Grupo 09

Carolyne Melo - 20210046
Marco Antônio Camargo - 20211019
Diogo Carvalho - 20210008

Introdução

O Uash App é um aplicativo de lavagem de automóveis que permite a ambos que pessoas requisitem uma lavagem de um de seus automóveis, e que outras possam aceitar e prestar essa lavagem, em troca de dinheiro, sendo chamadas “uashers”. Também é possível fazer o “rating” de lavagens, que serão atribuídas ao perfil do uasher que a realizou. Logo, a base de dados precisa:

- Permitir a requisição de lavagens de até vários veículos ao mesmo tempo, na hora ou com data marcada.
- Permitir a prestação dessas lavagens, com associação à pessoa que as realizar.
- Permitir também a avaliação das mesmas, que será agregada em um campo de “rating” no perfil do uasher.

Nota: Os tipos de dados seguem descritos da forma “tipoNaBD (tipoNoServidor)”.

Resolução das Tarefas

	Marco Moreira Pinho	Carolyne Melo	Diogo Carvalho
Planeamento construção da Base de Dados	35%	55%	10%
Planeamento e construção do servidor	35%	55%	10%
Implementação das diversas funcionalidades disponíveis ao end user no servidor	45%	45%	10%
Planeamento e construção da base do aplicativo	55%	35%	10%
Realização do backend do aplicativo	55%	35%	10%
Realização do frontend do aplicativo	45%	45%	10%
Redação da documentação necessária	33%	34%	33%

Descrição da App / Problema a Resolver

O objetivo da nossa aplicação é desenvolver uma plataforma onde as pessoas possam marcar remotamente um serviço de limpeza da sua viatura. Dentro disso existem alguns objetivos que queremos atingir, nomeadamente :

- Sistema de login com perfis de utilizador tanto para clientes como para “uashers” (utilizadores que irão cumprir o serviço da limpeza)
- Sistema de avaliação destes “uashers”, onde as pessoas podem pontuar a prestação dos mesmos.
- Sistema de coordenadas GPS para o cliente encontrar o “uasher” mais próximo, discriminar localização da viatura e por fim, acordar entre os utilizadores e uashers os pontos de lavagem de carros mais próximos.
- Tabulação de preços, com níveis diferentes de serviço.

Descrição dos Objetivos / Motivação do Trabalho

O objetivo da aplicação é ser o ponto de ligação entre um utilizador e o uasher, através do mesmo será possível ter a localização atual para que o serviço seja completado. Comparado a outros países, a lavagem de automóveis em Portugal é dificultada devido à prevalência do estacionamento na rua e relativa ausência de postos de lavagem facilmente acessíveis em várias áreas do país (Ex: Oeiras), que é um problema relativamente difícil de solucionar e relativamente despercebido. A ideia do “Uash” é ajudar a solucionar esse problema ao permitir que indivíduos com experiência e/ou interesse na atividade possam prestar lavagens e serem pagos por tal atividade, atenuando este problema de forma significativa.

Aplicações semelhantes

Encontrámos algumas aplicações/sites, que fornecem serviços semelhantes ou que serviram de inspiração, assim como: Heywash, Sideline, Uber, Glovo, LowClean. As aplicações Heywash, Sideline e LowClean dentro destas são as que mais se aproximam daquilo que envisionámos para a nossa aplicação.

A aplicação Heywash funciona através de agendamento com um prestador de serviço em que o solicitador da viatura não tem que estar presente. Aquilo que é requisitado do solicitador é que facilite o acesso à chave da viatura, seja por terceiros (i.e um porteiro que fique com a chave até à chegada do Heywasher), ou pessoalmente fornecido pelo dono do veículo. Os Heywashers têm que ter material necessário para a lavagem do carro e não dão a hipótese do veículo ser transportado para um posto de lavagem de carros. Eles não têm também em conta se o utilizador que requisita a lavagem do carro tem acesso a uma garagem privada (ou semelhante) para cumprir o serviço. Outras diferenças que verificamos são:

- As taxas aplicadas pela Heywash, onde o serviço pode ter acréscimo de valor dependendo da localização e tipo de sujidade que as viaturas apresentem (i.e lama, lixo, pelos de animais).
- A impossibilidade de recorrer ao mesmo Heywasher (a seleção é baseada na distância)
- A proibição de gorjetas a Heywashers (embora eles façam referência que os Heywashers mais bem avaliados recebem bonificações)

A aplicação Sideline funciona de uma maneira muito semelhante à Heywash, mas faz da sua maior diferenciação uma lavagem ecológica. O método de recolha de chave é novamente direta ou por terceiros, os operadores da Sideline são obrigados a ter o material de limpeza com eles e aplicam taxas maiores consoante o nível de sujidade da viatura. A uma primeira impressão não demonstram aumentos de preço consoante a zona nem a proibição de bonificações para os seus operadores. Esta aplicação foi contudo menos transparente nos seus serviços, dificultando a aquisição de detalhes sem uma requisição de serviço.

A aplicação LowClean segue a mesma linha de funcionalidade tendo como especialidade diferentes tipos de lavagem das viaturas. Se verificarmos a sua tabela de preços verificamos uma grande variedade de tipos de limpeza disponíveis, desde a mais básica, propriamente nomeada de "BASIC", à mais complexa que seria a "SuperDiamante", com uma valor muito maior do que as anteriores mas com uma atenção ao detalhe redobrada e incluindo lavagem da parte mecânica da viatura, podendo durar até cerca de 8.00h.

Por fim, a Uber e a Glovo foram a nossa maior inspiração para a nossa aplicação. Sem antes sabermos da existência das outras duas anteriores (Heywash e Sideline), foram estas aplicações de requisição de serviços remotos que nos trouxeram esta ideia, em que trabalhadores independentes poderiam agendar serviços através de uma plataforma que fizesse um ponto de ligação para os clientes. Apesar disto, estas aplicações não apresentam estes tipos de serviços, ficando assim apenas como fonte de inspiração e não como concorrência direta.

Solução Implementada

O aplicativo conta com um usuário local (a pessoa que faz login) e uma lista de todas as lavagens, também salvas localmente. Na ausência do servidor, o aplicativo aceita os dados da tela de registro como usuário local, e faz um populate das lavagens. A senha do usuário é salteada com o seu próprio ID do servidor e encriptada em SHA-256 antes de ser salva na base de dados, garantindo a segurança e permitindo que o próprio aplicativo realize a encriptação novamente em uma senha qualquer para verificar se essa e a senha na base de dados são as mesmas. O aplicativo possui uma página principal com um mapa GPS que exibe a posição atual em tempo real. Nesse mapa, é possível visualizar a localização real de todas as lavagens, que possuem seus próprios marcadores. Ao serem selecionadas, essas mostram todas as suas informações em uma página separada, onde também podem ser aceitas. Todos os objetos (User, Uasher, Veículos e Lavagens), ao serem criados, são enviadas ao servidor, e consequentemente a BD, de imediato. O restante das

funcionalidades estão completas na base de dados, no servidor e no backend do aplicativo, mas não chegaram a ficar prontas para o frontend, logo, seguirá uma explicação técnica.

Ao ser finalizada uma lavagem, uma lavagem pode ser atribuída a essa. Após isso, o servidor poderá atualizar a avaliação total do Uasher, que é uma média de todas as lavagens em que participou. (Ver “Atualizar rating do Uasher (put)” na documentação REST)

Enquadramento nas Unidades Curriculares

A aplicação foi feita no "Android Studio", relacionado à cadeira de Desenvolvimento Móvel pelo próprio aplicativo e disponibilidades como: mapas por GPS utilizando a biblioteca do Google, construção de layouts, uso de ListViews e seus adaptadores, activities em fragmentos, entre outros.

Java 17 também foi utilizada na elaboração do nosso projeto que está diretamente ligada à cadeira de Programação Orientada a Objetos. O Visual Studio Code foi utilizado em conjunto com o SpringBoot para estabelecer um servidor Apache Tomcat localmente e permitir a comunicação entre este, a base de dados e o aplicativo.

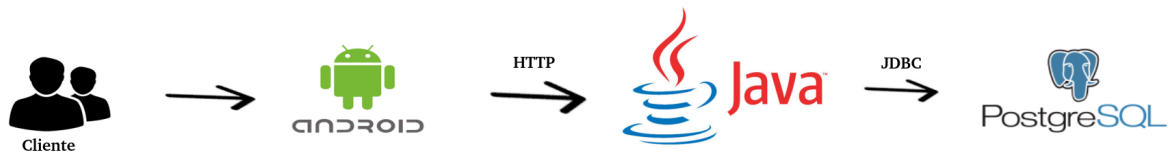
O servidor tem acesso a uma base de dados para armazenar e fazer a pesquisa de todos os tipos de dados, desenvolvida em conjunto com a sua respectiva cadeira (base de dados). Na implementação da Base de Dados utilizamos a linguagem SQL, e as ferramentas PostgresSQL e PgAdmin para gerenciamento dos dados.

Tecnologias Utilizadas

- PostgresSQL
- PgAdmin 4
- SpringBoot
- Android
- Java 17
- Apache Tomcat
- Biblioteca de mapa do Google

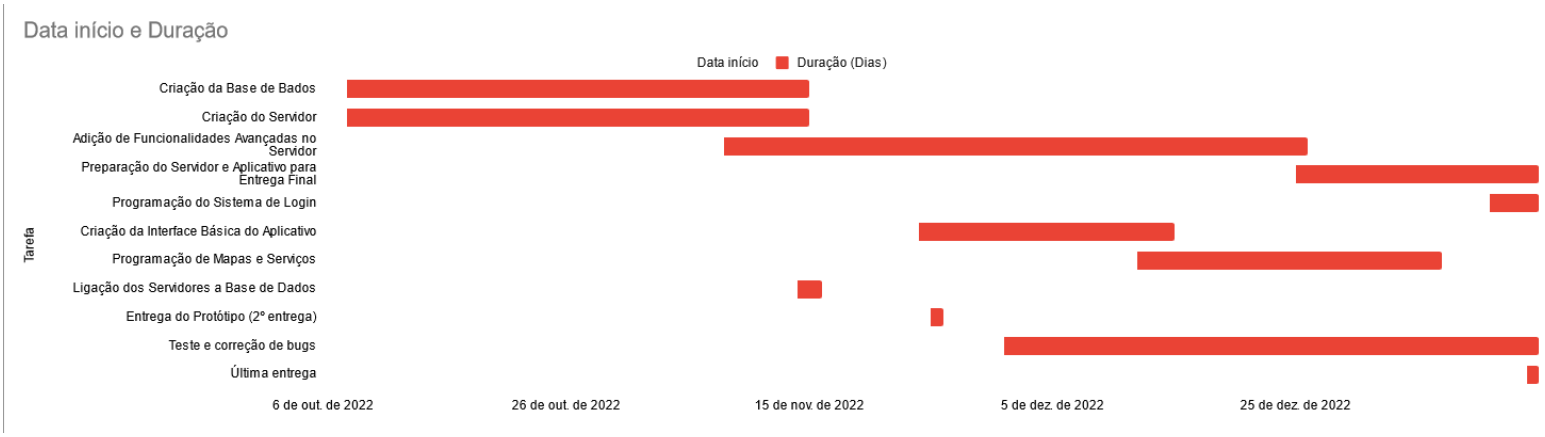
Com essas tecnologias, segundo o nosso planejamento atual, todos os recursos do app serão possíveis, mesmo que alguns não essenciais possam ser abandonados ou adicionados durante o percurso de elaboração do projeto.

Arquitetura da Solução



Calendarização

Tarefa	Data início	Duração (Dias)	Data conclusão
Criação da Base de Bados	8/10	38	14/11
Criação do Servidor	8/10	38	14/11
Adição de Funcionalidades Avançadas no Servidor	8/11	48	25/12
Preparação do Servidor e Aplicativo para Entrega Final	25/12	20	13/1
Programação do Sistema de Login	10/1	4	13/1
Criação da Interface Básica do Aplicativo	24/11	21	14/12
Programação de Mapas e Serviços	12/12	25	5/1
Ligação dos Servidores a Base de Dados	14/11	2	15/11
Entrega do Protótipo (2º entrega)	25/11	1	25/11
Teste e correção de bugs	1/12	44	13/1
Última entrega	13/1	1	13/1



Bibliografia

https://www.heywash.pt/pt

https://sideline.pt/

https://www.lowclean.pt/

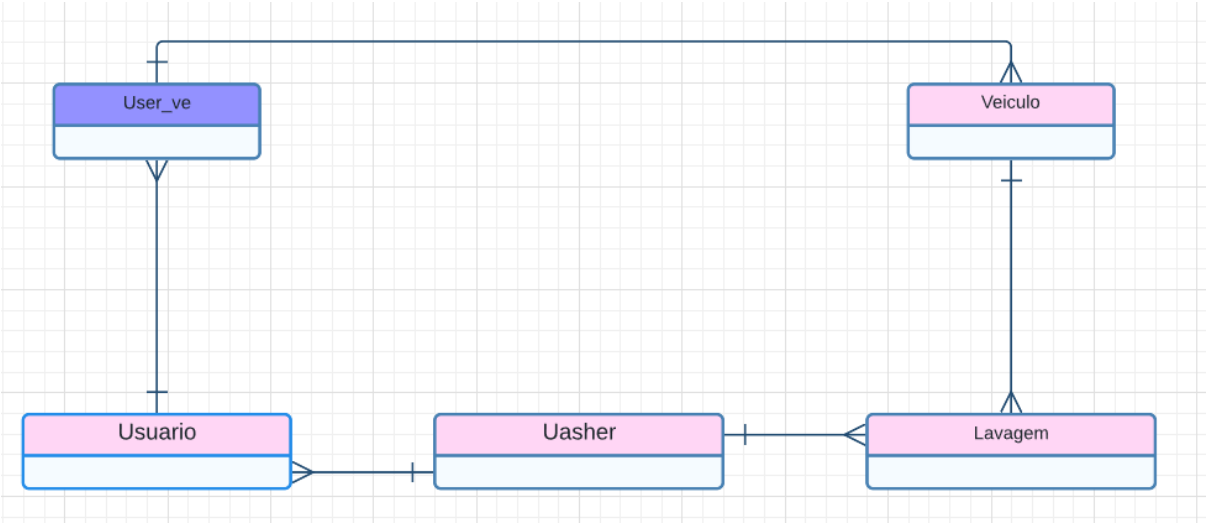
https://www.flaticon.com/br/icone-gratis/usuarios-masculinos_17283

https://www.freepnglogos.com/pics/android-logo

https://www.sintesys.us/fullscreen-page/comp-jiz1imzz/503b93d4-9b87-4f18-b50f-2c0799522cf9/4/%3Fi%3D4%26p%3Dijdas%26s%3Dstyle-jiz1ikom

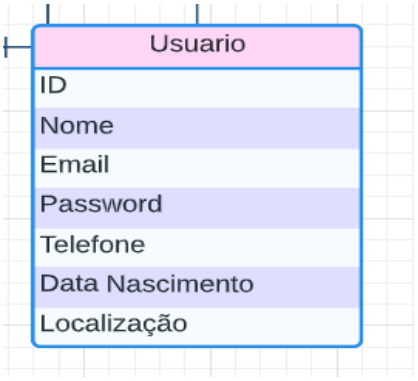
https://1000logos.net/postgresql-logo/

Dicionário de Dados



Modelo Entidade-Relação

Usuário:



ID - user_id : integer (int)
Nome - user_nome : varchar(255) (String)
Email - user_email : varchar(255) (String)
Password - user_pass : varchar(255) (String)
Telefone - user_tel : varchar(255) (int)
Data Nascimento - user_dt_nasc : timestamp (LocalDate)
Localização - user_loc : varchar(255) (String)

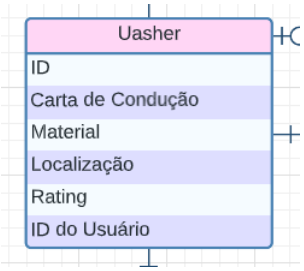
A localização do usuário é salva na forma de coordenadas em formato de texto, e a senha está encriptada em SHA-256.

```
select *
from user
```

Selecionar perfil Uasher de um ou todos usuário(s).

```
select uasher
from uasher, user
where uasher_id = :input / user_id
```

