Final Project: API Spec

CS 493: Cloud Application Development

Spring 2021

Oregon State University

Last Update: June 6, 2021. 11:00 am Pacific

URL Account Creation/Login: https://final-project-leonep-1041pm.wl.r.appspot.com

URL Deployed Application: http://35.236.62.86:8000

Change log

Version	Change	Date
1.0	Initial version.	June 6 2021

Data Models

Data Models presented in this API.

Users

Property Name	Notes
id	String. The id of the user. Datastore automatically generates it. Do not add it yourself as a property of the entity.
firstName	String. First Name of the User on the given Google Account
lastName	String. Last Name of the User on the given Google Account
uniqueId	String. The unique Id of the given user to define the owner of the boats created. Unique Id is generated based off the user's account Id of their Google account.

Boats

Property Name	Notes
id	String. The id of the boat. Datastore automatically generates it. Do not add it yourself as a property of the entity.
name	String. Name of the boat. E.g., Goddess, Zeus, etc.
type	String. Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer. The length of the boat. E.g., 100, 12, etc.
containers	Array. List of containers assigned to the boat. Otherwise, an empty array.

Containers

Property Name	Notes
id	The id of the container. Datastore automatically generates it. Do not add it yourself as a property of the entity.
number	Integer. The number of the container E.g., 1, 1001, 99, etc.
weight	Integer. The weight of the container E.g., 125, 101, 14, etc.
content	String. The contents of the container E.g., Toys, Weights, etc.
boat	Object. Contains the information of the boat the container is assigned to. Contains the name and the id of the boat. E.g., boat: { name: Goddess, id: 1112345 }

Relationships

Boats and Containers

The relationship of boats to containers is one to many. A boat can have many containers; however, a container can only be assigned one boat at a time. Boats are created without any containers and containers are created without a boat assigned. A boat can only be loaded, and a container can only be assigned through JWT authorization as the boats are owned by a user.

Users and Boats

The relationship of users and boats is one to many. A user can own none, or 1 or more boats. A boat can only be assigned to one user. It is a one to one relationship with users.

Users

Users are assigned two unique ids. The first id is assigned by the Google Datastore upon creation of the User entity. The second identification is the uniqueId of the user which is given through authentication using Google's Oauth. The uniqueId is generated as the user's account id of their Google Account used to authenticate. This is uniqueId is used to identify the owner of the boats upon creation of the boat after successful authentication of the JWT. The JWT is used to verify the paths to access full CRUD functionality of a user's boats.

Create a Boat

Allows you to create a new boat.

POST /boats

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request Authorization

Bearer Token: JWT provided upon successful logging in

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat. E.g., Goddess, Zeus, etc.	Yes
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat in feet. E.g., 100, 12, etc.	Yes
containers	Array	Array of the containers the boat carries	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 do not 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	415 Unsupported Media Type	Content-Type must be set to application/json or the boat will not be created.
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created.
- The `self` attribute will contain the live link to the REST resource corresponding to this boat. In other words, this is the URL to get this newly created boat. The `self` attribute is not stored in Datastore.

Success

Status: 201 Created

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  “containers”: [],
  "self": "http://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

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

Failure

Status: 415 Unsupported Media Type

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

Failure

Status: 401 Unauthorized

Get a Boat

Allows you to get an existing boat

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

MIME Type

application/json

Request Authorization

Bearer Token: JWT provided upon successful logging in

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No boat with this boat_id exists
Failure	406 Not Acceptable	Only Accepts application/json data
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

Success

Status: 200 OK

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "containers":[
     {"id": 5678,
       “number”: 3,
       “weight”: 120,
       “content”: “Anchors”,
      "self":"http://appspot.com/containers/5678"},
      {"id": 8765,
       “number”: 12,
       “weight”: 10,
       “content”: “Toys”,
      "self":"http://appspot.com/containers/8765"}
   ],
  "self": "http://<your-app>/boats/abc123"
}
```

Failure

Status: 404 Not Found

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

Failure

Status: 406 Not Acceptable

