Portfolio Assignment: Final Project

CS493: Cloud Application Development
Spring 2021
Oregon State University
https://ballesta-project.wl.r.appspot.com
2 pre-created users provided

Contents

Change Log

Version	Change	Date
1.0	Initial version	06/04/2020
2.0	Final version	06/06/2020

Data Model

Boat

Property Name	Notes
id	The id of the boat. Datastore automatically generates it. It is not needed to add it manually.
name	String. Required. Name of the boat. Letters and numbers only.
type	String. Required. Type of the boat. Letters and numbers only.
length	Integer. Required. Length of the boat in meters. Numbers only
loads	String. List. Not required. List of all loads on the boat. Otherwise, null. Load id taken from Datastore.
owner	String. Not required. Owner of the boat. Retrieved from encoding provided jwt.

Load

Property Name	Notes
id	The id of the load. Datastore automatically generates it. It is not needed to add it manually.
volume	Integer. Required. Volume of the load in cubic meters. Numbers only.
content	String. Required. Contents of the load. Letters and numbers only.
year	Integer. Required. Year load was made. Numbers only.
carrier	Integer. Not required. Boat the load belongs to. Otherwise, null. Boat id is taken from Datastore.

User

Property Name	Notes
id	The id of the user pertaining to Datastore. Datastore automatically generates this. It is not needed to add it manually.
user_id	String. Required. This property is automatically created when a new user first creates a boat. User is not required to do anything. Any boat the user creates thereafter does not create another user_id in Datastore.

Functionality

1) Add a boat
   - Create a new boat owned by a user.
2) View a boat
   - View details of a boat.
3) View all boats
   - View details of all boats.
   - If accessed by user, view all boats by user. If accessed by non-user, view all boats in collection.
4) Modify a boat
   - Modify through Put: All properties must be provided in request body to modify.

- Modify through Patch: Individual properties can be modified.

5) Delete a boat
   - Deletes a boat by valid user.
   - All loads on the boat will be removed. Load's carrier property will be modified to null.
6) Add a load
   - Create a new load. Not limited by user.
7) View a load
   - View details about a load.
8) View all loads
   - View details of all loads.
9) Modify a load
   - Modify through Put: All properties must be provided in request body to modify.
   - Modify through Patch: Individual properties can be modified.
10) Delete a load
    - Deletes a load.
    - If load is deleted, it's relevant carrier's loads property is modified to null if no other loads. Removed otherwise.
11) Add load to a boat
    - Add a specific load to a specific boat. Boats can hold multiple loads, but loads can only be on one boat.
12) Remove load from a boat
    - Remove a load from a boat.
    - Load's carrier property is changed to null. If it was the only load on the boat, the boat's loads property is changed to null. Otherwise, load is just removed from the boat.
13) User Login
    - The pre-created users will provide a jwt under "id_token" that can be used for authorization. Environment variables provided (jwt1 and jwt2)
    - 2 pre-created users available for use (See provided Postman Collection)
      - Username: foo@bar.com, Password: Foobar123
      - Username: wahlburg@cheese.com, Password: Cheese123
14) Get all users
    - View all users in Datastore
    - The unique identifier for a user is under user_id. It is the "sub" property given when encoding the jwt in a website such as jwt.io
    - The user entity shares a relationship with the boat entity only.

Create a Boat

Allows for the creation of a new boat. Protected entity. Bearer Token required.

```
POST /boats
```

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
name	String	Name of the boat.	Yes
type	String	Type of the boat. Speeder, Yacht, Sailboat, etc.	Yes
length	Integer	Length of the boat in meters	Yes

Request Body Example

```
{
   “name”:  “Odyssey”,
   “type”:  “Yacht”,
   “length”:  88
}
```

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 will not be created. 400 status code will be returned. Validation of the values of the attributes is not necessary. If the request contains any of the listed attributes, then the attribute’s values is valid. There will be no extraneous attributes.

Unauthorized	401	If the token is invalid or not found, 401 status code will be returned.

