|  |  |  |  |
| --- | --- | --- | --- |
| A picture containing icon  Description automatically generated | | Curso TeSP em  Programação de Sistemas de Informação  SIS | |
|  |  | |  |
| **Ano Letivo 2021/2022** | **Av. Periódica** | | **2º Ano, 1º Semestre** |

|  |
| --- |
| **PROJETO DE SERVIÇOS E INTEROPERABILIDADE DE SISTEMAS** |

**Relatório de Acompanhamento do**

**Ciclo de Desenvolvimento de Software do Projeto em Serviços e Interoperabilidade de Sistemas**



|  |  |  |
| --- | --- | --- |
| **Turno**: PL1 | **Grupo**: A | **Docente**: Mário Fernandes |
| **Nº 2202415** | Andreia Agostinho | |
| **Nº 2201126** | Diogo Pereira | |
| **Nº 2201131** | Gonçalo Ferreira | |

Instituto Politécnico de Leiria

Torres Vedras

Curso Técnico Superior Profissional de Programação de Sistemas de Informação

10 de Janeiro de 2022

Andreia Agostinho

Diogo Pereira

Gonçalo Ferreira

**Índice**

[1 Introdução 7](#_Toc92466931)

[2 Endpoints 8](#_Toc92466932)

[2.1 Cars 8](#_Toc92466933)

[2.1.1 Métodos de Implementação 8](#_Toc92466934)

[2.2 Companies 13](#_Toc92466935)

[2.2.1 Métodos de Implementação 13](#_Toc92466936)

[2.3 CompaniesList 16](#_Toc92466937)

[2.3.1 Métodos de Implementação 16](#_Toc92466938)

[2.4 Contributors 17](#_Toc92466939)

[2.4.1 Métodos de Implementação 17](#_Toc92466940)

[2.5 Login 20](#_Toc92466941)

[2.5.1 Métodos de Implementação 20](#_Toc92466942)

[2.6 Repairs 21](#_Toc92466943)

[2.6.1 Métodos de Implementação 21](#_Toc92466944)

[2.7 Schedules 26](#_Toc92466945)

[2.7.1 Métodos de Implementação 26](#_Toc92466946)

[2.8 Signup 30](#_Toc92466947)

[2.8.1 Métodos de Implementação 30](#_Toc92466948)

[2.9 User 31](#_Toc92466949)

[2.9.1 Métodos de Implementação 31](#_Toc92466950)

[2.10 Informações 34](#_Toc92466951)

[3 Mosquitto 35](#_Toc92466952)

[3.1 Introdução ao Mosquitto 35](#_Toc92466953)

[3.2 Aplicação Prática 35](#_Toc92466954)

[4 Conclusão 38](#_Toc92466955)

**Índice DE FIGURAS**

[Figura 1 - Cars, Método GET 8](#_Toc92467052)

[Figura 2 - Cars, Método POST 9](#_Toc92467053)

[Figura 3 - Cars, Método SET CUSTOM 9](#_Toc92467054)

[Figura 4 - Cars, Método TOTAL CUSTOM 10](#_Toc92467055)

[Figura 5 - Cars, Método TOTAL USER CUSTOM 10](#_Toc92467056)

[Figura 6 - Cars, Método DELETE 10](#_Toc92467057)

[Figura 7 - Cars, Método PUT 11](#_Toc92467058)

[Figura 8 - Cars, Método DELETE CUSTOM 11](#_Toc92467059)

[Figura 9 - Cars, Método POST CUSTOM 12](#_Toc92467060)

[Figura 10 - Cars, Método PUT CUSTOM 12](#_Toc92467061)

[Figura 11 - Companies, Método GET 13](#_Toc92467062)

[Figura 12 - Companies, Método POST 13](#_Toc92467063)

[Figura 13 - Companies, Método PUT 14](#_Toc92467064)

[Figura 14 - Companies, Método DELETE 14](#_Toc92467065)

[Figura 15 - Companies, Método TOTAL CUSTOM 15](#_Toc92467066)

[Figura 16 - Companies, Método SET CUSTOM 15](#_Toc92467067)

[Figura 17 - CompaniesList, Método GET 16](#_Toc92467068)

[Figura 18 - Contributors, Método GET 17](#_Toc92467069)

[Figura 19 - Contributors, Método POST 17](#_Toc92467070)

[Figura 20 - Contributors, Método PUT 18](#_Toc92467071)

[Figura 21 - Contributors, Método DELETE 18](#_Toc92467072)

[Figura 22 - Contributors, Método SET CUSTOM 19](#_Toc92467073)

[Figura 23 - Contributors, Método TOTAL CUSTOM 19](#_Toc92467074)

[Figura 24 - Login, Método DO CUSTOM 20](#_Toc92467075)

[Figura 25 - Repairs, Método GET 21](#_Toc92467076)

[Figura 26 - Repairs, Método HISTORY CUSTOM 22](#_Toc92467077)

[Figura 27 - Repairs, Método REPAIR CONTRIBUTOR CUSTOM 22](#_Toc92467078)

[Figura 28 - Repairs, Método POST CUSTOM 23](#_Toc92467079)

[Figura 29 - Repairs, Método PUT CUSTOM 23](#_Toc92467080)

