Final : Users, Boats, and Loads API Spec

By Brandon Withington

CS493 Cloud Application Development

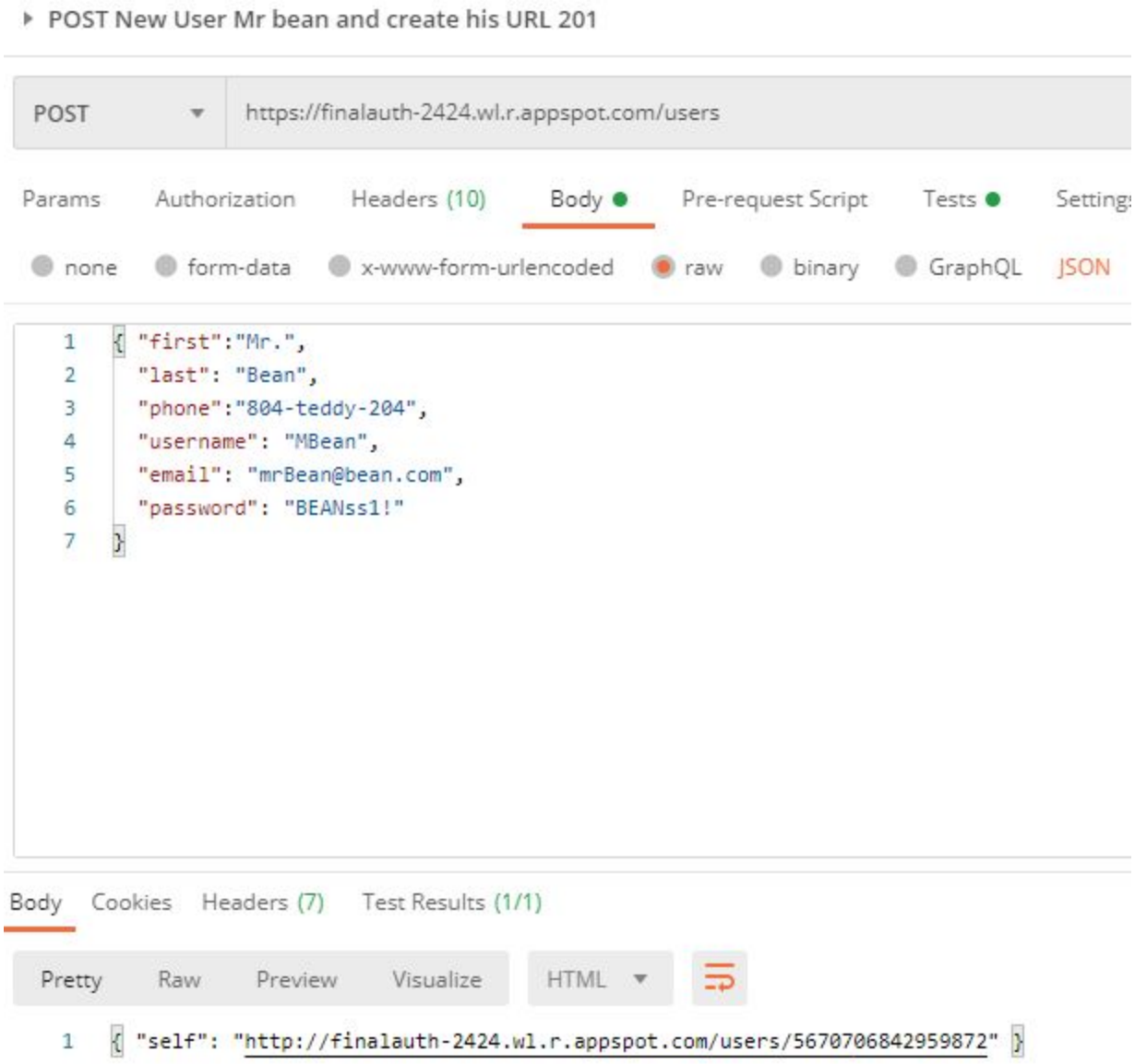