Response Examples

- Datastore automatically generates an ID with the entity being created.
- Self attribute contains the live link to the REST resource corresponding to this boat.

Success

```
Status: 201 created

{
  “id”:  “4905249181532160”,
  “name”:  “Odyssey”,
  “type”:  “Yacht”,
  “length”:  88,
  “owner”:  “auth0|60bc7091d2fc940071e1e554”,
  “self”:  http://localhost:8080/boats/4905249181532160
}
```

Failure

```
Status: 400 Bad Request

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

Unauthorized

```
<body>
  <pre>UnauthorizedError: invalid token<br>…</pre>
</body>
```

Get a Boat

Allows user to get an existing boat. Protected entity. Bearer Token required.

```
GET /boats/:boat_id
```

Request

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	
Failure	404 Not Found	No boat with this boat_id exists.
Failure	406 Not Acceptable	Not acceptable. Only application/json available.

Response Examples

Success

```
Status: 200 OK

{
   “name”:  “Sonic”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5538451646578688”,
   “self”:  http://localhost:8080/boats/5538451646578688
}
```

Failure

```
Status: 404 Not Found

{
   “Error”:  “No boat with this boat_id exists”
}
```

Failure

```
Status: 406 Not Acceptable

{
   “Error”:  “Not Acceptable.  Only application/json available”
}
```

Get All Boats

List all boats relevant to valid user. If non-user, lists all boats in collection. Pagination implemented. If there are more than 5 results, “next” will be available. Shows total count of results in “total”. Protected Entity. Bearer Token required.

```
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

```
{ “items”: [
{
   “name”:  “Sonic1”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5533423446578688”,
   “self”:  http://localhost:8080/boats/5533423446578688
},
{
   “name”:  “Sonic2”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5538451620914588”,
   “self”:  http://localhost:8080/boats/5538451620914588
},
{
   “name”:  “Sonic3”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5538451646578688”,
   “self”:  http://localhost:8080/boats/5538451646578688
},
{
   “name”:  “Sonic4”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5111151646578688”,
   “self”:  http://localhost:8080/boats/5111151646578688
},
{
   “name”:  “Sonic5”,
   “length”:  32,
   “owner”: “auth0|60bc8975142088006808c91c”,
   “type”:  “Speeder”,
   “id”:  “5123451646578688”,
   “self”:  http://localhost:8080/boats/5123451646578688
}],
   “next”:  http://localhost:8080/boats?cursor=Ci0SJ2oSbX5iYZXN0YS1Wcmq1z9ChELE=,
   “total”: 7
}
```

Edit a Boat (PUT)

Allows a user to edit a boat. Protected Entity. Bearer Token required.

```
PUT /boats/:boat_id
```

Request

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	Name of the boat	Yes
type	String	Type of the boat, eg. Speeder, Yacht, etc.	Yes
length	Integer	Length of the boat in meters.	Yes

Request Body Example

```
{
  “name”:  “Tails”,
  “type”:  “Raft”,
  “length”: 2
}
```

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 will not be updated. 400 status code will be returned.

		The request object is missing at least one of the required attributes.
Failure	403 Forbidden	Error: No boat with this boat_id exists or boat_id is owned by someone else

Response Examples

Success

```
Status: 200 OK

{
  “id”:  5537452203485,
  “name”:  “Tails”,
  “type”:  “Raft”,
  “length”:  2,
  “owner”:  “auth0|60bc892387492310c91c”,
  “self”:  http://localhost:8080/boats/5537452203485
}
```

Failure

```
Status: 400 Bad Request

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

Failure

```
Status: 403 Forbidden

{
{
  “Error”:  “No boat with this boat_id exists or boat_id is owned by someone else”
}
```

Edit a Boat (PATCH)

Allows a user to edit a boat. Protected Entity. Bearer Token required.

