https://hw6lestere.uw.r.appspot.com

DATA MODEL

Boat

Property Data	**Type**	**Notes**
id	Integer	The id of the boat.
name	String	Name of the boat.
type	String	Type of the boat.
length	Integer	The length of the boat in feet.
loads	Array	Array of loads on boat.
public	Bool	Always false statement to create desired interactions.
owner	String	The jwt sub that shows who owns the boat.

Loads

Property Data	**Type**	**Notes**
id	Integer	The id of the loads.
weight	Integer	
carrier	Array	
content	String	
radioactive	Bool	Says if the contents are radioactive.

Users

Property Data	**Type**	**Notes**
id	Integer	The id of the user.
email	Sting	Email of user.
sub	Integer	JWT sub

ENDPOINTS

---Accounts---

Are for creating and accessing your account.

GET /

This brings you to the welcome page where you can sign in with a google account.

Request:

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	

Response example:

200 Success

```
<!DOCTYPE html>
<style>
    div {
        margin: auto;
        width: 50%;
        padding: 10px;
        background: white;
        overflow-wrap: anywhere;
    }

    html {
        background: darkblue;
    }
</style>
<html>

<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width" />
    <title>Final CS493 Assignment</title>
</head>

<body>
    <div class="container">
        <section>


            <p>Welcome to the Auth page.<br></br>
            </p>
            <p>
                <a
                    href="https://accounts.google.com/o/oauth2/auth?response_type=code
&amp;client_id=748815447778-
gl3cdfm57s9u4cea04lhtrr1a5r683re.apps.googleusercontent.com&amp;redirect_uri=https%3A%
2F%2Fhw6lestere.uw.r.appspot.com%2Foauth&amp;scope=https%3A%2F%2Fwww.googleapis.com%2F
auth%2Fuserinfo.email+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile+openi
d&amp;state=QGQCgbeCKyO4IlNfYtv3TIPWdxZO02&amp;access_type=offline&amp;prompt=select_a
ccount">Google
                    Login</a>
            </p>


        </section>
    </div>
</body>

</html>
```

GET /oauth

This is the page you see once logged in and tells you your user_id and jwt. If a new account was made it will say “account created” if a new user is created and “welcome“ if a returning user signs in.

---Users---

These endpoints are for accessing user data.

GET /users/

Gets all users.

Request:

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	Will show no boats if not logged in.

Response example:

200 Success

```
[
    {
        "email": "lestere@oregonstate.edu",
        "id": 4765542485327872,
        "self": "https://hw6lestere.uw.r.appspot.com/users/4765542485327872",
        "sub": "101938265794005152347"
    },
    {
        "email": "pdxsites503@gmail.com",
        "id": 5794372540956672,
        "self": "https://hw6lestere.uw.r.appspot.com/users/5794372540956672",
        "sub": "101967599021812109405"
    }
]
```

GET /users/<uid>

{private}

Get a specific user info.

Request:

Path Parameters

Name	Description
boat_id	ID of the boat to be deleted

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Not authorized to see user list.
Failure	404 Not Found	No boat with this user id exists
Failure	406 Not Acceptable	

200 Success

```
[
    {
        "email": "lestere@oregonstate.edu",
        "id": 4765542485327872,
        "self": "https://hw6lestere.uw.r.appspot.com/users/4765542485327872",
        "sub": "101938265794005152347"
    }
]
```

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

404 Failure

```
{
    Error': 'This user does not exist!\n
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

---Boats---

These end pints are just to clear the database for testing or resetting.

GET /boats

{private}

Gets all boats owned by the user.

Request:

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	Will show no boats if not logged in.

Response example:

200 Success

