

DESENVOLVIMENTO WEB I

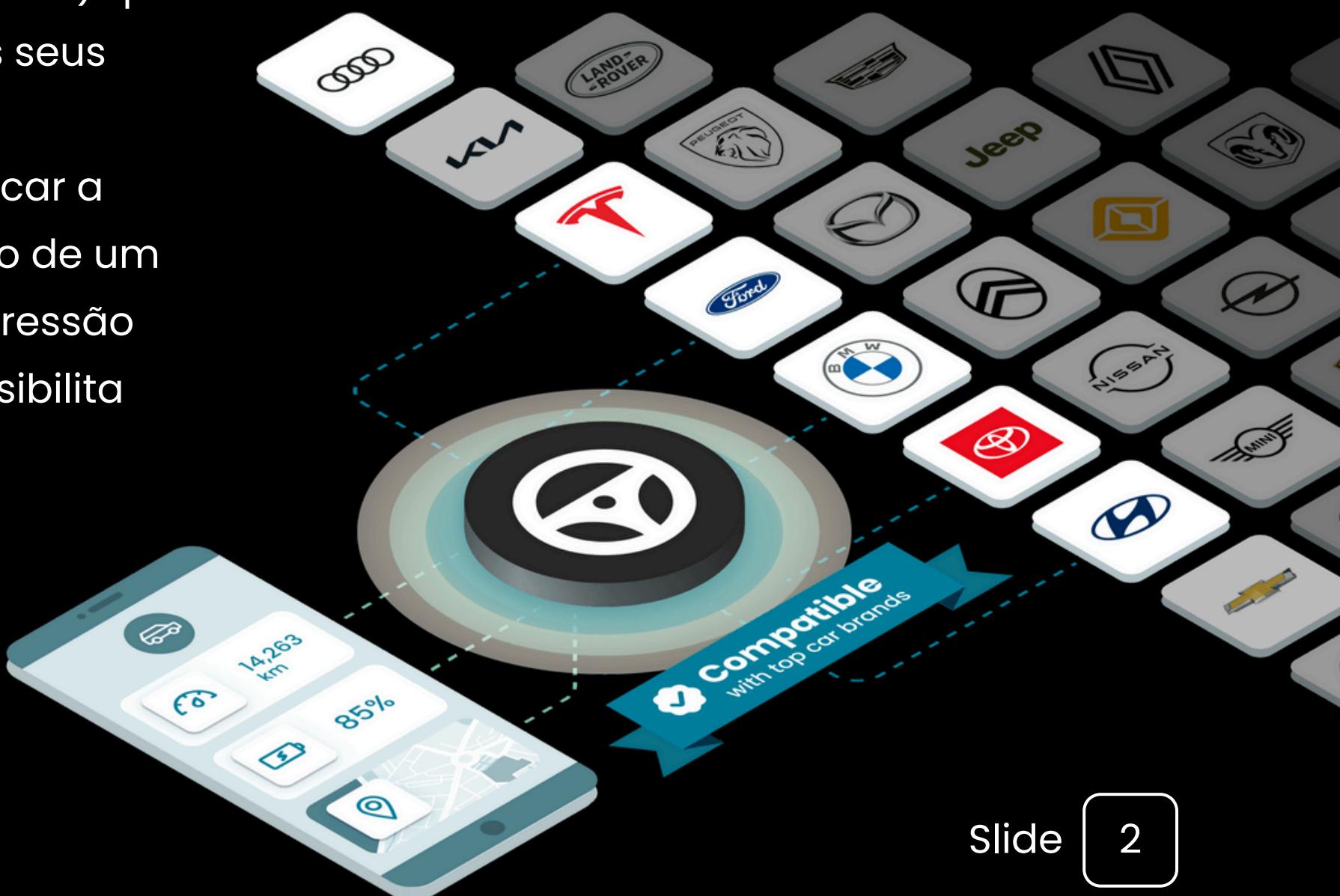
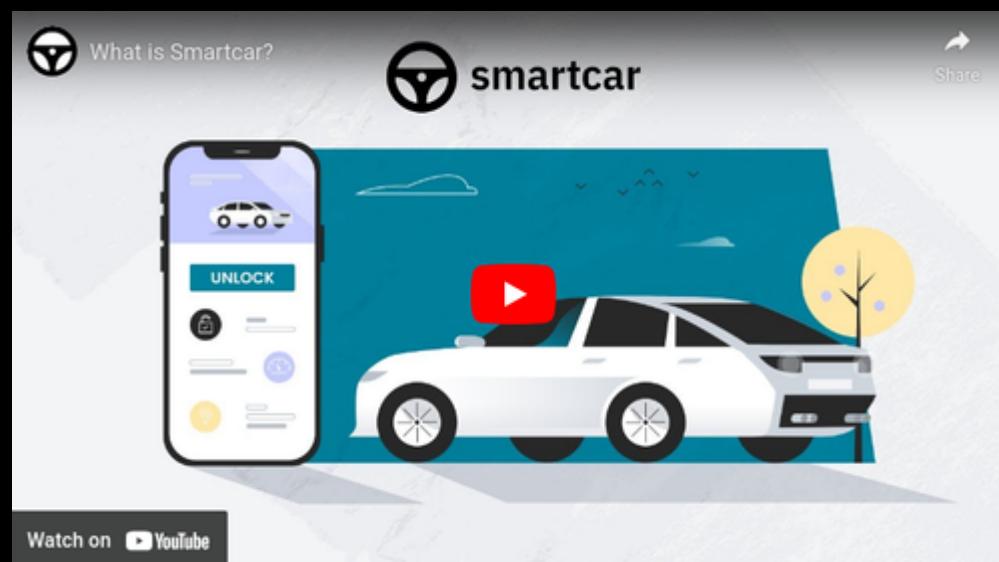
# SMARTCAR



