Portfolio Assignment - Final Project

CS 493: Cloud Application Development

Fall 2022

Oregon State University

Nick Francisco

Welcome URL: https://francnic-final-project.uw.r.appspot.com/

Login URL: https://francnic-final-project.uw.r.appspot.com/login

Table of Contents

[Data Model 8](#_Toc153226282)

[Car 8](#_Toc153226283)

[Parts 8](#_Toc153226284)

[Users 8](#_Toc153226285)

[Relationship 8](#_Toc153226286)

[Users 8](#_Toc153226287)

[Login Instruction 9](#_Toc153226288)

[Postman Environment Setup 9](#_Toc153226289)

[Create a Car (JWT Protected) 10](#_Toc153226290)

[Request 10](#_Toc153226291)

[Authorization 10](#_Toc153226292)

[Headers 10](#_Toc153226293)

[Path Parameters 10](#_Toc153226294)

[Request Body 10](#_Toc153226295)

[Request Body Format 10](#_Toc153226296)

[Request JSON Attributes 10](#_Toc153226297)

[Request Body Example 10](#_Toc153226298)

[Response 10](#_Toc153226299)

[Response Body Format 10](#_Toc153226300)

[Response Statuses 10](#_Toc153226301)

[Get all Cars (JWT Protected) 12](#_Toc153226302)

[Request 12](#_Toc153226303)

[Authorization 12](#_Toc153226304)

[Headers 12](#_Toc153226305)

[Path Parameters 12](#_Toc153226306)

[Request Body 12](#_Toc153226307)

[Response 12](#_Toc153226308)

[Response Body Format 12](#_Toc153226309)

[Response Statuses 12](#_Toc153226310)

[Response Examples 12](#_Toc153226311)

[Get a Car (JWT Protected) 14](#_Toc153226312)

[Request 14](#_Toc153226313)

[Authorization 14](#_Toc153226314)

[Headers 14](#_Toc153226315)

[Path Parameters 14](#_Toc153226316)

[Request Body 14](#_Toc153226317)

[Response 14](#_Toc153226318)

[Response Body Format 14](#_Toc153226319)

[Response Statuses 14](#_Toc153226320)

[Response Examples 14](#_Toc153226321)

[Edit a Car (Patch) (JWT Protected) 16](#_Toc153226322)

[Request 16](#_Toc153226323)

[Authorization 16](#_Toc153226324)

[Headers 16](#_Toc153226325)

[Path Parameters 16](#_Toc153226326)

[Request Body 16](#_Toc153226327)

[Request Body Format 16](#_Toc153226328)

[Request JSON Attributes 16](#_Toc153226329)

[Request Body Example 16](#_Toc153226330)

[Response 16](#_Toc153226331)

[Response Body Format 16](#_Toc153226332)

[Response Statuses 16](#_Toc153226333)

[Response Examples 17](#_Toc153226334)

[Edit a Car (Put) (JWT Protected) 18](#_Toc153226335)

[Request 18](#_Toc153226336)

[Authorization 18](#_Toc153226337)

[Headers 18](#_Toc153226338)

[Path Parameters 18](#_Toc153226339)

[Request Body 18](#_Toc153226340)

[Request Body Format 18](#_Toc153226341)

[Request JSON Attributes 18](#_Toc153226342)

[Request Body Example 18](#_Toc153226343)

[Response 18](#_Toc153226344)

[Response Body Format 18](#_Toc153226345)

[Response Statuses 18](#_Toc153226346)

[Response Examples 19](#_Toc153226347)

[Put part on a Car (JWT Protected) 20](#_Toc153226348)

[Request 20](#_Toc153226349)

[Authorization 20](#_Toc153226350)

[Headers 20](#_Toc153226351)

[Path Parameters 20](#_Toc153226352)

[Request Body 20](#_Toc153226353)

[Request Body Format 20](#_Toc153226354)

[Request JSON Attributes 20](#_Toc153226355)

[Request Body Example 20](#_Toc153226356)

[Response 20](#_Toc153226357)

[Response Body Format 20](#_Toc153226358)

[Response Statuses 20](#_Toc153226359)

[Response Examples 21](#_Toc153226360)

[Delete part on a Car (JWT Protected) 22](#_Toc153226361)

[Request 22](#_Toc153226362)

[Authorization 22](#_Toc153226363)

[Headers 22](#_Toc153226364)

[Path Parameters 22](#_Toc153226365)

[Request Body 22](#_Toc153226366)

[Request Body Format 22](#_Toc153226367)

[Request JSON Attributes 22](#_Toc153226368)

[Request Body Example 22](#_Toc153226369)