```
{
    "boats": [
        {
            "id": 5168126949851136,
            "length": 99,
            "loads": [],
            "name": "Odyssey",
            "owner": "101938265794005152347",
            "public": "false",
            "self": "https://hw6lestere.uw.r.appspot.com/boats/5168126949851136",
            "type": "Yatch"
        },
        {
            "id": 5643550469390336,
            "length": 99,
            "loads": [],
            "name": "Odyssey",
            "owner": "101938265794005152347",
            "public": "false",
            "self": "https://hw6lestere.uw.r.appspot.com/boats/5643550469390336",
            "type": "Yatch"
        },
        {
            "id": 5731076903272448,
            "length": 1,
            "loads": [],
            "name": "spyboat",
            "owner": "101938265794005152347",
            "public": "false",
            "self": "https://hw6lestere.uw.r.appspot.com/boats/5731076903272448",
            "type": "unknown"
        },
        {
            "id": 5755374237908992,
            "length": 9449,
            "loads": [],
            "name": "Odysdssey",
            "owner": "101938265794005152347",
            "public": "false",
            "self": "https://hw6lestere.uw.r.appspot.com/boats/5755374237908992",
            "type": "Yatasdch"
        },
        {
            "id": 5765243099676672,
            "length": 9449,
            "loads": [],
            "name": "Odysdssey",
            "owner": "101938265794005152347",
            "public": "false",
            "self": "https://hw6lestere.uw.r.appspot.com/boats/5765243099676672",
            "type": "Yatasdch"
        }
    ],
    "next": "https://hw6lestere.uw.r.appspot.com/boats?limit=5&offset=5"
}
```

POST /boats

{private}

Post a boat owned by the account logged in.

Request:

Path Parameters

None

Request Body

Required

JSON

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat.	Yes
length	Length of the boat.	Yes
public	Legacy from when needed, no matter what is entered it is false(private)	Yes

```
{
  "name": "Odysdssey",
  "type": "Yatasdch",
  "length": 9449,
  "public": true
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	201 Created	Sends back created boat data.
Failure	400 Bad Request	
Failure	401 Unauthorized	

Response example:

201 Success

```
{
    "id": 5473899261198336,
    "length": 9449,
    "name": "Odysdssey",
    "owner": "101938265794005152347",
    "public": "false",
    "self": "https://hw6lestere.uw.r.appspot.com/boats/5473899261198336",
    "type": "Yatasdch"
}
```

400 Failure

```
{
    "Error": "not enough attributes"
}
```

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

DELETE /boats/<boat_id>

{private}

Delete a Specific Boat.

Request:

Path Parameters

Name	Description
boat_id	ID of the boat to be deleted

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	Not authorized to see boat.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

404 Failure

```
{
    "Error": "boat does not exist"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

GET /boats/<boat_id>

{private}

Get a Specific Boat.

Request:

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Not authorized to see boat.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	

200 Success

```
{
    "id": 5473899261198336,
    "length": 9449,
    "name": "Odysdssey",
    "owner": "101938265794005152347",
    "public": "false",
    "self": "https://hw6lestere.uw.r.appspot.com/boats/5473899261198336",
    "type": "Yatasdch"
}
```

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

404 Failure

```
{
    "Error": "No boat with this boat_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

PUT /boats/<boat_id>

{private}

Change a Specific Boat.

Request:

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

Required

JSON

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat.	Yes
length	Length of the boat.	Yes

```
{
  "name": "OdysdsseyI",
  "type": "Ytasdch",
  "length": 949,
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	202 Accpeted	
Failure	401 Unauthorized	Not authorized to see boat.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	

202 Accepted

```
{
    "id": 6599799168040960,
    "length": 260,
    "name": "Odyssey III",
    "owner": "101938265794005152347",
    "public": "false",
    "self": "https://hw6lestere.uw.r.appspot.com/boats/6599799168040960",
    "type": "Mega Yatch"
}
```

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

404 Failure

```
{
    "Error": "No boat with this boat_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

PATCH /boats/<boat_id>

{private}

Change a Specific Boat.

Request:

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

Required

JSON

Name	Description	Required?
name	The name of the boat.	No*
type	The type of the boat.	No*
length	Length of the boat.	No*

*At least one is required.

```
{
  "name": "OdysdsseyI",
  "type": "Ytasdch",
  "length": 949,
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	202 Accpeted	
Failure	401 Unauthorized	Not authorized to see boat.
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	

202 Accepted

```
{
    "id": 6599799168040960,
    "length": 260,
    "name": "Odyssey III",
    "owner": "101938265794005152347",
    "public": "false",
    "self": "https://hw6lestere.uw.r.appspot.com/boats/6599799168040960",
    "type": "Mega Yatch"
}
```

401 Failure

```
{
    "Error": "no JWT given"
}
```

Or

```
{
    "Error": "Could not verify JWT\n"
}
```

404 Failure

```
{
    "Error": "No boat with this boat_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

GET /owners/<owner_id>/boats

{private}

Gets the boats by a specific user.

#this is still in my code from previous assignments but I do not believe is required for this assignment, and since it is private it should not be a problem keeping this functionality.

---Loads---

These end points relate to loads and are not private.

This means you can create and put loads on ships without authorization.

GET /loads

Gets all loads in database.