# SMARTCAR

A plataforma Smartcar é uma API (Application programming interface) que permite às empresas e clientes conectarem-se de forma fácil aos seus veículos.

Caso estes precisem de rastrear a localização de um veículo, verificar a quilometragem, partilhar uma chave virtual, iniciar o carregamento de um veículo elétrico, monitorizar o nível de combustível, inspecionar a pressão dos pneus ou verificar a vida útil do óleo do motor, a Smartcar possibilita tudo isto com um único pedido.



# FUNCIONAMENTO

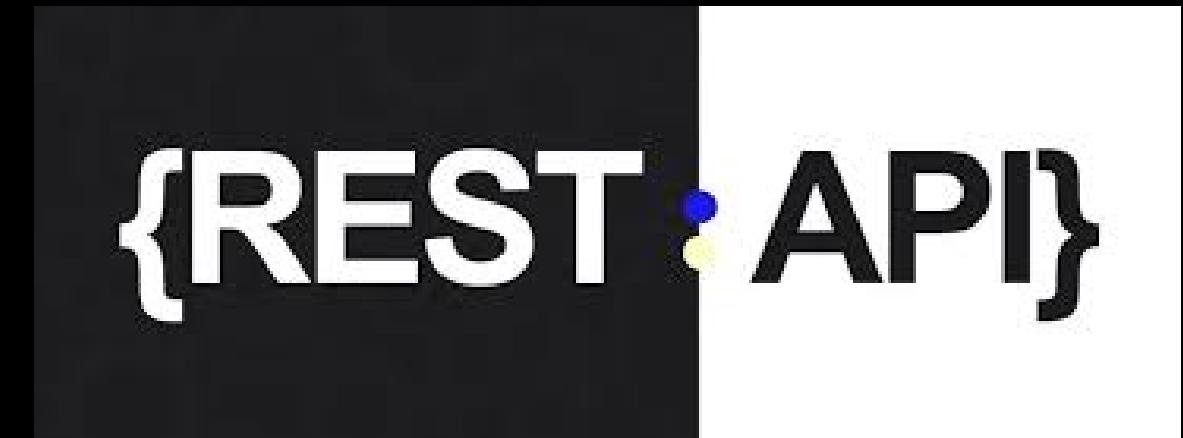
A Smartcar funciona através de uma interface RESTful, utilizando chamadas HTTP para acessar e modificar informações do veículo.

