Final Assignment: API Spec

CS 493: Cloud Application Development

Fall 2019
Oregon State University
Ryan Wallerius
https://wallerirfinalproject.appspot.com/

To begin the authentication process you will proceed to the following URL. If that does not work I attached it above. There you will give my application your account permission. On the screen you will be displayed some information along with an ID token. You will use this token to gain permissions to certain tests.

I do notice that sometimes these tokens expire, or they stop working. What I do is recreate the users, copy those ID tokens and paste them into the proper environment variables that I have created.

At the end of this document, I will go over the things in my program that are not working properly, however I will give an explanation as to why in order to gain some points. There're only a few examples to go over. The main functionality for all my functions work, but there's some minor things that are not working as intended.

Below is how I designed the entities for this project. The underlined properties are those related to just that entity. The ones that are *italicized* relate to another entity.

Boats	**Cargo**	**Users**
Id	Id	Id
name	weight	First Name
type	content	Last Name
length	delivery_date	Email
Owner	*carrier*	
Cargo	self	
self		

The reason why I picked these entities is because this is what we've been working on throughout the term. I didn't want to confuse myself by coming up with another data scheme. In the description it mentioned starting from assignment 4 so that's what I did, and what's why I have Boats and Cargo.

Create a Boat

Allows you to create a new boat. This is the entity that will be related to the user entity.

POST /boats

Request

Request Parameters

Key	Value	Description	Required?
Content-Type	Application/json	Only accept type JSON	Yes
Authorization (Bearer Token)	{{token}}	Proper authentication token to post	Yes

Make sure the token that is created after logging in, that it is properly copied over to the token property. I have had an issue with that, but I solved it with re pasting the code properly to the token variable in the environment.

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 Yatch	Yes
length	Integer	Length of the boat	Yes

Request Body Example

```
{
        “name”: “Main Boat”,
        “type”: “Yatch”,
        “length”: 100
}
```

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 and the user will encounter this status code.
Failure	401 Unauthorized	In order to create a boat you must be a registered user. You do this by going to the main page, giving account permissions and then copying the token that is posted to the token variable in the environment. This ensures that a owner is properly assigned to a boat
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will return a 406 not acceptable error code.

Response Examples

- The Datastore will automatically generate an ID and store it with the entity that is being created. This value will be seen in the response body shown in the example below.

- The *self*-attribute will contain a live link to the REST resource corresponding to the boat that was created. This attribute is not stored in the Datastore.

- The owner attribute corresponds with the sub property of the token that was created. This can be found by going to jwt.io and pasting the code in. In the decoded message, you will find a sub property that is the same as the owner property here.

Success

Status: 201 Created

{ "id": "5181872069935104",
"name": "Main Boat",
"type": "Yatch",
"length": 100,
"owner": "112167133362268180865",
"self": "https://wallerirfinalproject.appspot.com/boats/5181872069935104"
}

Failure

Status: 400 Bad Request

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

Status: 401 Unauthorized


```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Unauthorized</pre>
</body>
```


Status: 406 Not Acceptable

Server only accepts application/json data.

Get a Boat

Allows you to get an existing boat by ID. For my program I decided not to make this a protected entity. Anyone can view a specific boat by its ID.

GET /boats/:boat_id

Request

Request Parameters

Name	Type	Description	Required?
boat_id	String	ID of the boat	Yes

Request Body
None

Request Body Format
JSON

Response
Response Body Format
JSON

Response Statuses
Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No boat exists with this ID

Response Examples
Success
Status: 200 OK

```
[
  {
    "type": "Yatch",
    "cargo": [],
    "length": 100,
    "owner": "112167133362268180865",
    "name": "Main Boat",
    "self": "https://wallerirfinalproject.appspot.com/boats/4868884649738240"
  }
]
```

Failure

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


View all Boats
Allows you to view all boats, no matter who created it. I will provide what it looks like when less than 5 entities are retrieved and what it looks like when more than 5 are retrieved to fit assignment requirements.

GET /boats

Request

Request Parameters
None

Request Body
None

Request Body Format
JSON

Response
Response Body Format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples
- Here you will also see a cargo entity that will be empty (cargo: []). This is created when a user creates a boat, however with how the program is built, a user cannot assign cargo to it at the start.
- Below are two examples. The first success shows what the body looks like when less than 5 entities are retrieved from the datastore. The second shows that pagination is properly implemented.