Request:

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	

Response example:

200 Success

```
{
    "loads": [
        {
            "carrier": null,
            "content": "unknown",
            "id": 5056647047151616,
            "radioactive": true,
            "self": "https://hw6lestere.uw.r.appspot.com/loads/5056647047151616",
            "weight": 166
        },
        {
            "carrier": null,
            "content": "Beer",
            "id": 5221338004324352,
            "radioactive": false,
            "self": "https://hw6lestere.uw.r.appspot.com/loads/5221338004324352",
            "weight": 15
        },
        {
            "carrier": null,
            "content": "Crazy String",
            "id": 5331550992334848,
            "radioactive": false,
            "self": "https://hw6lestere.uw.r.appspot.com/loads/5331550992334848",
            "weight": 14
        },
        {
            "carrier": null,
            "content": "Beer",
            "id": 5338122023862272,
            "radioactive": false,
            "self": "https://hw6lestere.uw.r.appspot.com/loads/5338122023862272",
            "weight": 15
        },
        {
            "carrier": null,
            "content": "LEGO Blocks",
            "id": 5762344265187328,
            "radioactive": false,
            "self": "https://hw6lestere.uw.r.appspot.com/loads/5762344265187328",
            "weight": 5
        }
    ],
    "next": "https://hw6lestere.uw.r.appspot.com/loads?limit=5&offset=5"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

POST /loads

Post a load.

Request:

Path Parameters

None

Request Body

Required

JSON

Name	Description	Required?
weight	Weight of load	Yes
content	Consents of load	Yes
radioactive	Is consents radioactive.	Yes

```
{
    "weight": 5,
    "content": "Wood",
    "radioactive": false
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	201 Created	Sends back created boat data.
Failure	400 Bad Request	

Response example:

201 Success

```
{
    "carrier": null,
    "content": "Wood",
    "id": 6182546953994240,
    "radioactive": false,
    "self": "https://hw6lestere.uw.r.appspot.com/loads/6182546953994240",
    "weight": 5
}
```

400 Failure

```
{
    "Error": "The request object is missing at least one of the required attributes"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

DELETE /loads/<load_id>

Delete a Specific load.

Request:

Path Parameters

Name	Description
load_id	ID of the load to be deleted

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No boat with this load_id exists
Failure	406 Not Acceptable	

404 Failure

```
{
    "Error": "No load with this load_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

GET /loads/<load_id>

Get a Specific load.

Request:

Path Parameters

Name	Description
load_id	ID of the load

Request Body

None

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	
Failure	406 Not Acceptable	

404 Failure

```
{
    "Error": "No loads with this load_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

PUT /loads/<boat_id>

Change a Specific Load.

Request:

Path Parameters

Name	Description
load_id	ID of the load

Request Body

Required

JSON

Name	Description	Required?
weight	Weight of load	Yes
content	Loads contents	Yes
radioactive	Is boat radioactive	Yes

```
{
    "weight": 14,
    "content": "Crazy String",
    "radioactive": false
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	202 Accpeted	
Failure	404 Not Found	
Failure	406 Not Acceptable	

202 Accepted

```
{
    "content": "Crazy String",
    "id": 6208935233060864,
    "public": "false",
    "radioactive": false,
    "self": "https://hw6lestere.uw.r.appspot.com/loads/6208935233060864",
    "weight": 14
}
```

404 Failure

```
{
    "Error": "No loads with this load_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

PATCH /loads/<load_id>

Change a Specific Load.

Request:

Path Parameters

Name	Description
load_id	ID of the load

Request Body

Required

JSON

Name	Description	Required?
weight	Weight of load	No*
content	Loads contents	No*
radioactive	Is boat radioactive	No*

*At least one is required.

```
{
    "weight": 14,
    "radioactive": true
}
```

Response

Response Body Format

JSON

Outcome	Status Code	Notes
Success	202 Accpeted	
Failure	404 Not Found	
Failure	406 Not Acceptable	

202 Accepted

```
{
    "content": "Crazy String",
    "id": 6208935233060864,
    "public": "false",
    "radioactive": true,
    "self": "https://hw6lestere.uw.r.appspot.com/loads/6208935233060864",
    "weight": 14
}
```

404 Failure

```
{
    "Error": "No load with this load_id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

PUT /<boat_id>/loads/<load_id>

This endpoint allows you to put load on a boat

Request:

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Outcome	Status Code	Notes
Success	204 No Content	
Failure	403 Forbidden	When boat or load is already in use.
Failure	404 Not Found	No boat or load was found with that id.
Failure	406 Not Acceptable	