[Figura 30 - Repairs, Método DELETED CUSTOM 24](#_Toc92467081)

[Figura 31 - Repairs, Método TOTAL CUSTOM 24](#_Toc92467082)

[Figura 32 - Repairs, Método SET CUSTOM 25](#_Toc92467083)

[Figura 33 - Schedules, Método TOTAL CUSTOM 26](#_Toc92467084)

[Figura 34 - Schedules, Método SET CUSTOM 26](#_Toc92467085)

[Figura 35 - Schedules, Método POST CUSTOM 27](#_Toc92467086)

[Figura 36 - Schedules, Método DELETE CLIENT CUSTOM 27](#_Toc92467087)

[Figura 37 - Schedules, Método DELETE CONTRIBUTOR CUSTOM 28](#_Toc92467088)

[Figura 38 - Schedules, Método PUT CLIENT CUSTOM 28](#_Toc92467089)

[Figura 39 - Schedules, Método PUT CONTRIBUTOR CUSTOM 29](#_Toc92467090)

[Figura 40 - Signup, Método POST CUSTOM 30](#_Toc92467091)

[Figura 41 - User, Método ACCOUNT CUSTOM 31](#_Toc92467092)

[Figura 42 - User, Método TOTAL CUSTOM 31](#_Toc92467093)

[Figura 43 - User, Método SET CUSTOM 32](#_Toc92467094)

[Figura 44 - User, Método DELETED CUSTOM 32](#_Toc92467095)

[Figura 45 - User, Método PUT CUSTOM 33](#_Toc92467096)

[Figura 46 - Mosquitto, Função afterSave 35](#_Toc92467097)

[Figura 47- Mosquitto, Função FazPublish 36](#_Toc92467098)

[Figura 48 - Mosquitto, Função actionDo e Função FazSubscribe 37](#_Toc92467099)

# Introdução

No âmbito da unidade curricular de Serviços e Interoperabilidade de Sistemas do 1º Semestre do 2º ano do Curso Técnico Superior Profissional de Programação de Sistemas de Informação do Instituto Politécnico de Leiria, foi criado o projeto apelidado de “CarBuddy”, que consiste no desenvolvimento de uma plataforma de gestão de todo o tipo de veículos, desde as suas reparações às suas manutenções.

A nossa plataforma consiste numa combinação de três componentes, sendo eles um Website, uma Aplicação Móvel e uma API.

Assim, para a presente unidade curricular, o objetivo é o desenvolvimento da API, sendo esta uma API REST, para a unidade curricular de Plataformas de Sistemas de Informação, o objetivo é a criação e desenvolvimento do Website e para a unidade curricular de Acesso Móvel a Sistemas de Informação, o objetivo é a criação de uma aplicação móvel.

No website, é onde ocorre a gestão de contas de utilizador, onde irão ser vistos/registados os veículos, o seu estado e as suas reparações.

Na aplicação, é onde cada cliente tem listadas todas as reparações e os respetivos veículos, sendo também possível realizar todas as ações do website na ótica do cliente.

Por fim, para que haja uma interligação de dados entre ambos os componentes referidos anteriormente, procedemos ao desenvolvimento da API REST.

O objetivo, é também que mesmo a nível internacional, se possam registar e verificar o histórico de reparações, centralizando assim todos os dados.

O projeto é comum entre ambas as disciplinas, mas as fases que traçam este projeto são divididas pelas três, conforme a sua pertinência.

Podemos referir também o facto de que o projeto tem como objetivo geral a consolidação de todo o conhecimento adquirido ao longo do curso.

# Endpoints

## Cars

### Métodos de Implementação

**Método GET**

Este método serve para obter a listagem de todos os veículos registados na plataforma e apenas o administrador consegue utilizar este método.

Relativamente ao Token, este é variável consoante o utilizador.

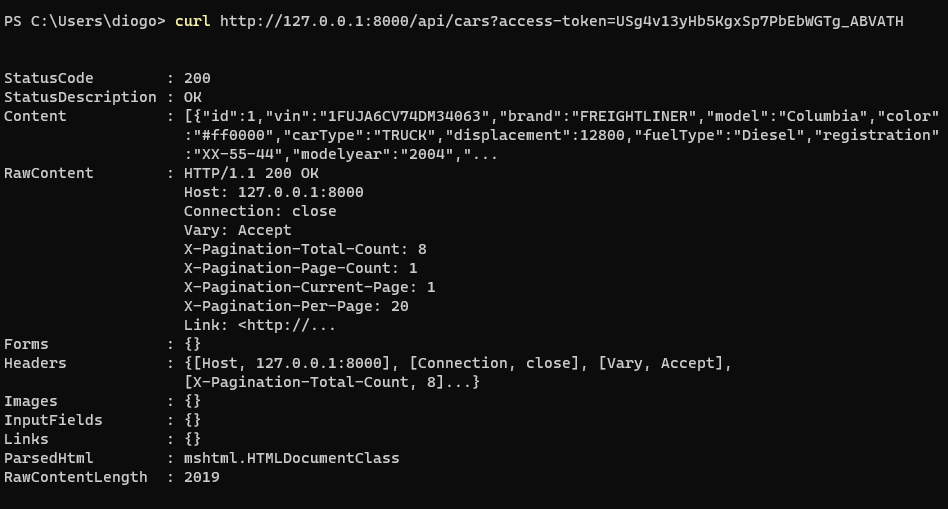


Figura 1 - Cars, Método GET

**URL:** curl <http://127.0.0.1:8000/api/cars?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método POST**

Este método serve para adicionar um novo veículo.

