Final Project

CS 493: Cloud Application Development
Spring 2021
Oregon State University

GCP URL (HOME PAGE AND APP URL)

- https://cs493-final-project-315721.uc.r.appspot.com

LOGIN

- https://cs493-final-project-315721.uc.r.appspot.com/login

PRE-CREATED USER INFO

- User 1
  - Email: newacc@test.com
  - PW: Fakepass1234
- User 2
  - Email: rachid@test.com
  - PW: Fakepass1234

INSTRUCTIONS

Users must be created manually if they do not exist as an entity already within datastore. In other words, if you are testing with accounts that you are creating instead of the ones shown above, then their entities will not exist in datastore, and they must be created.

If this is not done, ALL tests for protected endpoints will FAIL.

So, please, once you create each user and login, copy their JWT and Unique User ID into a jwt/user_id pair. From there, the test collection should create the entities if they do not exist already. If that doesn't work, then manually send a POST request to '/users' with the bearer type jwt in the authorization header.

Once you've created and logged in to one user account, go back to the home page to create and login to the second account

Data Model

- User
  - Example of a user object:
    - {
      id: <unique_id>,
      sub: <unique_sub>,
      boats: [<list of boats>]
      }
  - Properties
    - ID (string) – Not required on creation, it is randomly generated string of numbers by Datastore.
    - Sub (string) – Required, but is provided in your valid JWT. Sub must be unique.
    - Boats (list of boat objects) – Not required, empty on creation of user entity
  - The unique identifier of the User is the sub property. This is provided on creation of the user through the valid JWT, which is received when a user logs in on the webpage.

  - Requests to protected resources require that the user provide their valid JWT in the authorization header (bearer type). From there, the sub property in the JWT compared against the sub property in the user entity to make sure they are allowed access to the resource.

  - The JWT is mapped to the user entity in Datastore using the unique sub property that Auth0 provides in the JWT. Every time a request is made to a protected resource, the sub in the JWT is checked against the sub saved in datastore.

  - A user is related to boats. A user can own zero, one, or many boats.
    - On creation of a user entity, the user doesn't own any boats
    - The user can then create boats, which will automatically update the relationship between the user and the boat.
- Boats
  - Example of a boat object:
    - {
      id: <unique id>,
      name: "Boat 1",
      type: "Boat Type 1",
      length: 1,
      loads: [< list of load objects >],
      owner: { < user object > }
      }

    - Properties
        - ID (string) - Not required on creation, it is randomly generated string of numbers by Datastore.
        - Name (string) – Required, name of the boat
        - Type (string) – Required, type of the boat
        - Length (number) – Required, length of the boat
        - Loads (list of load objects) – Not Required, empty on creation
        - Owner (user object) – Required, but is extracted based on valid JWT
    - Boats are a protected resource.
        - All CRUD operations require a valid JWT and a user entity for that JWT to exist
    - A boat is related to a user.
        - A boat must be owned by 1 user, no more, no less.
        - On creation of a boat, the owner is assigned based on the JWT provided in the authorization header.
    - A boat is related to loads
        - A boat can have zero, one, or many loads
        - Once a boat is created, loads can be added to a boat
- Loads
    - Example of a load object:
        - {
            id: < unique id >,
            name: “Load 1”,
            volume: 1,
            destination: “Destination 1”,
            carrier: { < boat object > }
          }
    - Properties
        - ID (string) - Not required on creation, it is randomly generated string of numbers by Datastore.
        - Name (string) – Required, name of the load
        - Volume (number) – Required, volume of the load
        - Destination (string) – Required, destination of the load
        - Carrier (boat object) – Not required, the boat that the load is stored on
    - A load is related to boats
        - A load can be on either zero or one boat
        - Once a load is created, it can be assigned to a singular boat

Create a Boat

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

Allows you to create a new boat.