[Response 22](#_Toc153226370)

[Response Body Format 22](#_Toc153226371)

[Response Statuses 22](#_Toc153226372)

[Response Examples 23](#_Toc153226373)

[Delete a Car (JWT Protected) 24](#_Toc153226374)

[Request 24](#_Toc153226375)

[Authorization 24](#_Toc153226376)

[Headers 24](#_Toc153226377)

[Path Parameters 24](#_Toc153226378)

[Request Body 24](#_Toc153226379)

[Response 24](#_Toc153226380)

[Response Body Format 24](#_Toc153226381)

[Response Statuses 24](#_Toc153226382)

[Response Examples 24](#_Toc153226383)

[Create a Part 26](#_Toc153226384)

[Request 26](#_Toc153226385)

[Authorization 26](#_Toc153226386)

[Headers 26](#_Toc153226387)

[Path Parameters 26](#_Toc153226388)

[Request Body 26](#_Toc153226389)

[Request Body Format 26](#_Toc153226390)

[Request JSON Attributes 26](#_Toc153226391)

[Request Body Example 26](#_Toc153226392)

[Response 26](#_Toc153226393)

[Response Body Format 26](#_Toc153226394)

[Response Statuses 26](#_Toc153226395)

[Get all Parts 28](#_Toc153226396)

[Request 28](#_Toc153226397)

[Authorization 28](#_Toc153226398)

[Headers 28](#_Toc153226399)

[Path Parameters 28](#_Toc153226400)

[Request Body 28](#_Toc153226401)

[Response 28](#_Toc153226402)

[Response Body Format 28](#_Toc153226403)

[Response Statuses 28](#_Toc153226404)

[Response Examples 28](#_Toc153226405)

[Get a Part 30](#_Toc153226406)

[Request 30](#_Toc153226407)

[Authorization 30](#_Toc153226408)

[Headers 30](#_Toc153226409)

[Path Parameters 30](#_Toc153226410)

[Request Body 30](#_Toc153226411)

[Response 30](#_Toc153226412)

[Response Body Format 30](#_Toc153226413)

[Response Statuses 30](#_Toc153226414)

[Response Examples 30](#_Toc153226415)

[Edit a Part (Patch) 32](#_Toc153226416)

[Request 32](#_Toc153226417)

[Authorization 32](#_Toc153226418)

[Headers 32](#_Toc153226419)

[Path Parameters 32](#_Toc153226420)

[Request Body 32](#_Toc153226421)

[Request Body Format 32](#_Toc153226422)

[Request JSON Attributes 32](#_Toc153226423)

[Request Body Example 32](#_Toc153226424)

[Response 32](#_Toc153226425)

[Response Body Format 32](#_Toc153226426)

[Response Statuses 32](#_Toc153226427)

[Response Examples 33](#_Toc153226428)

[Edit a Part (Put) 34](#_Toc153226429)

[Request 34](#_Toc153226430)

[Authorization 34](#_Toc153226431)

[Headers 34](#_Toc153226432)

[Path Parameters 34](#_Toc153226433)

[Request Body 34](#_Toc153226434)

[Request Body Format 34](#_Toc153226435)

[Request JSON Attributes 34](#_Toc153226436)

[Request Body Example 34](#_Toc153226437)

[Response 34](#_Toc153226438)

[Response Body Format 34](#_Toc153226439)

[Response Statuses 34](#_Toc153226440)

[Response Examples 35](#_Toc153226441)

[Delete a Part 36](#_Toc153226442)

[Request 36](#_Toc153226443)

[Authorization 36](#_Toc153226444)

[Headers 36](#_Toc153226445)

[Path Parameters 36](#_Toc153226446)

[Request Body 36](#_Toc153226447)

[Response 36](#_Toc153226448)

[Response Body Format 36](#_Toc153226449)

[Response Statuses 36](#_Toc153226450)

[Response Examples 36](#_Toc153226451)

[Get all Users 37](#_Toc153226452)

[Request 37](#_Toc153226453)

[Authorization 37](#_Toc153226454)

[Headers 37](#_Toc153226455)

[Path Parameters 37](#_Toc153226456)

[Request Body 37](#_Toc153226457)

[Response 37](#_Toc153226458)

[Response Body Format 37](#_Toc153226459)

[Response Statuses 37](#_Toc153226460)

[Response Examples 37](#_Toc153226461)

# Data Model

The app stores three kinds of entities in Datastore; Cars, Parts, and Users.

