Mobile Mechanics

https://mechanic-on-the-go.herokuapp.com/

Carro

|  |
| --- |
| **Ir buscar um carro dado um id**  Devolve o carro correspondente ao id com informação |
| Url: /api/cars/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do carro a obter |
| Resultado:  {  "id": 1,  "carLicensePlate": "dsad13",  "carYear": 2020,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "automatico",  "carInformation": null,  "carFuel": "gasoleo",  "carBrand": {  "id": 1,  "brandName": "mercedes"  },  "carEngine": {  "id": 2,  "engineName": "2.0",  "engineOil": "5w30",  "engineDisplacement": "2000cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): O carro não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "Car with id 1000 not found.",  "path": "/api/cars/1000"  } |

|  |
| --- |
| **Adicionar um carro**  Recebe a informação do carro |
| Url: /api/cars **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do carro a adicionar |
| Resultado:  {carClientId=2, carModelId=2, carBrandId=1, carLicensePlate=, carTransmission=Automatica, carEngineId=1, carFuel=Gasoleo, carYear=2021} |
| Erros:  {  "timestamp": "2021-12-22T17:15:00.383+00:00",  "status": 400,  "message": "Required request body is missing: public com.iade.mobilemechanics.models.Car com.iade.mobilemechanics.controllers.CarController.saveCar(com.iade.mobilemechanics.models.request.CarRequest)",  "path": "/api/cars"  } |

|  |
| --- |
| **Remover um carro dado um id** |
| Url: /api/cars/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do carro a apagar |
| Resultado:  {  Deleted  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): O carro não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "Car with id 1000 not found.",  "path": "/api/cars/1000"  } |

|  |
| --- |
| **Ir buscar todos os carros com o id do cliente**  Devolve todos os carros do cliente, cada um com a sua informação |
| Url: /api/cars/client/{id} **(get)** |
| Resultado:  [  {  "id": 1,  "carLicensePlate": "dsad13",  "carYear": 2020,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "automatico",  "carInformation": null,  "carFuel": "gasoleo",  "carBrand": {  "id": 1,  "brandName": "mercedes"  },  "carEngine": {  "id": 2,  "engineName": "2.0",  "engineOil": "5w30",  "engineDisplacement": "2000cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  },  {  "id": 19,  "carLicensePlate": "89ti56",  "carYear": 2019,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "Automatica",  "carInformation": null,  "carFuel": "Gasoleo",  "carBrand": {  "id": 2,  "brandName": "bmw"  },  "carEngine": {  "id": 3,  "engineName": "2.4",  "engineOil": "5w40",  "engineDisplacement": "2400cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  }  ] |

Marca

|  |
| --- |
| **Ir buscar todas as marcas**  Devolve todas as marcas na base de dados, cada uma com a sua informação. |
| Url: /api/brands **(get)** |
| Resultado:  [ {  "id": 2,  "brandName": "bmw"  },  {  "id": 3,  "brandName": "toyota"  },  {  "id": 1,  "brandName": "mercedes"  }] |

|  |
| --- |
| **Ir buscar uma marca dado um id**  Devolve a marca correspondente ao id com informação |
| Url: /api/brands/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da marca a obter |
| Resultado:  {  "id": 1,  "brandName": "mercedes"  }  Erros:  404 (HttpStatus.NOT\_FOUND): A marca não foi encontrada  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "brand with id 1000 not found.",  "path": "/api/brands/1000"  } |

|  |
| --- |
| **Adicionar uma marca**  Recebe a informação da marca |
| Url: /api/brands **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da marca a adicionar |
| Resultado:  {  "id": 35,  "brandName": "brand"  } |
| Erros:  500: Erro do servidor  {  "timestamp": "2021-01-22T13:44:49.731+00:00", "status": 500,  "error": "Internal Server Error",  "message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",  "path": "/api/brands"  }  409:(HttpStatus.*CONFLICT): A marca já existe*  {  "timestamp": "2021-11-13T08:56:37.495+00:00",  "status": 409,  "error": "Conflict",  "message": "Brand with name bmw already exist ",  "path": "/api/brands"  } |

|  |
| --- |
| **Remover uma marca dado um id** |
| Url: /api/brands/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da marca a apagar |
| Resultado:  {  Deleted  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): A marca não foi encontrada  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "brand with id 1000 not found.",  "path": "/api/brands/1000"  } |