```
PATCH /boats/:boat_id
```

Request

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	Name of the boat	No
type	String	Type of the boat, eg. Speeder, Raft, Yacht, etc.	No
length	Integer	Length of the boat in meters.	No

Request Body Example

```
{
  “name” :  “Knuckles”
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request is missing at least 1 of the required attributes, the boat will not be updated. 400 status code will be returned. The request object is missing at least one of the required attributes.
Failure	403 Forbidden	Error: No boat with this boat_id exists or boat_id is owned by someone else
Failure	405 Method Not Allowed	Error: Method not allowed. See Accept Header.

Success

```
Status: 200 OK

{
  “id”:  5538456164678688,
  “name”:  “Knuckles”,
  “type”:  “Raft”,
  “length”:  2,
  “owner”:  “auth0|60bc897513647880805c91c”,
  “self”:  http://localhost:8080/boats/5538456164678688
}
```

Failure

```
Status: 400 Bad Request

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

Failure

```
Status: 403 Forbidden

{
  “Error”:  “No boat with this boat_id exists or boat_id is owned by someone else”
}
```

Failure

```
Status: 405 Method Not Allowed

{
  “Error”:  “Method not allowed.  See Accept Header”
}
```

Delete a Boat

Allows the user to delete a boat. If a boat holds a load, the load’s carrier property will become null. Protected Entity. Bearer token required.

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

Request

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	
Failure	403 Forbidden	Error: No boat with this boat_id exists or boat_id is owned by someone else.

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
  “Error”:  “No boat with this boat_id exists or boat_id is owned by someone else”
}
```

Create a Load

Allows for the creation of a load. Not protected.

```
POST /loads
```

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
volume	Integer	Volume of the load in cubic meters.	Yes
content	String	Contents of the load	Yes
year	Integer	Year contents were created	Yes

Request Body Example

```
{
  “volume”:  1,
  “content”:  “silicon”,
  “year”:  2021
}
```

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 load will not be created. 400 status code will be returned. Validation of the values of the attributes is not necessary. If the request contains any of the listed attributes, then the attribute’s values is valid. There will be no extraneous attributes.

Success

```
Status: 201 Created

{
  “id”:  “6382395930104”,
  “volume”:  1,
  “content”:  “silicon”,
  “year”:  2021,
  “self”:  http://localhost:8080/loads/6382395930104
}
```

Failure

```
Status: 400 Bad Request

{
  “Error”:  “The request object is missing the required volume or content”
}
```

Get a Load

Allows the user to get a load. Not protected.

```
GET /loads/:load_id
```

Request

Path Parameters

Name	Type	Description
boat_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	Error: No load with this load_id exists
Failure	406 Not Acceptable	Error: Not acceptable. Only application/json available

Response Examples

Success

```
Status: 200 OK

{
  “year”: 2021,
  “content”:  “silicon”,
  “volume”: 1,
  “id”:  “66435155236412”,
  “carrier”:  “348293865888324”,
  “self”:  http://localhost:8080/loads/66435155236412
}
```

Failure

```
Status: 404 Not Found

{
  “Error”:  “No boat with this boat_id exists”
}
```

Failure

```
Status: 406 Not Acceptable
{
  “Error”:  “Not acceptable.  Only application/json available”
}
```

Get All Loads

List all the loads. Pagination implemented. If there are more than 5 results, “next” will be available. Shows total count of results in “total”. Not a Protected Entity.

```
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

{
“items”: [
  {
    “content”:  “silicon1”,
  “year”: 2021,
   “volume”:  1,
   “id”:  “6643523452312”,
   “carrier”:  “8989455113234”,
   “self”:  http://localhost:8080/6643523452312
  },
 {
   “content”:  “silicon2”,
  “year”: 2021,
   “volume”:  1,
   “id”:  “66435234231312”,
   “carrier”:  “8989455113234”,
   “self”:  http://localhost:8080/66435234231312
  },
 {
   “content”:  “silicon3”,
  “year”: 2021,
   “volume”:  1,
   “id”:  “66435234211123”,
   “carrier”:  “8989455113234”,
   “self”:  http://localhost:8080/66435234211123
  },
 {
   “content”:  “silicon4”,
  “year”: 2021,
   “volume”:  1,
   “id”:  “89325234231312”,
   “carrier”:  “8989455113234”,
   “self”:  http://localhost:8080/89325234231312
  },
 {
   “content”:  “silicon5”,
  “year”: 2021,
   “volume”:  1,
   “id”:  “66435234298765”,
   “carrier”:  “8989455113234”,
   “self”:  http://localhost:8080/66435234298765
  }
 ],
   “next”:  http://localhost:8080/loads?cursor=Col8diJ2oSbX5iYZXN0YS1Wcmq1z9ChELE=,
   “total”: 6
}
```