## Car

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Data Type** | **Required** | **Notes** |
| id | Integer | No | The id of the car. Datastore automatically generates it. |
| make | String | Yes | Make of the car. E.g. Honda, Toyota |
| model | String | Yes | Model of the car. E.g., Civic, Tacoma |
| year | Integer | Yes | Model year of the car. |
| owner | String | No | Owner of vehicle using user id. |
| parts | String Array | No | List of part ids installed on car. |

## Parts

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Data Type** | **Required** | **Notes** |
| id | Integer | No | The id of the car. Datastore automatically generates it. |
| name | String | Yes | Name of the part. |
| component | String | Yes | Part of the car the part is installed on. |
| install\_date | String | Yes | Date the parts was installed. |
| car | String | No | Car id that the part is installed on. |

## Users

|  |  |  |  |
| --- | --- | --- | --- |
| **Property** | **Data Type** | **Required** | **Notes** |
| id | Integer | No | The id of the car. Datastore automatically generates it. |
| user\_id | String | Yes | Unique identifier from Auth0. |

# Relationship

The relationship between cars and parts is that a car can have a part installed on it. To represent this relationship a car has a list of part ids that are installed on the car as a property, and the part has a car property with the car id that the parts are installed on.

# Users

The users are modeled in the datastore by adding new users to the database whenever a new account is created. When a new account is created the Auth0 unique identifier (sub) is saved to the database to keep track of all users. To gain access to protected resources a valid JWT token for the user must be supplied to the request authorization header. The API then checks for valid JWT before accessing protected resources. There is a relationship between Users and Cars, but there is not a relationship between Users and Parts.

# Login Instruction

To login navigate to <https://francnic-final-project.uw.r.appspot.com/login> and enter you account credentials or create a new account. After creating an account or logging in, your JWT and unique identifier will be displayed. You can logout by navigating to <https://francnic-final-project.uw.r.appspot.com/logout>.

# Postman Environment Setup

Make sure to add two JWT tokens to the environment variables for the postman collection tests to work. To do this enter in the username and password for two account in the body if the first two test in the postman collection to get a valid JWT, or manually enter two JWT tokens into the postman environment and skip the first two test in the postman collection.

# Create a Car (JWT Protected)

Allows you to create a new car.

|  |
| --- |
| POST /cars |

## Request

### Authorization

Authorization: JWT Token

### Headers

None

### Path Parameters

None

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| make | Make of the car. E.g. Honda, Toyota | Yes |
| model | Model of the car. E.g., Civic, Tacoma | Yes |
| year | Model year of the car. | Yes |

### Request Body Example

|  |
| --- |
| {  " make": "Honda",  "model": "S2000",  "year": 2004  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 201 Created |  |
| Failure | 400 Bad Request | Missing attributes |
| Failure | 403 Forbidden | Invalid JWT |

#### Success

|  |
| --- |
| Status: 201 Created  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  } |

#### Failure

|  |
| --- |
| Status: 400 Bad Request  {  "Error": "The request object is missing at least one of the required attributes"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

# Get all Cars (JWT Protected)

List all the cars.

|  |
| --- |
| GET /cars |

## Request

### Authorization

Authorization: JWT Token

### Headers

Accept: application/json

### Path Parameters

None

### Request Body

None

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "cars": [  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  },  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2000,  "parts": [],  "model": "Civic",  "make": "Honda",  "id": "5470083618963456",  "self": "http://localhost:8080/cars/5470083618963456"  },  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 1991,  "parts": [],  "model": "NSX",  "make": "Acura",  "id": "6028455875444736",  "self": "http://localhost:8080/cars/6028455875444736"  },  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2016,  "parts": [],  "model": "Tacoma",  "make": "Toyota",  "id": "6033033572384768",  "self": "http://localhost:8080/cars/6033033572384768"  },  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 1995,  "parts": [],  "model": "Miata",  "make": "Mazda",  "id": "6518167308337152",  "self": "http://localhost:8080/cars/6518167308337152"  }  ],  "next": "http://localhost:8080/cars?cursor=CjQSLmoaenV3fmZyYW5jbmljLWZpbmFsLXByb2plY3RyEAsSA0NhchiAgIC40YfKCwwYACAA",  "count": 5  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Get a Car (JWT Protected)

Allows you to get an existing car

|  |
| --- |
| GET /cars/:car\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

Accept: application/json

## Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |

### Request Body

None

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 401 Unauthorized | Invalid Token |
| Failure | 403 Forbidden | Not owner of the car |
| Failure | 404 Found | No car with car\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  } |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Edit a Car (Patch) (JWT Protected)

Allows you to edit the properties of a car.

|  |
| --- |
| PATCH /cars/:car\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