Fall 2020

Oregon State University

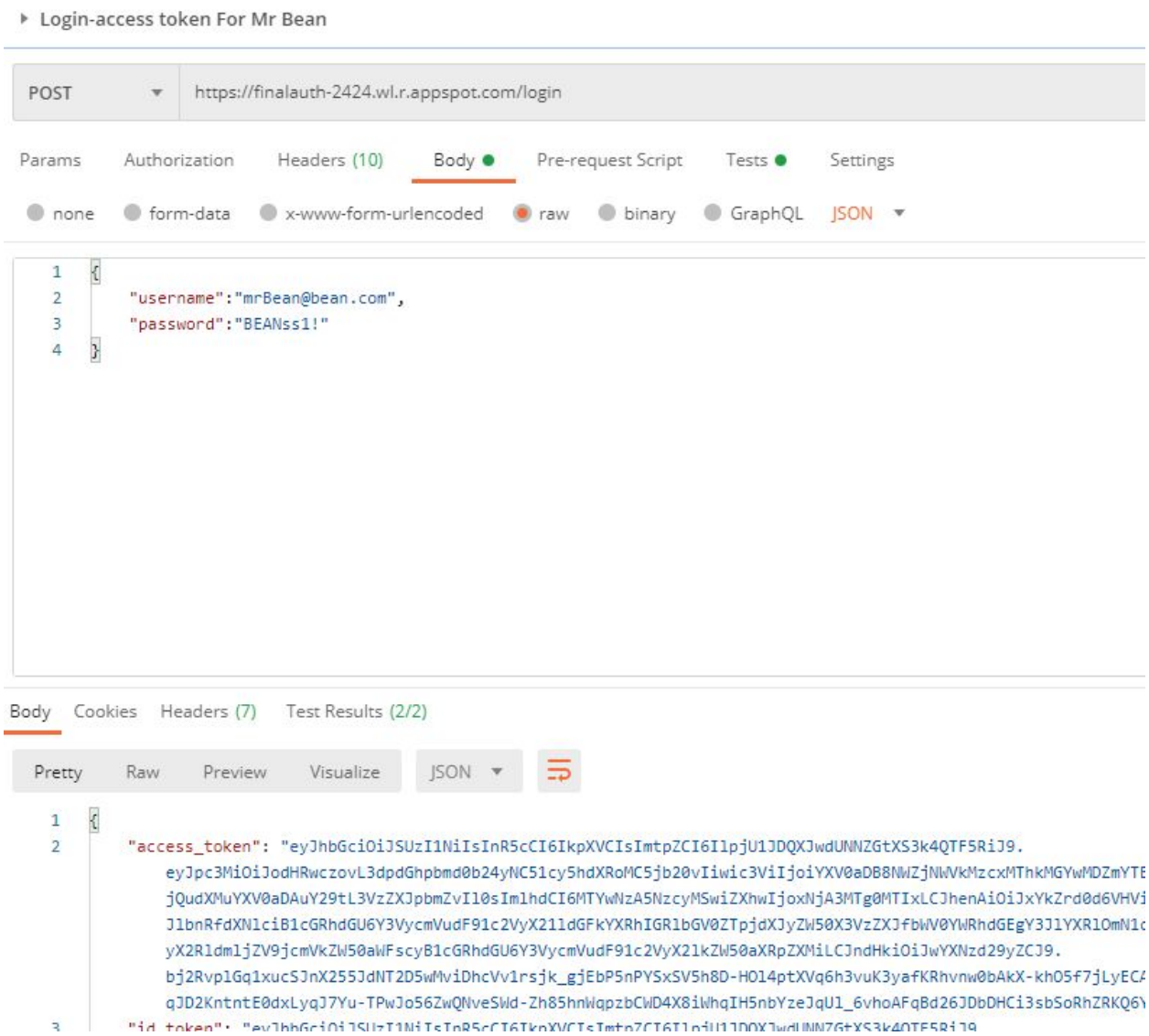
Last Updated : Dec, 3, 2020. 3:51am.