Uasher:



ID - uasher_id : integer (int)
Carta de Condução - uasher_cartaConducao : varchar(255) (String)
Materiais de Limpeza - uasher_mat : boolean (boolean)
Localização - uasher_loc : varchar(255) (String)
Rating - uasher_rat : integer (Integer)
ID do Usuário - uasher_user : integer (int)

```
select *
from uasher
```

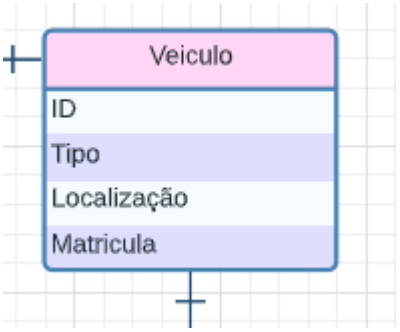
Selecionar todas as lavagens de um ou todos Uasher(s):

```
select lavagem
from uasher, lavagem
where uasher_id = :input / lavagem_uasher
```

Essa querry é utilizada no servidor para determinar o “rating” de um Uasher.

```
select uasher_rat
from uasher, lavagem
where uasher_id = :input
```

Veículo:



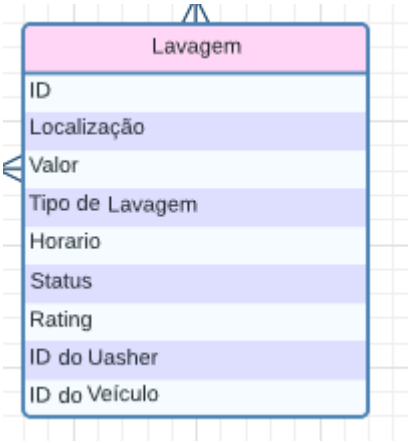
ID - veiculo_id : integer (int)
Tipo - veiculo_tipo : varchar(255) (String)
Matrícula - veiculo_mat : varchar(255) (String)
Localização - veiculo_loc : varchar(255) (String)

```
select *
from veiculo
```

Mostrar todos os veículos de um usuário.

```
select *
from veiculo
INNER JOIN veiculo_users ON veiculo_veiculo_id = veiculo_id
where users_user_id = :input
```

Lavagem:



ID - lavagem_id : integer (int)
Localização - lavagem_loc : varchar(255) (String)
Valor - lavagem_val : integer (int)
Tipo de Lavagem - lavagem_tipo: varchar(255) (String)
Horário - lavagem_hora: timestamp (**LocalDateTime*)**
Tipo de Status - lavagem_sta: varchar(255) (String)
Placeholder do “rating” - lavagem_rat : integer (int)
ID do Uasher - lavagem_uasher : integer (int)
ID do Veículo - lavagem_veic : integer (int)

***** Note que o tipo é “LocalDateTime”, e não apenas “LocalDate”. Isto permite salvar também o horário específico da lavagem requisitada, ao contrário das datas de nascimento.

```
select *
from lavagem
```

Guia de Dados

Nota: Todas as localizações são apenas coordenadas em formato de character. Os nomes entre parênteses estão apenas na Guia de Dados para servir de orientação.

Usuário:

Um usuário possui seu nome, email, senha (que será encriptada por SHA-256), telefone, data de nascimento e localização. Um usuário também pode ter um perfil uasher (One to one or zero), e pode ter vários veículos (Many to many) - ambos sendo opcionais. Esse também é atribuído um ID único. Na tabela de usuário, temos os dados que serão necessários para fazer o cadastro do utilizador, será um formulário que será preenchido quando inicia o aplicativo pela primeira vez e deverá conter obrigatoriamente os campos indicados acima.

Atualmente temos alguns usuários, como é o exemplo do Alexandre, Íris, Joaquim, Luiza, Francisco e Vitor Hugo. Com seus perfis criados e com suas informações pessoais preenchidas, seus dados terão esse aspecto:

user_id	1	2	3	4	5
user_name	Alexandre Carvalho	Íris Ferreira Santos	Joaquim Ramos	Luíza Araújo	Francisco Barbosa
user_email	nutellaking@gmail.com	irissantos24@yahoo.com	jocar45@hotmail.com	luiza_ara@gmail.com	fran_bar@gmail.com
user_pass	6823cc2d4ac2e848a53f64	d59849704e18653cb6a8	"d59849704e18653cb6a8"	"f79559704r19753op3a5"	"p98565664o19633rt3a2"
user_tel	910599351	910378312	911415432	955030276	989001466
user_dt_nasc	1997-03-15	2000-09-27	1994-11-25	2002-05-03	2003-10-09
user_loc	38.7043515, -9.2509783 (Cruz Quebrada)	38.7022849, -9.2290415 (Algés)	38.7079173, -9.156786 (Santos)	38.5289659,-8.8799924 (Setúbal)	38.7436266, -9.1602032 (Lisboa)

Uasher:

Digamos também que o Alexandre está interessado em prestar lavagens, para ajudar a pagar o seu mestrado, e se registar como um uasher. Adicionalmente ao seu perfil de usuário, ele precisará fornecer a sua carta de condução (caso seja requisitada a dirigir algum veículo para outro local), sua localização preferencial de lavagens (diferente da dos usuários, que são a localização de residência), e se possui ou não os materiais necessários para a realização das lavagens. Digamos que, por enquanto, ainda lhe faltam produtos para lavagens à seca, esse campo será então apresentado como “false”.
Já o Joaquim, estava a procura de trabalho mas não encontrava algo do seu agrado e decidiu se inscrever no aplicativo, uma vez que tem uma maior liberdade para gerir seus horários.
Para ambos o campo “rating” será zero por default, já que ainda não prestou nenhuma lavagem, e o formato de seus dados serão esses:

uasher_id	1	2	3
uasher_cartaConducao	3098761/B	6547531/B	2069739/B
uasher_mat	false	true	true
uasher_loc	38.7043515, -9.2509783 (Cruz Quebrada)	38.7079173, -9.156786 (Santos)	38.7436266, -9.1602032 (Lisboa)
uasher_rat	0	5	0
uasher_user	1	3	5

Veículo:

Alexandre, por sua vez, recentemente comprou um Honda Civic 2003 de segunda mão. Para registrar o seu carro, Alexandre atualmente precisa apenas informar o aplicativo com o tipo de veículo, a localização atual deste e a sua matrícula (placa/registro). Há a hipótese de um veículo haver vários donos, além de um único usuário ter vários veículos. Alexandre, por exemplo, tem um namorado que não possui uma carta de condução, mas ainda pretende ajudar a cuidar do carro - mas ainda não se registrou no aplicativo. Caso ele decida registrar-se, é importante que o mesmo veículo possa ter vários donos, para evitar conflitos de informação.

Por isso, uma ligação Many to Many é necessária, com uma estrutura “User_ve” servindo de intermediária, contendo os IDs do veículo e de um dos donos.

veiculo_id	3
veiculo_tipo	Honda Civic 2003
veiculo_mat	15VA73
veiculo_loc	38.7043515,-9.2509783 (Cruz Quebrada)

Lavagem:

Com o seu carro registrado, Alexandre agora pode requisitar a sua lavagem. Para isso, ele precisará fornecer o local desejado da lavagem (visto que pode ser na sua própria garagem, em um posto de lavagem, ou em outro local desejado), o valor da lavagem, o tipo de lavagem (lavagem seca, em local fechado, etc, seguindo as leis de Portugal segundo o uso de água em via pública), e o horário desejado (já que é possível agendar a lavagem para uma data posterior, ou requisitar essa na hora). O “status” da lavagem é gerida automaticamente por uma combinação do servidor e aplicativo, mas começa como “solicitada”.

O Alexandre marcou sua lavagem em sua garagem para o dia seguinte, dia 12/01/2023, ao meio-dia. Íris, por sua vez, vê a lavagem imediatamente, por estar na sua zona de preferência. Com isso, ela aceita a lavagem, mudando o status dessa para “agendada”.

RESTRIÇÃO: Os preços e tipos de lavagem são fixos e tabelados, mas atualmente são campos abertos na base de dados.

lavagem_id	1
lavagem_loc	38.7043515,-9.2509783 (Cruz Quebrada)
lavagem_val	45
lavagem_tipo	Garagem Privada
lavagem_hora	2023-01-18 15:00:00
lavagem_sta	Concluído
lavagem_rat	5
lavagem_uasher	2
lavagem_veic	3

NOTA IMPORTANTE: Adicionamos 10 usuários de exemplo na base de dados, usamos apenas alguns deles para que não ficasse muito extenso.

Inserts e Creates

```
CREATE TABLE .lavagem (
    lavagem_id integer NOT NULL,
    lavagem_hora timestamp without time zone,
    lavagem_loc character varying(255),
    lavagem_rat integer,
    lavagem_sta character varying(255),
    lavagem_tipo character varying(255),
    lavagem_val integer,
    lavagem_uasher integer,
    lavagem_veic integer
);

CREATE TABLE .uasher (
    uasher_id integer NOT NULL,
    uasher_carta character varying(255),
    uasher_loc character varying(255),
    uasher_mat boolean,
    uasher_rat integer,
    uasher_user integer,
    user_uashers_fav integer
);

CREATE TABLE .usuario (
    user_id integer NOT NULL,
    user_dt_nasc date,
    user_email character varying(255),
    user_loc character varying(255),
    user_nome character varying(255),
    user_pass character varying(255),
    user_tel integer
);

CREATE TABLE .veiculo (
    veiculo_id integer NOT NULL,
    veiculo_loc character varying(255),
    veiculo_mat character varying(255),
    veiculo_tipo character varying(255)
);
```

```
INSERT INTO .lavagem (lavagem_id, lavagem_hora, lavagem_loc,
lavagem_rat, lavagem_sta, lavagem_tipo, lavagem_val, lavagem_uasher,
lavagem_veic) VALUES (1, 2023-01-18 15:00:00, '38.7043515,-9.2509783',
5, 'Concluído', 'Garagem Privada', 45, 2, 3);
INSERT INTO .lavagem (lavagem_id, lavagem_hora, lavagem_loc,
lavagem_rat, lavagem_sta, lavagem_tipo, lavagem_val, lavagem_uasher,
lavagem_veic) VALUES (2, 2023-01-21 17:00:00, '38.707729,-9.1819063',
0, 'Solicitado', 'Lavagem a Seco', 25, 1, 1);



INSERT INTO.uasher (uasher_id, uasher_carta, uasher_loc, uasher_mat,
uasher_rat, uasher_user, user_uashers_fav) VALUES (1, '3098761/B',
'38.7043515,-9.2509783', false, 0, 1, NULL);
INSERT INTO.uasher (uasher_id, uasher_carta, uasher_loc, uasher_mat,
uasher_rat, uasher_user, user_uashers_fav) VALUES (2, '6547531/B',
'38.7079173,-9.156786', true, 0, 3, NULL);
INSERT INTO.uasher (uasher_id, uasher_carta, uasher_loc, uasher_mat,
uasher_rat, uasher_user, user_uashers_fav) VALUES (3, '2069739/B',
'38.7436266,-9.1602032', true, 0, 5, NULL);


INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (1, '1997-03-15',
'nutellaking@gmail.com', '38.7043515,-9.2509783', 'Alexandre Carvalho',
'6823cc2d4ac2e848a53f64', 910599351);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (2, '2000-09-27',
'irissantos24@yahoo.com', '38.7022849,-9.2290415', 'Íris Ferreira
Santos', 'd59849704e18653cb6a8', 910378312);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (3, '1994-11-25',
'jocar45@hotmail.com', '38.7079173,-9.156786', 'Joaquim Ramos',
'd59849704e18653cb6a8', 911415432);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (4, '2002-05-03',
'luiza_ara@gmail.com', '38.5289659,-8.8799924', 'Luiza Araújo',
'f79559704r19753op3a5', 955030276);
```

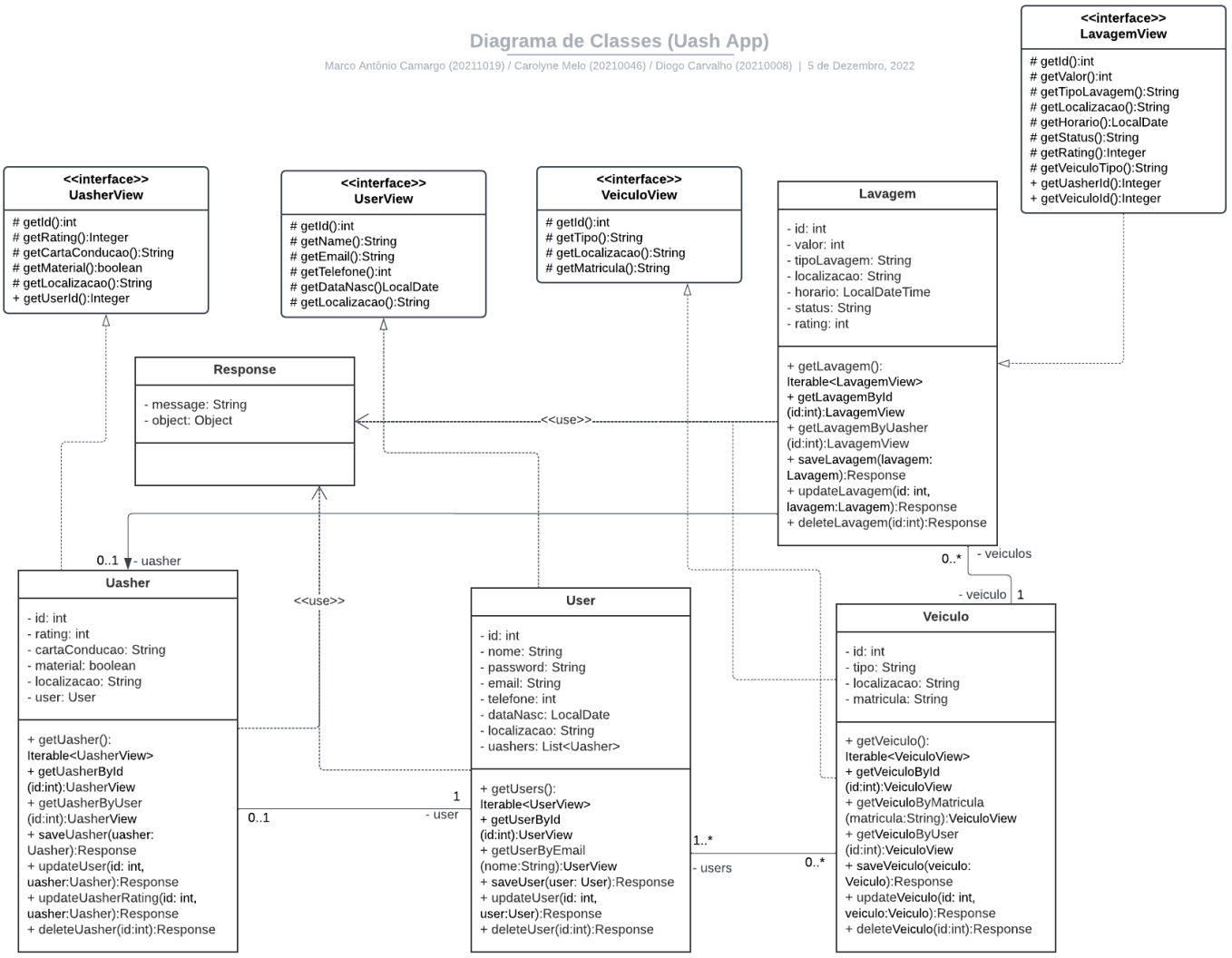
```
INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (5, '2003-10-09',
'fran.bar@gmail.com', '38.7436266,-9.1602032', 'Francisco Barbosa',
'p98565664o19633rt3a2', 989001466);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (6, '1998-12-27',
'vitor_h_1998@gmail.com', '38.6974652,-9.4237401', 'Vitor Hugo Moura',
'h98565951t15313df8s2', 960423115);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (7, '1978-01-07',
'carlos_eduardo@outlook.com', '38.7436266,-9.1602032', 'Carlos Eduardo
Cunha', 'm65565951p15313df8d8', 930600835);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (8, '1992-06-06',
'lara_das_neves11@hotmail.com', '38.7079173,-9.156786', 'Lara das
Neves', 'l385984951b18523kk8f6', 916377555);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (9, '1996-11-07',
'emanu.p@gmail.com', '38.707729,-9.1819063', 'Emanuel Porto',
'k35943951c18523vc8t9', 986066979);

INSERT INTO.usuario (user_id, user_dt_nasc, user_email, user_loc,
user_nome, user_pass, user_tel) VALUES (10, '2001-07-20',
'luiz-mig-ara@gmail.com', '38.7463133,-9.1131452', 'Luiz Miguel
Araújo', 'p85243951d88523yc8r7', 986066979);



INSERT INTO public.veiculo (veiculo_id, veiculo_loc, veiculo_mat,
veiculo_tipo) VALUES (1, '38.7043515,-9.2509783', '15VA73', 'Honda
Civic 2003');

INSERT INTO public.veiculo (veiculo_id, veiculo_loc, veiculo_mat,
veiculo_tipo) VALUES (2, '38.7079173,-9.156786', '87CM11', 'Fiat
Panda');
```

```
INSERT INTO public.veiculo (veiculo_id, veiculo_loc, veiculo_mat,
veiculo_tipo) VALUES (3, '38.707729,-9.1819063', '15VA73', 'Honda Civic
2003');


INSERT INTO public.veiculo_users (veiculo_veiculo_id, users_user_id)
VALUES (2, 8);
INSERT INTO public.veiculo_users (veiculo_veiculo_id, users_user_id)
VALUES (3, 1);



SELECT pg_catalog.setval('public.lavagem_lavagem_id_seq', 2, true);



SELECT pg_catalog.setval('public.uasher_uasher_id_seq', 3, true);



SELECT pg_catalog.setval('public.usuario_user_id_seq', 10, true);



SELECT pg_catalog.setval('public.veiculo_veiculo_id_seq', 3, true);



ALTER TABLE ONLY .lavagem
    ADD CONSTRAINT lavagem_pkey
    PRIMARY KEY (lavagem_id);



ALTER TABLE ONLY .uasher
    ADD CONSTRAINT uasher_pkey
    PRIMARY KEY (uasher_id);



ALTER TABLE ONLY .usuario
    ADD CONSTRAINT usuario_pkey
    PRIMARY KEY (user_id);
```

```
ALTER TABLE ONLY .veiculo
    ADD CONSTRAINT veiculo_pkey
    PRIMARY KEY (veiculo_id);



ALTER TABLE ONLY .uasher
    ADD CONSTRAINT fk1ssv3j4qqk1i6h9g1ofd58hrr
    FOREIGN KEY (user_uashers_fav)
    REFERENCES .usuario(user_id);



ALTER TABLE ONLY .lavagem
    ADD CONSTRAINT fk2t37p5lnfm78dal884soccmkh
    FOREIGN KEY (lavagem_uasher)
    REFERENCES .uasher(uasher_id);



ALTER TABLE ONLY .lavagem
    ADD CONSTRAINT fk7bgn910is69lwfclf9a56xx4p
    FOREIGN KEY (lavagem_veic)
    REFERENCES .veiculo(veiculo_id);



ALTER TABLE ONLY .veiculo_users
    ADD CONSTRAINT fkh3ilsjlk4gqq3gj3brgs1s4ol
    FOREIGN KEY (veiculo_veiculo_id)
    REFERENCES .veiculo(veiculo_id);



ALTER TABLE ONLY public.uasher
    ADD CONSTRAINT fki5301tw5aovmj1bhykmm2rdov
    FOREIGN KEY (uasher_user)
    REFERENCES public.usuario(user_id);
```

```
ALTER TABLE ONLY public.veiculo_users
    ADD CONSTRAINT fkrk66ra39qhr7n8yfqeqa3clt2
    FOREIGN KEY (users_user_id)
    REFERENCES public.usuario(user_id);



REVOKE USAGE ON SCHEMA public FROM PUBLIC;
GRANT ALL ON SCHEMA public TO PUBLIC;
```

