Russell Feathers
CS 493: Cloud Applications
Final Assignment: Full Marina
06/11/23

API Documentation Table of Contents

Base URL & Login URL Address

https://marina-388119.ue.r.appspot.com

DataStore Data Model

The app stores three kinds of entities in Datastore:

Boats

Property	Reqd	Data Type	Notes	Example
id	No	Int	The id of the boat. Datastore automatically generates it. May not be changed.	5645760532054016
name	Yes	Str	Name of the boat. Only lowercase letters, spaces, and uppercase letters.	Odessa
type	Yes	Str	Type of the boat. E.g., Sailboat, Catamaran, etc. Only lowercase letters, spaces, and uppercase letters.	Sailboat
length	Yes	Int	The length of the boat in feet.	200
owner	No	Str	Unique id of the user that owns the boat	auth0|6466ceea29c87bf8f14f3e29
loads	No	List [Int]	list of loads, ea. load contains id (see loads for description). Set to an empty list on creation.	[7645760532054016, 7645760532054017]
self	No	Str	Resource location. May not be changed.	{base_url}/boats/5645760532054016

Loads

Property	Reqd	Data Type	Notes	Example
id	No	Int	The id of the load. Datastore automatically generates it. May not be changed.	8645760532054016
volume	Yes	Int	Volume of the load	1940
item	Yes	Str	name/ description. Only lowercase letters, spaces, and uppercase letters.	Rangers Hockey Pucks
creation_ date	Yes	Str	date load was created. Must be valid data format as YYYY-MM-DD	1994-06-14
carrier	No	int	Boat id that owns the boat, set to None on creation	5645760532054016
self	No	Str	Resource location. May not be changed.	{base_url}/loads/8645760532054016

Users

Property	Reqd	Data Type	Notes	Example
id	No	Int	Unique id of the user. JWT sub value	auth0|6466ceea29c87bf8f14f3e29
nickname	Yes	Str	Nickname of the user.	wallace
email	Yes	Str	Email of the user	wallace@cheese.com
id_token	Yes	Str	Id_Token of last valid JWT	eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InBzRlJyU2h0Z….

Relationship between User and Boats

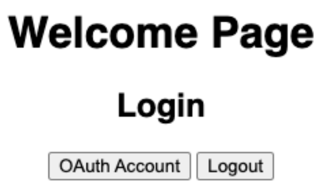
Users own boats. Only one user may own a given boat at a time. User does not store the reference to the boat, but the boat does store the reference to the user, aka owner.

Relationship between Boats and Loads

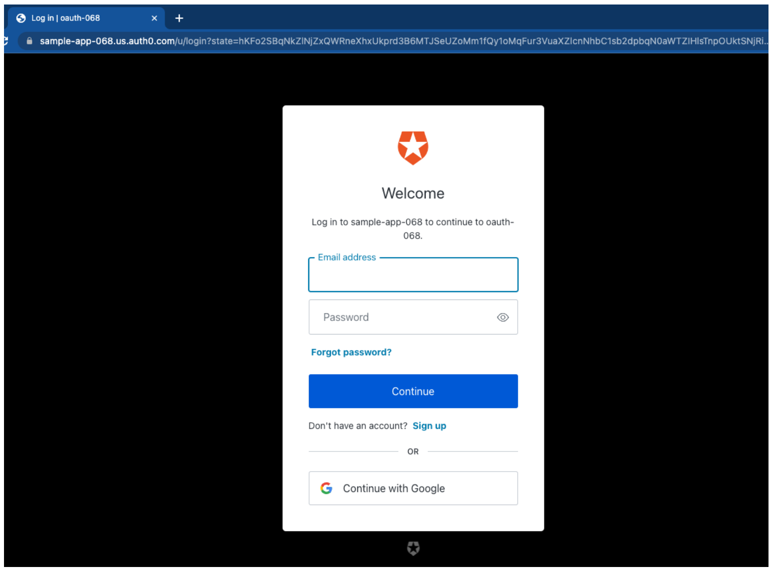
Boats contain loads. A boat can contain multiple loads, and a load may only be on one boat. Each entity contains a reference to the other. The stored reference is merely the id i.e. a boat stores the id of the loads, and a load stores the id of the boat that is carrying it.