Success
Status: 200 OK


```
{
  "items": [
    {
      "type": "Fishing Boat",
      "cargo": [],
      "length": 200,
      "owner": "111594992691666983750",
      "name": "Main Boat Other Owner",
      "id": "5115214177501184",
      "self": "https://wallerirfinalproject.appspot.com/boats/5115214177501184"
    },
    {
      "owner": "112167133362268180865",
      "name": "Main Boat",
      "type": "Yatch",
      "cargo": [],
      "length": 100,
```

```
            "id": "5181872069935104",
            "self": "https://wallerirfinalproject.appspot.com/boats/5181872069935104"
        },
        {
            "name": "Main Boat",
            "type": "Yatch",
            "cargo": [],
            "owner": "112167133362268180865",
            "id": "5710855106723840",
            "self": "https://wallerirfinalproject.appspot.com/boats/5710855106723840"
        }
    ]
}
```


Status: 200 OK


```
{
    "items": [
        {
            "type": "Yatch",
            "cargo": [],
            "length": 100,
            "owner": "112167133362268180865",
            "name": "Main Boat",
            "id": "4867664006610944",
            "self": "https://wallerirfinalproject.appspot.com/boats/4867664006610944"
        },
        {
            "owner": "112167133362268180865",
            "name": "Main Boat",
            "type": "Yatch",
            "cargo": [],
            "length": 100,
            "id": "4868884649738240",
            "self": "https://wallerirfinalproject.appspot.com/boats/4868884649738240"
        },
        {
            "cargo": [],
            "length": 200,
            "owner": "111594992691666983750",
            "name": "Main Boat Other Owner",
            "type": "Fishing Boat",
            "id": "5115214177501184",
            "self": "https://wallerirfinalproject.appspot.com/boats/5115214177501184"
        },
        {
            "name": "Main Boat",
```

```
        "type": "Yatch",
        "cargo": [],
        "length": 100,
        "owner": "112167133362268180865",
        "id": "5168618539057152",
        "self": "https://wallerirfinalproject.appspot.com/boats/5168618539057152"
    },
    {
        "type": "Yatch",
        "cargo": [],
        "length": 100,
        "owner": "112167133362268180865",
        "name": "Main Boat",
        "id": "5181872069935104",
        "self": "https://wallerirfinalproject.appspot.com/boats/5181872069935104"
    }
],
"next":
"https://wallerirfinalproject.appspot.com/boats?cursor=CjISLGoWbX53YWxsZXJpcmZpbmFscHJvamVjdHISCxIFQm
9hdHMYgICA4KmcmgkMGAAgAA=="
}
```


Delete a Boat

Allows you to delete a boat. If there is currently cargo on that boat the cargo should be removed. See note below at the end of response examples.

DELETE /boats/:boat_id

Request

Request Parameters

Key	Value	Description	Required?
Authorization (Bearer Token)	{{token}}	Proper authentication token to post	Yes

Make sure the token that is created after logging in, that it is properly copied over to the token property. I have had an issue with that, but I solved it with re pasting the code properly to the token variable in the environment.

Request Body

None

Request Body Format

None

Request JSON Attributes
None

Response
Response Body Format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	401 Unauthorized	This will occur when either no token is provided, or an invalid token is provided. For my test I did not provide a token and received the 401-error code.
Failure	403 Forbidden	The way this test is set up, is I created 2 users. I created a boat with the first user. I then tried to delete that boat, however the access token I gave belongs to the second user I created. You will get a 403-status code when this happens.
Failure	404 Not Found	This occurs when the boat that is trying to be deleted doesn't exist or the boat_id is invalid.
Failure	405 Method Not Allowed	This will occur when you try and delete all boats in the list. You would supply the argument DELETE /boats

Updating the token properties will stop the tests from behaving in a weird way. Sometime when I get run the test that is supposed to return the 403-status code, I may get a 401. If that happens, I create the two users, properly assign those tokens to the token variables in the environment. I will then create a boat using the first users token, then run this test. Then you will receive the proper response.