POST /boats

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat.	Yes
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat in feet.	Yes

Request Body Example

```
{
   "name": "Sea Witch",
 "type": "Catamaran",
   "length": 28
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes or they have a value of null, the boat is not be created, and a 400 status code is returned.
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	JWT may be valid, but it either doesn't have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If

		the request doesn't accept it, then this error is thrown
Failure	415 Unsupported media type	Server only accepts application/json data. Other media types are rejected

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created.
- The returned object will only contain the ID and the self link
- The relationship between user and boat will be added automatically

Success

Status: 201 Created

```
{
   "id": "abc123",
   "self": "https://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

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

Status: 401 Unauthorized

No response body

Status: 403 Forbidden

```
{
"Error":  "You must first create a user entity with this JWT."
}
```

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Status: 415 Unsupported media type

```
{
"Error":  "Server only accepts application/json data."
}
```

Get a Boat

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

Allows you to get an existing boat

GET /boats/:boat_id

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	A boat can only be accessed by its owner
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	JWT may be valid, but it either doesn't have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

- Loads is created when creating the boat. It defaults to an empty array. Loads represents the loads that are stored on the boat.
- The user object stored in owner contains the unique sub identifier and a self link for the user
- A boat can only be accessed by its owner

Success

Status: 200 OK


```
{
    "id": "abc123",
    "name": "Sea Witch",
  "type": "Catamaran",
    "length": 28,
  “loads”: [],
  “owner”: {< user object >}
    "self": "https://<your-app>/boats/abc123"
}
```


Failure

Status: 400 Bad Request


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


Status: 401 Unauthorized

No response body

Status: 403 Forbidden


```
{
"Error":  “You must first create a user entity with this JWT."
}
```


Status: 404 Not Found


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


Status: 406 Not Acceptable


```
{
"Error":  "Not acceptable."
}
```


Status: 415 Unsupported media type


```
{
"Error":  "Server only accepts application/json data."
}
```

List all Boats

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

List all the boats.

GET /boats

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Only returns boats owned by the user identified by the JWT
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	JWT may be valid, but it either doesn't have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

Success

- Only a maximum five boats will be listed at a time. If there are more than five boats in the datastore, then a "next" attribute will exist with a link to view the next five boats
- Only boats associated with the JWT will be shown
- Total represents the total number of boats owned by the user

Status: 200 OK


```
{
  “total”: 2
  “items”: [{
     "id": "abc123",
     "name": "Sea Witch",
     "type": "Catamaran",
     "length": 28,
     “loads”, [],
     “owner”: { < user object > },
     "self": "https://<your-app>/boats/abc123"
    },
    {
     "id": "def456",
     "name": "Adventure",
     "type": "Sailboat",
     "length": 50,
     “loads”, [],
     “owner”: { < user object > },
     "self": "https://<your-app>/boats/def456"
    }
  ],

}
```


Status: 401 Unauthorized

No response body

Status: 403 Forbidden


```
{
"Error":  “You must first create a user entity with this JWT."
}
```


Status: 406 Not Acceptable


```
{
"Error":  "Not acceptable."
}
```


Status: 415 Unsupported media type


```
{
"Error":  "Server only accepts application/json data."
}
```

Edit a Boat (PATCH)

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

Allows you to edit a subset of a boat

PATCH /boats/:boat_id

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat.	No
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	No
length	Integer	Length of the boat in feet.	No

Any other attributes are ignored.

Request Body Example

```
{
   "name": "Sea Witch",
    "type": "Catamaran"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	A boat can only be updated by its owner
Failure	400 Bad Request	If the value of any attribute is null, the result will be a 400 error.
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	JWT may be valid, but it either doesn't have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	404 Not Found	No boat with this boat_id exists

Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown
Failure	415 Unsupported Media Type	Request body must be 'application/json'. Other media types are not supported.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this boat.

Success

Status: 201 Created

```
{
   "id": "abc123",
   "self": "https://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "At least one attribute with invalid value of null"
}
```

Status: 401 Unauthorized

No response body

Status: 403 Forbidden

```
{
"Error":  "You must first create a user entity with this JWT."
}
```

Status: 404 Not Found

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

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Status: 415 Unsupported Media Type

```
{
"Error":  "Server only accepts application/json data"
}
```

Edit a Boat (PUT)

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

Allows you to edit a boat.

PUT /boats/:boat_id

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat.	Yes
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat in feet.	Yes

Any other attributes are ignored.

Request Body Example

```
{
   "name": "Sea Witch",
  "type": "Catamaran",
  “length”: 77
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	A boat can only be edited by its owner.
Failure	400 Bad Request	All three attributes are required. Any missing attribute will result in a 400 error. Additionally, if the value of any attribute is null, the result will be a 400 error
Failure	403 Forbidden	JWT may be valid, but it either doesn’t have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	404 Not Found	No boat with this boat_id exists

Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown
Failure	415 Unsupported Media Type	Request body must be 'application/json'. Other media types are not supported.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this boat.