Authorization Documentation

Landing Page

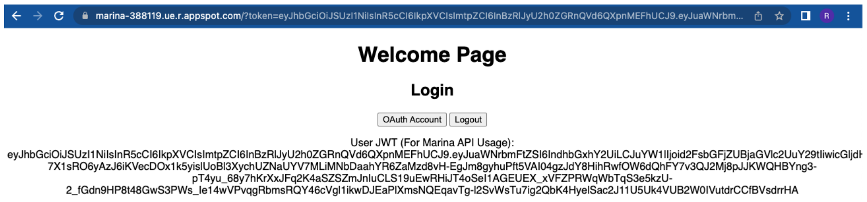


Signing Up and Log In


Log in | oauth-068
sample-app-068.us.auth0.com/u/login?state=hKFo2SBqNkZlNjZxQWRneXhxUkprd3B6MTJSeUZoMm1fQy1oMqFur3VuaXZlcnNhbC1sb2dpbqN0aWTZIHlsTnpOUktSNjRi...
Welcome
Log in to sample-app-068 to continue to oauth-068.
Email address
Password
Forgot password?
Continue
Don't have an account? Sign up
OR
Continue with Google

JWT Usage

Some API Routes require a JWT id_token. The token is displayed on the landing page after logging in. Also the last valid JWT for each user is available from the get users api route.


marina-388119.ue.r.appspot.com/?token=eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InBzRlJyU2h0ZGRnQVd6QXpnMEFhUCJ9.eyJuaWNrbm...
Welcome Page
Login
OAuth Account
Logout

User JWT (For Marina API Usage):

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InBzRlJyU2h0ZGRnQVd6QXpnMEFhUCJ9.eyJuaWNrbmFtZSI6IndhbGxhY2UiLCJuYW1lIjoid2FsbGFjZUBjaGVlc2UuY29tIiwicGljdH
7X1sRO6yAzJ6iKVecDOx1k5yislUoBl3XychUZNaUYV7MLiMNbDaahYR6ZaMzd8vH-EgJm8gyhuPft5VAI04gzJdY8HihRwfOW6dQhFY7v3QJ2Mj8pJJKWQHBYng3-
pT4yu_68y7hKrXxJFq2K4aSZSZmJnIuCLS19uEwRHiJT4oSeI1AGEUEX_xVFZPRWqWbTqS3e5kzU-
2_fGdn9HP8t48GwS3PWs_Ie14wVPvqgRbmsRQY46cVgl1ikwDJEaPlXmsNQEqavTg-l2SvWsTu7ig2QbK4HyelSac2J11U5Uk4VUB2W0IVutdrCCfBVsdrrHA
```

API Documentation - Users

NOTE: Default Response and Request is JSON unless otherwise noted

Get users

- GET {base_url}/users
- Gets all users (no pagination)
- Request Parameters & Body:
  - None
- Successful Status Code: 200
  - Sample Response:

```
[
  {
    "email": "wallace@cheese.com",
    "id_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InBzRlJyU2h0ZGRnQVd6QXpnMEFhUCJ9.
eyJuaWNrbmFtZSI6IndhbGxhY2UiLCJuYW1lIjoid2FsbGFjZUBjaGVlc2UuY29tIiwicGljdHVyZSI6
Imh0dHBzOi8vcy5ncmF2YXRhci5jb20vYXZhdGFyL2Q2NjBhMjE4ODNhNGNjYjM3OWFkMWNl
MWIxOTg0M2U5P3M9NDgwJnI9cGcmZD1odHRwcyUzQSUyRiUyRmNkbi5hdXRoMC5jb20lMk
ZhdmF0YXJzJTJGd2EucG5nIiwidXBkYXRlZF9hdCI6IjIwMjMtMDYtMDNUMDM6MDM6MDUuO
DI3WiIsImVtYWlsIjoid2FsbGFjZUBjaGVlc2UuY29tIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlL
CJpc3MiOiJodHRwczovL3NhbXBsZS1hcHAtMDY4LnVzLmF1dGgwLmNvbS8iLCJhdWQiOiJV
UVp5VWxLUFZmdEoxSmpXVTR2bXhuT1JIREtUcXhKZCIsImlhdCI6MTY4NTc2MzE2MiwiZXh
wIjoxNjg1Nzk5MTYyLCJzdWIiOiJhdXRoMHw2NDY2Y2VlYTI5Yzg3YmY4ZjE0ZjNlMjkiLCJzaW
QiOiJRREdHckVDTGlORjlyaHI2R0RkVndfVTNEM0pHY2JFTyIsIm5vbmNlIjoiRUxpRm9ielgwZn
JaUlJNVHlpdVUifQ.IdhaQ4pQsCdomZtkWS-FLvUdZAcK3KBA_9ytM4xuQP3iIoBrwuDK6ECLH
qmInYQdhwcWI-gDIPneZbuSDK0Y9bRvw00Q6wxCc0NttMW_0duEs82k2DACghDSllwoNf-Gitc
CVuGx09AEDgdvm2Pf_qkHgMeKAJtE_ZEhqrwcyhlHL5N2AXO-LyU7_P6OpKHobqx9FGgMpTr
bZ3ZNAm1SpK1ZvJiiwWTTSRr4BAVjXnw70SBxaFnC7y1dcVAD3Ak4RRefKyauWefJuVOG8F-
5KfupI9rnBdyVy8vaItX_zMt6S9PnkBmru8P3BTKMpJqb4tWeoi3wzBbxjFILflvG8A",
    "nickname": "wallace",
    "sub": "auth0|6466ceea29c87bf8f14f3e29"
  },
  {
    "email": "nowhere@somewhere.com",
    "id_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6InBzRlJyU2h0ZGRnQVd6QXpnMEFhUCJ9.
eyJuaWNrbmFtZSI6Im5vd2hlcmUiLCJuYW1lIjoibm93aGVyZUBzb21ld2hlcmUuY29tIiwicGljdHV
yZSI6Imh0dHBzOi8vcy5ncmF2YXRhci5jb20vYXZhdGFyLzNkMmE3NGMwNzI4ZmZiMDE1OTY
1NWMxZTA3MzJiZDJhP3M9NDgwJnI9cGcmZD1odHRwcyUzQSUyRiUyRmNkbi5hdXRoMC5jb
20lMkZhdmF0YXJzJTJGbm8ucG5nIiwidXBkYXRlZF9hdCI6IjIwMjMtMDYtMDNUMDM6MzM6Mj
UuMjQ0WiIsImVtYWlsIjoibm93aGVyZUBzb21ld2hlcmUuY29tIiwiZW1haWxfdmVyaWZpZWQiO
mZhbHNlLCJpc3MiOiJodHRwczovL3NhbXBsZS1hcHAtMDY4LnVzLmF1dGgwLmNvbS8iLCJhd