O processo envolve os seguintes passos:

**Autenticação OAuth2:** Para aceder à API de um veículo, o utilizador deve primeiro autorizar a aplicação via OAuth2. Isto envolve redirecionar o utilizador para uma página de login do fabricante do veículo.

**Pedidos HTTP:** Após a autorização, a aplicação pode fazer chamadas HTTP com os tokens apropriados para consultar dados do veículo ou acionar comandos.

**Resposta:** A API retorna respostas JSON contendo os dados requisitados ou a confirmação de comandos executados.



# AUTENTIFICAÇÃO

**Token Name**

Nome do token que será gerado. Serve para identificar o token. Neste caso, "SmartCarAuthToken" é o nome dado.

**Callback URL**

URL para onde o utilizador será redirecionado após o processo de autenticação.

**Authorize using browser**

Esta opção, se marcada, indica que o fluxo de autenticação será feito diretamente através do navegador.

**Auth URL**

URL onde o pedido de autenticação será iniciado. Este URL redireciona o usuário para a página de login, onde ele pode autorizar o app. Neste caso, o URL da SmartCar.

**Access Token URL:**

URL onde o app pode trocar o código de autorização por um token de acesso. Após o utilizador autorizar, a aplicação envia um pedido a esta URL para obter o token.

**Client ID**

Identificador exclusivo da aplicação fornecido pela SmartCar ao registrar o app. Este ID é usado para identificar a aplicação durante o processo de autenticação.

**Client Secret**

Uma chave secreta usada em conjunto com o Client ID para autenticar a aplicação de forma segura. Deve ser mantido em segredo e nunca compartilhado publicamente.

**Client Authentication:**

Método de autenticação do cliente. Neste caso, a aplicação está configurada para enviar o Client ID e o Client Secret como um cabeçalho de autenticação básica (Basic Auth header).

**Auth Type**

O tipo de autenticação a ser utilizado. Neste caso, "OAuth 2.0"

**Refresh Token URL:**

Este é o URL onde o aplicativo pode enviar um pedido para renovar ou obter um novo token de acesso quando o token atual expirar. O "refresh token" é usado para evitar que o utilizador precise repetir o processo de autenticação sempre que o token de acesso expira.

# CONFIGURATION - SMARTCAR

**CLIENT ID**

248c5c93-68b1-4aa3-985e-7bca2cc5ac97

( DIFERENTE - Conta Exemplo )

**CLIENT SECRET**

ⓘ Please store your client secret in a safe place. For security purposes, we will not show it again.

48b7f16f-aa69-4742-9e01-bfa920e98f11

 Copy  Regenerate

( DIFERENTE - Conta Exemplo )

**MANAGEMENT API TOKEN**

19c0e987-face-42bb-9387-6f356870a5c8

 Copy  Regenerate

( DIFERENTE - Conta Exemplo )

**REDIRECT URIS**

<https://oauth.pstmn.io/v1/callback>

 Copy  Delete

( IGUAL - Padrão do Postman )

# CONFIGURATION - POSTMAN



Auth Type

OAuth 2.0

( DIFERENTE - Consoante Tipo de Authetificação )

Token

SmartCarAuthToken

19c0e987-face-42bb-9387-  
6f356870a5c8

( DIFERENTE - Cada Conta na Api SmartCar tem um )

Token Name

SmartCarAuthToken

( ESCOLHA - Cada Conta na Api SmartCar tem um )