Response Examples

Success

Status: 204 No Content

Failure

Status: 401 Unauthorized

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Unauthorized</pre>
</body>
```

Status: 403 Forbidden

```
Boat is owned by another person
```

Status: 404 Not Found

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

Status: 405 Method Not Allowed

```
Not allowed to delete all boats
```

Edit a Boat

Allows you to edit a boat via PUT or PATCH.

PUT /boats/:boat_id

Request

Request Parameters

Key	Value	Description	Required?
Content-Type	Application/json	Only accept type JSON	Yes
Authorization (Bearer Token)	{{token}}	Proper authentication token to post	Yes

Make sure the token that is created after logging in, that it is properly copied over to the token property. I have had an issue with that, but I solved it with re pasting the code properly to the token variable in the environment.

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 Yatch	Yes
length	Integer	Length of the boat	Yes

Request Body Example
```
{
        "name": "New Boat",
        "type": "New Type",
        "length": 150
}
```

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 edited.
Failure	401 Unauthorized	If an invalid token is submitted, the user will get an unauthorized code. You must be an owner in order to edit properties of the boat.
Failure	403 Forbidden	This occurs if another user is trying to access

		and edit a boat not created by them. If tokens are updated properly this message will be received.
Failure	405 Method Not Allowed	A user is not able to edit all boats in a list. In order to receive this, you will have to run the URL PUT /boats.
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will return a 406 not acceptable error code.

Response Examples

- If the request is successful, they will see all the updated properties with id and owner and the self link staying the same.
- I touch on this below, but I was not able to keep track of the cargo properly so when a boat is updated and you send a get request, that boat that was updated will not have a cargo property. A user is still able to assign cargo to that boat. Once assigned, that property will show back up.

Success

Status: 200 OK


```
{
    "id": "5729600659259392",
    "name": "New Boat",
    "type": "New Type",
    "length": 150,
    "owner": "112167133362268180865",
    "self": "https://wallerirfinalproject.appspot.com/boats/5729600659259392"
}
```


Failure

Status: 400 Bad Request

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

Status: 401 Unauthorized

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8">
    <title>Error</title>
</head>

<body>
    <pre>Unauthorized</pre>
</body>
```

Status: 403 Forbidden

Boat is owned by another person

Status: 405 Method Not Allowed

Not allowed to edit (put) all boats

Status: 406 Not Acceptable

Server only accepts application/json data.

PATCH /boats/:boat_id

Request

Request Parameters

Key	Value	Description	Required?
Content-Type	Application/json	Only accept type JSON	Yes
Authorization (Bearer Token)	{{token}}	Proper authentication token to post	Yes

Make sure the token that is created after logging in, that it is properly copied over to the token property. I have had an issue with that, but I solved it with re pasting the code properly to the token variable in the environment.

Patch allows you to be able to supply any of the arguments and have the property still be updated. I'll show below with examples.

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 Yatch	Yes
length	Integer	Length of the boat	Yes

Request Body Example

```
{
        "name": "Patch name",
        "type": "Patch type",
        "length": 200
}
```

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	401 Unauthorized	If an invalid token is submitted, the user will get an unauthorized code. You must be an owner in order to edit properties of the boat
Failure	404 Not Found	If the user supplies the request with an invalid boat_id then they will receive this error message.
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will return a 406 not acceptable error code.

Response Examples

- When a user updates the boat via PATCH, nothing will show up in the response window. They can get the updated boat by sending a get request to boats and will see the updated boat in the list that is returned.
- I will show examples of different request bodies with not all the properties and show they are still updated correctly. The same thing happens to cargo in this function as PUT. However, cargo can still be added to it.

Success

Status: 200 OK

{
"name": "Patch name",
"type": "Patch type",
}

Result:


```
{
    "items": [
        {
            "length": 150,
            "owner": "112167133362268180865",
            "name": "Patch name",
            "type": "Patch type",
            "id": "5729600659259392",
            "self": "https://wallerirfinalproject.appspot.com/boats/5729600659259392"
        }
    ]
}
```


{
"name": "New Name",
"type": "new yatch"
}


```
{
    "items": [
        {
            "length": 150,
            "owner": "112167133362268180865",
            "name": "New Name",
            "type": "new yatch",
            "id": "5729600659259392",
            "self": "https://wallerirfinalproject.appspot.com/boats/5729600659259392"
        }
    ]
}
```


{

"name": "Patch name",
"type": "Patch type",
"length": 200
}


```
{
    "items": [
        {
            "length": 200,
            "owner": "112167133362268180865",
            "name": "Patch name",
            "type": "Patch type",
            "id": "5729600659259392",
            "self": "https://wallerirfinalproject.appspot.com/boats/5729600659259392"
        }
    ]
}
```


Failure

Status: 401 Unauthorized

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <title>Error</title>
</head>

