Boats Final Project API Doc

Deployment URL: https://boats-api-mariast.wl.r.appspot.com/

Login/Logout information in “working with users” sub-section

Data Model

The app stores two kinds of entities in Datastore: Boats and Loads. Users are managed by Auth0.

Boats

Property	**Required?**	**Data Type**	**Notes**
id	yes	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
name	yes	String	Name of the boat. NOTE: must be unique.
type	yes	String	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	yes	Integer	The length of the boat in feet.
public	no	bool	Whether the boat can be seen by all users or not. (if left blank, default: true)
owner	yes	string	This is the id for the user, found in the “sub” in the Auth0 jwt.
loads	no	Integer Array	The loads present in the boat. This is not initially generated upon the creation of a new boat, but upon the first time a new load is added to the boat. This array contains a set of ids of the loads present.

self	no	string	The url pointing to the specific entity. This will be automatically generated.

Loads

Property	**Required?**	**Data Type**	**Notes**
id	yes	Integer	The id of the load. Datastore automatically generates it. Don't add it yourself as a property of the entity.
volume	yes	Integer	The volume the load takes up.
weight	yes	Integer	The weight of the load.
cost	yes	float	The cost of the load, format in dollar amount “0.00”
public	no	bool	Whether the boat can be seen by all users or not. (if left blank, default: true)
self	no	string	The url pointing to the specific entity. This will be automatically generated.
owner	yes	string	This is the id for the user, found in the “sub” in the Auth0 jwt.

Table Relationships

Boats have a zero-to-many relationship with loads; a boat can hold many loads.

Loads have a zero-to-one relationship with boats; a load can only be in a single boat at a time.

Both boats and loads have a one-and-only-one relationship with users; a boat and a load can only have one user.

Users have a zero-to-many relationship with boats and loads; they can own no or many of them.

A user can add a load that they do not own to a boat so long as they own the boat, this is for simplicity's sake.

Working with Users

All users are created and authenticated through Auth0.

Endpoints

Logging in / sign up:

https://boats-api-mariast.wl.r.appspot.com/login

Logging out:

https://boats-api-mariast.wl.r.appspot.com/logout

Getting all users:

https://boats-api-mariast.wl.r.appspot.com/users

When it comes to getting all the users, I'm not entirely sure if I got it right, as the request takes forever. The API request has a particular configuration of values which I set under user_options in server.js. These are correct based on Auth0's documentation. I think my request using Axios might be faulty. For fixing it, I would continue testing by changing the configuration settings and recreating the request, then I would also tweak the Axios GET request to see if I could get feedback from the Auth0 server.

JWT's

This application maps a JWT to the user through Auth0's express openid connect. Once a user is authenticated, they will receive a JWT.

Pre-created users

You can create a new user, but I have two pre-created test users as well

User 1:

Email: “bob@bob.com”, Password: “1234Bob1234”

User 2:

Email: “jim@jim.com”, Password: “1234Jim1234”

Create a Boat

Allows you to create a new boat.

POST /boats

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

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

public	Whether the boat can be seen by all users or not. (if left blank, default: true)	no

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, the boat must not be created, and 400 status code must be returned. You don't need to validate the values of the attributes and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute that is not listed above).
Failure	403	Boat does not have a unique name
Failure	401	User is not authenticated

Failure	415	Content type is not json

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. Your app must send back this value in the response body as shown in the example.

Success

Status: 201 Created


```
{
"id": 123,
"name": "Sea Witch",
"type": "Catamaran",
"length": 28,
“Public”: true,
“Self”: “some_url”
“Owner”: “some_user_sub”
}
```


Failure

Status: 400 Bad Request


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


Status: 403


```
{
"Error": "Request is missing one or more attributes"
}
```


Status: 401


```
{
"Error": "Not Authenticated"
}
```

Get a Boat

Allows you to get an existing boat

GET /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No boat with this boat_id exists
Failure	401	User is not authenticated
Failure	403	User does not own boat

Failure	406	Client does not accept json
Failure	500	Client does not accept anything

Response Examples

Success

Status: 200 OK


```
{
"id": 123,
"name": "Sea Witch",
"type": "Catamaran",
"length": 28,
"Public": true,
"Self": "some_url"
"Owner": "some_user_sub"
}
```


Failure

Status: 404 Not Found


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

List all Boats

List all the boats.

GET /boats

Request

User can make the request signed in or otherwise. Being signed in will allow them to access all the boats that are protected under their account. If not signed in, client can only view the unprotected boats.

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Regardless of whether signed in or not, the status code will be the same

Response Examples

Success

Status: 200 OK


```
[
{
"id": 123,
"name": "Sea Witch",
"type": "Catamaran",
"length": 28,
“Public”: true,
“Self”: “some_url”
“Owner”: “some_user_sub”
},
{
"id": 456,
"name": "Adventure",
"type": "Sailboat",
"length": 50,
“Public”: true,
“Self”: “some_url”
“Owner”: “some_user_sub”
},
{
"id": 789,
"name": "Hocus Pocus",
```

```
"type": "Sailboat",
"length": 100,
“Public”: true,
“Self”: “some_url”
“Owner”: “some_user_sub”
}
]
```


Edit a Boat

Allows you to edit a boat.

