Portfolio Project: API Spec

CS 493: Cloud Application Development

Nicholas Ledbetter

ledbetni@oregonstate.edu

https://cs493portfolioproject.uc.r.appspot.com

Oregon State University Spring 2023

Last Update: June 12 11:15 am Pacific

Change log

Version	Change	Date
1.0	Initial version.	June 11, 2023

Important Information – Please Read

- Go to the URL above (https://cs493portfolioproject.uc.r.appspot.com) to create new users or log in. After registration/login users will be redirected to the '/profile' route which will display a JWT for the user and the "sub" property which will be added to Datastore database as the Users uniqueID. Please copy and paste JWT into Postman environment.

- **Navigate to '/logout' to log out and create a different new user. This instruction is also displayed on the webpage!**

- **Postman tests are provided for getting or refreshing JWT but USERS WILL ONLY BE ADDED TO DATABASE WHEN REGISTERED THROUGH THE WEBSITE! Postman tests may be used to refresh JWT after initial registration.**

- Relationships – Boats are owned by a User. Boats can have a Load assigned to them, and when this relationship occurs the Load will also have a Boat carrier assigned to them.

- Requests to protected Boat endpoints must have a valid JWT Authorization Bearer Token header! Invalid JWT tokens will be rejected, valid JWT tokens will be accepted and then protected routes will verify the "sub" property of a User against the "owner" property of any protected Boat resource

- When conducting Postman tests please be aware of the different request names. There are 2 POST requests for boats – one for USER1 and another for USER2 these will set the "boat_id_user1" and "boat_id_user2" environment variables. Tests that show functionality when USER1 requests a resource owned by USER2 or if USER2 requests a resource owned by USER1 are dependent on this variable.

Data Model

The app stores three kinds of entities in Datastore, Boats and Loads, and Users.

*****New Users may only be created in database by logging into/registering at website URL*****

All routes associated with Boat are protected and must have a valid JWT Authorization Bearer Token

Boat

Property	Data Type	Notes
id	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
name	String required	Name of the boat.
type	String required	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer required	The length of the boat in feet.
loads	Load	This is an array of the Load objects that a boat is carrying. Displays the id and self link of the Load
Owner	User required	The uniqueID of a User who owns the Boat

Load

Property	Data Type	Notes
id	Integer	The id of the load. Datastore automatically generates it. Don't add it yourself as a property of the entity.
volume	Integer required	The quantity of items
carrier	Boat	If there is a boat carrying the load, it will list the Boat objects id, and self link
item	String required	The name of the item
creation_date	String required	The date that this load was created

User

Property	Data Type	Notes
id	Integer	The id of the load. Datastore automatically generates it. Don't add it yourself as a property of the entity.
uniqueID	String	“sub” property of JWT

Create a Boat - Protected

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

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	401 Unauthorized	Request is missing a JWT Authorization Bearer Token

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,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 400 Bad Request

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

Status: 401 Unauthorized

UnauthorizedError: No authorization token was found

View a Boat - Protected

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	403 Forbidden	JWT does not match boat owner

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 404 Not Found

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

Status: 403 Forbidden

```
{
“Error”: “Incorrect owner – not authorized to view this resource”
}
```

View all Boats with Pagination - Protected

List all the boats.

GET /boats

Request

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

boats:[

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
},
{
  "id": 456,
  "name": "Adventure",
  "type": "Sailboat",
  "length": 50,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/456

},
{
  "id": 789,
  "name": "Hocus Pocus",
  "type": "Sailboat",
  "length": 100,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
```

```
  “self”: http://localhost:8080/boats/789

}
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
},
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
}
], “next”: “http://localhost:8080/boats/?cursor=123456987”
```

Add a Load to a Boat and Carrier to Load

Allows you to edit the load of a boat.

PUT /boats/:boat_id/loads/:load_id

Request

Path Parameters

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

Request Body

None

Request Body Format

JSON

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 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

Response Examples

Success

Status: 204 OK

Failure

Status: 404 Bad Request

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

Edit a Boat – individual attributes - Protected

Allows you to create a new boat.

PATCH /boats/id

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

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

Request Body Example

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

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 Created	Can edit individual attributes without altering others
Failure	404 Invalid ID	If the ID of the boat does not exist, return status code 404 Boat name must be unique. Request must be JSON and Response must be JSON
Failure	403 Unauthorized	JWT does not match

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: 200 Created

```
{
  "id": 123,
  "name": "Sea Witch Edited",
  "type": "Catamaran Floating Device",
  "length": 28,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 404 Bad Request

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

}
```

Status: 403 Unauthorized

```
{
“Error”: “Incorrect owner – not authorized to view this resource”
}
```

Edit a Boat – all attributes - Protected

Allows you to create a new boat.

PUT /boats/id

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

Request Body Example

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

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	303 Created	Location header set to the new URL
Failure	400 Bad Request	If the request is missing any of the 3 required attributes, the boat must not be edited, and 400 status code must be returned. Boat name must be unique. Request must be JSON and Response must be JSON
Failure	403 Unauthorized	JWT does not match

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: 303 Created

```
{
  "id": 1234,
  "name": "Sea Witch2",
  "type": "Catamaran2",
  "length": 282,
  “loads”: [],
  “owner”: “auth0|647af34cf0fdsc”,
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 400 Bad Request

```
{
"Error":  "Must specify all attributes. Use PATCH to update individual attributes"
}

}
```

Status: 403 Unauthorized

```
{
“Error” : “Incorrect owner – not authorized to view this resource”
}
```

Edit a Load – individual attributes

Allows you to create a new boat.