```
{
"Error":  "Not Acceptable"
}
```

Failure

Status: 401 Unauthorized

List all Boats

List all the boats with pagination. Return a limit of 3 boats per page. There must be a “next” link to get the next page of boats, if there are more boats to be displayed.

GET /boats

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

MIME Type

application/json

Request Authorization

Bearer Token: JWT provided upon successful logging in

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

- The response will return 5 boats per page.
- There will be a next link to get the next page of results.
- There will not be a next link if the page being viewed is the last page.
- Containers on the boat will display the id, number, weight, and content of the container and a self link linking to the container itself.
- Each boat will have its own self link.
- The “boat” property of the items array contains the number of boats the owner has.

Failure

Status: 401 Unauthorized

Success

Status: 200 OK
items: [

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "self": "http://<your-app>/boats/abc123"
},
```

```
{
  "id": "def456",
  "name": "Adventure",
  "type": "Sailboat",
  "length": 50,
  "containers":[
     {"id": 5678,
       “number”: 101,
       “weight”: 200,
       “content”: “Barbells”,
      "self": “http://appspot.com/loads/5678”
     },
   ],
  "self": "http://<your-app>/boats/def456"
},
{
  "id": "def457",
  "name": "Goddess",
  "type": "Sailboat",
  "length": 50,
  "containers":[],
  "self": "http://<your-app>/boats/def456"
},
{
  "id": "def458",
  "name": "Batman",
  "type": "Sailboat",
  "length": 50,
  "containers":[
     {"id": 5677,
       “number”: 12,
       “weight”: 91,
       “content”: “Dishes”,
      "self":"http://appspot.com/loads/5677"},
   ],
  "self": "http://<your-app>/boats/def456"
},
{
  "id": "xyz123",
  "name": "Hocus Pocus",
  "type": "Sailboat",
  "length": 100,
  "self": "http://<your-app>/boats/xyz123"
},
{
  “boats”: 6
}
]
  "next": "http://<your-
app>/boats?cursor=CjMSLWoYendufmh3NC1nYWUtbGVvbmVwLTIzOHBtchELEgRMb2FkGICAgJim14ALDBgAIAA="
```

Edit a Boat

Allows you to edit a boat.

PATCH /boats/:boat_id

Request

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request Authorization

Bearer Token: JWT provided upon successful logging in

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat. E.g., Goddess, Zeus, etc.	No
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	No
length	Integer	Length of the boat E.g. 12, 1000, etc.	No

Request Body Example

```
{
  "name": "Sea Witch",
  "length": 99
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the body contains the “uniqueId” attribute to edit, the boat must not be updated, and 400 status code must be returned.
Failure	404 Not Found	No boat with this boat_id exists
Failure	403 Forbidden	If the owner of the boat does not match the authenticated user’s JWT the boat must not be updated.
Failure	415 Unsupported Media Type	If the Content-Type of the header is not application/json, the boat must not be created, and 415 status code to be returned.
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. It is not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99,
  “containers”: [
   “number”: 12,
   “weight”: 120,
   “content”: “Anchors”,
   “self”: "http://<your-app>/containers/abc123"
]
  "self": "http://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

Status: 400 Bad Request (Request Body Contains the uniqueId)

```
{
"Error": "Cannot update the owner of the boat."
}
```

Status: 404 Not Found

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

Status: 403 Forbidden

```
{
"Error": "Boat is owned by someone else"
}
```

Status: 415 Unsupported Media Type

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

Status: 401 Unauthorized

Edit a Boat Entity

Allows you to edit a boat.

PUT /boats/:boat_id

Request

Path Parameters

Name	Type	Description
boat_id	String	ID of the boat

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request Authorization

Bearer Token: JWT provided upon successful logging in

Request JSON Attributes

Name	Type	Description	Required?
name	String	The name of the boat. E.g., Goddess, Zeus, etc.	Yes
type	String	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat E.g. 12, 1000, etc.	Yes

Request Body Example

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

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. If the body contains the “uniqueId” attribute to edit, the boat must not be updated, and 400 status code must be returned.
Failure	404 Not Found	No boat with this boat_id exists
Failure	403 Forbidden	If the owner of the boat does not match the authenticated user’s JWT the boat must not be updated.

Failure	415 Unsupported Media Type	If the Content-Type of the header is not application/json, the boat must not be created, and 415 status code to be returned.
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. It is not store this attribute in Datastore.

Success

Status: 200 OK

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99,
  “containers”: [
   “number”: 12,
   “weight”: 120,
   “content”: “Anchors”,
   “self”: "http://<your-app>/containers/abc123"
]
  "self": "http://<your-app>/boats/abc123"
}
```