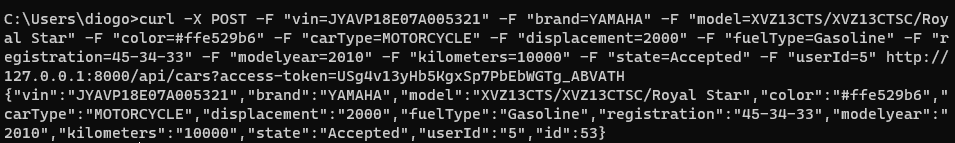


Figura 2 - Cars, Método POST

**URL:** curl -X POST -F "vin=JYAVP18E07A005321" -F "brand=YAMAHA" -F "model=XVZ13CTS/XVZ13CTSC/Royal Star" -F "color=#ffe529b6" -F "carType=MOTORCYCLE" -F "displacement=2000" -F "fuelType=Gasoline" -F "registration=45-34-33" -F "modelyear=2010" -F "kilometers=10000" -F "state=Accepted" -F "userId=5" <http://127.0.0.1:8000/api/cars?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método SET - CUSTOM**

Este método serve para mostrar uma quantidade específica de dados, neste caso, todos os veículos até um certo limite definido pelo utilizador.

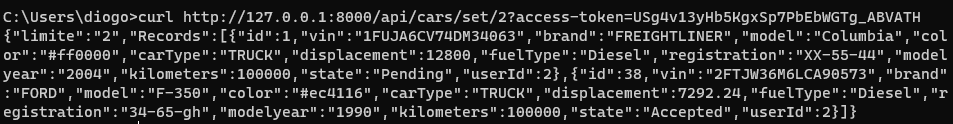


Figura 3 - Cars, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/cars/set/2?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método TOTAL - CUSTOM**

Este método só pode ser executado pelo administrador e serve para obter o número total de veículos registados na plataforma.



Figura 4 - Cars, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/cars/total?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método TOTAL USER - CUSTOM**

Este método serve para obter o número total de veículos registados na plataforma de um dado utilizador, através da utilização do Access Token.



Figura 5 - Cars, Método TOTAL USER CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/cars/totaluser?access-token=hg95zPz_0_IKUJVLRnf07jwT7cgE_ad2>

**Método DELETE**

Este método só pode ser executado pelo administrador e serve para eliminar o veículo selecionado pelo mesmo.



Figura 6 - Cars, Método DELETE

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/cars/58?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método PUT**

Este método serve para atualizar os dados de um veículo selecionado pelo administrador.

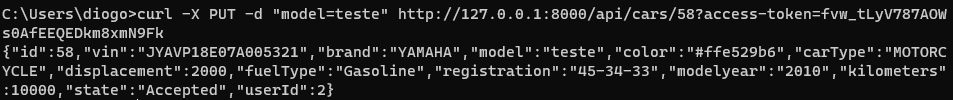


Figura 7 - Cars, Método PUT

**URL:** curl -X PUT -d "model=teste" <http://127.0.0.1:8000/api/cars/58?access-token=fvw_tLyV787AOWs0AfEEQEDkm8xmN9Fk>

**Método DELETE - CUSTOM**

Este método só pode apagar registos de veículos quando o utilizador mencionado no Access Token é o proprietário do mesmo. Este método retorna 1 quando apaga o registo com sucesso e 0 quando o pedido falha.



Figura 8 - Cars, Método DELETE CUSTOM

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/cars/deleted/61?access-token=3YEr44AJGN0CZHIXyvqKp9vYffzw0u1J>

**Método POST – CUSTOM**

Este método serve para inserir um registo de um veículo na ótica do cliente.

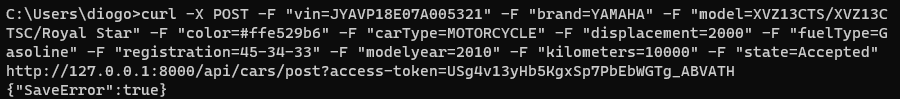


Figura 9 - Cars, Método POST CUSTOM

**URL:** curl -X POST -F "vin=JYAVP18E07A005321" -F "brand=YAMAHA" -F "model=XVZ13CTS/XVZ13CTSC/Royal Star" -F "color=#ffe529b6" -F "carType=MOTORCYCLE" -F "displacement=2000" -F "fuelType=Gasoline" -F "registration=45-34-33" -F "modelyear=2010" -F "kilometers=10000" -F "state=Accepted" <http://127.0.0.1:8000/api/cars/post?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método PUT – CUSTOM**

Este método serve para editar os dados de um veículo selecionado pelo utilizador. O veículo só pode ser atualizado se o utilizador for o proprietário.