Cliente

|  |
| --- |
| **Ir buscar todos os clientes**  Devolve todos os clientes na base de dados, cada uma com a sua informação. |
| Url: /api/clients **(get)** |
| Resultado:  [  {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  ] |

|  |
| --- |
| **Ir buscar um cliente dado o id da pessoa**  Devolve o cliente correspondente ao id com informação |
| Url: /api/clients/person/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do cliente a obter |
| Resultado:  {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  Erros:  404 (HttpStatus.NOT\_FOUND): O cliente não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "client with id 1000 not found.",  "path": "/api/clients/1000"  } |
|  |

|  |
| --- |
| **Adicionar um cliente**  Recebe a informação do cliente |
| Url: /api/clients **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do cliente a adicionar |
| Resultado: |
| Erros:  {  "timestamp": "2021-12-22T17:22:52.587+00:00",  "status": 400,  "message": "Required request body is missing: public com.iade.mobilemechanics.models.Client com.iade.mobilemechanics.controllers.ClientController.saveClient(com.iade.mobilemechanics.models.Client)",  "path": "/api/clients"  } |

Mecânico

|  |
| --- |
| **Adicionar um mecânico**  Recebe a informação do mecânico |
| Url: /api/mechanics **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do mecânico a adicionar |
| Resultado:  {  "id": 1,  "mechanicPersonId": 1,  "mechanicExperience": null,  "mechanicAboutMe": null  }  Erros:  {  "timestamp": "2021-12-22T19:27:02.711+00:00",  "status": 400,  "message": "Required request body is missing: public com.iade.mobilemechanics.models.Mechanic com.iade.mobilemechanics.controllers.MechanicController.saveMechanic(com.iade.mobilemechanics.models.Mechanic)",  "path": "/api/mechanics"  } |

|  |
| --- |
| **Remover um mecânico dado um id** |
| Url: /api/mechanics/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do mecânico a apagar |
| Resultado:  {  Deleted  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): O mecânico não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "mechanic with id 1000 not found.",  "path": "/api/mechanics/1000"  } |

|  |
| --- |
| **Ir buscar todos os mecânicos disponíveis**  Devolve todos os mecânicos disponíveis, cada um com a sua informação |
| Url: /api/mechanics/available**(get)** |
| Resultado:  [  {  "id": 1,  "mechanicPerson": {  "id": 3,  "personName": "personMechanic",  "personPhoneNumber": "987654322",  "personEmail": "@.m",  "personPassword": "1234",  "personIsMechanic": true  },  "mechanicExperience": null,  "mechanicAboutMe": "about me",  "mechanicAvailability": true  }  ] |

Modelo

|  |
| --- |
| **Adicionar um modelo**  Recebe a informação do modelo |
| Url: /api/models **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do modelo a adicionar |
| Resultado:  {  "id": 1,  "modelName": "Class A",  "modelBrandId": 1  }  Erros:  500: Erro do servidor  {  "timestamp": "2021-01-22T13:44:49.731+00:00", "status": 500,  "error": "Internal Server Error",  "message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",  "path": "/api/models"  } |

|  |
| --- |
| **Remover um modelo dado um id** |
| Url: /api/models/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do modelo a apagar |
| Resultado:  {  Deleted  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): O modelo não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "model with id 1000 not found.",  "path": "/api/models/1000"  } |

|  |
| --- |
| **Ir buscar todos os modelos com o id da marca**  Devolve todos os modelos da marca, cada um com a sua informação |
| Url: /api/models/brand/{id} **(get)** |
| Resultado:  [ {  "id": 1,  "modelName": "Class A",  "modelBrandId": 1  }  {  "id": 1,  "modelName": "Class C",  "modelBrandId": 1  }] |

Reparação

|  |
| --- |
| **Ir buscar as reparações com o id do carro**  Devolve as reparações, cada uma com a sua informação |
| Url: /api/repairs/car/{id} **(get)** |
| Resultado:    {  "id": 137,  "repairDate": "2021-12-22",  "repairPrice": null,  "repairMechanic": null,  "repairCar": {  "id": 1,  "carLicensePlate": "dsad13",  "carYear": 2020,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "automatico",  "carInformation": null,  "carFuel": "gasoleo",  "carBrand": {  "id": 1,  "brandName": "mercedes"  },  "carEngine": {  "id": 2,  "engineName": "2.0",  "engineOil": "5w30",  "engineDisplacement": "2000cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  },  "repairLat": null,  "repairLong": null  } |