PATCH/PUT /boats/:id

Request

Path Parameters

Name	Description
id	ID of the boat

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes (no if patch)
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes (no if patch)
length	Length of the boat in feet.	Yes (no if patch)
public	Can be viewed by non users	No

Request Body Example

```
{
"name": "Sea Witch",
"type": "Catamaran",
"length": 99,
“Public”: false
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the boat must not be updated, and 400 status code must be returned. You don't need to validate the values of the attributes and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than the ones that are listed).
Failure	404 Not Found	No boat with this boat_id exists
Failure	403	User does not own entity
Failure	401	User not authenticated

Response Examples

Success

Status: 200 OK


```
{
"id": 123,
"name": "Sea Witch",
"type": "Catamaran",
"length": 99
```

}

Failure

Status: 400 Bad Request


```
{
"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"
}
```

Delete a Boat

Allows you to delete a boat. If a boat has a load and is deleted, the load will not be deleted.

DELETE /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	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	
Failure	404 Not Found	No boat with this boat_id exists
Failure	401	Not authenticated

Failure	403	User does not own entity

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found


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


Create a Load

Allows you to create a new load.

POST /loads

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The size of the load	Yes
weight	The weight of the load	Yes
cost	The price of the load	yes
public	Whether the load can be seen by non-owners	no

Request Body Example

```
{
"volume": 5,
“Weight”: 5,
“Cost”: 1.99,
“Public”: true
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	201 Created	

Failure	400 Bad Request	If the request is missing the number attribute, the load must not be created, and 400 status code must be returned. You don't need to validate the values of this attribute and can assume that if the number attribute is specified, then its value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than number).
Failure	415	Content type is not json
Failure	401	User not authenticated

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. This value needs to be sent in the response body as shown in the example.

Success

Status: 201 Created

```
{
"id": 123,
“volume”: 5
“Weight”: 5,
“Cost”: 1.99,
“Public”: false,
“Self”: “some_url”
“Owner”: “some_user_sub”
}
```

Failure

Status: 400 Bad Request

```
{
"Error": "The request object is missing the required parameter"
}
```

Get a Load

Allows you to get an existing load.

GET /loads/:load_id

Request

Path Parameters

Name	Description
load_id	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	401	User is not authenticated
Failure	403	User does not own load

Failure	406	Client does not accept json
Failure	500	Client does not accept anything

Response Examples

Success

Status: 200 OK


```
{
"id": 123,
"volume": 5,
"Weight": 5,
"Cost": 1.99,
"Public": false,
"Self": "some_url"
"Owner": "some_user_sub"
}
```


Failure

Status: 404 Not Found


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

List all Loads

List all the loads.

GET /loads

Request

Depending on whether the user is signed in or not will change which loads the server will send back. If the user is signed in, they will get all the loads that are unprotected and the ones protected under their account. If not signed in, they will only get the unprotected loads.

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 OK


```
[
{
"id": 123,
"volume": 5,
“Weight”: 5,
“Cost”: 1.99,
“Public”: false,
“Self”: “some_url”
“Owner”: “some_user_sub”
},
{
"id": 456,
"volume": 4,
“Weight”: 5,
“Cost”: 1.99,
“Public”: false,
“Self”: “some_url”
“Owner”: “some_user_sub”
}
]
```

Delete a Load

DELETE /loads/:load_id

Request

Path Parameters

Name	Description
load_id	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
Failure	403	User does not own load

Failure	401	User not authenticated

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found


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


Edit a Load

Allows you to edit a boat.

PATCH/PUT /loads/:id

Request

Path Parameters

Name	Description
id	ID of the boat

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	The size of the load	Yes
weight	The weight of the load	Yes
cost	The price of the load	yes
public	Whether the load can be seen by non-owners	no

Request Body Example

```
{
"volume": 5,
“Weight”: 5,
“Cost”: 1.99,
“Public”: true
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes

Success	204	
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the load must not be updated, and 400 status code must be returned. You don't need to validate the values of the attributes and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than the ones that are listed).
Failure	404 Not Found	No boat with this id exists
Failure	403	User does not own entity
Failure	401	User not authenticated

Response Examples

Success

Status: 204

Failure

Status: 400 Bad Request

{

"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"

}

Add a Load to a Boat

Adding a load to a boat.

PUT /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load
boat_id	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	200	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and the load is not already assigned.
Failure	403 Forbidden	Load cannot be assigned because it is already in another boat or user does not own boat.
Failure	404 Not Found	The specified boat and/or load does not exist
Failure	401	Not authenticated

Response Examples

Success

Status: 200 OK

```
{
"id": 123,
"name": "Sea Witch",
"type": "Catamaran",
"length": 28,
"Public": true,
"Self": "some_url"
"Owner": "some_user_sub"
"Loads": [ {456} ]
}
```

Failure

Status: 403 Forbidden

```
{
"Error": "The load has already been assigned."
}
```

Status: 404 Not Found

```
{
"Error": "The specified boat and/or load does not exist"
}
```

Unloading a load from a boat

DELETE /boats/:boat_id/loads/:load_id

Request

Path Parameters

Name	Description
load_id	ID of the load
boat_id	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	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and the load is found in the boat.
Failure	404 Not Found	Boat does not exist or load does not exist or boat does not contain the load

Response Examples

Success

Status: 204

Failure

Status: 404 Not Found

```
{
"Error": "Load not present in boat."
}
```

View all loads for a given boat

List out all the loads associated with the boat.

GET /boats/:boat_id/loads

Request

Path Parameters

Name	Description
boat_id	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	200	Returns all the loads associated with the boat
Failure	404 Not Found	The boat id is not found

Response Examples

Success

Status: 200

loads:[

{

456,

789

}]

Failure

Status: 404 Not Found

{

"Error": " No boat with this boat_id."

}