Page Number	Change to the page	Date Changed
0	Created Page 1	12 / 1 / 2020
1 - 10	Added general API specs	12 / 2 / 2020
10 - 16	Added create, delete, view boat pages.	12 / 3 / 2020
16 - 24	Added load spec pages	12 / 3 / 2020

Website URL:
https://finalauth-2424.wl.r.appspot.com
It redirects you to an oauth portal where you can make an account.
The tests in postman utilize already-setup accounts eg:



After that, the user can then log into the application via post-method :



API spec :

This API spec is an upgraded form of the API spec seen in assignment four. The interactions between boats and loads are markedly similar in how a boat can take on a load and how a load is applied and removed from a boat. The relationship between the user, the user's boats and the user's loads works in the following fashion:

- A boat can only have one owner.
- (In this spec) one owner can only have one boat.
- A load can be assigned to a boat.
- A boat can have that load be removed.
- A boat can be edited by the owner.

- A load can be edited by the owner.
- A boat can be removed only by its owner.
- A load can only be removed by its owner.
- A user that does not own the boat in question cannot delete it.
- A user that does not own the load in question cannot delete it.
- A user cannot evoke the delete command on all boats.
- A person without a JWT can only get the list of boats and loads.

Entities that are present are below :

User Entities : first, last, phone, username, boats, user_id.
Load Entities : weight, content, delivery_date, carrier, owner
Boat Entities : id, name, type, brand, model, size, color, owner, load

Entity examples :

Users


```
{
    "username": "jsmith",
    "first": "Saden",
    "boats": [],
    "phone": "503-243-4905",
    "last": "Jmith",
    "id": "5139943911325696",
    "self": "http://localhost:8080/users/5139943911325696"
}
```


Loads


```
{
    "weight": "189435",
    "delivery_date": "11/24/199999997",
    "content": "Elite gamerz",
    "carrier": null,
    "id": "4683990183182336"
}
```


Boats


```
[
    {
        "type": "CRUUUUZZER",
        "size": "very lorge",
        "brand": "bugotti",
        "color": "amber",
```

```
        "owner": null,
        "load": null,
        "model": "BmW",
        "id": "5103757536788480"
    }
]
```


What does not completely work:

1.) When the request to GET all users is done, it does not correctly print the boats that each user owns. To get this to do what I want it to do, I would have to do something similar to what I had done for to get the loads from boats in assignment four. I would have to make a function that retrieves the boats for each user through datastore and match it to the user that owns that boat.

2.) Loads are not properly being displayed on the boats that they are assigned to. Something weird has happened to my implementation from assignment four. The loads are not being displayed properly, yet they are being assigned correctly. This would fulfill the requirement to have a non-user entity to another non-user entity relationship substantially. In order to fix this I would have to reprogram the way that a load gets PUT onto a boat by retrieving both the boat and load from datastore, assigning them together using their API spec; load->carrier, boat->load.

LOGIN URL

POST login credentials

At this URL, the user can login and get their JWT token.

POST /login

Headers : Content-Type: application/json.

Request Body:

```
{
    "username":"smelly@smelly.com",
    "password":"Password123!"
}
```