Diagrama de Classes



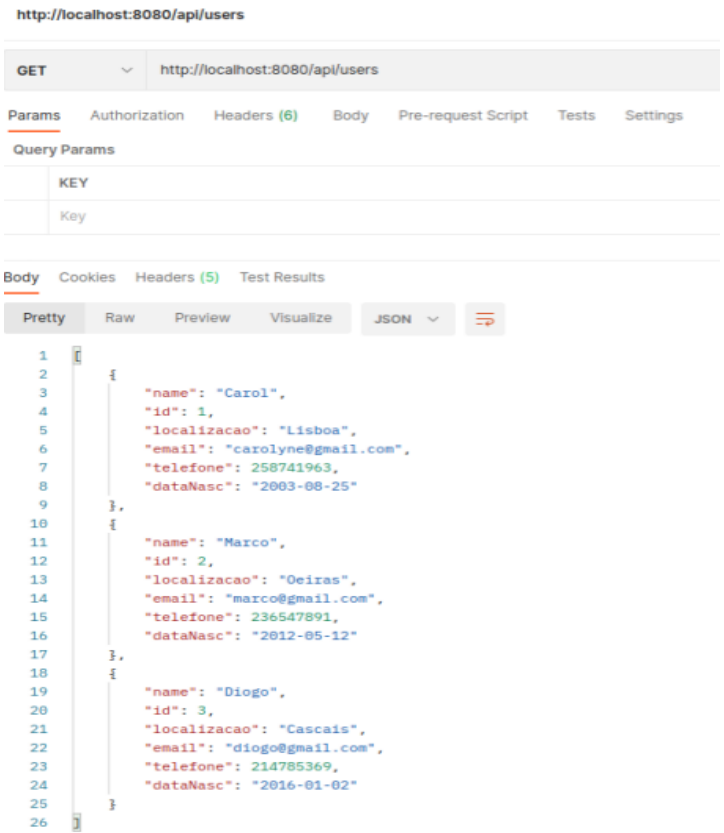
Documento REST

Erros “default”:
400 - Bad Request
404 - Not Found
405 - Method Not Allowed
500 - Internal Server Error

Usuários (/api/users)

Exibir todos os Users (get): /api/users/
Sem parâmetros adicionais.
Devolve uma tabela com todos os usuários e seus dados, exceto suas senhas.

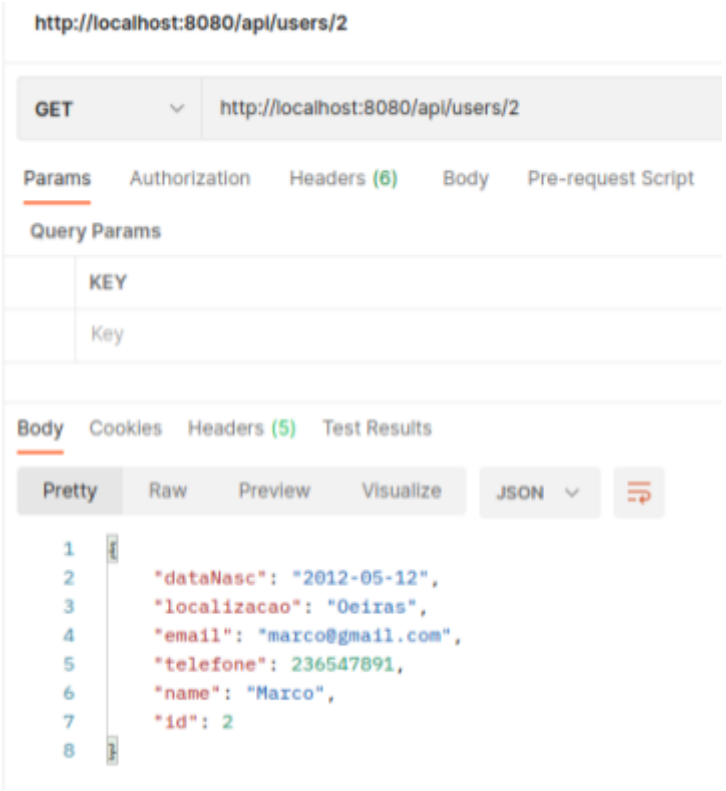
Resultado Esperado / Exemplo



Sem códigos de erro programados

Exibir User por Id (get): /api/users/{id:[0-9]+}
Id: O Id do User desejado.
Devolve os dados do usuário desejado, exceto a senha.

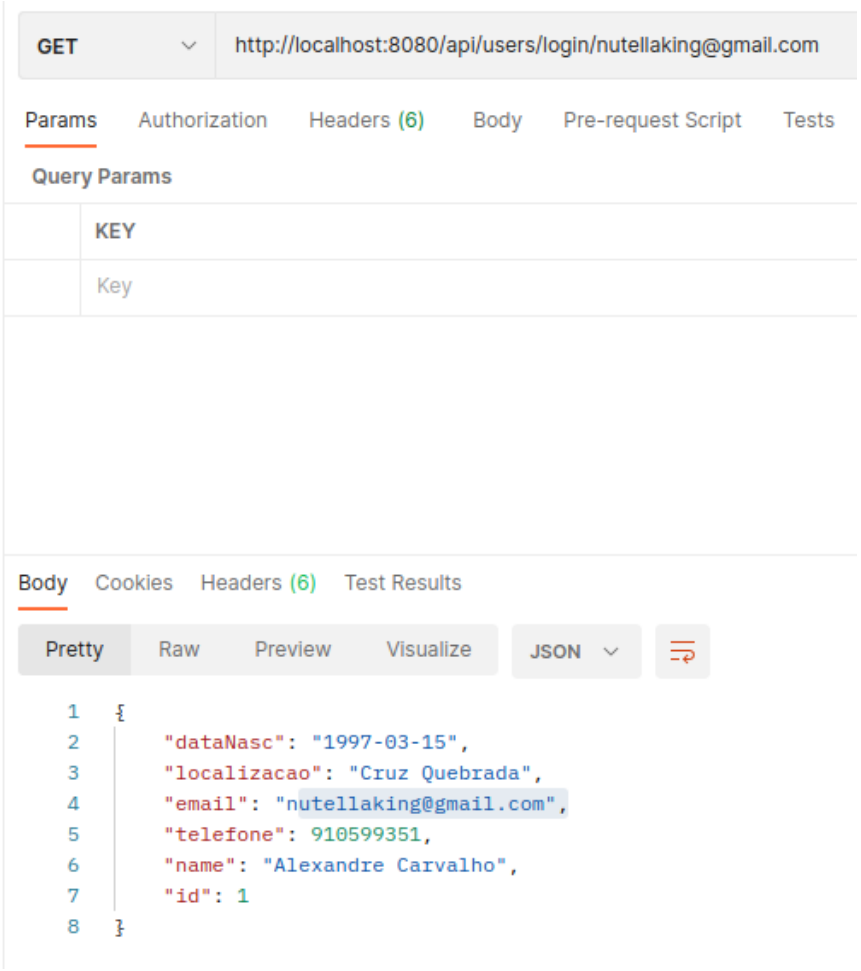
Resultado Esperado / Exemplo



Sem códigos de erro programados

Exibir User por Email (get): `api/login/{nome}`
Id: O Id do User desejado.
Devolve as informações do usuário através da busca pelo email.

Resultado Esperado / Exemplo



Sem erro de códigos programados

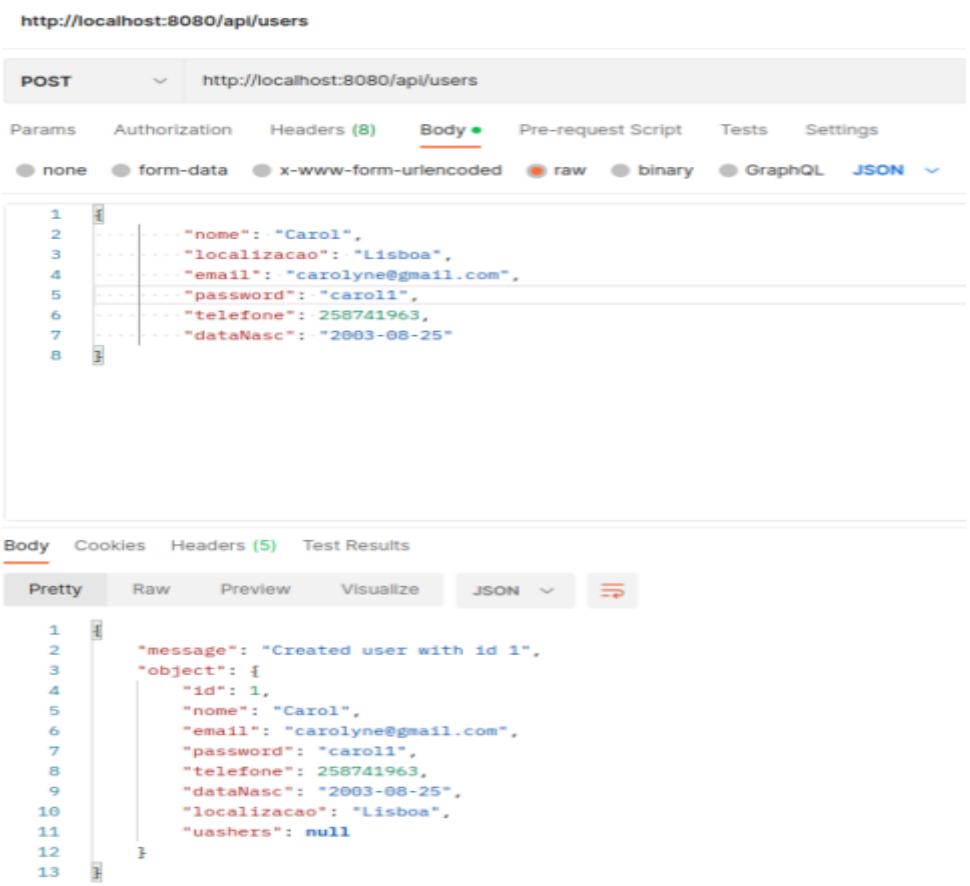
Adicionar User (post): /api/users/
Sem parâmetros adicionais. Um JSONObject User deverá ser providenciado.

Resultado Esperado


```
{
        “message”: “Created user with id 1”,
        “object”: {
                “id”: 1,
                “nome”: “Carol”,
                “email”: “carolyne@gmail.com”,
                “password”: “carol1”,
                “telefone”: 258741963,
                “dataNasc”: “2003-08-25”,
                “localizacao”: “Lisboa”,
                “uashers”: null
        }
}
```


Sem códigos de erro programados

Exemplo



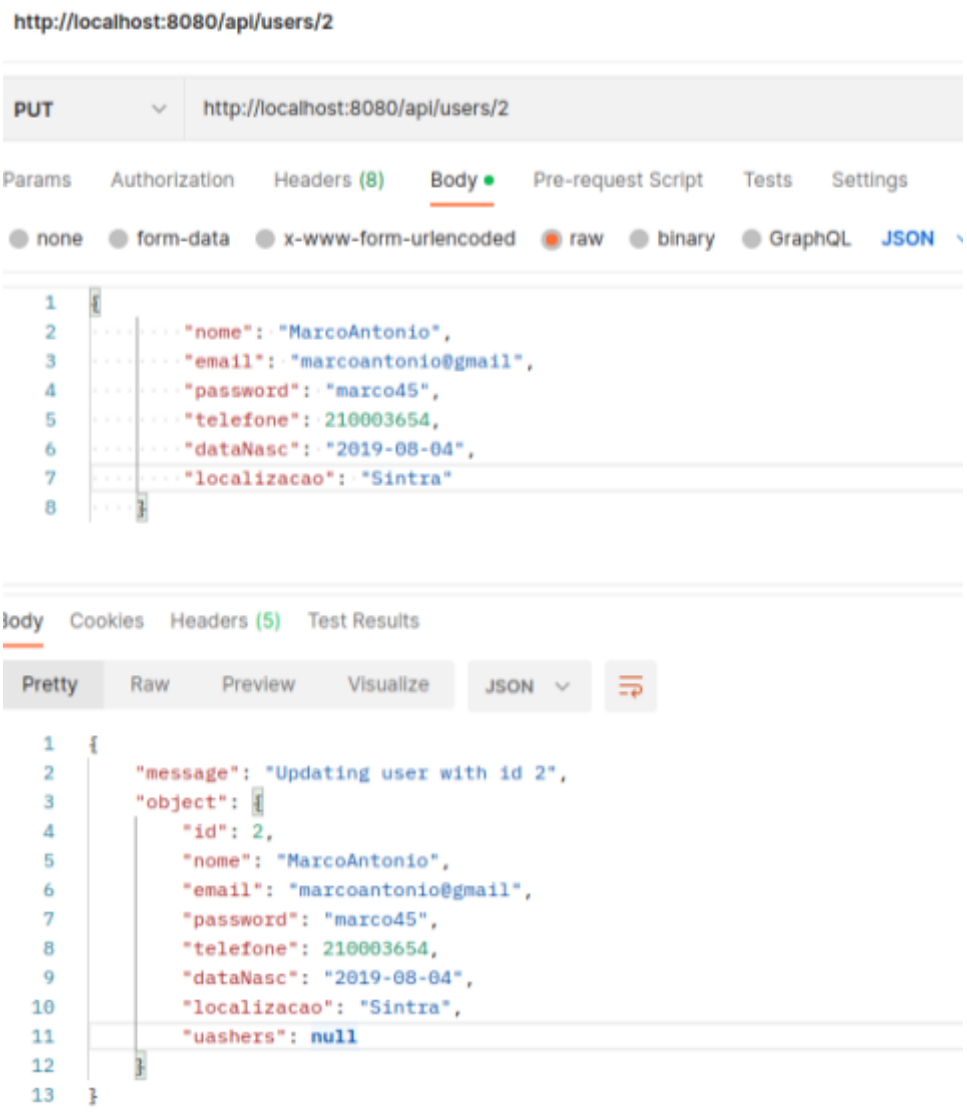
Atualizar User (put): /api/users/{id:[0-9]+}
Id: O Id do User desejado.
Atualiza o usuário com os dados providenciados, que devem ser em formato JSONArray.