|  |
| --- |
| **Adicionar uma reparação**  Recebe a informação da reparação |
| Url: /api/repairs **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da reparação a adicionar |
| Resultado:  {  "id": 1,  "repairDate": "2021-11-09",  "repairPrice": "20",  "repairMechanicId": 1,  "repairCarId": 1  }  Erros:  500: Erro do servidor  {  "timestamp": "2021-01-22T13:44:49.731+00:00", "status": 500,  "error": "Internal Server Error",  "message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",  "path": "/api/repairs"  } |

|  |
| --- |
| **Remover uma reparação dado um id** |
| Url: /api/repairs/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da reparação a apagar |
| Resultado:  {  Deleted  } |
| Erros:  404 (HttpStatus.NOT\_FOUND): O modelo não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "model with id 1000 not found.",  "path": "/api/repairs/1000"  } |

Tipos de reparações

|  |
| --- |
| **Ir buscar todos os tipos de reparações**  Devolve todos os tipos de reparações na base de dados, cada um com a sua informação |
| Url: /api/typeRepair **(get)** |
| Resultado:  [  {  "id": 1,  "typeRepairName": "Battery replacement"  },  {  "id": 2,  "typeRepairName": "Mudança de Oleo"  },  {  "id": 3,  "typeRepairName": "Mudança de Escovas"  },  {  "id": 4,  "typeRepairName": "Troca de travões"  },  {  "id": 5,  "typeRepairName": "Rodar os pneus"  },  {  "id": 6,  "typeRepairName": "diagnostico"  }  ] |

|  |
| --- |
| **Adicionar um tipo de reparação**  Recebe a informação do modelo |
| Url: /api/typeRepair **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da reparação a adicionar |
| Resultado:  {  "id": 2,  "typeRepairName": "battery replacement"  }  Erros:  500: Erro do servidor  {  "timestamp": "2021-01-22T13:44:49.731+00:00", "status": 500,  "error": "Internal Server Error",  "message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",  "path": "/api/models"  }  409:(HttpStatus.*CONFLICT): o tipo de reparação já existe*  {  "timestamp": "2021-11-13T08:56:37.495+00:00",  "status": 409,  "error": "Conflict",  "message": "type of repair with name oil change already exist ",  "path": "/api/typeRepair"  } |

Preço

|  |
| --- |
| **Ir buscar o preço com o id do modelo e do tipo de reparação**  Devolve o preço |
| Url: /api/prices/price?modelId=1&typeRepairId=1 **(get)** |
| Resultado:    {  "id": 1,  "priceModelId": 1,  "priceTypeRepairId": 1,  "pricePrice": 10.0  } |

Review

|  |
| --- |
| **Ir buscar a review com o id da reparação**  Devolve a review da reparação, cada uma com a sua informação |
| Url: /api/reviews/repair/{id} **(get)** |
| Resultado:    {  "id": 1,  "reviewComment": "Perfecto repairo",  "reviewStars": 5,  "reviewClientId": 1,  "reviewRepairId": 1  } |

|  |
| --- |
| **Adicionar uma review**  Recebe a informação da review |
| Url: /api/reviews **(post)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da review a adicionar |
| Resultado:  {  {  "id": 4,  "reviewComment": "good",  "reviewStars": 4,  "reviewClientId": 1,  "reviewRepairId": 1  }  }  Erros:  500: Erro do servidor  {  "timestamp": "2021-01-22T13:44:49.731+00:00", "status": 500,  "error": "Internal Server Error",  "message": "could not execute statement; SQL [n/a]; constraint [null]; nested exception is org.hibernate.exception.ConstraintViolationException: could not execute statement",  "path": "/api/models"  } |

Type Repair Repair

|  |
| --- |
| **Ir buscar os ids por id** |
| Url: /api/typeRepairRepair/1  **(get)** |
| Parâmetros:  id |
| Resultado:  {  "id": 35,  "typeRepairRepairTypeRepair": {  "id": 1,  "typeRepairName": "Battery replacement"  },  "typeRepairRepairRepair": {  "id": 55,  "repairDate": "2021-12-20",  "repairPrice": null,  "repairMechanic": null,  "repairCar": {  "id": 1,  "carLicensePlate": "dsad13",  "carYear": 2020,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "automatico",  "carInformation": null,  "carFuel": "gasoleo",  "carBrand": {  "id": 1,  "brandName": "mercedes"  },  "carEngine": {  "id": 2,  "engineName": "2.0",  "engineOil": "5w30",  "engineDisplacement": "2000cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  },  "repairLat": null,  "repairLong": null  }  }  Erros:  {  "timestamp": "2021-12-22T18:27:29.648+00:00",  "status": 404,  "message": "typeRepair with id 5 not found.",  "path": "/api/typeRepairRepair/5"  } |