Success

Status: 201 Created

```
{
   "id": "abc123",
   "self": "https://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "At least one attribute with invalid value of null"
}
```

Status: 401 Unauthorized

No response body

Status: 403 Forbidden

```
{
"Error":  "You must first create a user entity with this JWT."
}
```

Status: 404 Not Found

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

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Status: 415 Unsupported Media Type

```
{
"Error":  "Server only accepts application/json data"
}
```

Delete a Boat

PROTECTED – REQUIRES VALID JWT AND A USER ENTITY FOR THAT JWT

Allows you to delete a boat. All relationships are updated on deletion.

```
DELETE /boats/:boat_id
```

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that corresponds to a user entity.	Yes

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	A boat can only be deleted by its owner.
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	JWT may be valid, but it either doesn't have a user entity created yet or the JWT is trying to perform a CRUD operation on a resource it does not have access to
Failure	404 Not Found	No boat with this boat_id exists

Response Examples

- A boat can only be deleted by its owner
-

Success

```
Status: 204 No Content

No response body
```

Failure

Status: 401 Unauthorized No response body

Status: 403 Forbidden
{
"Error": "The request object is missing at least one of the required attributes"
}

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

Create a Load

NOT PROTECTED

Allows you to create a new load. All newly created loads are unassigned to any boat.

POST /loads

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	Name of the load	Yes
volume	Integer	The volume of the load	Yes
destination	String	Destination of the load	Yes

Request Body Example

```
{
  "volume": 1,
  “name”: “Load 1”,
   “destination”: “Destination 1”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes or they have a value of null, the load is not be created, and a 400 status code is returned.
Failure	406 Not Acceptable	Server response can only send ‘application/json’ data. If the request doesn’t accept it, then this error is thrown
Failure	415 Unsupported media type	Server only accepts application/json data. Other media types are rejected

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created
- The value of the attribute carrier is an object with the id of the boat where the load is currently stored. If the load is not stored on a boat, the value of carrier will be an empty object { }. On creation of a load, carrier will always be an empty object.

- The value of the attribute self is a live link to the REST resource corresponding to this load.

Success

Status: 201 Created


```
{
   "id": "123abc",
   "self": "https://<your-app>/loads/123abc"
}
```


Failure

Status: 400 Bad Request


```
{
"Error":  "At least one attribute with invalid value of null"
}
```


Status: 406 Not Acceptable


```
{
"Error":  "Not acceptable."
}
```


Status: 415 Unsupported Media Type


```
{
"Error":  "Server only accepts application/json data"
}
```

Get a Load

NOT PROTECTED

Allows you to get an existing load.

GET /load/:load_id

Request

Path Parameters

Name	Type	Description
load_id	String	ID of the load

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

- The carrier attribute contains the ID and self live link of the boat that the load is stored on. A load can only be stored on one boat at a time. A load does not need to be stored on a carrier, and so that object can be empty.

Success

Status: 200 OK

```
{
   "id": "123abc",
   "volume": 1,
    "name": "Load 1",
     "destination": "Destination 1",
    "carrier: {
       "id": xyz123,
        "self": "https://<your-app>/boats/xyz123"
 },
   "self": "https://<your-app>/loads/123abc"
}
```

Failure

Status: 404 Not Found
 	
{
"Error": "No load with this load_id exists"
}

Status: 406 Not Acceptable
 	
{
"Error": "Not acceptable."
}

List all Loads

NOT PROTECTED

List all the loads.

GET /loads

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Returns all loads in an object under the “items” attribute.

Response Examples

- Only a maximum five loads will be listed at a time. If there are more than five loads in the datastore, then a “next” attribute will exist with a link to view the next five loads
- Total number of loads is stored in “total”

Success

Status: 200 OK