Client ID ⓘ

Client ID

( DIFERENTE - Cada Conta na Api SmartCar tem um )

Client Secret ⓘ

Client Secret

( DIFERENTE - Cada Conta na Api SmartCar tem um )

Refresh Token URL ⓘ

<https://auth.smartcar.com/oauth/tok...>

( IGUAL - Padrão dA SmartCar )

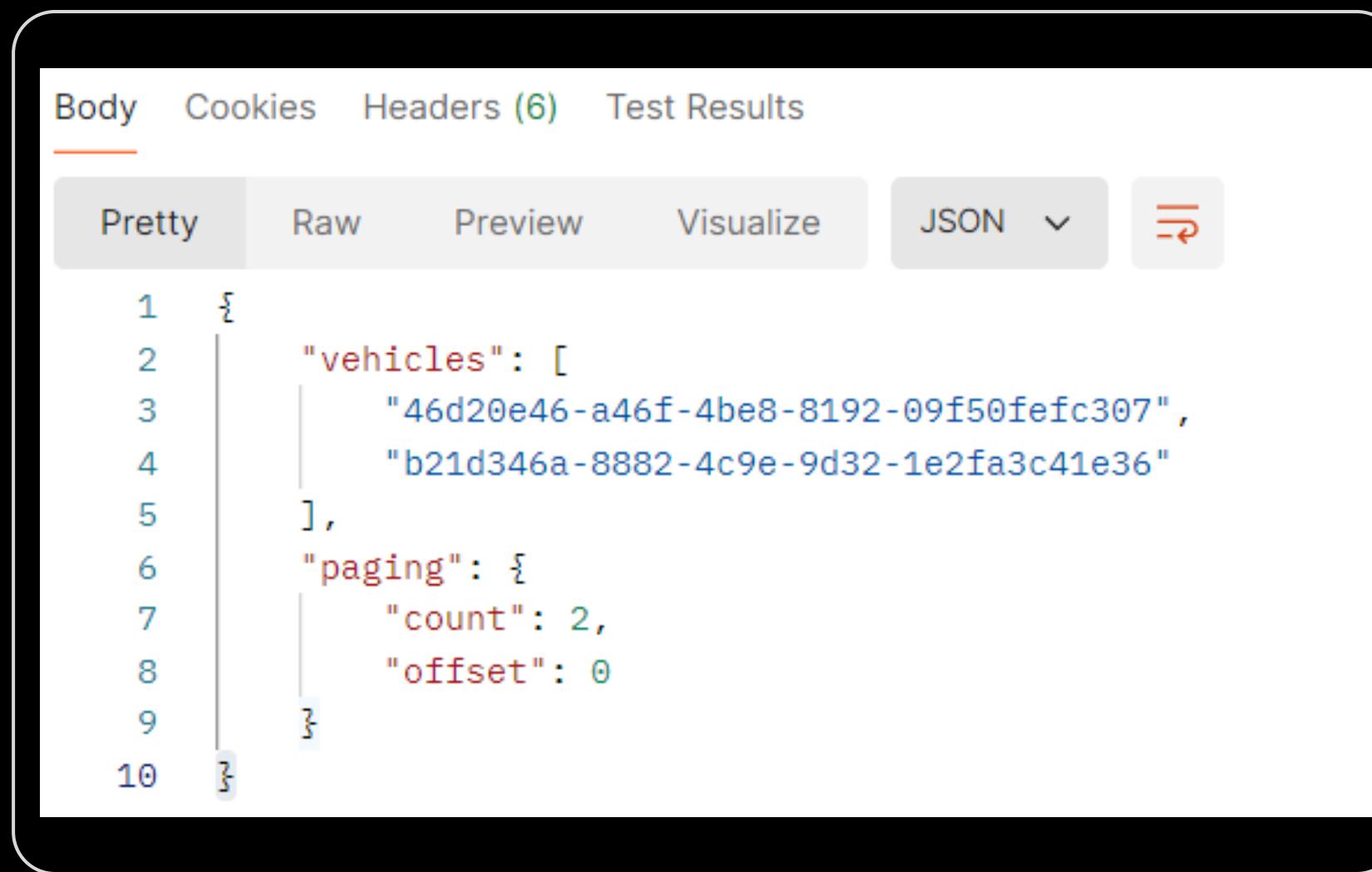
Callback URL ⓘ

<https://oauth.pstmn.io/v1/callback>

( IGUAL - Padrão do Postman )