Failure

Status: 400 Bad Request

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

Status: 400 Bad Request (Request Body Contains the uniqueId)

```
{
"Error": "Cannot update the owner of the boat."
}
```

Status: 404 Not Found

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

Status: 403 Forbidden

```
{
"Error": "Boat is owned by someone else"
}
```

Status: 415 Unsupported Media Type
 	
{
"Error": “Server only accepts application/json data.”
}

Status: 401 Unauthorized

Delete a Boat

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

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

Request Authorization

Bearer Token: JWT provided upon successful logging in

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No boat with this boat_id exists
Failure	403 Forbidden	Cannot delete a boat that is not owned by the authenticated user.
Failure	401 Unauthorized	Invalid or Expried JWT

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

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

Failure

Status: 403 Not Found

```
{
"Error":  "Boat is owned by someone else"

}
```

Failure

Status: 401 Unauthorized

Create a Container

Allows you to create a new container.

POST /containers

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request JSON Attributes

Name	Type	Description	Required?
number	Integer	The number of the container	Yes
weight	Integer	The weight of the container	Yes
content	String	The content of the container	Yes

Request Body Example

```
{
   "number": 7,
   “weight”: 125
   "content": "Candy",
}
```

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 container must not be created, and 400 status code must be returned. You do not 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	Unsupported Media Type	If the Content-Type of the header is not application/json, the boat must not be created, and 415 status code to be returned.

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.
- The value of the attribute boat is the ID of the boat currently assigned to this container. If there is no boat assigned to this container then it is left null until updated. Contains the name of the boat and a self link. Both are null until assigned.
- The value of the attribute `self` is a live link to the REST resource corresponding to this container. In other words, this is the URL to get this newly created container. You must not store this attribute in Datastore.

Success

Status: 201 Created

```
{
  "id": "123abc",
  "number": 7,
  “weight”: 125
  “content”: “Candy”
  "boat": {
     “id”: null,
     “name”: null,
     “self”: null
}
  "self": "http://<your-app>/containers/123abc"
}
```

Failure

Status: 400 Bad Request

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

Failure

Status: 415 Unsupported Media Type

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

Get a Container

Allows you to get an existing container.

GET /containers/:container_id

Request

Path Parameters

Name	Type	Description
container_id	String	ID of the container

Request Body

None

Response

Response Body Format

JSON

MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No container with this container_id exists
Failure	406 Not Acceptable	Invalid Accept Headers

Response Examples

Success

```
Status: 200 OK
{ "id": 5678,
  "number": 5,
  “weight”: 10,
  “content”: “Lego Blocks”,
  "boat": {
    "id": 1234,
    "name": "Sea Witch",
    "self": "http://<your-app>/boats/1234"
  },
  "self":"http://<your-app>/containers/5678"
}
```

Failure

```
Status: 404 Not Found

{
"Error":  "No container with container_id exists"
}
```

Failure

```
Status: 406 Not Acceptable

{
"Error":  "Not Acceptable"
}
```

List all Containers

List all the loads. Implements similar pagination of limit of 5 per page to view all containers. Contains a next link if there are more items to be viewed. Contains a "containers" property that counts the number of containers available. Contains a self link of each container and the self link of the boat if the container is assigned a boat.