403 Failure

```
{
    "Error": "Load already assigned to boat"
}
```

404 Failure

```
{
    "Error": "No boat/load with this id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

DELETE /<bid>/loads/<lid>

This endpoint allows you to delete a load on a boat

Request:

Path Parameters

Name	Description
boat_id	ID of the boat
load_id	ID of the load

Request Body

None

Response

Response Body Format

Success: No body

Failure: JSON

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No boat or load was found with that id.
Failure	406 Not Acceptable	

404 Failure

```
{
    "Error": " This load is not on the boat"
}
```

or

```
{
    "Error": "No boat/load with this id exists"
}
```

406 Failure

```
{
    "Error": "only accept JSON request"
}
```

---Delete---

These end points are just to clear the database for testing or resetting.

DELETE /delete

This command automatically deletes all boats and loads.

DELETE /deleteusers

This command deletes all users.

POSTMAN

There are two tests included since one of them became too complex and would have even been more complicated if added upon. The collection users jwt1, and therefore inheriting from the parent means user1 and bearer token {{jwt2}} means user2.

Test 1. lestere_project

This test creates boats and loads for two users and tests the availability of non-user entity when using one or the other accounts authentication. Goals of this test include:

- showing that boats created by a user will only be shown to that user
- loads assigned to boats cannot be assigned again, and will show up in boats “carrier”
- testing if only jsons and correct methods are accepted and return correct error codes.

Test 2. lestere_project2

This includes tests for creating and manipulating all non-user entities over one account connection. Starting with boats, then loads, and finally showing all users.(loads does not require authorization)

Notes:

I have found that my jwt tokens sometimes update without my control. I am not sure if this is when I log in again or if its just a time limit set. I will give jwt and user_id’s that work when I turn in the project, but please contact me if there are any problems.

I believe you could just make an account and put the jwt into my environment, and use /deleteusers if you did not want your info saved after grading.(though the first test requires 2 jwt tokens)

Provided test were all tests passed to show it does work.

RUBRIC

1.

All non-user entities have a root URL that returns the collection of that entity

The collection for an unprotected entity must show all the entities in the collection.

The collection for a protected entity must shown only the entities corresponding to the JWT.

Get /boat

Shows all boats that are owned by the signed in account and will show empty boat array if not logged in.

Get /users

Shows all users, even if not logged in.

Get /loads

Shows all loads, even if not logged in.

2.

All root non-user entity collections implement paging of 5 items at a time

Get /boat

Get /loads

Both implement 5 item paging

3.

All non-user entities have a self link that points to the most canonical representation of that entity instance

Boats and loads have a self link, as well as users.

4.

All non-user entities support at least 3 properties in addition to any properties modeling relationships, or the id and self property.

Boats – name, length, type

Loads – content, weight, radioactive

5.

Every non-user entity supports create operation

The endpoint must be protected for a protected entity.

Functions for Boats are all protected since it is a protected entity.

None of load’s points are protected so putting loads on boats is not protected.

6.

Every non-user entity supports read operation

The endpoint must be protected for a protected entity.

Functions for Boats are all protected since it is a protected entity.

None of load’s points are protected so putting loads on boats is not protected.

7.

Every non-user entity supports update operations

The endpoint must be protected for a protected entity

Functions for Boats are all protected since it is a protected entity.

None of load’s points are protected so putting loads on boats is not protected.

8.

Every non-user entity supports delete operation

The endpoint must be protected for a protected entity.

Functions for Boats are all protected since it is a protected entity.

None of load’s points are protected so putting loads on boats is not protected.

9.

Endpoints are provided to create and remove relationship between non-user entities

PUT /<bid>/loads/<lid>

DELETE /<bid>/loads/<lid>

Both can create and delete relationships to boats and loads (not private).

Also

DELETE / loads

DELETE / boats

Should both edit any relations it has with the other.

10.

200, 201 and 204 status codes are supported.

All of these status codes are used frequently.

11.

401, 403 and 405 status codes are supported.

I used all of these but 403 I used the least only showing up when you assign a load to a boat and either the said boat already has a load or the load in question already is assigned to a boat.

12.

Requests that do not accept JSON are rejected with status code 406

Besides / and /oauth that return html, all enpoints start with:

```
    if 'application/json' not in request.accept_mimetypes:
        return (jsonify({"Error" : "only accept JSON request"}), 406)
```

to make sure only json requests are accepted.

13.

Users can create new accounts.

Pressing google login on https://hw6lestere.uw.r.appspot.com **should allow you to create a new account through google.**

14.

Users can login, and is shown a JWT and their user id

Once signed in the user is greeted by their user id(under users in entity) and their jwt token for authorization.

15.

An endpoint is provided to show the collection of all users

Get /users

returns all users.