Resultado Esperado

```
{
        “message”: “Updating user with id 1”,
        “object”: {
                “id”: 2,
                “nome”: “MarcoAntonio”,
                “email”: “marcoantonio@gmail”,
                “password”: “marco45”,
                “telefone”: 210003654,
                “dataNasc”: “2019-08-04”,
                “localizacao”: “Sintra”, // Era “Oeiras”
                “uashers”: null
        }
}
```

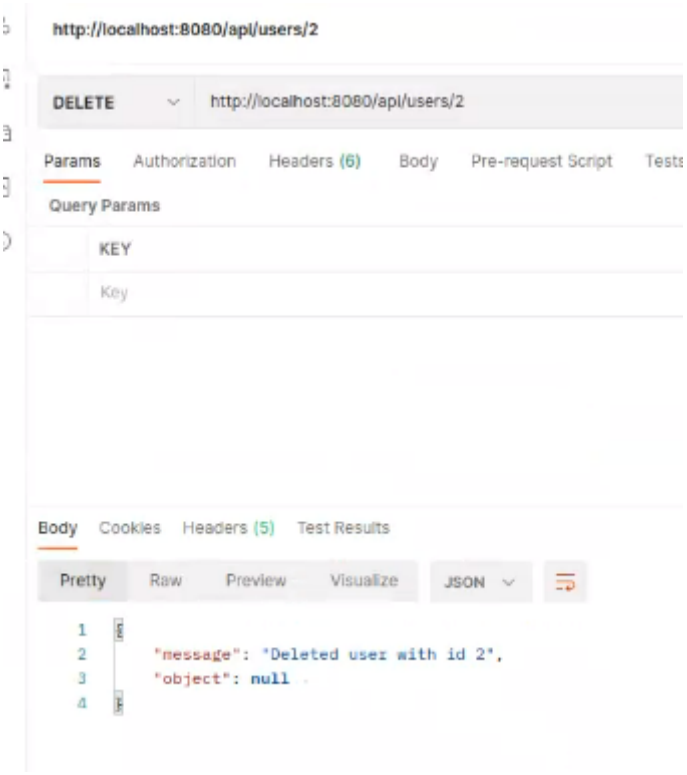
Sem códigos de erro programados

Exemplo



Apagar User (delete): /api/users/{id:[0-9]+}
Id: O Id do User desejado.
Apaga o usuário desejado.

Resultado Esperado / Exemplo

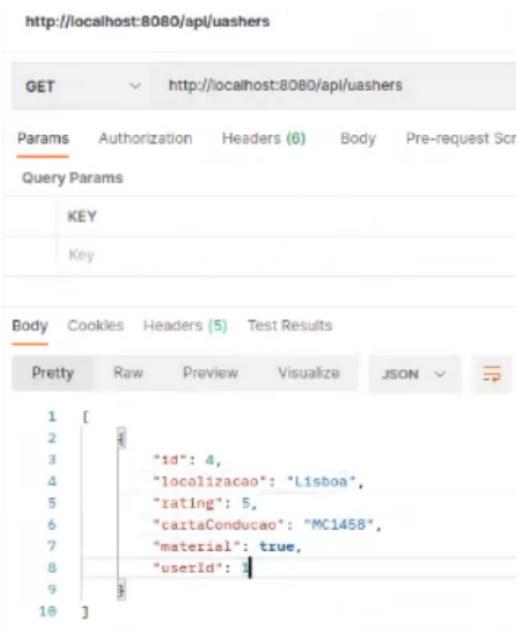


Sem códigos de erro programados

Uashers (/api/uashers)

Exibir todos os Uashers (get): /api/uashers/
Sem parâmetros adicionais.
Devolve uma tabela com todos os uashers e seus dados.

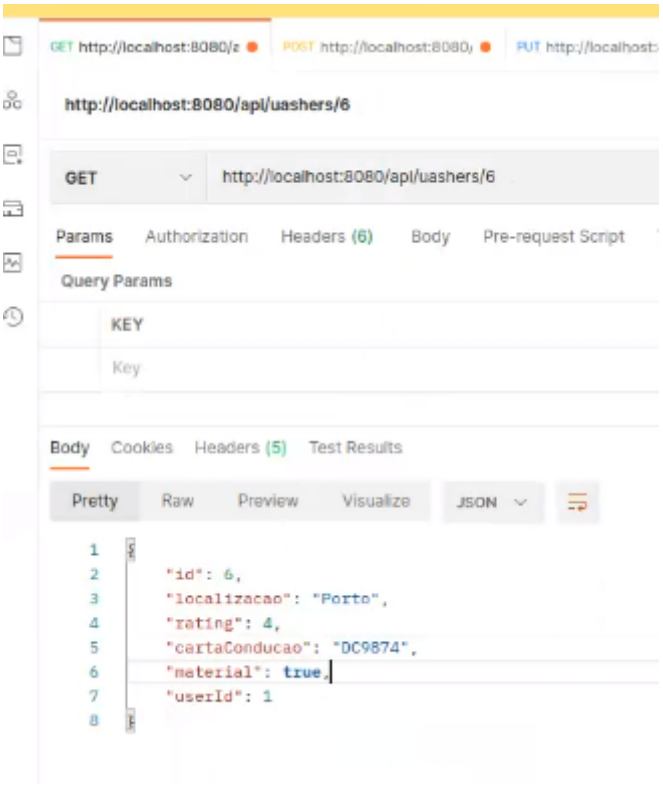
Resultado Esperado / Exemplo



Sem códigos de erro programados

Exibir Uasher por Id (get): /api/uashers/{id:[0-9]+}
Id: O Id do Uasher desejado.
Devolve os dados do uasher desejado.

Resultado Esperado / Exemplo

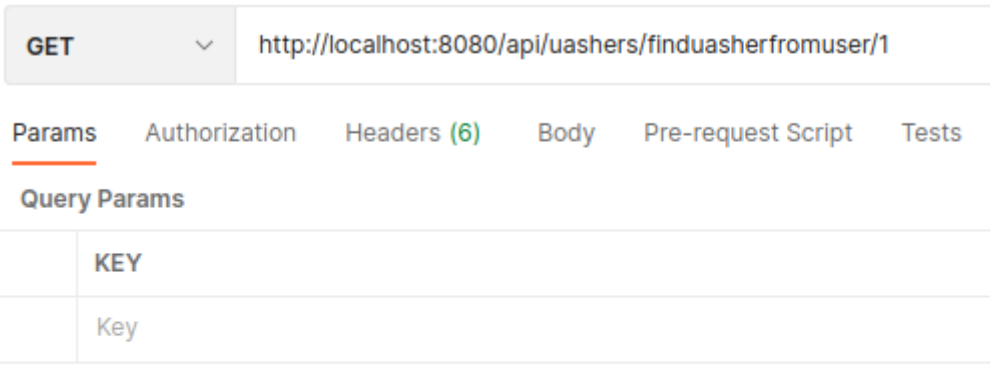


Sem códigos de erro programados

Exibir Uasher por User(get): /api/uashers/finduasherfromuser/{id:[0-9]+}

Id: O Id do Uasher desejado.
Utilizado logo após o login para verificar se o usuário já tem perfil uasher

Resultado Esperado / Exemplo



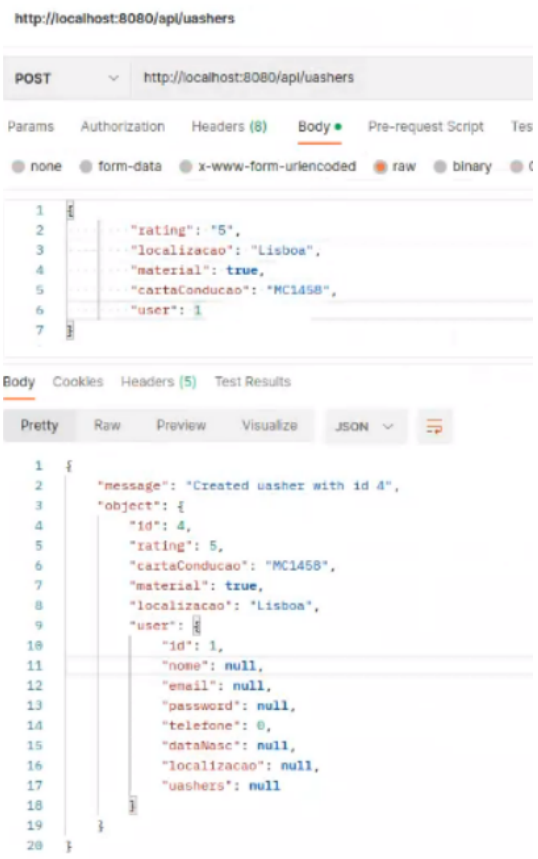
Adicionar Uasher (post): /api/uashers/
Sem parâmetros adicionais. Um JSONObject Uasher deverá ser providenciado.

Resultado Esperado


```
{
        “message”: “Created uasher with id 4”,
        “object”: {
                “id”: 4,
                “rating”: 5,
                “cartaConducao”: “MC1458”,
                “material”: true,
                “localizacao”: “Lisboa”,
                “user”: { … }
        }
}
```

Sem códigos de erro programados

Exemplo



Atualizar Uasher (put): /api/uashers/{id:[0-9]+}
Id: O Id do Uasher desejado.
Atualiza o uasher com os dados providenciados, que devem ser em formato JSONArray.

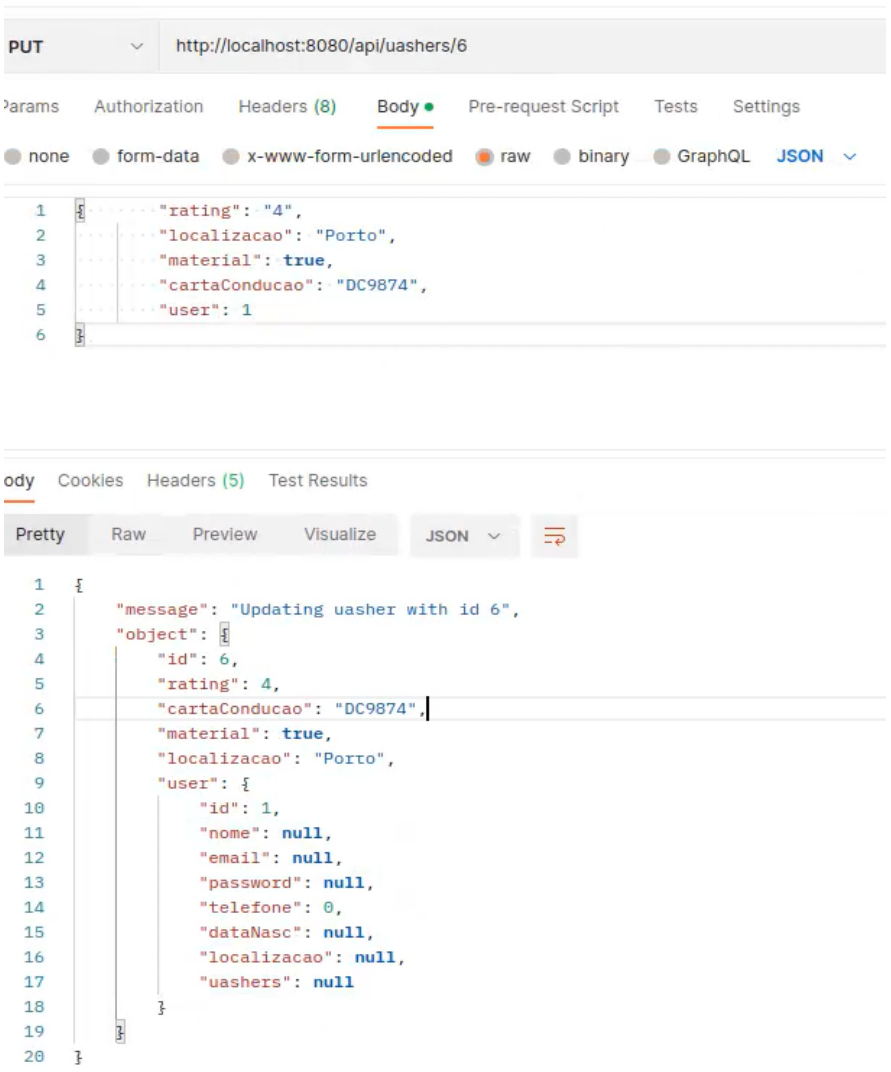
Resultado Esperado

```
{
        “message”: “Updating uasher with id 6”,
        “object”: {
                “id”: 6,
                “rating”: 4,
                “cartaConducao”: “DC9874”,
                “material”: true,
                “localizacao”: “Porto”,
                “user”: { … }
        }
}
```

Sem códigos de erro programados

Exemplo

http://localhost:8080/api/uashers/6



Atualizar rating do Uasher (put): /api/users/updaterating/{id:[0-9]+}

Id: O Id do Uasher desejado.
Atualiza a avaliação do uasher de acordo com a satisfação do usuário.