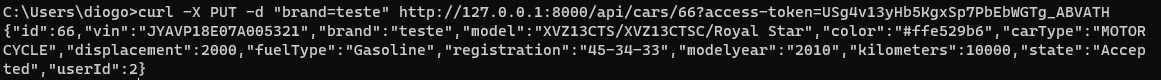


Figura 10 - Cars, Método PUT CUSTOM

**URL:** curl -X PUT -d "brand=teste" <http://127.0.0.1:8000/api/cars/66?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

## Companies

### Métodos de Implementação

**Método GET**

Este método serve para obter uma lista completa de todas as empresas registadas na plataforma e apenas pode ser executado pelo administrador.

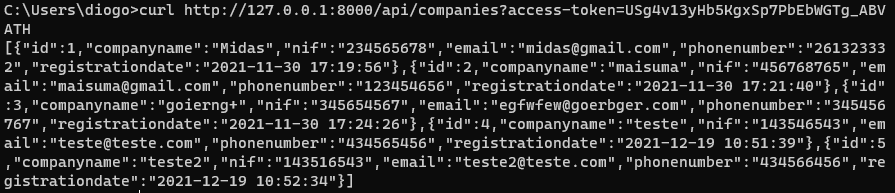


Figura 11 - Companies, Método GET

**URL:** curl <http://127.0.0.1:8000/api/companies?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método POST**

Este método serve para registar uma nova empresa na plataforma e apenas pode ser executado pelo administrador.

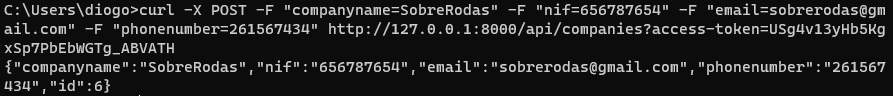


Figura 12 - Companies, Método POST

**URL:** curl -X POST -F "companyname=SobreRodas" -F "nif=656787654" -F "email=sobrerodas@gmail.com" -F "phonenumber=261567434" <http://127.0.0.1:8000/api/companies?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método PUT**

Este método serve para editar um registo de uma empresa na plataforma e apenas pode ser executado pelo administrador.

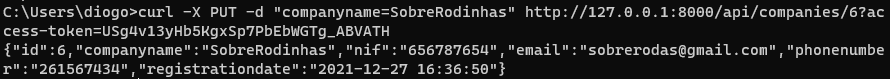
****

Figura 13 - Companies, Método PUT

**URL:** curl -X PUT -d "companyname=SobreRodinhas" <http://127.0.0.1:8000/api/companies/6?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método DELETE**

Este método serve para eliminar um registo de uma empresana plataforma e apenas pode ser executado pelo administrador.

****

Figura 14 - Companies, Método DELETE

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/companies/6?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método TOTAL – CUSTOM**

Este método serve para obter o número total de empresas registadas na plataforma e apenas pode ser executado pelo administrador.



Figura 15 - Companies, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/companies/total?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método SET – CUSTOM**

Este método serve para obter uma lista até um dado limite das empresas registadas na plataforma e apenas pode ser executado pelo administrador.

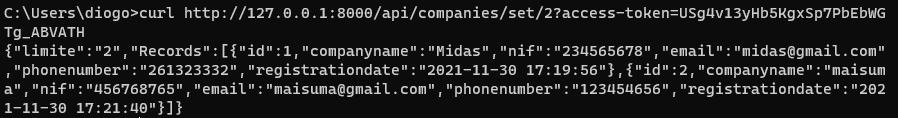
****

Figura 16 - Companies, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/companies/set/2?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

## CompaniesList

### Métodos de Implementação

**Método GET**

Este método serve para obter uma lista completa de todas as empresas registadas na plataforma e pode ser executado por qualquer utilizador, incluído o guest. Não requer Access Token.

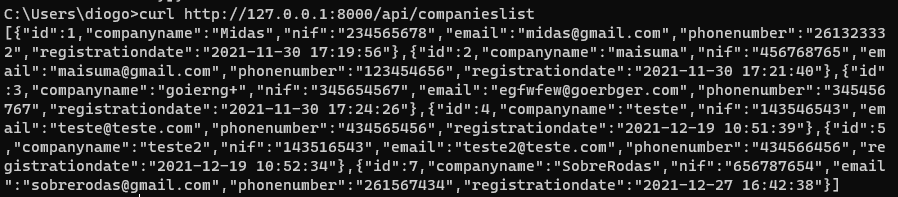


Figura 17 - CompaniesList, Método GET

**URL:** curl <http://127.0.0.1:8000/api/companieslist>

## Contributors

### Métodos de Implementação

**Método GET**

Este método serve para obter uma lista completa de todos os colaboradores registados na plataforma e apenas pode ser executado pelo administrador.

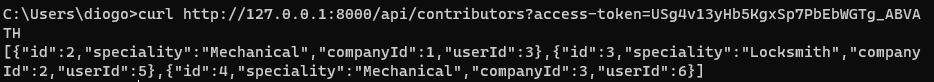
****

Figura 18 - Contributors, Método GET

**URL** curl <http://127.0.0.1:8000/api/contributors?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método POST**

Este método serve para registar um novo colaborador na plataforma e apenas pode ser executado pelo administrador.



Figura 19 - Contributors, Método POST

**URL:** curl -X POST -F "speciality=Painter" -F "companyId=1" -F "userId=2" <http://127.0.0.1:8000/api/contributors?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método PUT**

Este método serve para editar um registo de um colaborador na plataforma e apenas pode ser executado pelo administrador.