```
{
   “total”: 2,
  “items”: [{
     "id": "123abc",
     "volume": 1,
     “carrier”: {},
     "self": "https://<your-app>/loads/123abc"
   },
   {
     "id": "456def",
     “volume”: 2,
     “carrier”: {
        “id”: “xyz123”,
         “self”: “https://<your-app>/boats/xyz123”
     },
     “self”: “https://<your-app>/loads/456def”

   }]
}
```

Delete a Load

NOT PROTECTED

Allows you to delete a load. All relationships are updated on deletion.

```
DELETE /loads/:load_id
```

Request

Path Parameters

Name	Type	Description
load_id	String	ID of the load

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

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

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found


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

Edit a Load (PATCH)

NOT PROTECTED

Allows you to edit a subset of a load

PATCH /loads/:load_id

Request

Path Parameters

Name	Type	Description
load_id	String	ID of the load

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	Name of the load	No
volume	Integer	The volume of the load	No
destination	String	Destination of the load	No

Any other attributes are ignored.

Request Body Example

```
{
   "name": "Load 2",
    "volume": 2
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the value of any attribute is null, the result will be a 400 error.
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown
Failure	415 Unsupported Media Type	Request body must be 'application/json'. Other media types are not supported.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this boat.

Success

Status: 201 Created

```
{
   "id": "123abc",
   "self": "https://<your-app>/loads/123abc"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "At least one attribute with invalid value of null"
}
```

Status: 404 Not Found

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

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Status: 415 Unsupported Media Type

```
{
"Error":  "Server only accepts application/json data"
}
```

Edit a Load (PUT)

NOT PROTECTED

Allows you to edit a subset of a load

PATCH /loads/:load_id

Request

Path Parameters

Name	Type	Description
load_id	String	ID of the load

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	Name of the load	Yes
volume	Integer	The volume of the load	Yes
destination	String	Destination of the load	Yes

Any other attributes are ignored.

Request Body Example

```
{
   "name": "Load 2",
    "volume": 2,
    “destination”: “Destination 2”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	
Failure	400 Bad Request	If the value of any attribute is undefined or null, the result will be a 400 error.
Failure	404 Not Found	No load with this load_id exists
Failure	406 Not Acceptable	Server response can only send ‘application/json’ data. If the request doesn’t accept it, then this error is thrown
Failure	415 Unsupported Media Type	Request body must be ‘application/json’. Other media types are not supported.

Response Examples

- The self attribute will contain the live link to the REST resource corresponding to this boat.

Success

Status: 201 Created

```
{
   "id": "123abc",
   "self": "https://<your-app>/loads/123abc"
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "At least one attribute with invalid value of null"
}
```

Status: 404 Not Found

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

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Status: 415 Unsupported Media Type

```
{
"Error":  "Server only accepts application/json data"
}
```

Add Load to Boat

NOT PROTECTED

Store a load on a boat

PUT /boats/:boat_id/loads/:load_id

Request

Path Parameters

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

Request Body

None

Note: Set Content-Length to 0 in your request when calling out to this endpoint.

Response

No body

Response Body Format

Success: JSON

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and the load is not already stored on a boat.
Failure	403 Forbidden	The load is already stored on a boat
Failure	404 Not Found	No boat with this boat_id exists, and/or no load with this load_id exits.

Response Examples

- The boat that was modified is sent in the response body with the newly added load in the “loads” array.

Success

Status: 200 OK

```
{
  "name": "Death Star 2",
  "type": "Unfinished Rowboat",
  "loads": [{
      "id": “123def”,
      "self": "http://<your-app>/loads/123def”
  }],
  "length": 750,
  “owner”: {< user object >},
  "id": “123abc”,
  "self": "http://<your-app>/loads/123abc”
}
```

Failure

Status: 403 Forbidden

```
{
"Error":  "This load is already assigned to a boat”
}
```

Status: 404 Not Found