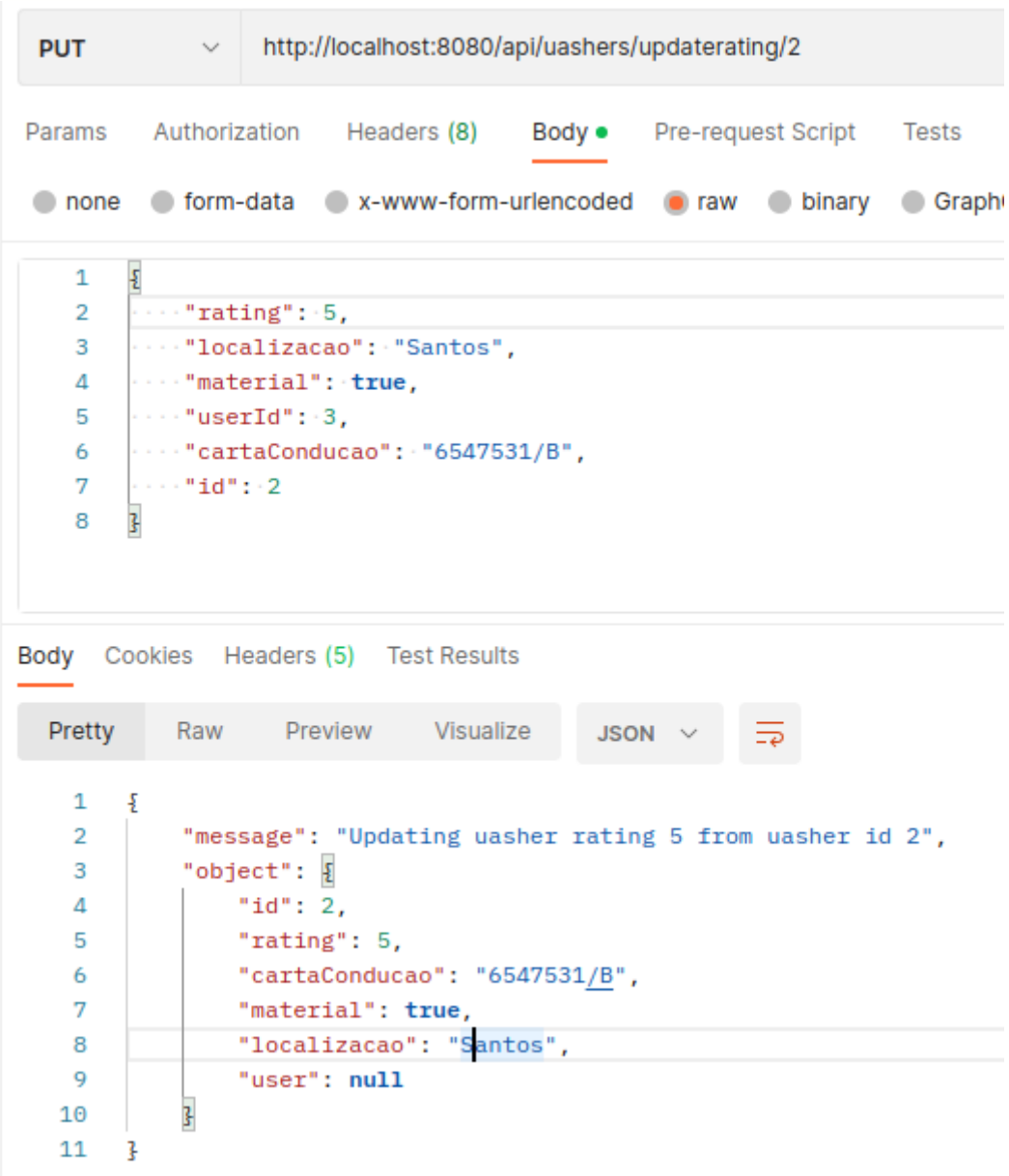
Resultado Esperado


```
"message": "Updating uasher rating 5 from uasher id 2",
   "object": {
       "id": 2,
       "rating": 5,
       "cartaConducao": "6547531/B",
```

```
        "material": true,
        "localizacao": "Santos",
        "user": null
    }
}
```

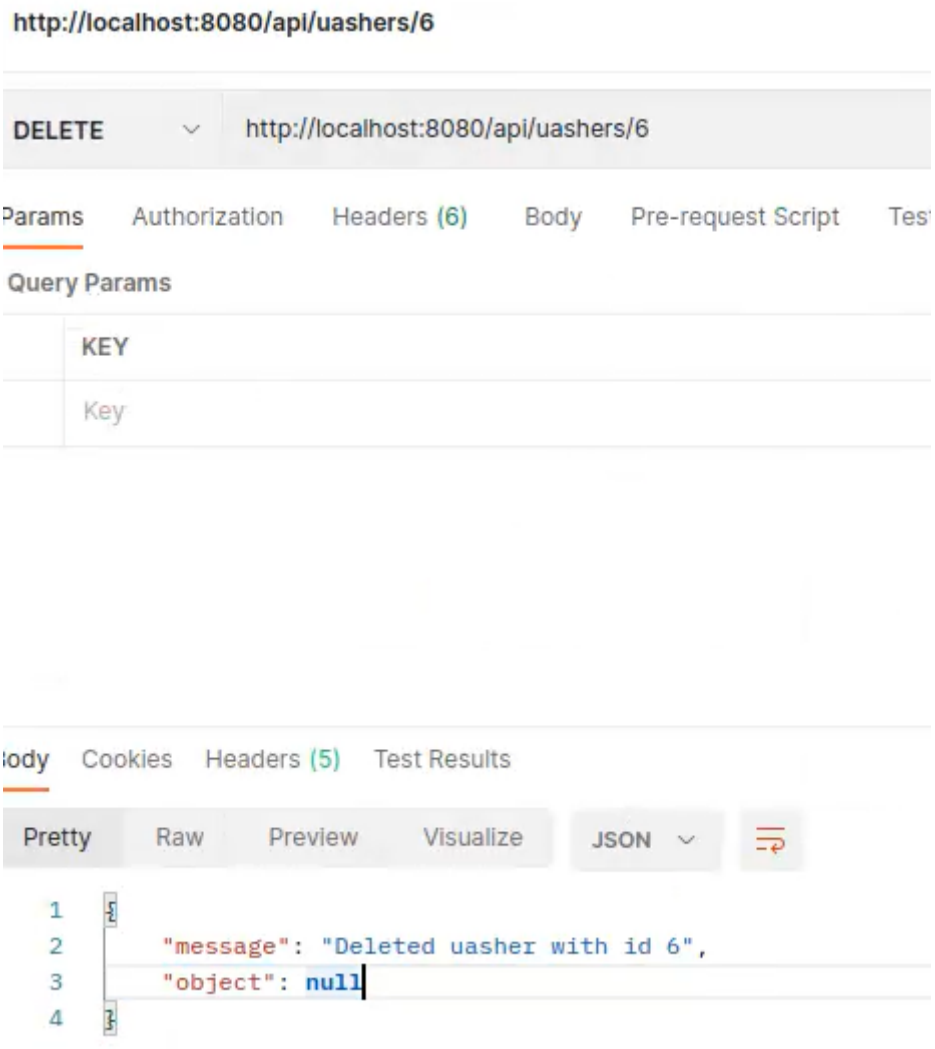

Sem códigos de erro programados

Exemplo



Apagar Uasher (delete): /api/uashers/{id:[0-9]+}
Id: O Id do Uasher desejado.
Apaga o uasher desejado.

Resultado Esperado / Exemplo

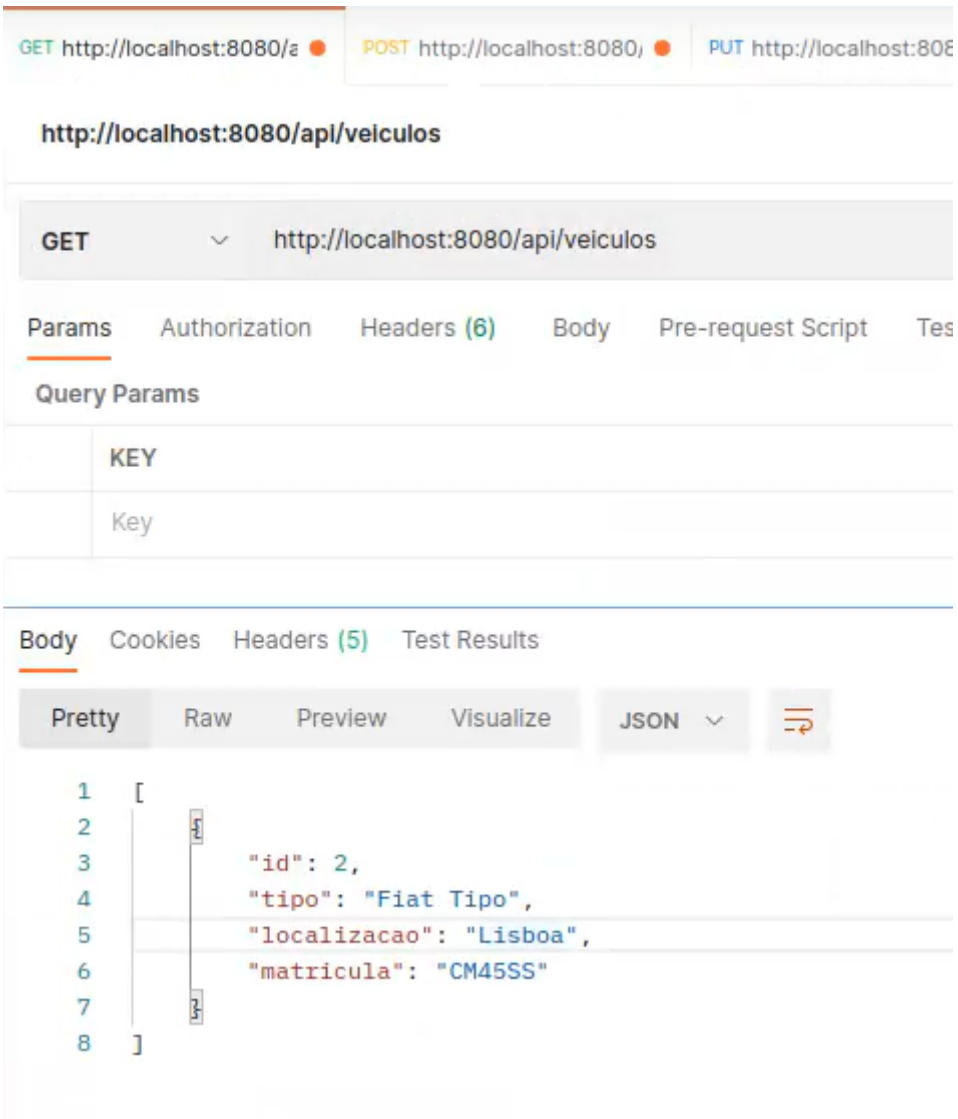


Sem códigos de erro programados

Veículos (/api/veiculos)

Exibir todos os Veículos (get): /api/veiculos/
Sem parâmetros adicionais.
Devolve uma tabela com todos os veículos e seus dados.

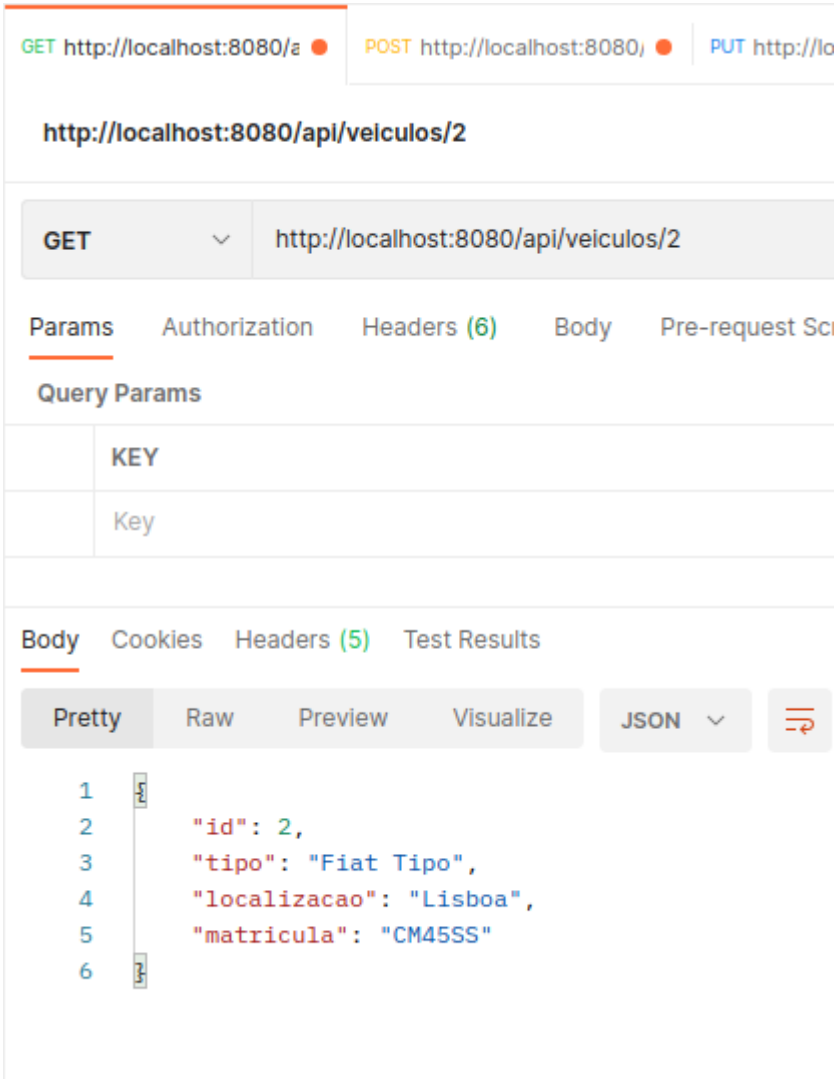
Resultado Esperado / Exemplo



Sem códigos de erro programados

Exibir Veículo por Id (get): /api/veiculos/{id:[0-9]+}
Id: O Id do Veículo desejado.
Devolve os dados do veículo desejado.

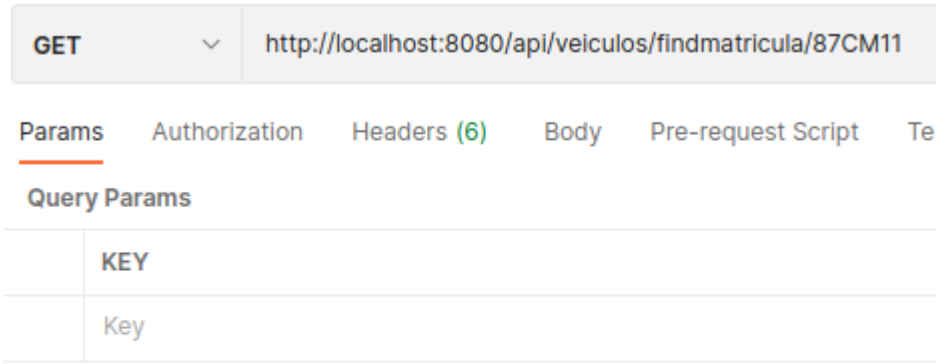
Exemplo



Exibir Veículo por matricula (get): /api/veiculos/findmatricula/{matricula}

Id: O Id do Veículo desejado.
Devolve o veículo com a matrícula requisitada

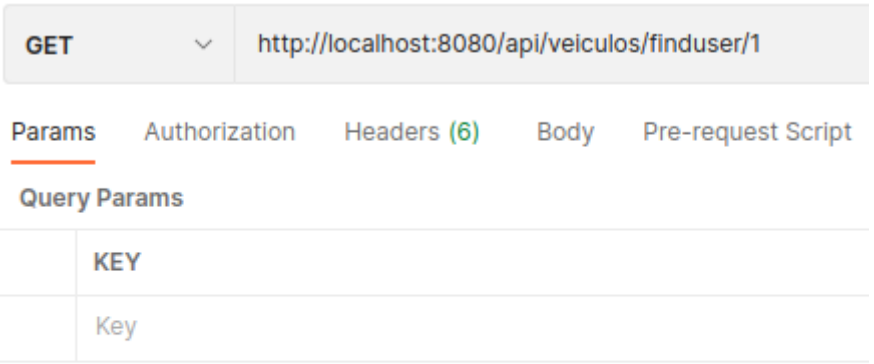
Resultado Esperado/Exemplo

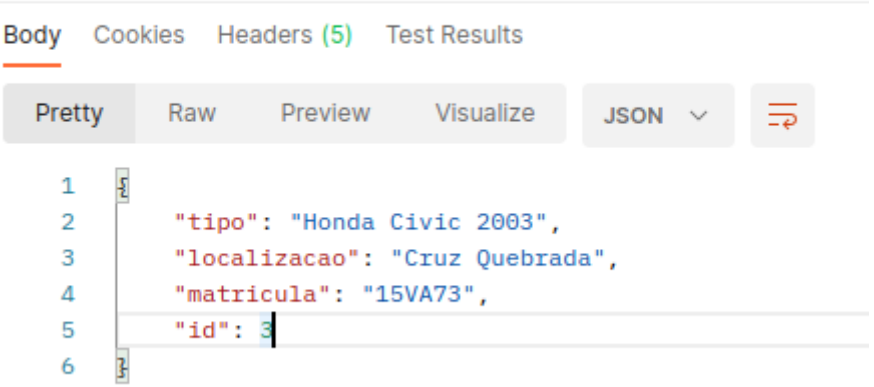


Exibir Veículo por usuario (get): /api/veículos/finduser/{id:[0-9]+}

Id: O Id do Veículo desejado.
Devolve o id do usuário proprietário do veículo

Resultado Esperado/Exemplo





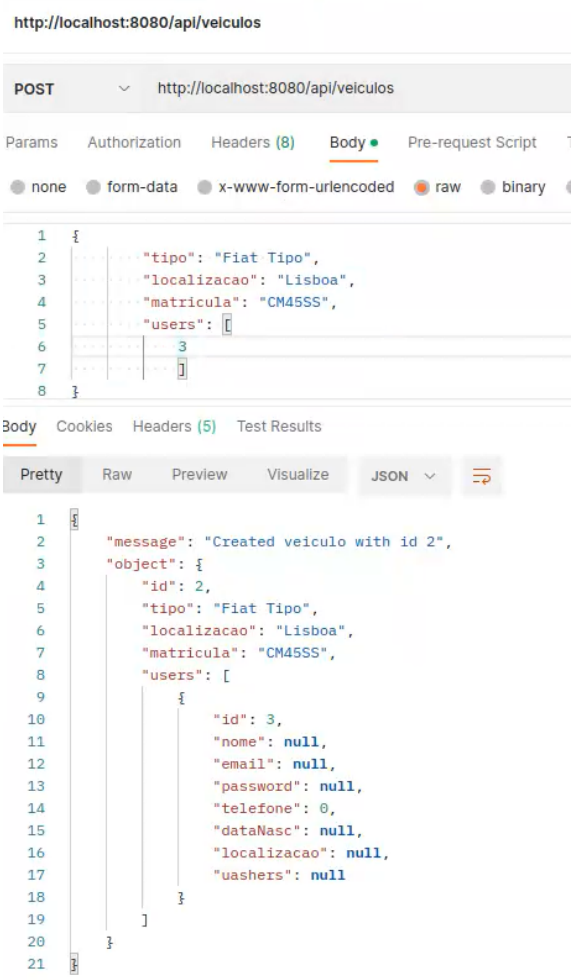
Adicionar Veiculo (post): /api/veiculos/
Sem parâmetros adicionais. Um JSONObject Veiculo deverá ser providenciado.