Response example :

```
{
    "access_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IlpjU1JDQXJwdUNNZGtXS3k4QTF5RiJ9.eyJpc3MiOiJodH
RwczovL3dpdGhpbmd0b24yNC51cy5hdXRoMC5jb20vIiwic3ViIjoiYXV0aDB8NWZjNWVjYzMwYmQ0YzIwMDY4ZjcxN2
E1IiwiYXVkIjpbImh0dHBzOi8vd2l0aGluZ3RvbjI0LnVzLmF1dGgwLmNvbS9hcGkvdjIvIiwiaHR0cHM6Ly93aXRoaW
5ndG9uMjQudXMuYXV0aDAuY29tL3VzZXJpbmZvIl0sImlhdCI6MTYwNjk0MzQ1MCwiZXhwIjoxNjA3MDI5ODUwLCJhen
AiOiJxYkZrd0d6VHVieGJub2k5MzJVcmUxN3I0TUw4Ukp0OCIsInNjb3BlIjoib3BlbmlkIHByb2ZpbGUgZW1haWwgYW
RkcmVzcyBwaG9uZSByZWFkOmN1cnJlbnRfdXNlciB1cGRhdGU6Y3VycmVudF91c2VyX21ldGFkYXRhIGRlbGV0ZTpjdX
JyZW50X3VzZXJfbWV0YWRhdGEgY3JlYXRlOmN1cnJlbnRfdXNlcl9tZXRhZGF0YSBjcmVhdGU6Y3VycmVudF91c2VyX2
RldmljZV9jcmVkZW50aWFscyBkZWxldGU6Y3VycmVudF91c2VyX2RldmljZV9jcmVkZW50aWFscyB1cGRhdGU6Y3Vycm
VudF91c2VyX2lkZW50aXRpZXMiLCJndHkiOiJwYXNzd29yZCJ9.k8qbQMOGwhcp3rx57aUUacJ7OfjTNppSYVSMwCl0c
AyHZRvSF_UEjkAsvx0uP7zmYV-A3kcAiOYgKpiCDY5OkzgqwgZWK2-7d06XKergA8XTsZjcrRQm9ZyhDMAZeHyB8vaOm
GOGaN3xKPCEbr95NcOwLKgemy9AUqOmrNkcYnfpK8EZsoq-VZACeW16Mj1bRtlgz1DTekaNAsktfueASSG-kPeiY9OoQ
EO-obuo5VmTlmeV8nTUOUi8hsqeFLk06y5BAvrfiSXdXZq3Cw7tnjCgTacmDtvNevpHBI1XJ9CGQ9mnszuz8KDCIVQGk
EUWA_TdPJPM-GFO9rnDpv-SqA",
    "id_token":
"eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6IlpjU1JDQXJwdUNNZGtXS3k4QTF5RiJ9.eyJuaWNrbmFtZS
I6InNtZWxseSIsIm5hbWUiOiJzbWVsbHlAc21lbGx5LmNvbSIsInBpY3R1cmUiOiJodHRwczovL3MuZ3JhdmF0YXIuY2
9tL2F2YXRhci84MjZmNGJhMDc2NjRiMDllYWM2ODZlYzBlNGNkMGQ4Zj9zPTQ4MCZyPXBnJmQ9aHR0cHMlM0ElMkYlMk
ZjZG4uYXV0aDAuY29tJTJGYXZhdGFycyUyRnNtLnBuZyIsInVwZGF0ZWRfYXQiOiIyMDIwLTEyLTAyVDIxOjEwOjUwLj
c4NVoiLCJlbWFpbCI6InNtZWxseUBzbWVsbHkuY29tIiwiZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJpc3MiOiJodHRwcz
```

```
ovL3dpdGhpbmd0b24yNC51cy5hdXRoMC5jb20vIiwic3ViIjoiYXV0aDB8NWZjNWVjYzMwYmQ0YzIwMDY4ZjcxN2E1Ii
wiYXVkIjoicWJGa3dHelR1Ynhibm9pOTMyVXJlMTdyNE1MOFJKdDgiLCJpYXQiOjE2MDY5NDM0NTAsImV4cCI6MTYwNj
k3OTQ1MH0.fjU5IBL4uzyu0hdlwEWHH1Tlfq3bwpKo_aDZfMTAsEaEQnp8cMEMHGTMA2gQdB7XYZDxYtyMyi-BQNWOIG
yr4y1iecrxNgnHWWzPOiJVxgtEeBXUc74SojGSleJuMmIO-akxR2z7-EH-M0SDDVtGe_CmSYszAvzlUImAMBxnd0uQ1q
G5_7Bytt84fWYEtu5Mwn5A_zCvvhobs1GDUkCW2qFlNSGwtiO3C0ZPdS7f_vm_MsLyNnpnYmCzM_ffZoEcZ3dr-0GACo
QyMB5r3BvlKyHw6nuFGwNrcyhrsGiPJBxl0DnR6wRQrRj6NlXXHD5NAWa8qnd8jEGYThh9ewSLvg",
    "scope": "openid profile email address phone read:current_user
update:current_user_metadata delete:current_user_metadata create:current_user_metadata
create:current_user_device_credentials delete:current_user_device_credentials
update:current_user_identities",
    "expires_in": 86400,
    "token_type": "Bearer"
}
```


Response Code :

200

500 error.

User URLs

Get all users

At this URL, it returns all the users with pagination.

GET /users

Headers : Content-Type: application/json.