|  |
| --- |
| **Ir buscar todos os tipos de reparação reparação**  Devolve todos na base de dados, cada um com a sua informação. |
| Url: /api/typeRepairRepair  **(get)** |
| Resultado:  {  "id": 141,  "typeRepairRepairTypeRepair": {  "id": 1,  "typeRepairName": "Battery replacement"  },  "typeRepairRepairRepair": {  "id": 137,  "repairDate": "2021-12-22",  "repairPrice": null,  "repairMechanic": null,  "repairCar": {  "id": 1,  "carLicensePlate": "dsad13",  "carYear": 2020,  "carModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  },  "carPhotoId": null,  "carTransmission": "automatico",  "carInformation": null,  "carFuel": "gasoleo",  "carBrand": {  "id": 1,  "brandName": "mercedes"  },  "carEngine": {  "id": 2,  "engineName": "2.0",  "engineOil": "5w30",  "engineDisplacement": "2000cc"  },  "carClient": {  "id": 2,  "clientPerson": {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  },  "clientNif": null  }  },  "repairLat": null,  "repairLong": null  }  }  ] |

Pessoa

|  |
| --- |
| **Ir buscar todas as pessoas**  Devolve todas as pessoas na base de dados, cada uma com a sua informação. |
| Url: /api/persons **(get)** |
| Resultado:  [  {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  }, |

|  |
| --- |
| **Ir buscar uma pessoa dado um id**  Devolve a pessoa correspondente ao id com informação |
| Url: /api/persons/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id da pessoa a obter |
| Resultado:  {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  }  Response code: 200; Time: 192ms; Content length: 137 bytesErros:  404 (HttpStatus.NOT\_FOUND): A pessoa não foi encontrada  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "person with id 1000 not found.",  "path": "/api/brands/1000"  } |

|  |
| --- |
| **Ir buscar uma pessoa dado um mail e password**  Devolve a pessoa correspondente com informação |
| Url: /api/persons//log/{user}/{pass}  **(get)** |
| Parâmetros:  user- string  pass-string |
| Resultado:  {  "id": 1,  "personName": "personClient",  "personPhoneNumber": "987654321",  "personEmail": "@.c",  "personPassword": "1234",  "personIsMechanic": false  }  Erros:  404 (HttpStatus.NOT\_FOUND): A pessoa não foi encontrada  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "person with id 1000 not found.",  "path": "/api/brands/1000"  } |

Modelo Motor

|  |
| --- |
| **Ir buscar todas os modelos-motores**  Devolve todos na base de dados, cada uma com a sua informação. |
| Url: /api/modelengine **(get)** |
| Resultado:  {  "id": 2,  "modelEngineEngine": {  "id": 1,  "engineName": "1.4D",  "engineOil": "5w30",  "engineDisplacement": "1400cc"  },  "modelEngineModel": {  "id": 2,  "modelName": "Class C",  "modelBrand": {  "id": 1,  "brandName": "mercedes"  }  }  }, |

|  |
| --- |
| **Ir buscar um modelo-motor dado um id**  Devolve o modelo-motor correspondente ao id com informação |
| Url: /api/modelengine/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id a obter |
| Resultado:  [  {  "id": 1,  "engineName": "1.4D",  "engineOil": "5w30",  "engineDisplacement": "1400cc"  },  Erros:  404 (HttpStatus.NOT\_FOUND): não foi encontrada  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": " with id 1000 not found.",  "path": "/api/modelengine /1000"  } |

Motor

|  |
| --- |
| **Ir buscar um motor dado um id**  Devolve o motor correspondente ao id com informação |
| Url: /api/engines/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do motor a obter |
| Resultado:  {  "id": 1,  "engineName": "1.4D",  "engineOil": "5w30",  "engineDisplacement": "1400cc"  }  Erros:  404 (HttpStatus.NOT\_FOUND): O motor não foi encontrado  {  "timestamp": "2021-01-22T13:33:30.282+00:00",  "status": 404,  "error": "Not Found",  "message": "engine with id 1000 not found.",  "path": "/api/engines/1000"  } |