GET /containers

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

```
Status: 200 OK
{
   "items": [
      {
         "self": "http://app-url/containers/5069549128908800",
         "boat": {},
         "content": "Monkeys",
         “weight”: 230
         "number": 7,
         "id": "5069549128908800"
      },
      {
         "content": "Weights",
         "number": 10,
         "boat": {},
         “weight”: 300
         "self": “http://<app-url>/containers/5642368648740864",
         "id": "5642368648740864"
      },
```

```
    {
      "number": 12,
      "content": "Candy",
      "weight": 25,
      "boat": {
        "id": "5071211717459968",
         “name”: “Medusa”
        “self”: "http://app-url/boats/5071211717459968"
      },
      “self”: “http://app-url/boats/6195449035751424
      "id": "6195449035751424"
    }
    {
      "number": 1001,
      "content": "Toys",
      "weight": "66",
      "boat": {
        "id": "5071211717459968"
        “name”: “Zeus”
        “self”: "http://app-url/boats/5071211717459968"
      },
      “self”: "http://app-url/containers/6195449035751424"
      "id": "6195449035751424"
    }
    {
      "number": 77,
      "content": "Guns",
      "weight": 55,
      "boat": {
        “name”: “Batman”
        "id": "5071211717459968"
        “self”: "https://appspot.com/boats/1234"
      },
     “self”: "http://app-url/containers/6195449035751424"
      "id": "6195449035751424"
    },
    {
      “containers”: 6
    }
  ],
  "next": "http://appurl/containers?cursor=CjMSLWoYendufmh3NC1nYWUtbGVvbmVwLTIzOHBtchELEgRMb2FkGICAgJim14ALDBgAIAA="
}
```

Edit a Container

Allows you to edit a container. Can edit one or multiple properties of the container.

PATCH /containers/:container_id

Request

Path Parameters

Name	Type	Description
container_id	String	ID of the container

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request JSON Attributes

Name	Type	Description	Required?
number	Integer	The number of the container	No
weight	Integer	The weight of the container	No
content	String	The content of the container	No

Request Body Example

```
{
  "number": 9,
  "content": "Toys"
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	400 Bad Request	If the body contains the "boat" attribute to edit, the container must not be updated, and 400 status code must be returned.
Failure	404 Not Found	No container with this container_id exists
Failure	415 Unsupported Media Type	If the Content-Type of the header is not application/json, the container must not be updated, and 415 status code to be returned.

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. It is not stored in the Datastore.

Success

Status: 200 OK

```
{
        "number": 9,
        "content": "Toys",
        "weight": 55,
        "boat": {
           “name”: “Batman”
           "id": "5071211717459968"
           “self”: "https://app-url/boats/1234"
        },
       “self”: "http://app-url/containers/6195449035751424"
        "id": "6195449035751424"
},
```

Failure

Status: 400 Bad Request (Request Body Contains the boat property)

```
{
"Error": "Cannot update the boat property directly."
}
```

Status: 404 Not Found

```
{
"Error":  "No container with container_id exists."
}
```

Status: 415 Unsupported Media Type

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

Edit a Container Entity

Allows you to edit a container.

PUT /containers/:container_id

Request

Path Parameters

Name	Type	Description
container_id	String	ID of the container

Request Body

Required

Request Body Format

JSON

MIME Type

application/json

Request JSON Attributes

Name	Type	Description	Required?
number	Integer	The number of the container	Yes
weight	Integer	The weight of the container	Yes
content	String	The content of the container	Yes

Request Body Example

```
{
  "number": 9,
  “weight”: 55
  "content”: “Toys”
}
```

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. If the body contains the “boat” attribute to edit, the container must not be updated, and 400 status code must be returned.
Failure	404 Not Found	No container with this container_id exists
Failure	415 Unsupported Media Type	If the Content-Type of the header is not application/json, the container must not be updated, and 415 status code to be returned.

Response Examples

- The `self` attribute will contain the live link to the REST resource corresponding to this boat. It is not stored in the Datastore.

Success

Status: 200 OK

```
{
        "number": 9,
        "content": "Toys",
        "weight": 55,
        "boat": {
           “name”: “Batman”
           "id": "5071211717459968"
           “self”: "https://app-url/boats/1234"
        },
       “self”: "http://app-url/containers/6195449035751424"
        "id": "6195449035751424"
},
```

Failure

Status: 400 Bad Request

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

Status: 400 Bad Request (Request Body Contains the boat property)

```
{
"Error": "Cannot update the boat property directly."
}
```

Status: 404 Not Found