Resultado Esperado

```
{
        “message”: “Created veiculo with id 2”,
        “object”: {
                “id”: 2,
                “tipo”: “Fiat Tipo”,
                “localizacao”: “Porto”,
                “matricula”: “CM45SS”,
                “users”: { … }
        }
}
```

Sem códigos de erro programados

Exemplo



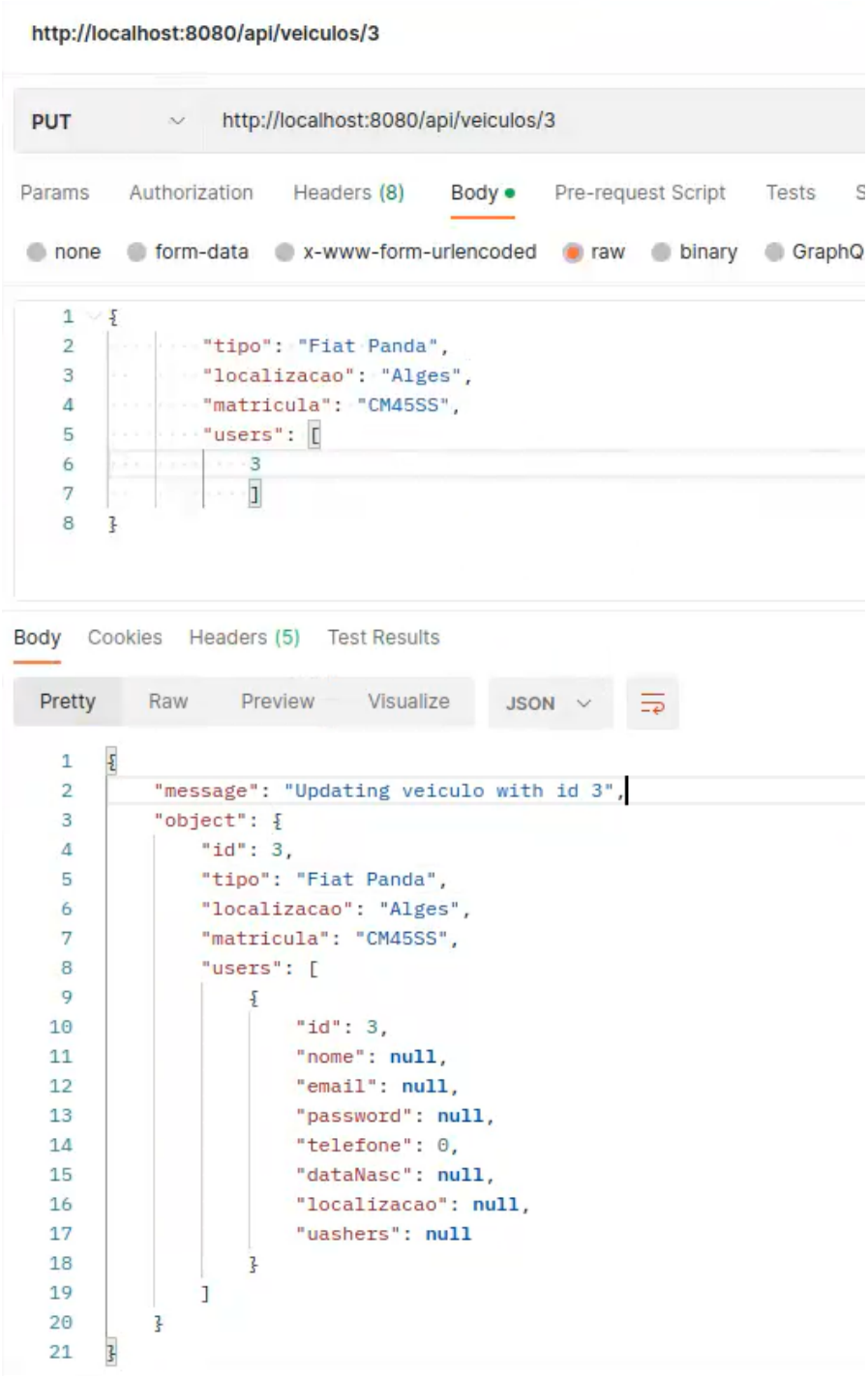
Atualizar Veiculo (put): /api/veiculos/{id:[0-9]+}
Id: O Id do Veiculo desejado.
Atualiza o veículo com os dados providenciados, que devem ser em formato JSONArray.

Resultado Esperado

```
{
        “message”: “Updating veiculo with id 2”,
        “object”: {
                “id”: 2,
                “tipo”: “Fiat Panda”,
                “localizacao”: “Alges”,
                “matricula”: “CM45SS”,
                “users”: { … }
        }
}
```

Sem códigos de erro programados

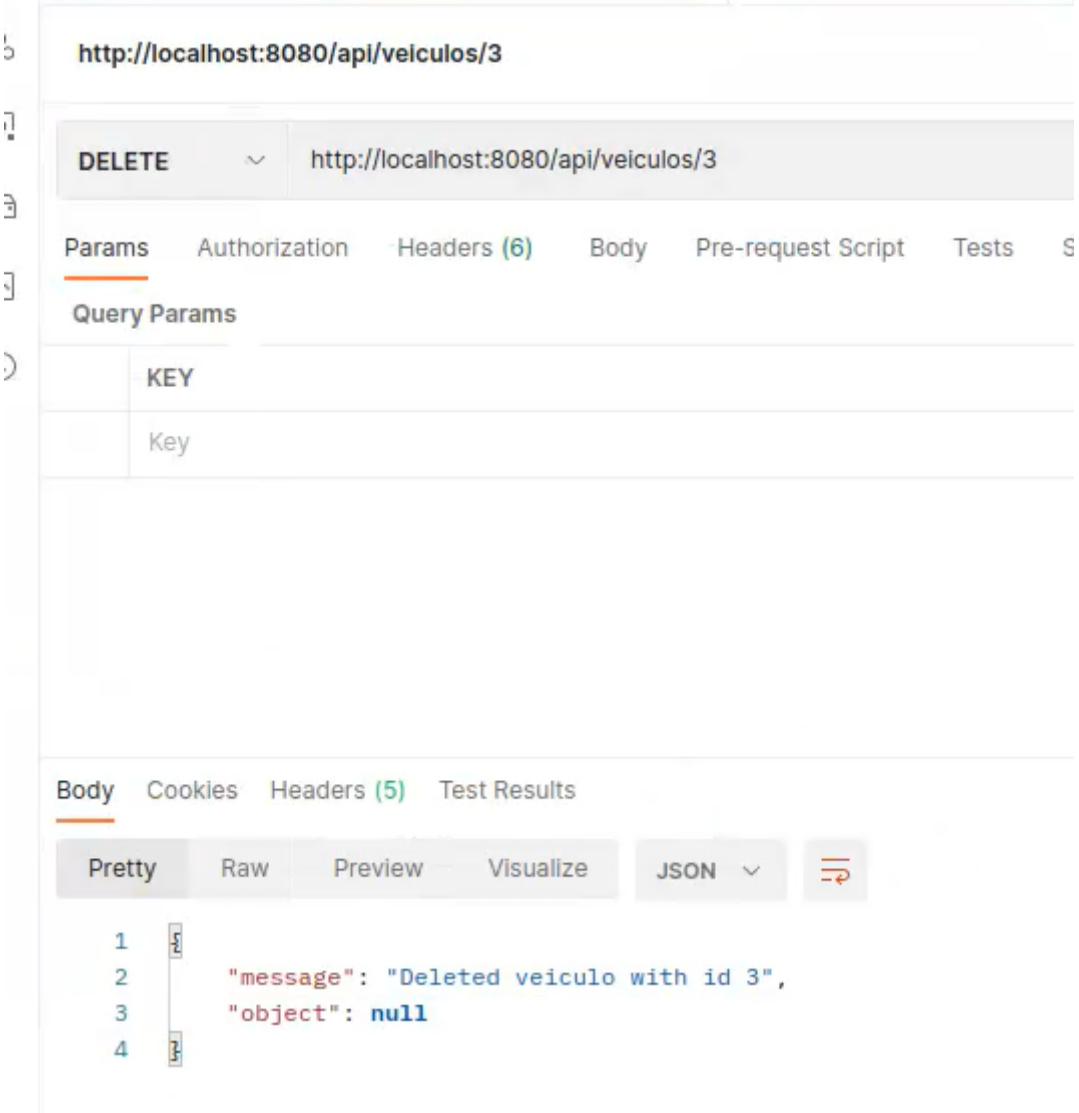
Exemplo



Apagar Veiculo (delete): /api/veiculos/{id:[0-9]+}

Id: O Id do Veiculo desejado.
Apaga o veículo desejado.

Resultado Esperado / Exemplo

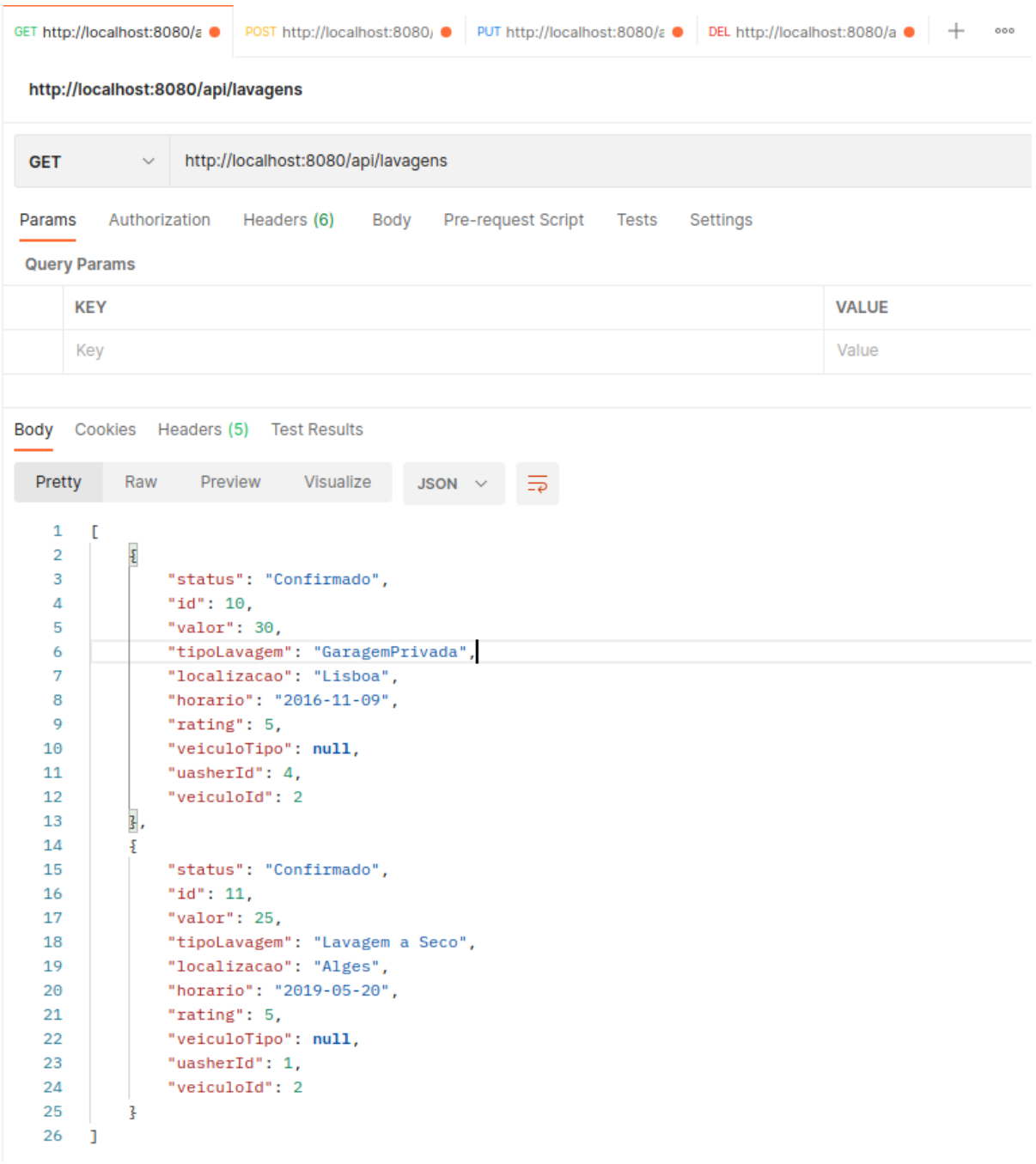


Sem códigos de erro programados

Lavagens (/api/lavagens)

Exibir todas as Lavagens (get): /api/lavagens/
Sem parâmetros adicionais.
Devolve uma tabela com todas as lavagens e seus dados.

