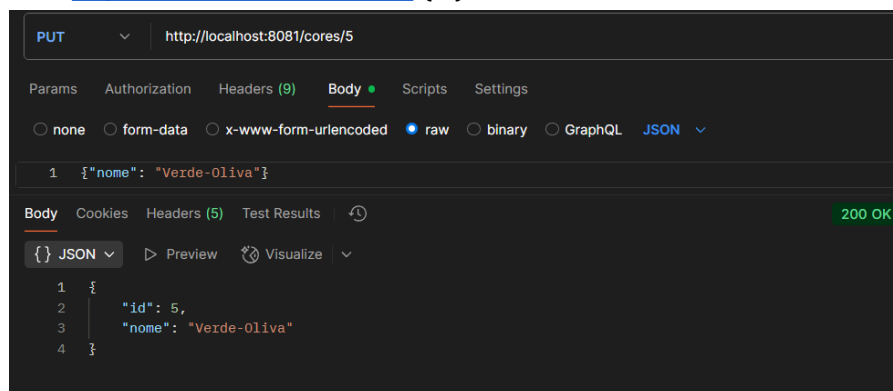
- a) GET <http://localhost:8081/cores> - retorna todas as cores.



- b) POST <http://localhost:8081/cores>- cria uma nova cor.
Required: Requer campo nome



- c) PUT <http://localhost:8081/cores/{id}>- atualiza a cor conforme id.



- d) DELETE <http://localhost:8081/cores/{id}>- deleta a cor conforme id.