```
{
"Error":  "No container with container_id exists."
}
```

Status: 415 Unsupported Media Type

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

Delete a Container

Allows you to delete a container. If the container being deleted has a boat, the container is removed from the boat.

DELETE /containers/:container_id

Request

Path Parameters

Name	Type	Description
container_id	String	ID of the container

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 container_id exists

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

```
{
"Error":  "No container with container_id exists"
}
```

Load Boat with Container

Loads boat with container and assigns container the boat of the specified id. If a container is already assigned to one boat and then is assigned to another boat without first being removed, it should return a 403 status code.

PUT /boats/:boat_id/containers/:container_id

Request

Path Parameters

Name	Type	Description
container_id	String	ID of the container
boat_id	String	ID of the boat

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: No body

Request Authorization

Bearer Token: JWT provided upon successful logging in

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a container exists with this container_id and the container is not assigned.
Failure	403 Forbidden	There is already a container assigned to a boat. Container with specified id already exists on the boat.
Failure	404 Not Found	No boat with boat_id exists, and/or no container with container_id exits.
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden

```
{
"Error":  "Container is already assigned a boat. Remove before assigning."
}
```

Failure

Status: 404 Not Found
```
{
"Error":  "No boat with boat_id and/or container with container_id exists"
}
```

Failure

Status: 401 Unauthorized

Comment

- A container cannot be assigned multiple boats.

Remove a Load From a Boat

Boat has removed the container and container is no longer assigned to that boat.

DELETE /boats/:boat_id/containers/:container_id

Request

Path Parameters

Name	Type	Description
containers_id	String	ID of the containers
boat_id	String	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: No body

Request Authorization

Bearer Token: JWT provided upon successful logging in

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 contains the load.
Failure	404 Not Found	No boat with this boat_id contains the load with this load_id.
Failure	401 Unauthorized	Invalid or Expired JWT

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

```
{
"Error":  "No boat with boat_id and/or container with container_id exists"
}
```

Failure

Status: 401 Unauthorized

Edit Boats

Cannot edit all boats.

PUT /boats

Request

Path Parameters

Name	Type	Description
N/A	N/A	N/A

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	Cannot update all boats.

Response Examples

Failure

Status: 405 Method Not Allowed

```
{
"Error":  "Cannot update all boats"
}
```

Delete Boats

Cannot delete all boats.

DELETE /boats

Request

Path Parameters

Name	Type	Description
N/A	N/A	N/A

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	Cannot delete all boats.

Response Examples

Failure

Status: 405 Method Not Allowed

```
{
"Error":  "Cannot delete all boats”
}
```

Edit Containers

Cannot edit all container.

PUT /containers

Request

Path Parameters

Name	Type	Description
N/A	N/A	N/A

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	Cannot update all boats.

Response Examples

Failure

Status: 405 Method Not Allowed

```
{
"Error":  "Cannot update all containers"
}
```

Delete Containers

Cannot delete all containers.

DELETE /containers

Request

Path Parameters

Name	Type	Description
N/A	N/A	N/A

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	Cannot delete all containers.

Response Examples

Failure

Status: 405 Method Not Allowed

```
{
"Error":  "Cannot delete all containers”
}
```

List all Users

List all the users that have created accounts.

GET /users

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 OK

```
{
  "id": "abc123",
  "firstName": "Paul",
  "lastName": "Leone",
  "uniqueId": “12223432”,
  "self": "http://<your-app>/users/abc123"
},

{
  "id": "def456",
  "firstName": "John",
  "lastName": "Doe",
  "uniqueId": “57830”,
  "self": "http://<your-app>/users/def456"
},
```

Get a User

Allows you to get an existing user

GET /users/:user_id

Request

Path Parameters

Name	Type	Description
user_id	String	ID of the user

Request Body

None

Response

Response Body Format

JSON

MIME Type

application/json

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No user with user_id exists

Response Examples

Success

Status: 200 OK

```
{
  "id": "def456",
  "firstName": "John",
  "lastName": "Doe",
  "uniqueId": “57830”,
  "self": "http://<your-app>/users/def456"
},
```

Failure

Status: 404 Not Found

```
{
"Error": "No user with user_id exists"
}
```