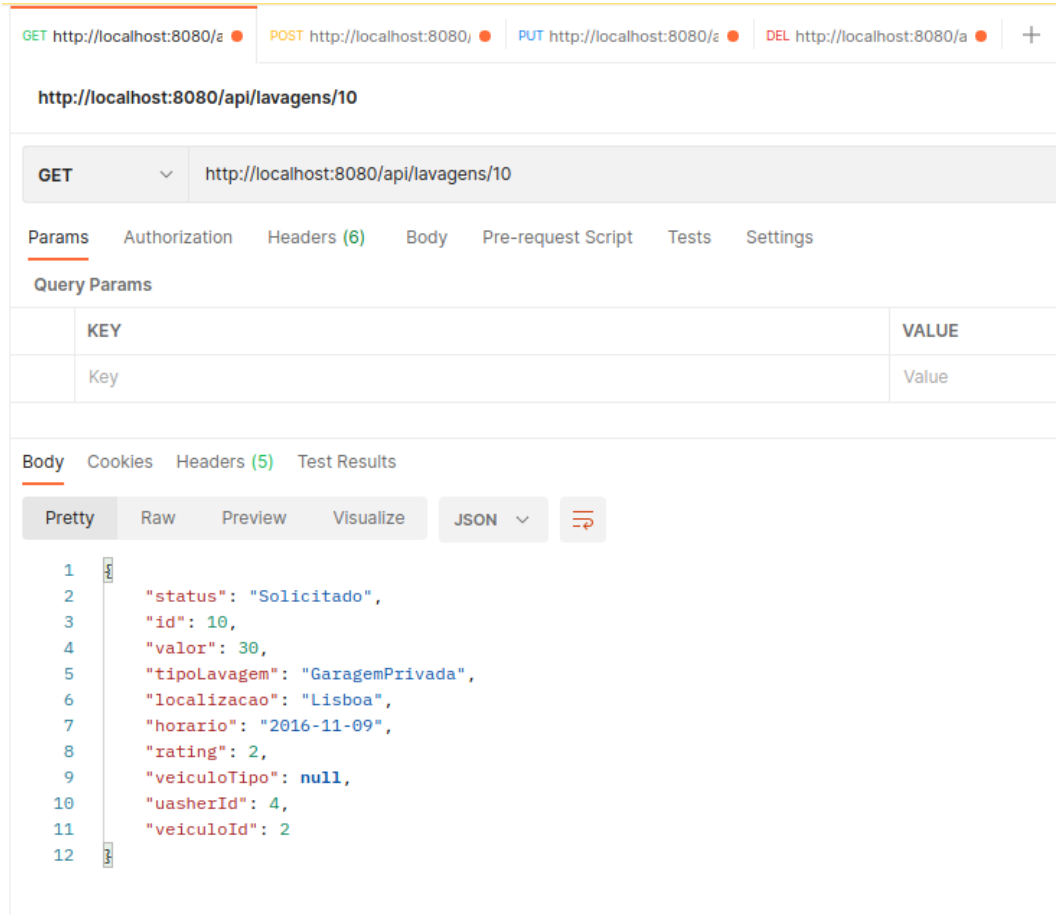
Resultado Esperado / Exemplo



Sem códigos de erro programados

Exibir Lavagem por Id (get): /api/lavagens/{id:[0-9]+}
Id: O Id da Lavagem desejada.
Devolve os dados da lavagem desejada.

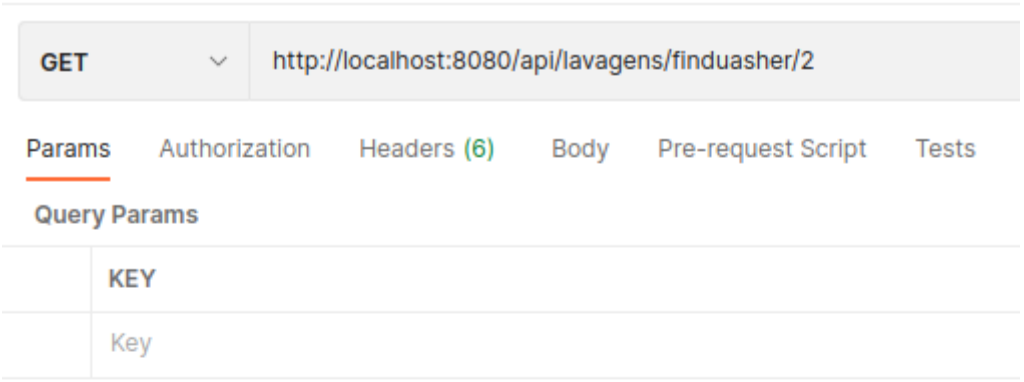
Resultado Esperado / Exemplo

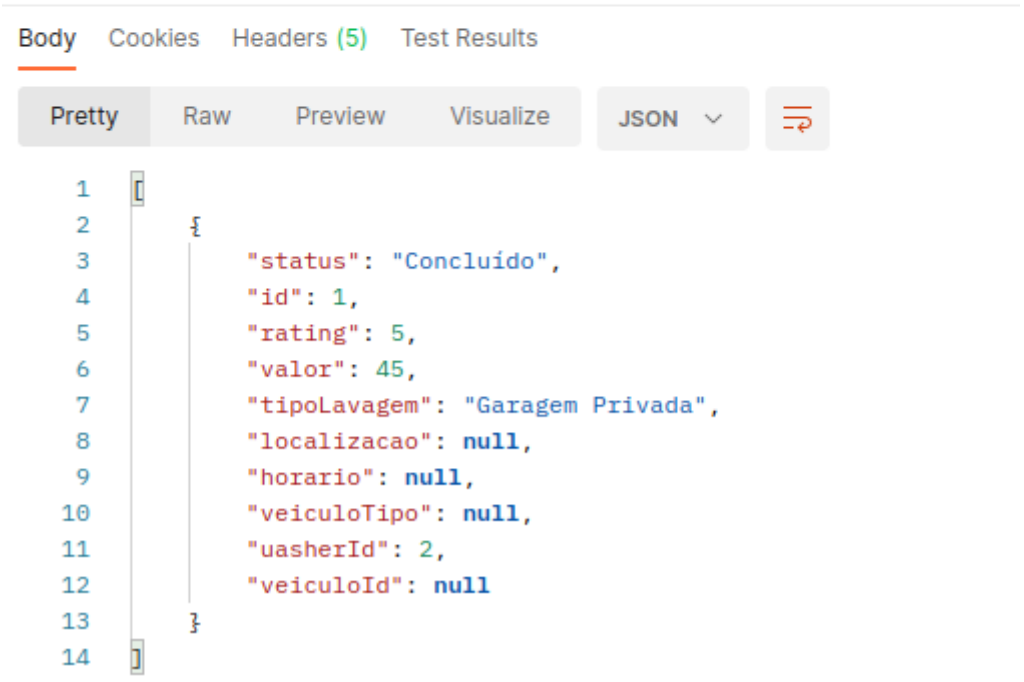


Exibir Lavagem por uasher (get): /api/lavagens/`finduasher/{id:[0-9]+}`

Id: O Id da Lavagem desejada.
Devolve as lavagens realizadas pelos uashers.

Resultado Esperado / Exemplo





Sem erros no código

Adicionar Lavagem (post): /api/lavagens/
Sem parâmetros adicionais. Um JSONObject Lavagem deverá ser providenciado.

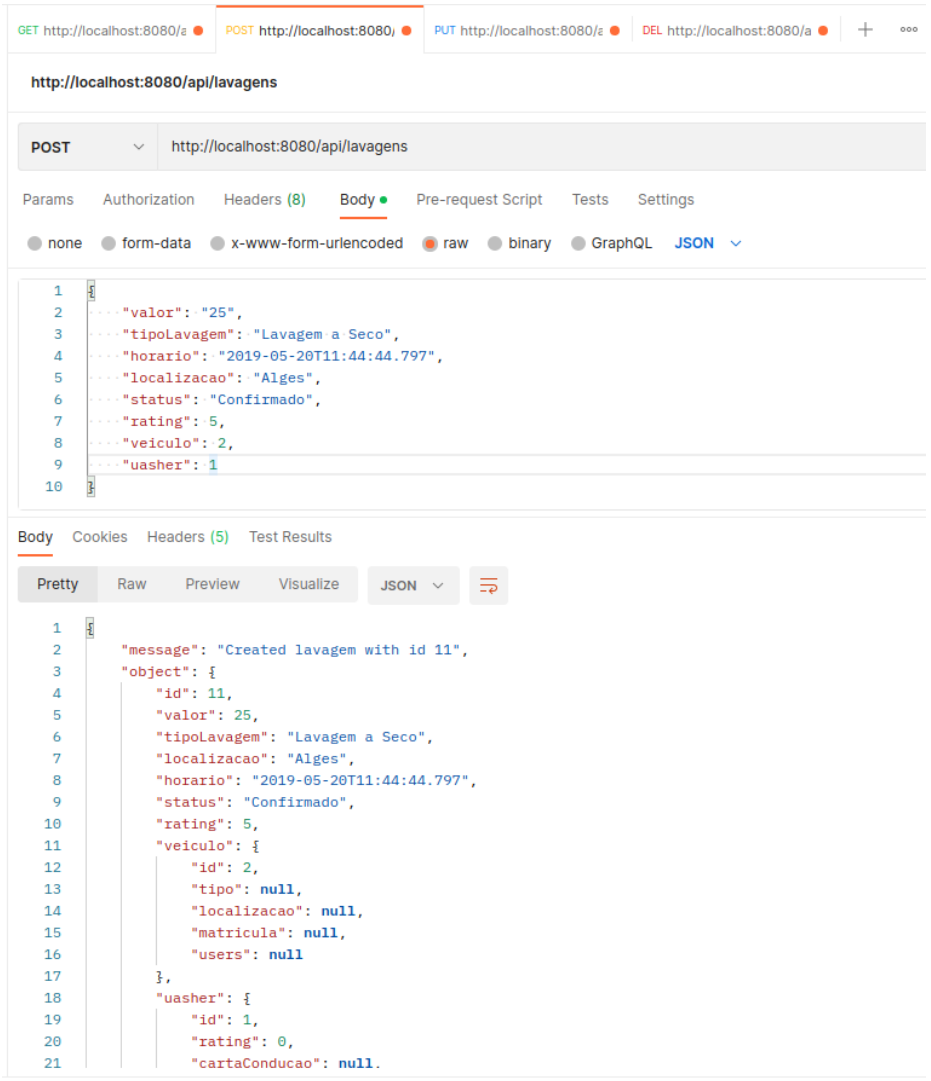
Resultado Esperado


```
{
  "valor": "25",
  "tipoLavagem": "Lavagem a Seco",
  "horario": "2019-05-20T11:44:44.797",
```

```
  "localizacao": "Alges",
  "status": "Confirmado",
  "rating": 5,
  "veiculo": 2,
  "uasher": 1
}
```


Sem códigos de erro programados

Exemplo



Sem códigos de erro programados

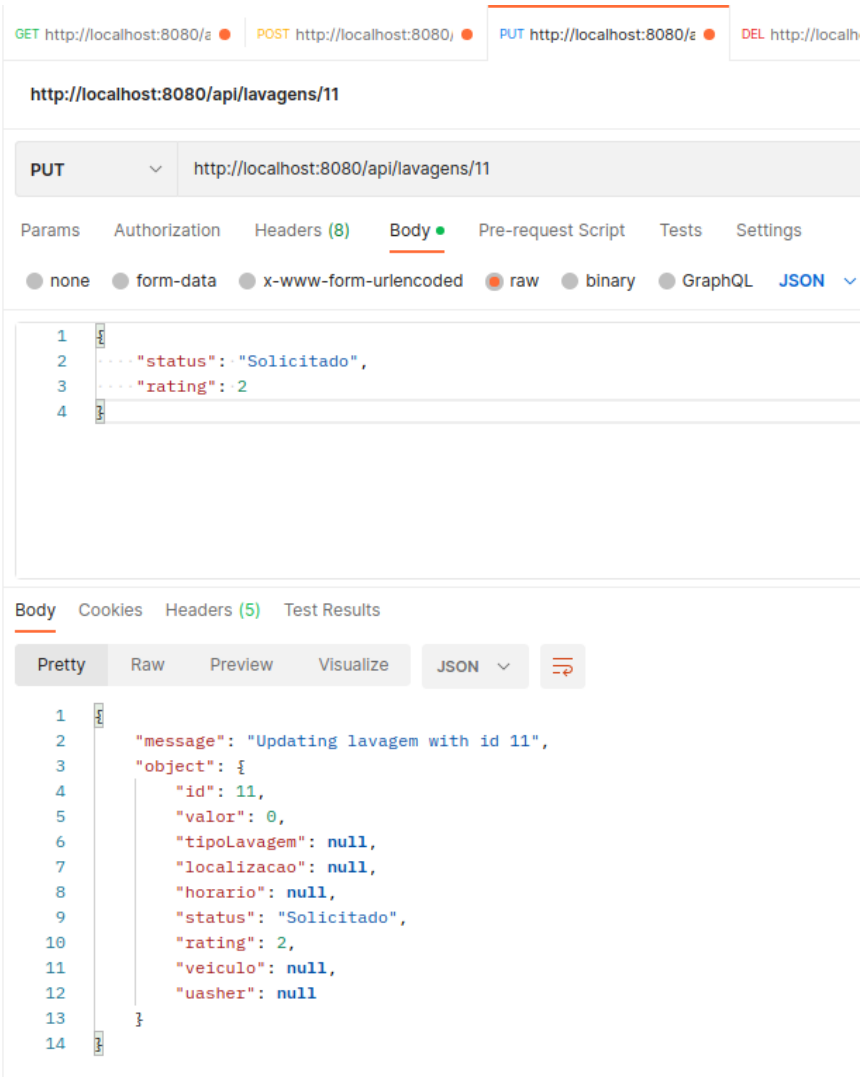
Atualizar Lavagem (put): /api/lavagens/{id:[0-9]+}
Id: O Id da Lavagem desejada.
Atualiza a lavagem com os dados providenciados, que devem ser em formato JSONArray.

Resultado Esperado

```
{
  "status": "Solicitado", //Antes estava Confirmado
  "rating": 2 //Antes estava 5
}
```