Request Body: N/A

Response example :


```
"items": [
        {
            "boats": [],
            "username": "MBean",
            "first": "Mr.",
            "phone": "804-teddy-204",
            "last": "Bean",
            "id": "4801335601922048",
            "self": "http://localhost:8080/users/4801335601922048"
        },
        {
            "username": "MBean",
            "phone": "804-teddy-204",
            "boats": [],
            "last": "Bean",
            "first": "Mr.",
            "id": "5080600515969024",
            "self": "http://localhost:8080/users/5080600515969024"
        },
        {
            "phone": "804-teddy-204",
            "username": "MBean",
            "boats": [],
            "last": "Bean",
            "first": "Mr.",
            "id": "5082810578632704",
            "self": "http://localhost:8080/users/5082810578632704"
        },
```

```
        {
            "first": "Mr.",
            "last": "Bean",
            "username": "MBean",
            "phone": "804-teddy-204",
            "boats": [],
            "id": "5085025104035840",
            "self": "http://localhost:8080/users/5085025104035840"
        },
        {
            "username": "MBean",
            "phone": "804-teddy-204",
            "boats": [],
            "first": "Mr.",
            "last": "Bean",
            "id": "5087093063680000",
            "self": "http://localhost:8080/users/5087093063680000"
        }
    ],
    "next":
"http://localhost:8080/users?cursor=CisSJWoQbX5maW5hbGF1dGgtMjQyNHIRCxIEVXNlchiAgIC48tWECQwY
ACAA",
    "total": 52
}
```


Response Codes :

200 Good

406 Not acceptable

POST a user

At this URL, the user can create a user that interacts with the system.

```
POST /users
```

Headers : Content-Type: application/json.

Request Body:

```
{ "first":"Jaden",
  "last": "Smith",
  "phone":"503-243-4905",
  "username": "jsmith",
  "email": "smelly@smelly.com",
  "password": "Password123!"
}
```

Response example :

```
{ "self": "http://localhost:8080/users/5139943911325696" }
```

Response Code :

201

EDIT (PUT) a user

At this URL, the user can create a user that interacts with the system.

```
PUT /users/{{user_id}}
```

Headers : Content-Type: application/json.

Request Body:

```
{
    "first":"Saden",
   "last": "Jmith"
}
```

Response example :

```
Status code is 200.

GET /users/{{user_id}}

{
    "username": "jsmith",
    "first": "Saden",
    "boats": [],
    "phone": "503-243-4905",
    "last": "Jmith",
    "id": "5139943911325696",
    "self": "http://localhost:8080/users/5139943911325696"
}
```

Response Code :

200

PUT add a boat to a user

At this URL, the user can add a boat they made to their account persay.

PUT /users/{{user_id}} / boats / {{boat_id}}

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :

```
Status code is 201.
{ "self": "http://localhost:8080/users/5144701594238976" }
```

Response Code :

- 201 Good.
- 403 Forbidden
- 403 Boat already has an owner

DELETE remove a boat from a user

At this URL, the user can remove a boat they own.

DELETE /users/{{user_id}} / boats / {{boat_id}}

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :

```
Status code is 204
```

Response Code :

- 204 Good.
- 403 Forbidden.

BOAT URLs

GET all boats

At this URL, the user can get all boats

```
GET /boats
```

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :


```
"items": [
        {
            "brand": "bugotti",
            "type": "CRUUUUZZER",
            "color": "amber",
            "load": null,
            "owner": "6046718076387328",
            "model": "BmW",
            "size": "very lorge",
            "id": "4675146040213504",
            "self": "http://localhost:8080/boats/4675146040213504"
        },
        {
            "load": null,
            "name": "THE BEAN",
            "owner": null,
            "type": "BEAN BOAT",
            "length": "APROX. 20 BEANS",
            "id": "4685230422097920",
            "self": "http://localhost:8080/boats/4685230422097920"
        },
        {
            "name": "The Smith boat",
            "length": "18",
```

```
            "owner": "smelly@smelly.com",
            "type": "cruiser",
            "load": null,
            "id": "4824727671537664",
            "self": "http://localhost:8080/boats/4824727671537664"
        },
        {
            "model": "BmW",
            "owner": "smelly@smelly.com",
            "color": "amber",
            "type": "CRUUUUZZER",
            "size": "very lorge",
            "load": null,
            "brand": "bugotti",
            "id": "4825967910453248",
            "self": "http://localhost:8080/boats/4825967910453248"
        },
        {
            "owner": "smelly@smelly.com",
            "size": "very lorge",
            "brand": "bugotti",
            "type": "CRUUUUZZER",
            "color": "amber",
            "load": null,
            "model": "BmW",
            "id": "4837131872632832",
            "self": "http://localhost:8080/boats/4837131872632832"
        }
    ],
    "next":
"http://localhost:8080/boats?cursor=CiwSJmoQbX5maW5hbGF1dGgtMjQyNHISCxIFQm9hdHMYgICAuIjrywgM
GAAgAA=="
}
```