# ALL VEHICLES (GET)

Obtém o código dos veículos da Lista/Marca.



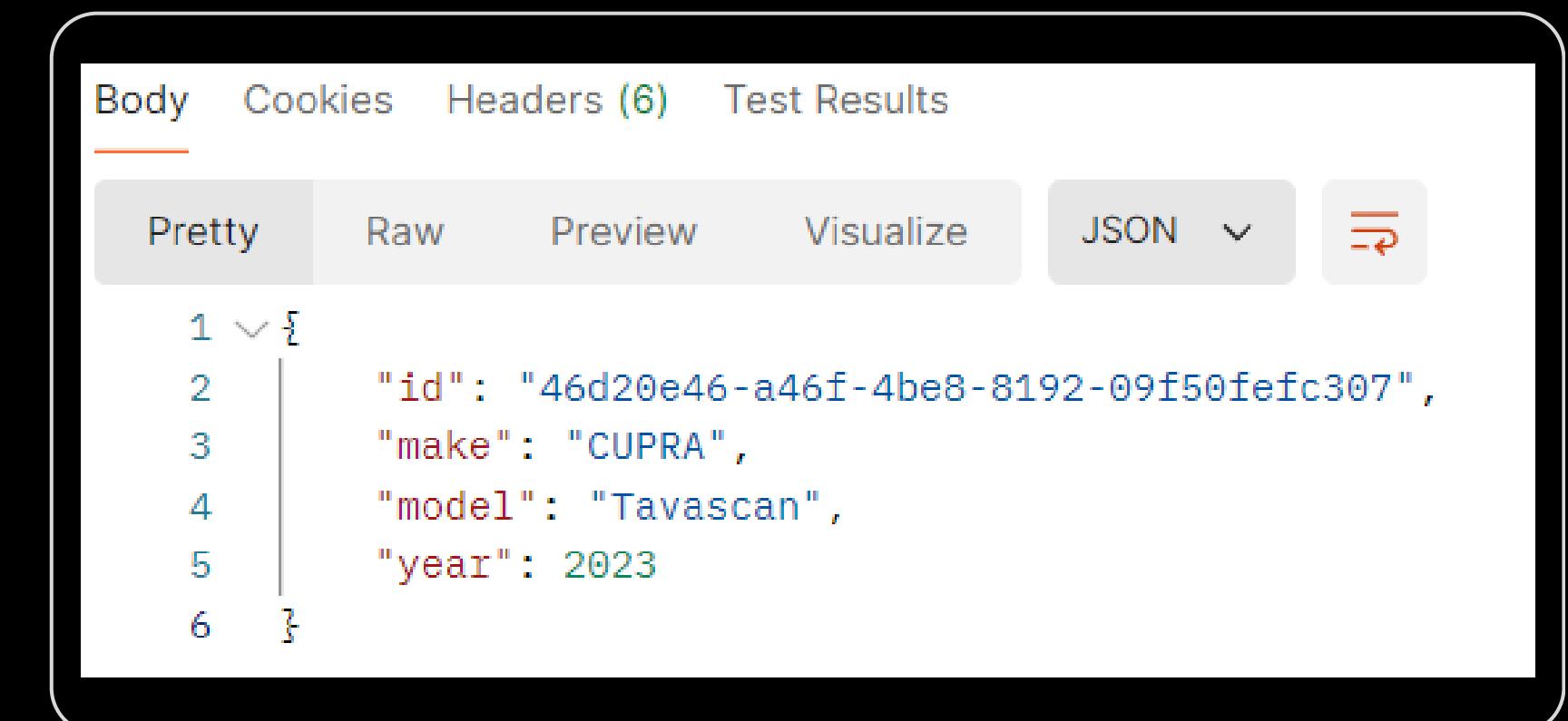
Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "vehicles": [  
3     "46d20e46-a46f-4be8-8192-09f50fefc307",  
4     "b21d346a-8882-4c9e-9d32-1e2fa3c41e36"  
5   ],  
6   "paging": {  
7     "count": 2,  
8     "offset": 0  
9   }  
10 }
```

# VEHICLE ATTRIBUTES (GET)

Obtém os atributos de um veículo



Body Cookies Headers (6) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "id": "46d20e46-a46f-4be8-8192-09f50fefc307",  
3   "make": "CUPRA",  
4   "model": "Tavascan",  
5   "year": 2023  
6 }
```