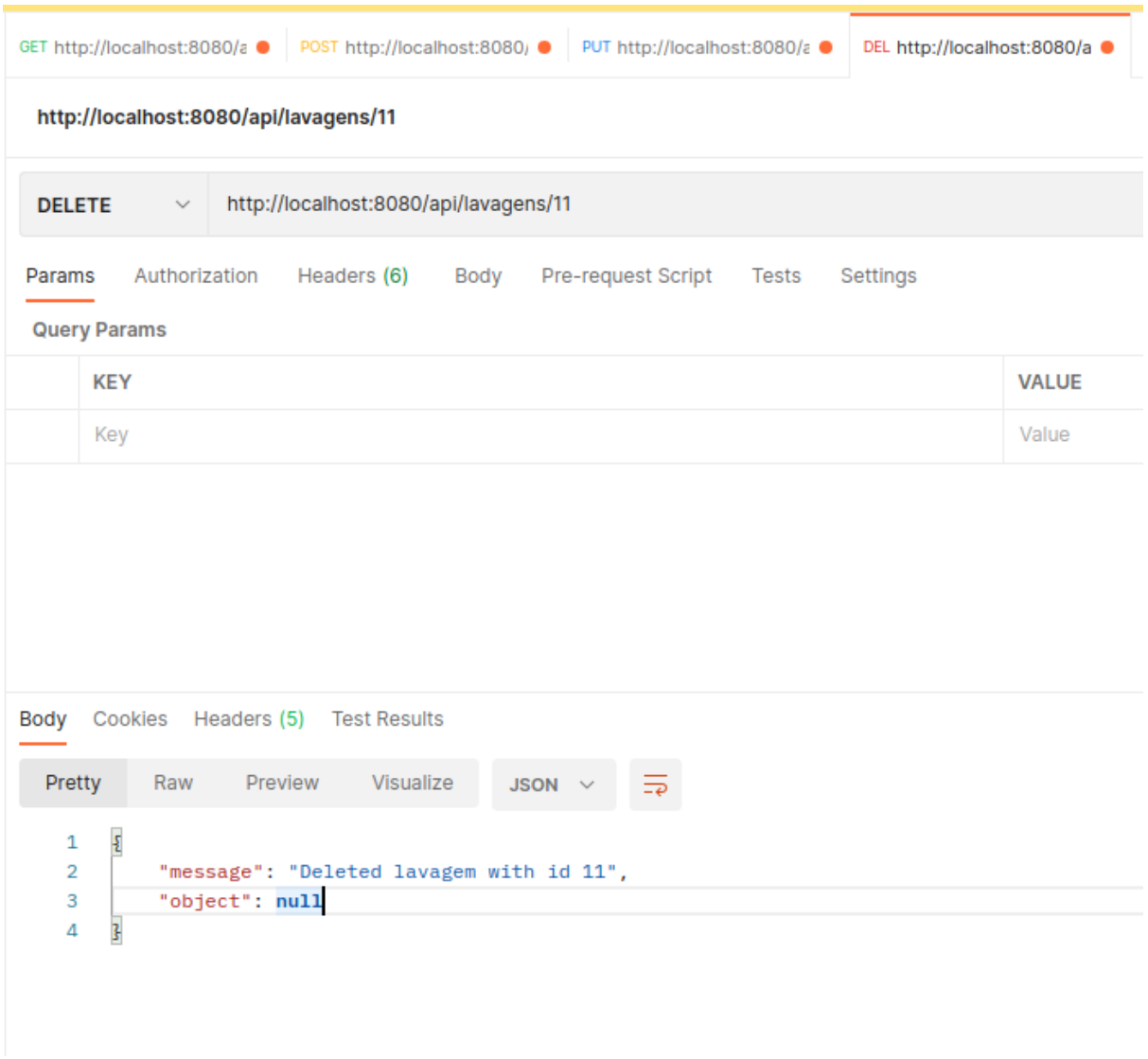
Sem códigos de erro programados

Exemplo



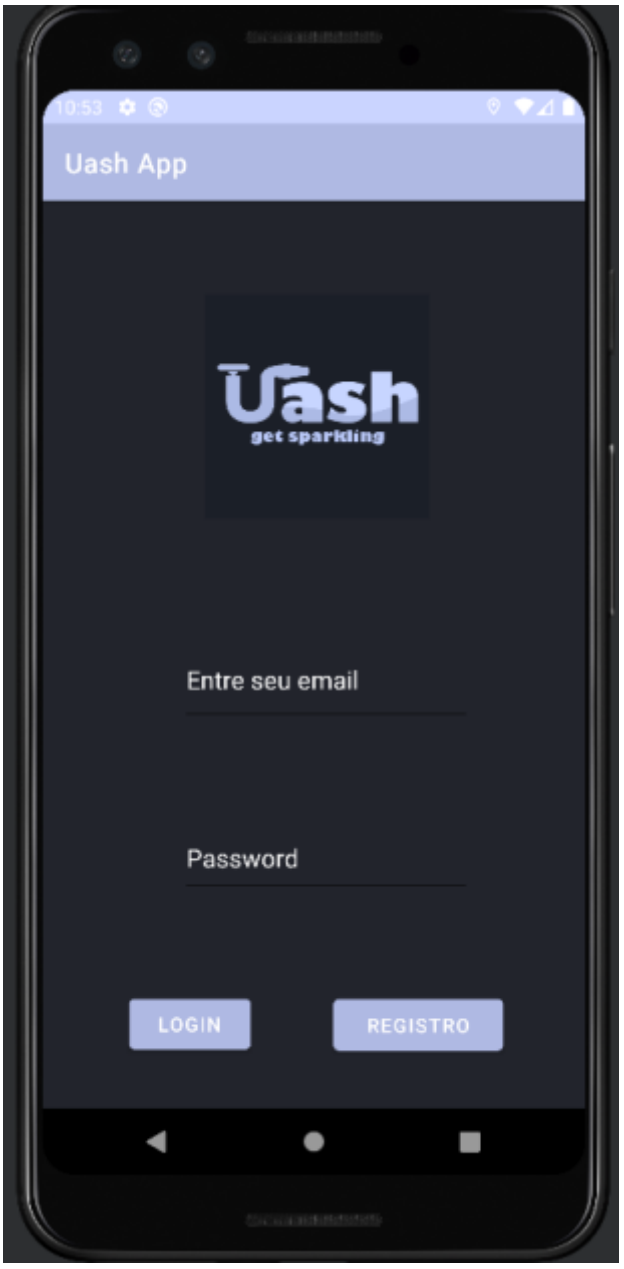
Apagar Lavagem (delete): /api/lavagens/{id:[0-9]+}
Id: O Id da Lavagem desejada.
Apaga a lavagem desejada.

Resultado Esperado/ Exemplo

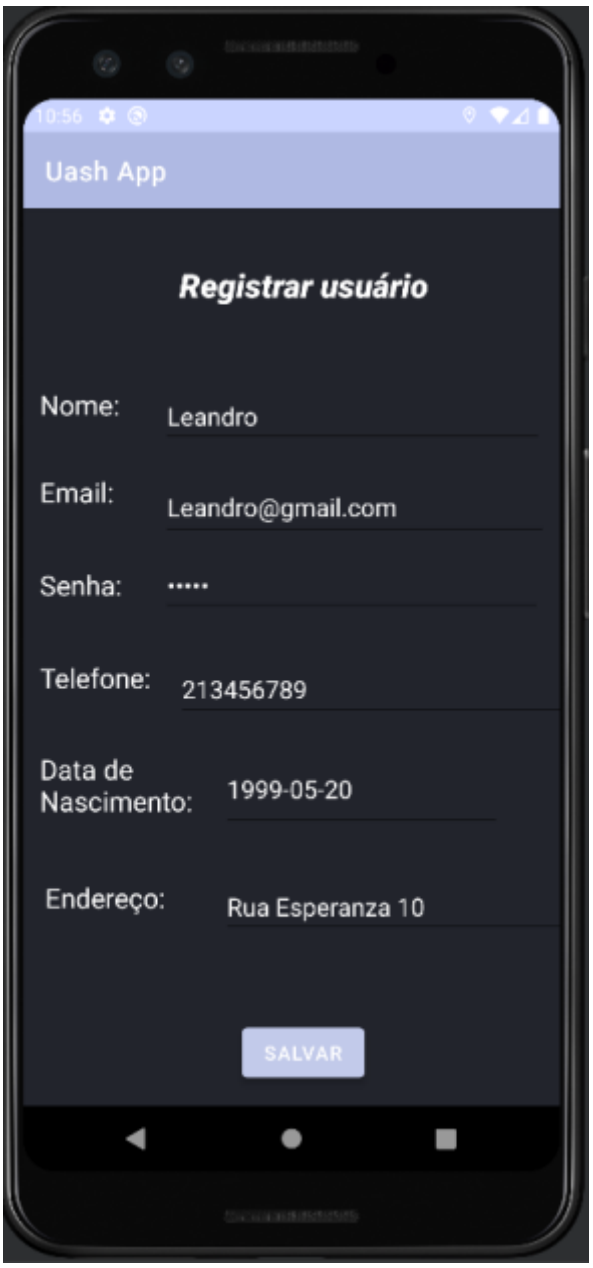


Sem códigos de erro programados

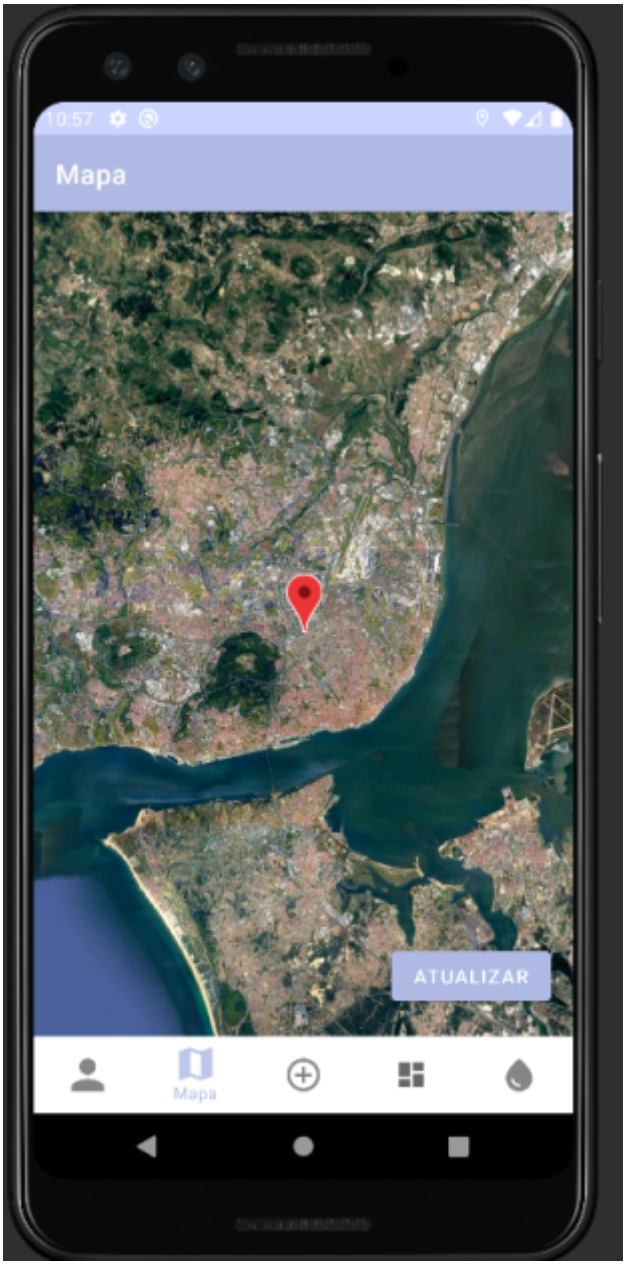
Guia do Aplicativo



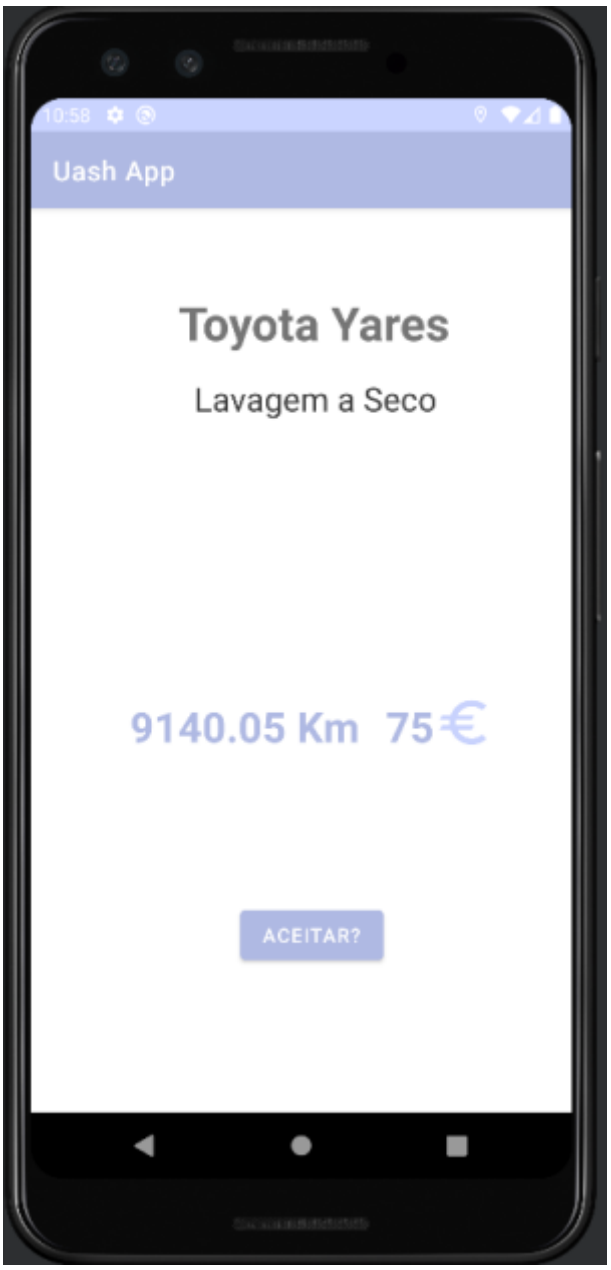
Página de Login, onde o usuário que já estiver cadastrado terá acesso a sua conta. Caso o usuário não tenha a conta registrada, terá acesso ao botão de registro.



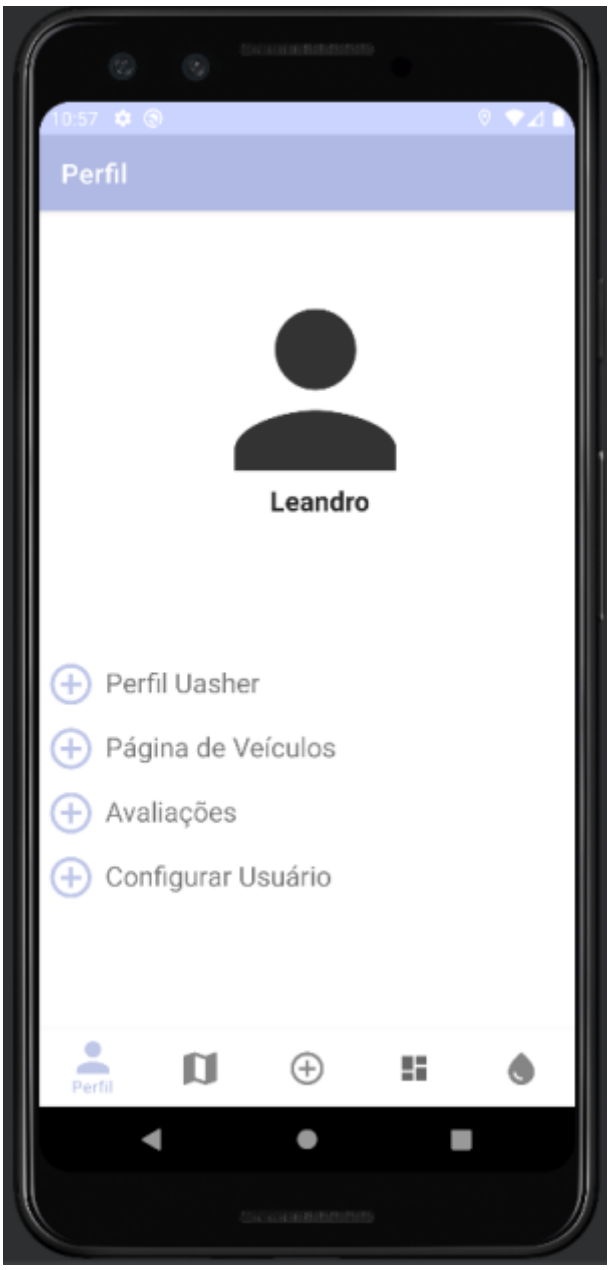
Neste página, será necessário preencher o formulário com as informações necessárias.



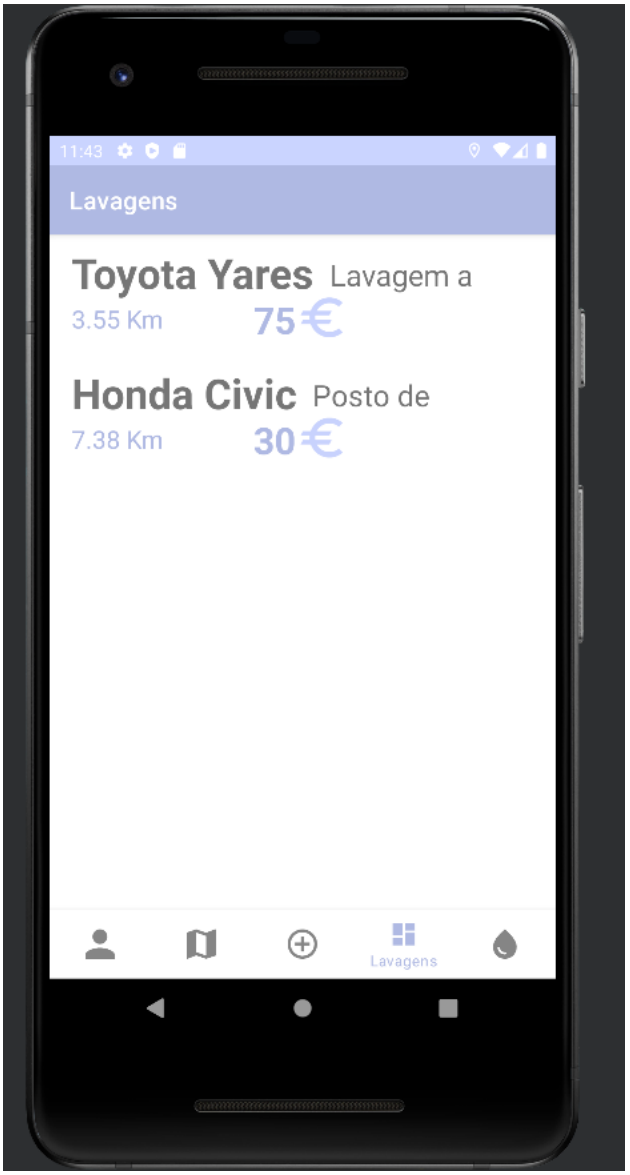
Logo em seguida, poderá ver o mapa com a sua localização atual. O uasher deverá então carregar no marcador do mapa. Apenas lavagens que deverão ser realizadas nos próximos 30 minutos aparecerão no mapa.



Assim que o uasher seleciona o marcador no mapa, mostra detalhes da lavagem, como por o tipo da lavagem, valor e etc.



No menu principal, podemos ver onde algumas opções do utilizador, como por exemplo configurar sua conta para ser um uasher.



Na página de lavagens, é possível ver todas as lavagens pendentes em forma de lista, independente da hora em que estiverem agendadas.