Accept: application/json

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| make | Make of the car. E.g. Honda, Toyota | No |
| model | Model of the car. E.g., Civic, Tacoma | No |
| year | Model year of the car. | No |

### Request Body Example

|  |
| --- |
| {  " make": "Honda",  "model": "S2000",  "year": 2004  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 403 Forbidden | Not owner of the car |
| Failure | 404 Not Found | No car with car\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  } |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Edit a Car (Put) (JWT Protected)

Allows you to edit all properties of a car.

|  |
| --- |
| PUT /cars/:car\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

Accept: application/json

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| make | Make of the car. E.g. Honda, Toyota | Yes |
| model | Model of the car. E.g., Civic, Tacoma | Yes |
| year | Model year of the car. | Yes |

### Request Body Example

|  |
| --- |
| {  " make": "Honda",  "model": "S2000",  "year": 2004  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 400 Bad Request | Missing attributes |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 403 Forbidden | Not owner of the car |
| Failure | 404 Not Found | No car with car\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  } |

#### Failure

|  |
| --- |
| Status: 400 Bad Request  {  "Error": "The request object is missing at least one of the required attributes"  } |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Put part on a Car (JWT Protected)

Allows you to add a part to a car by creating a relationship between the two entities.

|  |
| --- |
| PUT /:car\_id/parts/:part\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

None

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |
| part\_id | ID of the part |

### Request Body

None

### Request Body Format

None

### Request JSON Attributes

None

### Request Body Example

None

## Response

### Response Body Format

Success: No body

Failure: JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 204 OK |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 403 Forbidden | Not owner of the car or part on a car already |
| Failure | 404 Not Found | No car with car\_id exists or part with part\_id exists |

### Response Examples

#### Success

|  |
| --- |
| Status: 204 No Content |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

# Delete part on a Car (JWT Protected)

Allows you to remove a part from a car by removing the relationship between the two entities.

|  |
| --- |
| Delete /:car\_id/parts/:part\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

None

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |
| part\_id | ID of the part |

### Request Body

None

### Request Body Format

None

### Request JSON Attributes

None

### Request Body Example

None

## Response

### Response Body Format

Success: No body

Failure: JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 204 OK |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 403 Forbidden | Not owner of the car or part on a car already |
| Failure | 404 Not Found | No car with car\_id exists or part with part\_id exists |

### Response Examples

#### Success

|  |
| --- |
| Status: 204 No Content |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

# Delete a Car (JWT Protected)

Allows you to delete a car. Note that if the car has parts, deleting the car makes the parts car property empty.

|  |
| --- |
| DELETE /cars/:car\_id |

## Request

### Authorization

Authorization: JWT Token

### Headers

None

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| car\_id | ID of the car |

### Request Body

None

## Response

### Response Body Format

Success: No body

Failure: JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 204 No Content |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 403 Forbidden | Not owner of the car |
| Failure | 404 Not Found | No car with car\_id exists |

### Response Examples

#### Success

|  |
| --- |
| Status: 204 No Content |

#### Failure

|  |
| --- |
| Status: 401 Unauthorized  {  "Error": "Invalid Token"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No car with this car\_id exists"  } |

# Create a Part

Allows you to create a new part.

|  |
| --- |
| POST /parts |

## Request

### Authorization

None

### Headers

None

### Path Parameters

None

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| name | Name of the part. | Yes |
| component | Part of the car the part is installed on. | Yes |
| install\_date | Date the parts was installed. | Yes |

### Request Body Example

|  |
| --- |
| {  "name": "Coilover",  "component": "Suspension",  "install\_date": "6/20/2015"  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 201 Created |  |
| Failure | 400 Bad Request | Missing attributes |
| Failure | 403 Forbidden | Invalid JWT |

#### Success

|  |
| --- |
| Status: 201 Created  {  "component": "Suspension",  "car": null,  "name": "Coilover",  "install\_date": "6/20/2015",  "id": "5695708787113984",  "self": "https://francnic-final-project.uw.r.appspot.com/parts/5695708787113984"  } |

#### Failure

|  |
| --- |
| Status: 400 Bad Request  {  "Error": "The request object is missing at least one of the required attributes"  } |

#### Failure

|  |
| --- |
| Status: 403 Forbidden  {  "Error": "Forbidden"  } |

# Get all Parts

List all the parts.