Figura 20 - Contributors, Método PUT

**URL:** curl -X PUT -d "speciality=Electrician" <http://127.0.0.1:8000/api/contributors/5?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método DELETE**

Este método serve para eliminar um registo de um colaboradorna plataforma e apenas pode ser executado pelo administrador.

****

Figura 21 - Contributors, Método DELETE

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/contributors/5?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método SET – CUSTOM**

Este método serve para obter uma lista até um dado limite dos colaboradores registados na plataforma e apenas pode ser executado pelo administrador.

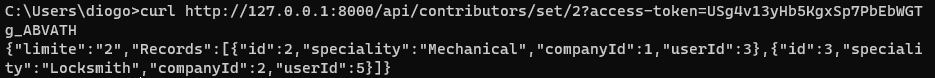


Figura 22 - Contributors, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/contributors/set/2?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método TOTAL – CUSTOM**

Este método serve para obter o número total de colaboradores registados na plataforma e apenas pode ser executado pelo administrador.

****

Figura 23 - Contributors, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/contributors/total?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

## Login

### Métodos de Implementação

**Método DO – CUSTOM**

Este método serve para validar a autenticação dos utilizadores e retornar os seus dados, de forma que este receba o token para ter acesso aos outros métodos. Além disso, este método também retorna a informação da ultima reparação efetuada a um veiculo desse utilizador, guardada no serviço Mosquitto, para que quando um utilizador fizer login este possa receber logo a informação de como se encontra o seu veiculo na oficina. Caso não haja essa informação no serviço Mosquitto, será enviado uma chave chamada “repair” com o valor true.

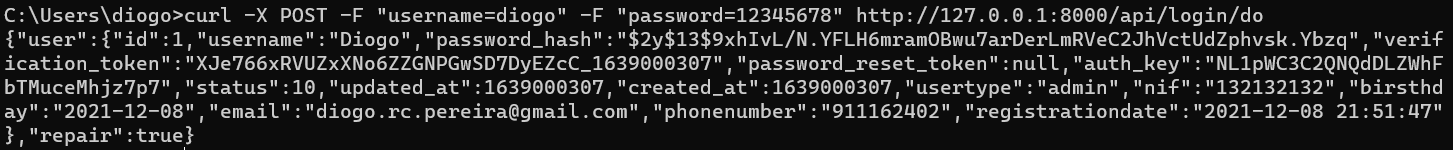


Figura 24 - Login, Método DO CUSTOM

**URL:** curl -X POST -F “username=diogo” -F “password=12345678” <http://127.0.0.1:8000/api/login/do>

## Repairs

### Métodos de Implementação

**Método GET**

Este método serve para obter uma lista completa de todas as reparações registadas na plataforma e apenas pode ser executado pelo administrador.

Uma imagem com texto

Descrição gerada automaticamente

Figura 25 - Repairs, Método GET

**URL:** curl <http://127.0.0.1:8000/api/repairs?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método HISTORY – CUSTOM**

Este método serve para obter uma lista completa de todas as reparações de um determinado veículo registado na plataforma e apenas pode ser executado pelo proprietário do veículo.

Uma imagem com texto

Descrição gerada automaticamente

Figura 26 - Repairs, Método HISTORY CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/repairs/history/1?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método REPAIR CONTRIBUTOR – CUSTOM**

Este método serve para retornar todas as reparações de um determinado colaborador e só pode ser executado por este mesmo colaborador.

Uma imagem com texto, garrafa, captura de ecrã

Descrição gerada automaticamente

Figura 27 - Repairs, Método REPAIR CONTRIBUTOR CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/repairs/repaircontributor?access-token=hg95zPz_0_IKUJVLRnf07jwT7cgE_ad2>

**Método POST – CUSTOM**

Este método serve para registar uma nova reparação na plataforma e apenas pode ser executado por um colaborador.

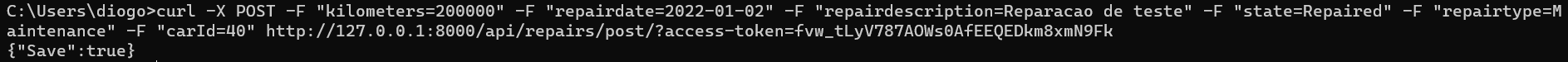
****

Figura 28 - Repairs, Método POST CUSTOM

**URL:** curl -X POST -F "kilometers=200000" -F "repairdate=2022-01-02" -F "repairdescription=Reparacao de teste" -F "state=Repaired" -F "repairtype=Maintenance" -F "carId=40" <http://127.0.0.1:8000/api/repairs/post/?access-token=fvw_tLyV787AOWs0AfEEQEDkm8xmN9Fk>

**Método PUT – CUSTOM**

Este método serve para alterar uma reparação na plataforma e apenas pode ser executado pelo colaborador que a criou.