```
{
"Error":  “No such boat or load exists”
}
```

•

Remove Load from a Boat

NOT PROTECTED

Remove a load from a boat without deleting the load or the boat

```
DELETE /boats/:boat_id/loads/:load_id
```

Request

Path Parameters

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

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and the load is on the boat.
Failure	404 Not Found	No boat with this boat_id or load with this load_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

{
"Error": "No such boat or load exists"
}
```

Create a User

PROTECTED - REQUIRES VALID JWT TO CREATE A USER ENTITY

Create a user entity

POST /users

Request

Request Header Attributes

Name	Type	Description	Required?
Authorization	Bearer	Valid JWT (Bearer token type) that will correspond to a user entity.	Yes

Path Parameters

None

Request Body

None

Response

Response Body Format

Success: JSON

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	Creates a user entity with the JWT sub property as the unique identifier.
Failure	401 Unauthorized	Invalid or missing JWT provided in Authorization header
Failure	403 Forbidden	If the JWT is valid, and the user already exists as an entity, 403 is returned
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

- Sub is the unique identifier for the user, which is taken from the valid JWT
- Only one user per JWT
- Boats is empty at first, but is populated when the user creates boats.

Success

Status: 201 Created

```
{
   "id": "abc123",
    "sub":  < sub from valid JWT >,
   "boats": [ < List of Boat Objects > ]
}
```

Failure

Status: 401 Unauthorized No response body

Status: 403 Forbidden
 	
{
"Error": "User already exists"
}

Status: 406 Not Acceptable
 	
{
"Error": "Not acceptable."
}

Get a User

NOT PROTECTED

Get a user entity

GET /users/:sub

Request

Path Parameters

Name	Type	Description
Sub	String	Unique sub identifier of the user (taken from JWT on user creation)

Request Body

None

Response

Response Body Format

Success: JSON

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No user with this unique sub exists
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

- Sub is the unique identifier for the user, which is taken from the valid JWT
- Only one user per JWT

Success

Status: 200 OK

```
{
  "id": "abc123",
  "sub": < sub from valid JWT >,
  "boats": [ < List of Boat Objects > ],
  "self": https://<app_url>/users/<unique_sub>
}
```

Failure

Status: 401 Unauthorized

No response body

Status: 404 Not Found

```
{
"Error": "No user with this unique sub exists."
}
```

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Get all Users

NOT PROTECTED

Get all user entities

GET /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

Success: JSON

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	406 Not Acceptable	Server response can only send 'application/json' data. If the request doesn't accept it, then this error is thrown

Response Examples

- Sub is the unique identifier for the user, which is taken from the valid JWT
- Only one user per JWT

Success

Status: 200 OK

```
{
 [
      {
        "id": "abc123",
        "sub":  < sub from valid JWT >,
        "boats": [ < List of Boat Objects > ]
      },
      {
         "id": "xyz123",
          "sub":  < sub from valid JWT >,
         "boats": [ < List of Boat Objects > ]
      }
 ]
}
```

Failure

Status: 406 Not Acceptable

```
{
"Error":  "Not acceptable."
}
```

Edit Users (PATCH)

NOT PROTECTED

Edit users – NOT SUPPORTED

PATCH /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

Success: None

Failure: None

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	User entities cannot be edited, PATCH not supported Only GET and POST supported for '/users'

Response Examples

Failure

Status: 405 Method Not Allowed

No Response Body

Edit Users (PUT)

NOT PROTECTED

Edit users – NOT SUPPORTED

PUT /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

Success: None

Failure: None

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	User entities cannot be edited, PUT not supported Only GET and POST supported for '/users'

Response Examples

Failure

Status: 405 Method Not Allowed No Response Body

Delete a User

NOT PROTECTED

Delete a user – NOT SUPPORTED

```
DELETE /users/:sub
```

Request

Path Parameters

Path Parameters

Name	Type	Description
Sub	String	Unique sub identifier of the user (taken from JWT on user creation)

Request Body

None

Response

Response Body Format

Success: None

Failure: None

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	User entities cannot be deleted, DELETE not supported Only GET and POST supported for '/users'

Response Examples

Failure

Status: 405 Method Not Allowed

No Response Body