Mobile Mechanics

Carro

| **Ir buscar todos os carros**  Devolve todos os carros na base de dados, cada um com a sua informação |
| --- |
| Url: /api/cars **(get)** |
| Resultado:  [ {  "id": 2,  "carLicensePlate": "01-AA-00",  "carYear": 2000,  "carClientId": 1,  "carModelEngineId": 1,  "carPhotoId": 1  },  {  "id": 7,  "carLicensePlate": "05-ab-qe",  "carYear": 2020,  "carClientId": 2,  "carModelEngineId": 3,  "carPhotoId": 3  },  {  "id": 3,  "carLicensePlate": "02-AA-00",  "carYear": 2000,  "carClientId": 2,  "carModelEngineId": 2,  "carPhotoId": 2  }] |

| **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": 2,  "carLicensePlate": "01-AA-00",  "carYear": 2000,  "carClientId": 1,  "carModelEngineId": 1,  "carPhotoId": 1  } |
| 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:  {  "id": 10,  "carLicensePlate": "01-BB-00",  "carYear": 2000,  "carClientId": 1,  "carModelEngineId": 3,  "carPhotoId": 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/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": 7,  "carLicensePlate": "05-ab-qe",  "carYear": 2020,  "carClientId": 2,  "carModelEngineId": 3,  "carPhotoId": 3  },  {  "id": 3,  "carLicensePlate": "02-AA-00",  "carYear": 2000,  "carClientId": 2,  "carModelEngineId": 2,  "carPhotoId": 2  }  ] |

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,  "clientPersonId": 4,  "clientNif": "235798568"  },  {  "id": 1,  "clientPersonId": 2,  "clientNif": "247911500"  }] |

| **Ir buscar um cliente dado um id**  Devolve o cliente correspondente ao id com informação |
| --- |
| Url: /api/clients/{id} **(get)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do cliente a obter |
| Resultado:  {  "id": 1,  "clientPersonId": 2,  "clientNif": "247911500"  }  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:  {  "id": 1,  "clientPersonId": 2,  "clientNif": "247911500"  } |
| 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/clients"  } |

| **Remover um cliente dado um id** |
| --- |
| Url: /api/clients/{id} **(delete)** |
| Parâmetros:  id - inteiro positivo que corresponde ao id do cliente a apagar |
| Resultado:  {  Deleted  } |
| 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"  } |

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:  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/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"  } |

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

| **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"  } |

| **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  } |

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": "oil change"  },  {  "id": 2,  "typeRepairName": "battery replacement"  }  ] |

| **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  } |