# FUEL TANK (GET)

Obtém o estado de combustível do veículo.

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "range": 234.52,  
3   "percentRemaining": 0.92,  
4   "amountRemaining": 121.97  
5 }
```

Documentation

Description

Returns the status of the fuel remaining in the vehicle's gas tank. Note: The fuel tank API is only available for vehicles sold in the United States.

Permission

read\_fuel

Response Body

Name	Type	Description
range	number	The estimated remaining distance the car can travel (in kilometers by default or in miles using the sc-unit-system).
percentRemaining	number	The remaining level of fuel in the tank (in percent).
amountRemaining	number	The amount of fuel in the tank (in liters by default or in gallons (U.S.) using the sc-unit-system).

# TRANCAR PORTAS DO VEÍCULO (POST)

Tranca as portas do Veículo.

```
1  {
2    "action": "LOCK"
3 }
```

Body Cookies Headers (6) Test Results

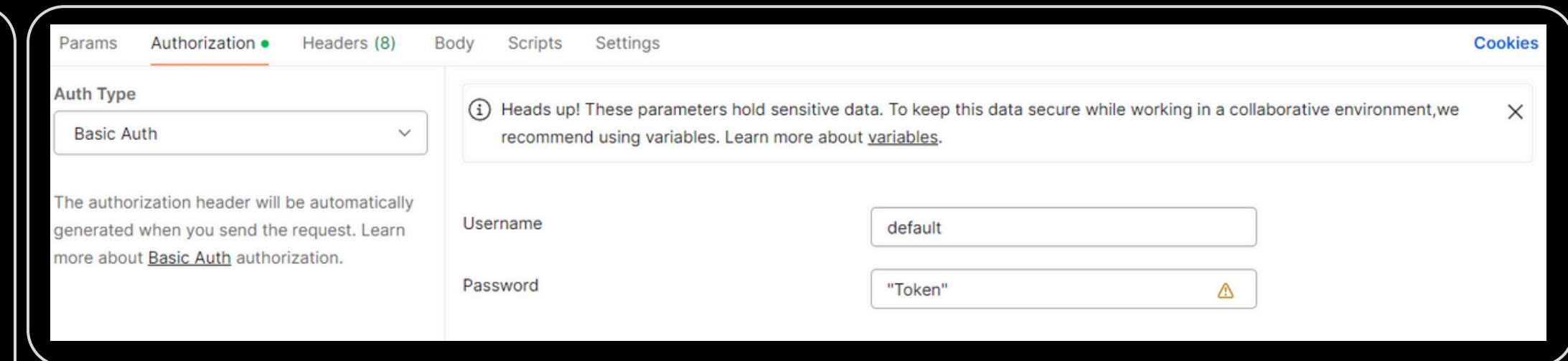
Pretty Raw Preview Visualize JSON ↗

```
1  {
2    "status": "success",
3    "message": "Successfully sent request to vehicle"
4 }
```

# VEHICLES CONNECTIONS (GET)

Indica os veículos Conectados