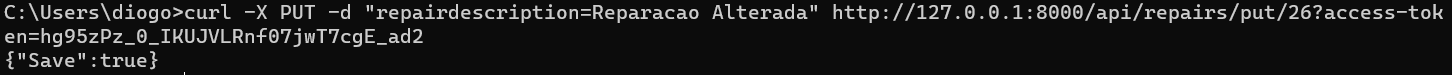


Figura 29 - Repairs, Método PUT CUSTOM

**URL:** curl -X PUT -d "repairdescription=Reparacao Alterada" <http://127.0.0.1:8000/api/repairs/put/26?access-token=hg95zPz_0_IKUJVLRnf07jwT7cgE_ad2>

**Método DELETED – CUSTOM**

Este método serve para apagar uma reparação na plataforma e apenas pode ser executado pelo colaborador que a criou.

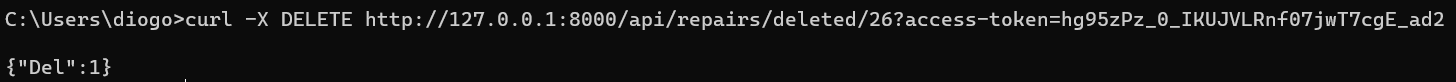


Figura 30 - Repairs, Método DELETED CUSTOM

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/repairs/deleted/26?access-token=hg95zPz_0_IKUJVLRnf07jwT7cgE_ad2>

**Método TOTAL – CUSTOM**

Este método serve para dar a informação ao administrador de quantas reparações foram registadas na plataforma.



Figura 31 - Repairs, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/repairs/total?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método SET – CUSTOM**

Este método serve para obter uma lista até um dado limite das reparações registadas na plataforma e apenas pode ser executado pelo administrador.

Uma imagem com texto, exterior

Descrição gerada automaticamente

Figura 32 - Repairs, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/repairs/set/10?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

## Schedules

### Métodos de Implementação

**Método TOTAL – CUSTOM**

Este método serve para obter o número total de agendamentos registados na plataforma e apenas pode ser executado pelo administrador.



Figura 33 - Schedules, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/schedules/total?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método SET – CUSTOM**

Este método serve para obter uma lista até a um dado limite dos agendamentos registados na plataforma e apenas pode ser executado pelo administrador.

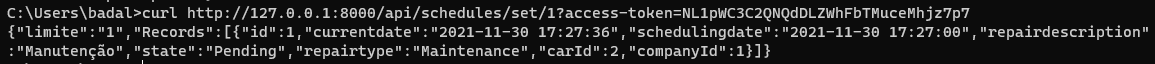


Figura 34 - Schedules, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/schedules/set/1?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método POST – CUSTOM**

Este método serve para adicionar um novo agendamento na plataforma e pode ser executado por todos os utilizadores à exceção do guest.

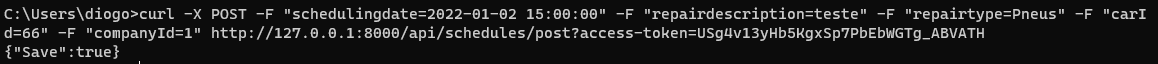


Figura 35 - Schedules, Método POST CUSTOM

**URL:** curl -X POST -F "schedulingdate=2022-01-02 15:00:00" -F "repairdescription=teste" -F "repairtype=Pneus" -F "carId=66" -F "companyId=1" <http://127.0.0.1:8000/api/schedules/post?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método DELETE CLIENT – CUSTOM**

Este método serve para eliminar um registo de um agendamento na plataforma e apenas pode ser executado pelo responsável da marcação do agendamento.



Figura 36 - Schedules, Método DELETE CLIENT CUSTOM

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/schedules/deleteclient/3?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método DELETE CONTRIBUTOR – CUSTOM**

Este método serve para eliminar um registo de um agendamento na plataforma e apenas pode ser executado pelo responsável da empresa do agendamento.



Figura 37 - Schedules, Método DELETE CONTRIBUTOR CUSTOM

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/schedules/deletecontributor/2?access-token=USg4v13yHb5KgxSp7PbEbWGTg_ABVATH>

**Método PUT CLIENT – CUSTOM**

Este método serve para editar um registo de um agendamento na plataforma e apenas pode ser executado pelo responsável da marcação do agendamento.

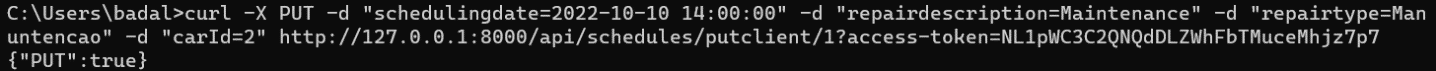
****

Figura 38 - Schedules, Método PUT CLIENT CUSTOM

**URL:** curl -X PUT -d "schedulingdate=2022-10-10 14:00:00" -d "repairdescription=Maintenance" -d "repairtype=Manuntencao" -d "carId=2" <http://127.0.0.1:8000/api/schedules/putclient/1?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método PUT CONTRIBUTOR – CUSTOM**

Este método serve para editar um registo de um agendamento na plataforma e apenas pode ser executado pelo responsável da empresa do agendamento.