Response Codes :

200 Good.

406 Not acceptable.

GET an individual boats

At this URL, the user can a boat (that is owned by another person)

```
GET /boats / {{boat_id}}
```

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :
Successful:


```
[
    {
        "load": null,
        "type": "cruiser",
        "length": "18",
        "name": "The Smith boat",
        "owner": null,
        "id": "5980884817674240"
        "self": "http://localhost:8080/boats/5103757536788480"

    }
]
```


Unsuccessful:
401 Unauthorized code :

```
UnauthorizedError: No authorization token was found<br>    at middleware
```

401 invalid JWT :

```
UnauthorizedError: No authorization token was found<br>    at middleware
```

POST a BOAT

At this URL, the user can remove a boat they own.

POST /boats

Headers : Content-Type: application/json.

Request Body:

```
{
    "brand": "bugotti",
    "type": "cruiser",
    "size": "very lorge",
    "model": "BmW",
    "color": "amber",
    "name":"The Smith boat",
    "length": "18",
    "public": "public"
}
```

Response example :

```
Status code is 201
{
    "id": "5103757536788480",
    "name": "The Smith boat",
    "type": "cruiser",
    "length": "18",
    "owner": null,
    "public": "public",
    "self": "http://localhost:8080/boats/5103757536788480"
}
```

Response Code :

201 Good.

403 Forbidden.

PUT edit a BOAT

At this URL, the user can remove a boat they own.

PUT /boats / {{boat_id}}

Headers : Content-Type: application/json.

Request Body:

```
{
    "brand": "bugotti",
    "type": "CRUUUUZZER",
    "size": "very lorge",
    "model": "BmW",
    "color": "amber",
    "name":"The Smith boat",
    "length": "18",
    "public": "public"
}
```

Response example :

```
Status code is 200
[
    {
        "type": "CRUUUUZZER",
        "size": "very lorge",
        "brand": "bugotti",
        "color": "amber",
        "owner": null,
        "load": null,
        "model": "BmW",
        "id": "5103757536788480"
        "self": "http://localhost:8080/boats/5103757536788480"

    }
]
```

Response Code :

200 Good.

403 Forbidden.

PUT a load onto a boat

At this URL, the user can put a load onto a boat.

PUT /boats / {{boat_id}} / loads / {{load_id}}

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :

```
Status code is 204
{ "self": "http://localhost:8080/boats/5144701594238976" }
```

Response Code :

204 Good.

403 Forbidden.

Response Examples

Success

Status: 204 No content

Failure

Status: 403 Forbidden

{
“Error”: “The load has already been assigned”
}

Status: 404 Not Found (In the case that an invalid load is assigned to a boat)
{
“Error”:The load / boat from the request does not exist”
}

Status: 404 Not Found (In the case that the boat cannot be found from the ID)
{
“Error”:The boat does not exist”
}

Status: 404 Not Found (In the case that the boat and the load cannot be found from the IDs)
{
“Error”: The load / boat from the request does not exist”
}

View all loads for a given Boat

This allows the user to view a list of the loads that have been assigned to a particular boat at the address of the id.

Get / boats / :boat_id / loads

Requests

Request Parameters

None

Request Body

None

Response

Response Body format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 found


```
[
 {
    "content": "Elite gamerz",
    "delivery_date": "11/24/199999997",
    "weight": "189435",
    "carrier": null,
    "id": "4683990183182336"
}

 {
    "content": "Elite gamerz",
    "delivery_date": "11/24/199999997",
    "weight": "189435",
```

```
    "carrier": null,
    "id": "4683990183182336"
},

 {
    "content": "Elite gamerz",
    "delivery_date": "11/24/199999997",
    "weight": "189435",
    "carrier": null,
    "id": "4683990183182336"
}
]
```

Delete a Boat

This allows the user to delete an existing boat. It unloads the loads that are currently on the boat, not deleting the loads outright.