|  |
| --- |
| GET /parts |

## Request

### Authorization

None

### Headers

Accept: application/json

### Path Parameters

None

### Request Body

None

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "parts": [  {  "component": "Wheels and Tires",  "car": null,  "name": "Summer Tires",  "install\_date": "4/5/2017",  "id": "4766396177186816",  "self": "http://localhost:8080/parts/4766396177186816"  },  {  "component": "Engine",  "car": null,  "name": "Intake",  "install\_date": "10/20/2015",  "id": "4920093460922368",  "self": "http://localhost:8080/parts/4920093460922368"  },  {  "component": "Suspension",  "car": null,  "name": "Coilover",  "install\_date": "6/20/2015",  "id": "6595983525806080",  "self": "http://localhost:8080/parts/6595983525806080"  }  ]  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Get a Part

Allows you to get an existing part.

|  |
| --- |
| GET /parts/:part\_id |

## Request

### Authorization

None

### Headers

Accept: application/json

## Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| part\_id | ID of the part |

### Request Body

None

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 404 Found | No part with part\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "owner": "auth0|65721420a1027c81f5dbdebb",  "year": 2004,  "parts": [],  "model": "S2000",  "make": "Honda",  "id": "4899245085687808",  "self": "http://localhost:8080/cars/4899245085687808"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No part with this part\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Acceptable  {  "Error": "Not Acceptable”  } |

# Edit a Part (Patch)

Allows you to edit the properties of a part.

|  |
| --- |
| PATCH /parts/:part\_id |

## Request

### Authorization

None

### Headers

Accept: application/json

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| part\_id | ID of the part |

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| name | Name of the part. | No |
| component | Part of the part the part is installed on. | No |
| install\_date | Date the parts was installed. | No |

### Request Body Example

|  |
| --- |
| {  "install\_date": "7/20/2016"  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 404 Not Found | No part with part\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {  "component": "Suspension",  "car": null,  "name": "Coilover",  "install\_date": "7/20/2016",  "id": "6595983525806080",  "self": "http://localhost:8080/parts/6595983525806080"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No part with this part\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Found  {  "Error": "Not Acceptable”  } |

# Edit a Part (Put)

Allows you to edit all properties of a part.

|  |
| --- |
| PUT /parts/:part\_id |

## Request

### Authorization

None

### Headers

Accept: application/json

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| part\_id | ID of the part |

### Request Body

Required

### Request Body Format

JSON

### Request JSON Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Required?** |
| name | Name of the part. | Yes |
| component | Part of the part the part is installed on. | Yes |
| install\_date | Date the parts was installed. | Yes |

### Request Body Example

|  |
| --- |
| {  "name": "Bucket Seat",  "component": "Interior",  "install\_date": "5/25/2019"  } |

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 400 Bad Request | Missing attributes |
| Failure | 404 Not Found | No part with part\_id exists |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  {{  "component": "Interior",  "car": null,  "name": "Bucket Seat",  "install\_date": "5/25/2019",  "id": "6595983525806080",  "self": "http://localhost:8080/parts/6595983525806080"  } |

#### Failure

|  |
| --- |
| Status: 400 Bad Request  {  "Error": "The request object is missing at least one of the required attributes"  } |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No part with this part\_id exists"  } |

#### Failure

|  |
| --- |
| Status: 406 Not Found  {  "Error": "Not Acceptable”  } |

# Delete a Part

Allows you to delete a part. Note that if car has this part installed, deleting the part will remove the part from the cars parts list property.

|  |
| --- |
| DELETE /parts/:part\_id |

## Request

### Authorization

None

### Headers

None

### Path Parameters

|  |  |
| --- | --- |
| **Name** | **Description** |
| part\_id | ID of the part |

### Request Body

None

## Response

### Response Body Format

Success: No body

Failure: JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 204 No Content |  |
| Failure | 404 Not Found | No part with part\_id exists |

### Response Examples

#### Success

|  |
| --- |
| Status: 204 No Content |

#### Failure

|  |
| --- |
| Status: 404 Not Found  {  "Error": "No part with this part\_id exists"  } |

# Get all Users

List all the parts.

|  |
| --- |
| GET /users |

## Request

### Authorization

None

### Headers

None

### Path Parameters

None

### Request Body

None

## Response

### Response Body Format

JSON

### Response Statuses

|  |  |  |
| --- | --- | --- |
| **Outcome** | **Status Code** | **Notes** |
| Success | 200 OK |  |
| Failure | 401 Unauthorized | Invalid token |
| Failure | 406 Not Acceptable | Server only returns JSON |

### Response Examples

#### Success

|  |
| --- |
| Status: 200 OK  [  {  "user\_id": "auth0|65768d17e94488fb5f792d32",  "id": "5072058866204672",  "self": "http://localhost:8080/users/5072058866204672"  },  {  "user\_id": "auth0|65721420a1027c81f5dbdebb",  "id": "5706627130851328",  "self": "http://localhost:8080/users/5706627130851328"  }  ] |