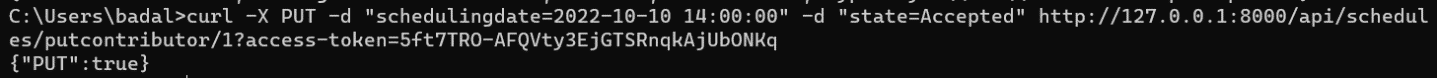


Figura 39 - Schedules, Método PUT CONTRIBUTOR CUSTOM

**URL:** curl -X PUT -d "schedulingdate=2022-10-10 14:00:00" -d "state=Accepted" <http://127.0.0.1:8000/api/schedules/putcontributor/1?access-token=5ft7TRO-AFQVty3EjGTSRnqkAjUbONKq>

## Signup

### Métodos de Implementação

**Método POST – CUSTOM**

Este método serve para registar qualquer tipo de utilizador, não sendo obrigatório o Access Token.

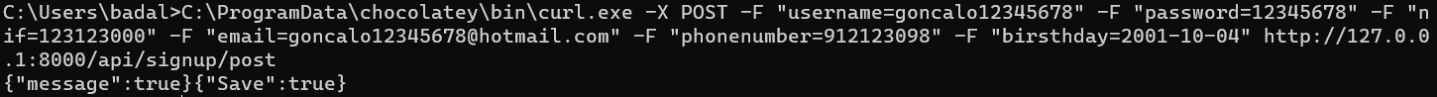


Figura 40 - Signup, Método POST CUSTOM

**URL:** curl -X POST -F "username=goncalo12345678" -F "password=12345678" -F "nif=123123000" -F "email=goncalo12345678@hotmail.com" -F "phonenumber=912123098" -F "birsthday=2001-10-04" <http://127.0.0.1:8000/api/signup/post>

## User

### Métodos de Implementação

**Método ACCOUNT – CUSTOM**

Este método serve para obter os dados da conta do utilizador associado ao Token utilizado no pedido. Este método requer Token e não está disponível para utilizadores guest.

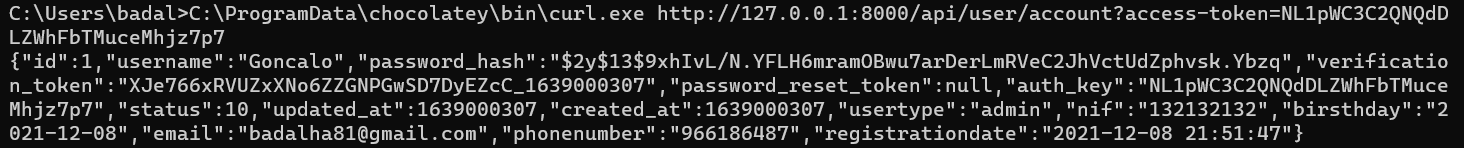


Figura 41 - User, Método ACCOUNT CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/user/account?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método TOTAL – CUSTOM**

Este método serve para obter o número total de utilizadores registados na plataforma e apenas pode ser executado pelo administrador.

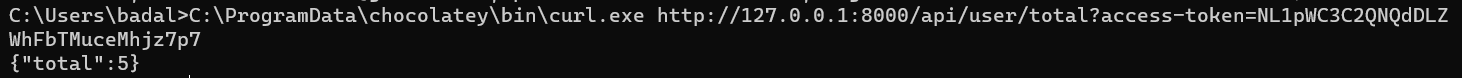


Figura 42 - User, Método TOTAL CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/user/total?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método SET – CUSTOM**

Este método serve para obter uma lista até a um dado limite dos utilizadores registados na plataforma e apenas pode ser executado pelo administrador.

Uma imagem com texto

Descrição gerada automaticamente

Figura 43 - User, Método SET CUSTOM

**URL:** curl <http://127.0.0.1:8000/api/user/set/2?access-token=NL1pWC3C2QNQdDLZWhFbTMuceMhjz7p7>

**Método DELETED - CUSTOM**

Este método serve para eliminar um registo de um utilizadorna plataforma e pode ser executado por qualquer utilizador, exceto os guest, visto que não possuem Access Token. Não requer o Id do utilizador, visto que o método utiliza o Access Token do utilizador em questão.

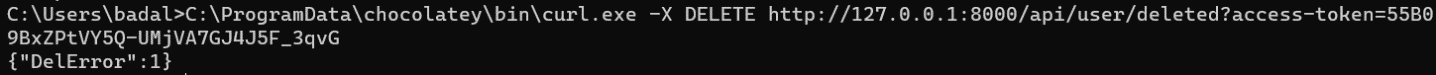


Figura 44 - User, Método DELETED CUSTOM

**URL:** curl -X DELETE <http://127.0.0.1:8000/api/user/deleted?access-token=55B09BxZPtVY5Q-UMjVA7GJ4J5F_3qvG>

**Método PUT - CUSTOM**

Este método serve para editar um registo de um utilizador na plataforma e pode ser executado por todos os utilizadores, exceto o guest.