Edit a Load (PUT)

Allows for the modification of a load. Not Protected Entity.

```
PUT /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?
volume	Integer	Volume of the load in cubic meters	Yes
content	String	Contents of the load	Yes
year	Integer	Year content was created.	Yes

Request Body Example

```
{
  “volume”:  5,
  “content”:  “basketballs”,
  “year”:  2010
}
```

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 load will not be updated. 400 status code will be returned. The request object is missing at least one of the required attributes.
Failure	404 Not found	Erorr: No load with this load_id exists

Success

```
Status: 200 OK
{
  “id”:  666432235143232,
  “volume”:  5,
  “content”:  “basketballs”,
  “year”:  2010,
  “carrier”:  64340913121843,
  “self”:  “http://localhost:8080/loads/64340913121843
}
```

Failure

```
Status: 400 Bad Request

{
  “Error”  “The request object is missing at least one of the required attributes.”
}
```

Failure

```
Status: 404 Not Found

{
  “Error”:  “No load with this load_id exists.”
}
```

Edit a Load (PATCH)

Allows for the modification of a load. Not Protected Entity.

```
PATCH /loads/:load_id
```

Request

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
volume	Integer	Volume of the contents in cubic meters	No
content	String	Contents of the load	No
year	Integer	Year content was created	No

Request Body Example

```
{
  “volume”:  8
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the request is missing at least one of the 3 required attributes, the load will not be updated. 400 status code will be returned. The request object is missing at least one of the required attributes.
Failure	404 Not Found	Error: No load with this load_id exists

Success

```
Status: 200 OK

{
  “id”:  6324548804785104,
  “volume”: 8,
  “content”:  “basketballs”,
  “year”:  2010,
  “carrier”:  “45412838945032”,
  “self”:  http://localhost:8080/loads/6324548804785104
}
```

Failure

```
Status: 400 Bad Request

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

Failure

```
Status: 404 Not found

{
  “Error”:  “No load with this load_id exists”
}
```

Delete a Load

Allows for the deletion of a load. If a boat holds a load, the boat’s loads property will become null if there are no other loads on the boat. Not Protected entity.

```
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	Error: 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”
}
```

Add Load to a Boat

Allows the user to add a load to a boat. A boat can hold more than 1 load. However, a load can only be associated to 1 boat. Not protected Entity.

```
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

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	403 Forbidden	Error: The load is already on a carrier
Failure	404 Not Found	Error: The specified boat and/or load does not exist

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
  "Error": "The load is already on a carrier"
}
```

Failure

```
Status: 404 Not Found

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

Remove Load from a Boat

Allows the user to remove a load from a boat. The load will still exist. The load’s carrier property will become null. If the load was the only load on the boat, the boat’s loads property will become null. Not Protected Entity.

```
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	
Failure	403 Forbidden	Error: The specified load is not on this boat
Failure	404 Not Found	Error: The specified boat and/or load does not exist

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 403 Forbidden

{
  “Error”:  “The specified load is not on this boat”
}
```

Failure

```
Status: 404 Not Found

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

Get all Users

Allows for the viewing of all users in Datastore. Not Protected Entity.

```
GET /users
```

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

[
 {
  “user_id”:  “auth0|60bc879645372480707c91c”,
  “id”  “4545929533194240”
 },
 {
  “user_id”:  “auth0|60bc7091d2fc940071e1e554”,
  “id”:  “6517507594649600”
 }
]
```