WQiOiJVUVp5VWxLUFZmdEoxSmpXVTR2bXhuT1JIREtUcXhKZCIsImlhdCI6MTY4NTc2MzIw
NSwiZXhwIjoxNjg1Nzk5MjA1LCJzdWIiOiJhdXRoMHw2NDdhYWI3OTdmMGE1MjE2MmNmYT
E0ZjciLCJzaWQiOiJkT1BPVTNYRVVobIJKUUp1Sk5jQWtYTW1mazhHa0dDcyIsIm5vbmNlIjoid
UVwMUw2SVM0SXZCenk4R3NRSkYifQ.Rm-1YIvf1OId0ut8nUde9Gc16Imt8eVyITYNjdQsfJf4iT
2rW0_-H52qzsCH8CjOV4PIxHtEpS5dUGTkmXxqUOLZ7oTb1LmAuACPDIUS5iTjTxiQkuZYNh
Ucm4VDWYjnXkKIC0gxOxJGgOQyddeoTgcaOU0qiIoxyeTcdIhserFViByVIkipmwDNH7-XdS_I_
Wp828F-se1VpdMksCMkWstAV4qgCc01_8Yzyiae2E8P7Tjcc2SiGTheoJg42IHtT32Syxe6NuQC
jO6xjzkAUo0JNnKbxvnXJZOuCQBnmU7KsxTwbqE7CtRWGoTthIZUYL5cSrcwh4AXjNLT1SRY
Fw",
      "nickname": "nowhere",
      "sub": "auth0|647aab797f0a52162cfa14f7"
  }
]
```

API Documentation - Boats

NOTE: Default Response and Request is JSON unless otherwise noted

Create Boat (JWT)

- POST {base_url}/boats
- Create a new boat and sends back created boat in response
- Request Parameters:
    - None
- Request Headers:
    - JWT
- Request JSON Body:
    - Required:
        - “name”
        - “type”
        - “length”
    - Optional: None
    - Sample Request:
      ```
      {
      "name": "Sea Witch",
      "type": "Catamaran",
      "length": 28
      }
      ```
- Successful Status Code: 201
    - Sample Response:
      ```
      {
        "name": "Sea Witch",
        "type": "Catamaran",
        "length": 28,
        "owner": "auth0|6466ceea29c87bf8f14f3e29",
        "loads": [],
        "id": 5634161670881280,
        "self": "{base_url}/boats/5634161670881280"
      }
      ```
- Unsuccessful Status Code:
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 401, 406

Get Boat by Id (JWT)

- GET {base_url}/boats/{boatId}
- Retrieve information about a specific boat.
- Request Parameters:
    - boatId (int): ID of the boat to retrieve information for
- Request Headers:
    - JWT
- Request Body:
    - None
- Response Format:
    - JSON
- Successful Status Code: 200
    - Sample Response:

```
{
  "owner": "auth0|6466ceea29c87bf8f14f3e29",
  "name": "Sea Witch",
  "length": 99,
  "loads": [
     5702893864747008
  ],
  "type": "Catamaran",
  "id": 5631671361601536,
  "self": "{base_url}/boats/5631671361601536"
}
```

- Unsuccessful Status Code:
    - See General Unsuccessful Status Codes
    - Applicable Codes: 401, 403, 404, 406

View all boats (JWT, supports pagination)

- GET {base_url}/boats
- Retrieve information about all boats with support for pagination.
- Request Headers:
  - JWT
- Request Parameters:
  - page (int): Page number to retrieve (optional, default is 0)
  - limit (int): Number of boats per page (optional, default is 5, max is 20)
  - sample format:
    - {base_url}/boats/?limit=1&offset=1
- Request Body:
  - None
- Response Format:
  - JSON
  - Link to Next Page (See Sample)
- Successful Status Code: 200

```
{
  "boats": [
     {
        "id": 5632499082330112,
        "length": 28,
        "loads": [],
        "name": "Sea Witch",
        "owner": "auth0|6466ceea29c87bf8f14f3e29",
        "self": "{base_url}/boats/5632499082330112",
        "type": "Catamaran"
     },
     {
        "id": 5634161670881280,
        "length": 28,
        "loads": [],
        "name": "Sea Witch",
        "owner": "auth0|6466ceea29c87bf8f14f3e29",
        "self": "{base_url}/boats/5634161670881280",
        "type": "Catamaran"
     },
     {
        "id": 5636645067948032,
        "length": 28,
        "loads": [],
        "name": "Sea Witch",
        "owner": "auth0|6466ceea29c87bf8f14f3e29",
        "self": "{base_url}/boats/5636645067948032",
        "type": "Catamaran"
        }
    ],
    "next": "{base_url}/boats/?limit=3&offset=3"
}
```

- Unsuccessful Status Code:
    - See General Unsuccessful Status Codes
    - Applicable Codes: 401, 406

Delete Boat (JWT)

- DELETE {base_url}/boats/{boatId}
- Deletes a boat with the specified ID.
- Request Headers:
    - JWT
- Request Parameters:
    - boatId (int): ID of the boat to delete
- Request Body:
    - None
- Response Format:
    - No Content
- Successful Status Code: 204
    - No Body Response
- Unsuccessful Status Code
    - See General Unsuccessful Status Codes
    - Applicable Codes: 401, 403, 404

Edit - Patch Boat (JWT)

- PATCH {base_url}/boats/{boatId}
- Partially edits a boat with the specified ID
- Request Headers:
  - JWT
- Request Parameters:
  - boatId (int): ID of the boat to edit
- Request Body:
  - Optional (at least one required)l:
    - "name"
    - "type"
    - "length"
  - Sample Request:
    ```
    {
    "name": "Sailfish",
    "length": 40
    }
    ```
- Response Format:
  - JSON
- Successful Status Code: 201
  - Sample Response:
    ```
    {
      "owner": "auth0|6466ceea29c87bf8f14f3e29",
      "name": "Sea Witch Patched",
      "length": 99,
      "loads": [],
      "type": "Catamaran",
      "id": 5646488461901824,
      "self": "{base_url}/boats/5646488461901824"
    }
    ```
- Unsuccessful Status Codes:
  - See General Unsuccessful Status Codes
  - Applicable Codes: 400, 401, 403, 404, 406

Edit - Put a boat (JWT)

- PUT {base_url}/boats/{boatId}
- Full edit of a boat with the specified ID
- Request Headers:
  - JWT
- Request Parameters:
  - boatId (int): ID of the boat to edit
- Request Body:
  - Required:
    - "name"
    - "type"
    - "length"
  - Sample Request:
    ```
    { "name": "Sailfish",
    "type": "Sailboat",
    "length": 40 }
    ```
- Response Format:
  - JSON
- Successful Status Code: 201
  - Sample Response:
    ```
    {
      "owner": "auth0|6466ceea29c87bf8f14f3e29",
      "name": "Sea Witch Putted",
      "length": 99,
      "loads": [],
      "type": "Catamaran Putted",
      "id": 5631671361601536,
      "self": "{base_url}/boats/5631671361601536"
    }
    ```

- Unsuccessful Status Codes:
  - See General Unsuccessful Status Codes
  - Applicable Codes: 400, 401, 403, 404, 406

API Documentation - Loads

NOTE: Default Response and Request is JSON unless otherwise noted

Create Load

- POST {base_url}/loads
    - Create a new load with the provided data.
- Request Parameters:
    - None
- Request Body:
    - Required
        - “volume”
        - “item”
        - “creation_date”
    - Sample Request:

```
{
   "volume": 100,
   "item": "Boxes of Apples",
   "creation_date": "2023-05-01"
}
```

- Response Format:
    - JSON
- Successful Status Code: 201
    - Sample Response:

```
{
  "item": "Boxes of Apples",
  "volume": 100,
  "creation_date": "2023-05-01",
  "carrier": null,
  "id": 5638358357245952,
  "self": "{base_url}/loads/5638358357245952"
}
```

- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 406

Get Load by ID

- GET {base_url}/loads/{loadId}
- Retrieve information about a specific load.
- Request Parameters:
    - loadId (int): ID of the load to retrieve information
- Response Format:
    - JSON
- Successful Status Code: 200
    - Sample Response:
```
{
  "volume": 100,
  "item": "Boxes of Apples",
  "carrier": null,
  "creation_date": "2023-05-01",
  "id": 5638358357245952,
  "self": "{base_url}/loads/5638358357245952"
}
```
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 404, 406

Get all Loads (supports pagination)

- GET {base_url}/loads
  - Retrieve a list of all loads.
- Request Parameters
  - page (int): Page number to retrieve (optional, default is 0)
  - limit (int): Number of boats per page (optional, default is 5, max is 20)
  - sample format:
    - {base_url}/boats/?limit=1&offset=1
- Response Format:
  - JSON
- Successful Status Code: 200
  - Sample Response:

```
{
  "loads": [
    {
      "carrier": null,
      "creation_date": "2023-05-01",
      "id": 5633378543992832,
      "item": "Boxes of Apples",
      "self": "{base_url}/loads/5633378543992832",
      "volume": 100
    },
    {
      "carrier": null,
      "creation_date": "2023-05-01",
      "id": 5635008819625984,
      "item": "Boxes of Apples",
      "self": "{base_url}/loads/5635008819625984",
      "volume": 100
    },
    {
      "carrier": null,
      "creation_date": "2023-05-01",
      "id": 5638358357245952,
      "item": "Boxes of Apples",
      "self": "{base_url}/loads/5638358357245952",
      "volume": 100
    }
  ],
  "next": "{base_url}/loads/?limit=3&offset=3"
}
```

- Unsuccessful Status Codes
  - See General Unsuccessful Status Codes
  - Applicable Codes: 406

Delete Load

- DELETE {base_url}/loads/{loadId}
- Delete a specific load.
- Request Parameters:
    - loadId (int): ID of the load to be deleted
- Response Format:
    - No content
- Successful Status Code: 204
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 403, 404, 406

Put Load

- PUT {base_url}/loads/{loadId}
- Edit all parameters in a specific load.
- Request Parameters:
    - loadId (int): ID of the load to be deleted
- Request Body:
    - Required
        - “volume”
        - “item”
        - “creation_date”
    - Sample Request:
      ```
      {
         "volume": 100,
         "item": "Boxes of Apples",
         "creation_date": "2023-05-01"
      }
      ```
- Successful Status Code: 201
    - Sample Response:
      ```
      {
        "volume": 168,
        "item": "Boxes of Oranges",
        "carrier": null,
        "creation_date": "2023-05-31",
        "id": 5702893864747008,
        "self": "{base_url}/loads/5702893864747008"
      }
      ```
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 403, 404, 406

Patch Load

- PUT {base_url}/loads/{loadId}
- Edit all parameters in a specific load.
- Request Parameters:
    - loadId (int): ID of the load to be deleted
- Request Body:
    - Optional, but at least 1 Required
        - “volume”
        - “item”
        - “creation_date”
    - Sample Request:
      ```
      {
         "volume": 100,
         "creation_date": "2023-05-01"
      }
      ```
- Successful Status Code: 201
    - Sample Response:
      ```
      {
        "volume": 168,
        "item": "Boxes of Oranges",
        "carrier": null,
        "creation_date": "2023-05-31",
        "id": 5702893864747008,
        "self": "{base_url}/loads/5702893864747008"
      }
      ```
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 403, 404, 406

API Documentation - Boats and Loads

NOTE: Default Response and Request is JSON unless otherwise noted

Add Load to boat (JWT)

- PUT {base_url}/boats/{boatId}/loads/{loadId}
- Adds a load with the specified ID to a boat with the specified ID.
- Request Headers:
    - JWT
- Request Parameters:
    - boatId (int): ID of the boat to add the load to
    - loadId (int): ID of the load to add to the boat
- Request Body:
    - None
- Response Format:
    - No Content
- Successful Status Code: 204
    - No Content Returned
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 401, 403, 404, 406

Delete Load to boat (JWT)

- DELETE {base_url}/boats/{boatId}/loads/{loadId}
- Deletes a load with the specified ID from a boat with the specified ID.
- Request Headers:
    - JWT
- Request Parameters:
    - boatId (int): ID of the boat to remove the load from
    - loadId (int): ID of the load to remove from the boat
- Request Body:
    - None
- Returns:
    - No Content
- Successful Status Code: 204
    - No Content Returned
- Unsuccessful Status Codes
    - See General Unsuccessful Status Codes
    - Applicable Codes: 400, 401, 403, 404, 406

Unsuccessful Status Codes

- 400
    - Sample Response:
      {"Error" : "The request is missing at least one required attribute"}
      {"Error" : "Invalid Name. Either No input, too long (more than 255 characters), contains invalid characters, or does not begin or end with a letter"}
      {"Error" : "Invalid Type. Either No input, too long (more than 255 characters), contains invalid characters, or does not begin or end with a letter"},
      {"Error" : "Invalid Length. Not an integer"},
      {"Error" : "Invalid id type"},
      {"Error" : "Invalid Date. YYYY-MM-DD format required."},
      {"Error" : "Invalid Type. Either No input, too long (more than 255 characters), contains invalid characters, or does not begin or end with a letter"},
      {"Error" : "Invalid Length. Not an integer"},
      {"Error" : "Invalid id type"},

- 403
    - Sample Response:
      {"Error" : "Boat does not belong to JWT owner"}
- 404
    - Sample Response:
      {"Error" : "No boat with this boat_id exists"}
      {"Error" : "No load with this load_id exists"}
- 405
    - Sample Response:
      {"Error" : "Method not supported at this address"}
- 406
    - Sample Response:
      {"Error" : “Unsupported Accept Header"}