DELETE /boats/:boat_id

Requests

Request Parameters

None

Request Body

None

Response

Response Body format
JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No content	
Failure	404 Not found	No boat with this boat_id exists

Response Examples

Success

Status: 204 No content

Failure

Status: 404 Not Found
 	
{
“Error”: “No boat with this boat_id exists”
}

LOAD URLs

GET all loads

At this URL, the user can get all loads

```
GET /loads
```

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :


```
"items": [
        {
            "weight": "189435",
            "delivery_date": "11/24/199999997",
            "carrier": null,
            "content": "Elite gamerz",
            "id": "4683990183182336"
        },
        {
            "carrier": "4795176081948672",
            "delivery_date": "11/24/1997",
            "weight": "189435",
            "content": "Gamer",
            "id": "4722489129172992"
        },
        {
            "content": "Gamer",
            "delivery_date": "11/24/1997",
            "weight": "189435",
            "carrier": null,
            "id": "4863226617528320"
        },
        {
```

```
            "content": "Gamer",
            "delivery_date": "11/24/1997",
            "weight": "189435",
            "carrier": null,
            "id": "4899245085687808"
        },
        {
            "delivery_date": "11/24/1997",
            "content": "Gamer",
            "carrier": "5107442887163904",
            "weight": "189435",
            "id": "5076651058659328"
        }
    ],
    "next":
"http://localhost:8080/loads?cursor=CiwSJmoQbX5maW5hbGF1dGgtMjQyNHISCxIFTG9hZHMYgICA+P6lggkMGAAgAA=="
}
```


Response Codes :

200 Good.

406 Not acceptable.

GET an individual load

At this URL, the user can a boat (that is owned by another person)

```
GET /loads / {{load_id}}
```

Headers : Content-Type: application/json.

Request Body:

```
N/A
```

Response example :
Successful:

```
{
    "weight": "1337",
    "delivery_date": "11/24/1997",
    "content": "Gamer",
    "carrier": null,
    "id": "4683990183182336"
    "self": "http://localhost:8080/loads/5992450292187136"

}
```

Unsuccessful:
401 Unauthorized code :

```
UnauthorizedError: No authorization token was found<br>    at middleware
```

401 invalid JWT :

```
UnauthorizedError: No authorization token was found<br>    at middleware
```

POST a load

At this URL, the user can remove a boat they own.

POST /loads

Headers : Content-Type: application/json.

Request Body:

```
{
  "weight": "1337",
  "contents": "Gamer",
  "delivery_date": "11/24/1997"
}
```

Response example :

```
Status code is 201
{
    "weight": "1337",
    "delivery_date": "11/24/1997",
    "content": "Gamer",
    "carrier": null,
    "id": "4683990183182336"
     "self": "http://localhost:8080/loads/5992450292187136"

}
```

Response Code :

201 Good.

403 Forbidden.

PUT edit a load

At this URL, the user can remove a boat they own.

PUT /loads / {{load_id}}

Headers : Content-Type: application/json.

Request Body:

```
{
  "weight": "189435",
  "content": "Elite gamerz",
  "delivery_date": "11/24/199999997"
}
```

Response example :

```
Status code is 200
{
    "weight": "189435",
    "delivery_date": "11/24/199999997",
    "content": "Elite gamerz",
    "carrier": null,
    "id": "4683990183182336",
    "self": "http://localhost:8080/loads/5992450292187136"

}
```

Response Code :

200 Good.

403 Forbidden.

Delete a Load

This allows the user to delete an existing load. It accesses this through the loads/:load_id endpoint. It updates the boat that was once carrying the load and deletes the load from that boat.

DELETE /loads/:load_id

Requests

Request Parameters

None

Request Body

None

Response

Response Body format
JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No content	
Failure	404 Not found	The load / boat from the request does not exist This occurs when Neither the boat ID or the load ID can be pinpointed.

Response Examples

Success

Status: 204 No content

Failure

Status: 404 Not Found
 	
{
“Error”: “The load / boat from the request does not exist”
}