```
{  
  "connections": [  
    {  
      "connectedAt": "2024-10-04T22:24:52.152Z",  
      "vehicleId": "46d20e46-a46f-4be8-8192-09f50fefc307",  
      "userId": "b65b8599-2ca0-42a6-bed6-aecc90eaac33",  
      "mode": "simulated"  
    },  
    {  
      "connectedAt": "2024-10-04T22:24:52.152Z",  
      "vehicleId": "b21d346a-8882-4c9e-9d32-1e2fa3c41e36",  
      "userId": "b65b8599-2ca0-42a6-bed6-aecc90eaac33",  
      "mode": "simulated"  
    },  
    {  
      "connectedAt": "2024-10-04T21:32:32.723Z",  
      "vehicleId": "e9f2fd5b-243a-465f-a2f6-c7611a0d4bcc",  
      "userId": "e1cad2b6-e738-494e-9ef5-d1c9d04f5e7f",  
      "mode": "simulated"  
    }  
  "paging": {  
    "cursor": null  
  }  
}
```

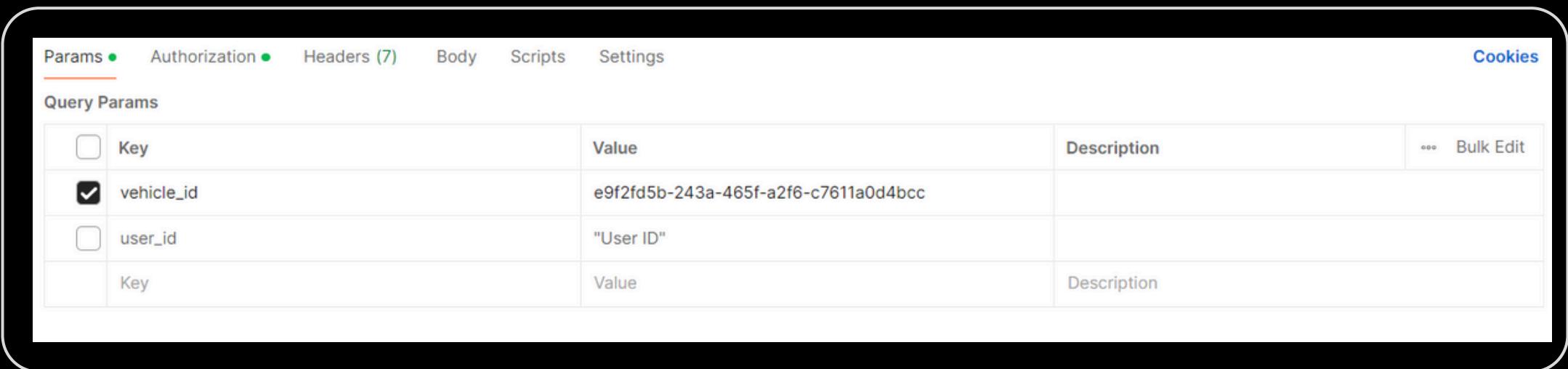


The screenshot shows the configuration for a GET request to the 'Vehicles Connections' endpoint. On the left, a code block displays the expected JSON response, which lists three vehicle connections with their respective details like connectedAt, vehicleId, userId, and mode. On the right, the API tool interface includes:

- Params**, **Authorization** (selected), **Headers (8)**, **Body**, **Scripts**, **Settings** tabs.
- Auth Type**: Set to **Basic Auth**.
- A note: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#)." with a close button.
- Username**: default
- Password**: "Token" with a warning icon.
- Cookies** tab.

# VEHICLES CONNECTIONS (DELETE)

Elimina um Veículo Conectado



The screenshot shows the 'Params' tab of a Postman request configuration. It includes sections for 'Query Params' and 'Cookies'. In the 'Query Params' section, there are two entries: 'vehicle\_id' with value 'e9f2fd5b-243a-465f-a2f6-c7611a0d4bcc' and 'user\_id' with value '"User ID"'. The 'Cookies' section is empty.



The screenshot shows the 'Body' tab of the Postman interface with a JSON response. The response is a single object with a 'connections' key, which contains an array of one element. This element is another object with 'vehicleId' and 'userId' keys, both of which have their values redacted (REDACTED).