PATCH /loads/id

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	Amount of item	No
item	Name of item	No
creation_date	Date order was created	No

Request Body Example

```
{
  "volume": 5
  "item": "Legos",
  "creation_date": “10/10/21”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 Created	Can edit individual attributes without altering others
Failure	404 Invalid ID	If the ID of the boat does not exist, return status code 404 Boat name must be unique. Request must be JSON and Response must be 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: 200 Created

```
{
  "id": 123,
  "volume": 5
  "item": "Legos",
  "creation_date": “10/10/21”,
  “carrier”: [],
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 404 Bad Request

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

}
```

Edit a Load – all attributes

Allows you to create a new boat.

PUT/loads/id

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
volume	Amount of item	Yes
item	Name of item	Yes
creation_date	Date order was created	Yes

Request Body Example

```
{
  "volume": 5
  "item": "Legos",
  "creation_date": “10/10/21”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 Created	Can edit individual attributes without altering others
Failure	404 Invalid ID	If the ID of the boat does not exist, return status code 404 Boat name must be unique. Request must be JSON and Response must be JSON
Failure	400 Bad request	Required attributes not included

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: 200 Created

```
{
  "id": 123,
  "volume": 5
  "item": "Legos",
  "creation_date": “10/10/21”,
  “carrier”: [],
  “self”: http://localhost:8080/boats/123
}
```

Failure

Status: 404 Bad Request

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

}
```

Status: 400 Bad Request

{
“Error”: “Must specify all attributes. Use PATCH to update individual attributes”
}

Delete a Boat - Protected

Allows you to delete a boat. Note that if the boat is currently in a load, deleting the boat makes the load empty.

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	403 Forbidden	JWT does not match boat owner
Failure	405 Method Not Allowed	Operation not allowed at this route

Response Examples

Success

Status: 204 No Content

Failure

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

Status: 403 Forbidden
 	
{
“Error”: “Incorrect owner – not authorized to view this resource”
}

Status 405 Method Not Allowed
 	
{
“Error”: “This request is not allowed to this route”
}

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 quantity of items.	Yes
item	The name of the item	Yes
creation_date	The date this load was created	Yes

Request Body Example

```
{
  "volume": 5,
  "item": "LEGO Blocks",
  "creation_date": "10/18/2021"
}
```

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).

Response Examples

Success

Status: 201 Created

```
{
  “id”: 5676228493180928
  "volume": 5,
  "item": “LEGO Blocks”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/5676228493180928
}
```

Failure

Status: 400 Bad Request

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

View 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

Response Examples

Success

Status: 200 OK

```
{
  “id”: 5676228493180928
  "volume": 5,
  "item": “LEGO Blocks”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/5676228493180928
}
```

Failure

Status: 404 Not Found

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

View all Loads with Pagination

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	

Response Examples

Success

Status: 200 OK

loads:[

```
{
  “id”: 567
  "volume": 5,
  "item": “LEGO Blocks”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/567
},
{
  “id”: 5676228493180928
  "volume": 5,
  "item": “Lincoln Logs”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/5676228493180928
},
{
  “id”: 567622
  "volume": 5,
  "item": “Elmo Toy”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/567622”
},
{
```

```
  “id”: 5676228493180928
  "volume": 5,
  "item": “Lincoln Logs”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/5676228493180928
},
{
  “id”: 5676228493180928
  "volume": 5,
  "item": “Lincoln Logs”,
  "creation_date": “10/18/2021”,
  “carrier”: null,
  “self”: “http://localhost:8080/loads/5676228493180928
},


], “next”: “http://localhost:8080/loads/?cursor=1234566”
```

Delete a Load

Allows you to delete a load. If the load being deleted has a boat, the boat is now considered “at sea.”

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

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

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

Remove a Load from a Boat and Carrier from Load

Load management.

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 No Content	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and this boat is at this load.
Failure	404 Not Found	No boat with this boat_id is at the load with this load_id. This could be because no boat with this boat_id exists, or because no load with load_id exists, or even if both boat_id and load_id are valid, the boat with this boat_id is not at this load with this load_id

Response Examples

Success

Status: 204 No Content

Failure

```
Status: 404 Not Found
{
"Error":  " No boat with this boat_id is loaded with the load with this load_id"
}
```