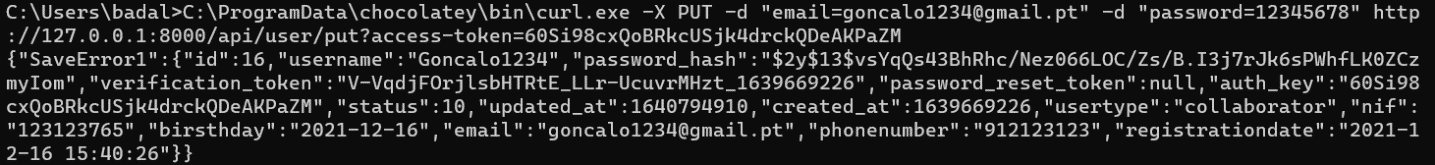


Figura 45 - User, Método PUT CUSTOM

**URL:** curl -X PUT -d "email=goncalo1234@gmail.pt" -d "password=12345678" <http://127.0.0.1:8000/api/user/put?access-token=60Si98cxQoBRkcUSjk4drckQDeAKPaZM>

## Informações

Alguns dos exemplos dos métodos referidos anteriormente necessitam, previamente, das credencias de autenticação, neste caso o access token do utilizador. Para esse efeito, para voltar a executar cada um desses métodos, o access token terá de ser alterado para um token de um utilizador que já se encontre registado na base de dados do projeto.

**Exemplo:** curl -X PUT -d "email=goncalo1234@gmail.pt" -d "password=12345678" [http://127.0.0.1:8000/api/user/put?access-token=**60Si98cxQoBRkcUSjk4drckQDeAKPaZM**](http://127.0.0.1:8000/api/user/put?access-token=60Si98cxQoBRkcUSjk4drckQDeAKPaZM)

**Explicação:** A zona a negrito do URL deve ser alterada para um novo token, salvo se, esse token estiver associado a um user já registado na base de dados.

O mesmo acontece com alguns métodos que necessitem do campo ID de um dado objeto. Por exemplo, alguns métodos de PUT ou Delete precisam do ID do objeto que queremos alterar, e mais uma vez esse ID terá que ser alterado para o ID de um objeto registado nessa tabela especifica.

**Exemplo:** curl -X PUT -d "schedulingdate=2022-10-10 14:00:00" -d "state=Accepted" [http://127.0.0.1:8000/api/schedules/putcontributor/**1**?access-token=5ft7TRO-AFQVty3EjGTSRnqkAjUbONKq](http://127.0.0.1:8000/api/schedules/putcontributor/1?access-token=5ft7TRO-AFQVty3EjGTSRnqkAjUbONKq)

**Explicação:** A zona a negrito do URL deve ser alterada para um novo ID, salvo se, esse ID estiver associado a um objeto desse tipo já registado na base de dados.

# Mosquitto

## Introdução ao Mosquitto

O Mosquitto é um serviço que utiliza o protocolo MQTT. O MQTT (Message Queuing Telemetry Transport), é um protocolo de mensagens leve para sensores e pequenos dispositivos móveis otimizado para redes TCP/IP. O esquema de troca de mensagens é fundamentado no modelo Publish-Subscribe, sendo extremamente simples e leve.

## Aplicação Prática

Tendo em conta o nosso projeto, o caso de uso do Mosquitto foi no modelo das reparações, intercetando a gravação do modelo das reparações utilizando a função aftersave.

Como se pode ver na figura abaixo, o objetivo desta função é publicar um pequeno objeto onde constem algumas das propriedades do objeto reparação, como por exemplo, a data e o estado.

Uma imagem com texto

Descrição gerada automaticamente

Figura 46 - Mosquitto, Função afterSave

Por fim, é chamada a função fazpublish, onde indicamos o nome do canal e a mensagem que queremos passar. O nome do canal será “REPAIR-X”, onde X representa o Id do utilizador do carro que está relacionado com esta reparação. A mensagem é constituída pelo JSON do objeto criado anteriormente, onde constam algumas das propriedades do objeto reparação, como por exemplo, a data e o estado.

Como podemos notar na figura abaixo, a função fazpublish vai publicar a mensagem descrita anteriormente, no seu respetivo canal.

Uma imagem com texto

Descrição gerada automaticamente

Figura 47- Mosquitto, Função FazPublish

A última mensagem fica guardada para posteriormente ser consultada pelo cliente, tendo em conta que, no método PUBLISH, a propriedade retain está definida como true.

Como podemos observar na imagem abaixo, no momento em que o utilizador faz a autenticação, a função fazsubscribe, entra no respetivo canal do utilizador, subscrevendo-o e verifica se tem ou não uma mensagem.

Uma imagem com texto

Descrição gerada automaticamente

Figura 48 - Mosquitto, Função actionDo e Função FazSubscribe

# Conclusão

Com a elaboração deste projeto, cujo tema seria a criação de uma API onde o objetivo é haver comunicação entre plataformas, conseguimos desenvolver a prática da programação orientada a objetos e a sistemas API REST.

A nível de dificuldades podemos referir, que inicialmente, não sabíamos por onde começar. Com o passar do tempo, e das aulas da Unidade Curricular de Serviços e Interoperabilidade de Sistemas, e após o desenvolvimento de algumas das fichas práticas, fomos ganhando prática no desenvolvimento de API’s REST, sendo também bastante útil toda a documentação da YII Framework. Bem perto do fim do projeto, sentimos dificuldades também na implementação da tecnologia Mosquitto, pois não sabíamos bem onde a implementar.

Enquanto grupo, o trabalho foi sempre divido e muito bem gerido entre nós, podendo assim afirmar que os três elementos do grupo trabalharam de igual modo. O grupo fica contente por conseguir concluir assim um projeto com esta dimensão pois, com isto, sentimos evolução pessoal e profissional.