```
1 <v>{  
2 <v>  "connections": [  
3 <v>    {  
4 <v>      "vehicleId": "e9f2fd5b-243a-465f-a2f6-c7611a0d4bcc",  
5 <v>      "userId": "e1cad2b6-e738-494e-9ef5-d1c9d04f5e7f"  
6 <v>    }  
7 <v>  ]  
8 <v>}
```

# RELAÇÕES DE TIPO

```
{  
  "connections": [  
    {  
      "connectedAt": "2024-10-04T22:24:52.152Z",  
      "vehicleId": "46d20e46-a46f-4be8-8192-09f50fefc307",  
      "userId": "b65b8599-2ca0-42a6-bed6-aecc90eaac33",  
      "mode": "simulated"  
    },  
    {  
      "connectedAt": "2024-10-04T22:24:52.152Z",  
      "vehicleId": "b21d346a-8882-4c9e-9d32-1e2fa3c41e36",  
      "userId": "b65b8599-2ca0-42a6-bed6-aecc90eaac33",  
      "mode": "simulated"  
    },  
    {  
      "connectedAt": "2024-10-04T21:32:32.723Z",  
      "vehicleId": "e9f2fd5b-243a-465f-a2f6-c7611a0d4bcc",  
      "userId": "e1cad2b6-e738-494e-9ef5-d1c9d04f5e7f",  
      "mode": "simulated"  
    }  
  "paging": {  
    "cursor": null  
  }  
}
```

Cada Carro “VehicleId” apenas pode ter um Utilizador “userId”  
UM PARA UM (1--1)

PORÉM

Cada Utilizador “userId” pode ter varios Carro “VehicleId”  
UM PARA MUITOS (1--\*)

# AVALIAÇÃO E ANÁLISE CRÍTICA

## Pontos Positivos

- Fácil documentação: A API é bem documentada, com exemplos claros.
- Integração com várias marcas: A compatibilidade com várias marcas e modelos de veículos.
- Compatibilidade por Marcas: Possui endpoints específicos para marcas populares, alinhando com a particularidades de cada fabricante.
- Ampla gama de funcionalidades: A API permite o controle e monitoramento de veículos, incluindo bateria, localização, tanque de combustível, odômetro, vida do óleo, pressão dos pneus, além de opções para bloquear/desbloquear e iniciar/parar carga.

VS

## Pontos a Melhorar

- Ausência de suporte para XML: Para empresas ou sistemas que utilizam XML, essa limitação pode ser um desafio.
- Carros mais recentes: Nos carros modernos, essas funcionalidades estão disponíveis, porém nos mais antigos não.
- Teoria e funcionalidade paga: Na teoria, é possível acessar e entender as funcionalidades, mas a maioria delas requer um pagamento para uso, e não é possível testar em um carro real sem esse pagamento.



# EXEMPLO DE MELHORIA

É possível utilizar um método PATCH na API Smartcar para atualizar os estados de bloqueio das janelas, portas e do porta-malas do veículo, permitindo que o utilizador controle individualmente cada parte, sem a necessidade de retornar todos os estados de bloqueio.

```
curl "https://api.smartcar.com/v2.0/vehicles/{id}/security" \
-H "Authorization: Bearer {token}" \
-X "PATCH" \
-H "Content-Type: application/json" \
```

# OBRIGADO

DESENVOLVIMENTO WEB I

Beatriz Almeida  
(A044416)

Carolina Fernandes  
(A044897)

Pedro Venda  
(A045464)