<body>
  <pre>Unauthorized</pre>
</body>
```

Status 404 Not Found

Something went wrong with the data

Status: 406 Not Acceptable

Server only accepts application/json data.

Create Cargo

Allows you to create a piece of cargo. This is not a protected entity. Anyone can create cargo.

POST /cargo

Request

Request Parameters

Key	Value	Description	Required?

Content-Type	Application/json	Only accept type JSON	Yes

Request Body
Required

Request Body Format
JSON

Request JSON Attributes

Name	Type	Description	Required?
weight	Integer	Weight of the cargo	Yes
content	String	What is in this piece of cargo	Yes
delivery_date	String	The expected arrival date of the cargo	Yes

Request Body Example
```
{
	"weight": 100,
	"content": "Powerful PC",
	"expected_delivery": "12/31/2019"
}
```

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 cargo will not be created and the user will encounter this status code.
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will return a 406 not acceptable error code.

Response Examples

- The Datastore will automatically generate an ID and store it with the entity that is being created. This value will be seen in the response body shown in the example below.

- The *self*-attribute will contain a live link to the REST resource corresponding to the boat that was created. This attribute is not stored in the Datastore.

Success

Status: 201 Created

{ "id": "5738476309839872",
"weight": "100",
"content": "Powerful PC",
"delivery date": "12/31/2019",
"self": "https://wallerirfinalproject.appspot.com/cargo/5738476309839872"
}

Failure

Status: 400 Bad Request

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

Status: 406 Not Acceptable

Server only accepts application/json data.

View all Cargo

Allows you to view all cargo. I will provide what it looks like when less than 5 entities are retrieved and what it looks like when more than 5 are retrieved to fit assignment requirements.

GET /cargo

Request

Request Parameters

None

Request Body

None

Request Body Format

JSON

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

- Here you will also see a carrier entity that will be empty (carrier: []). This is created when a user creates cargo, however with how the program is built, a user cannot assign a boat to it at the start. That is done through a PUT request that will be documented below.
- Below are two examples. The first success shows what the body looks like when less than 5 entities are retrieved from the datastore. The second shows that pagination is properly implemented.

Success

Status: 200 OK


```
{
  "items": [
    {
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "content": "Powerful PC",
      "weight": 100,
      "id": "5993563913453568",
      "self": "https://wallerirfinalproject.appspot.com/cargo/5993563913453568"
    }
  ]
}
```


Status: 200 OK


```
{
  "items": [
    {
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "content": "Powerful PC",
      "weight": 100,
      "id": "5110311572996096",
      "self": "https://wallerirfinalproject.appspot.com/cargo/5110311572996096"
    },
    {
      "content": "Powerful PC",
      "weight": 100,
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "id": "5652786310021120",
      "self": "https://wallerirfinalproject.appspot.com/cargo/5652786310021120"
    },
```

```
    {
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "content": "Powerful PC",
      "weight": 100,
      "id": "5665833682468864",
      "self": "https://wallerirfinalproject.appspot.com/cargo/5665833682468864"
    },
    {
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "content": "Powerful PC",
      "weight": 100,
      "id": "5993563913453568",
      "self": "https://wallerirfinalproject.appspot.com/cargo/5993563913453568"
    },
    {
      "content": "Powerful PC",
      "weight": 100,
      "expected_delivery": "12/31/2019",
      "carrier": [],
      "id": "6016868674437120",
      "self": "https://wallerirfinalproject.appspot.com/cargo/6016868674437120"
    }
  ],
  "next": "https://wallerirfinalproject.appspot.com/cargo?cursor=CjISLGoWbX53YWxsZXJpcmZpbmFscHJvamVjdHISCxIFQ2FyZ28YgICAwPaJ2AoMGAAgAA=="
}
```


Delete Cargo

Allows you to delete a piece of cargo. One of the side affects is if the cargo is on a ship, to remove that cargo from the ship. However, that functionality is not properly working. Will explain at the end of the document.

DELETE /cargo/:cargo_id

Request

Request Parameters

None

Request Body

None

Request Body Format

None

Request JSON Attributes

None

Response

Response Body Format

Success: No Body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	This occurs when the cargo that is trying to be deleted doesn't exist or the cargo_id is invalid.
Failure	405 Method Not Allowed	This will occur when you try and delete all caro in the list. You would supply the argument DELETE /cargo

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

```
{
   "Error": "No cargo with this cargo_id exists"
}
```

Status: 405 Method Not Allowed

Not allowed to delete all cargo

Edit Cargo

Allows you to edit cargo via PUT or PATCH.

PUT /cargo/:cargo_id

Request

Request Parameters

Key	Value	Description	Required?

Content-Type	Application/json	Only accept type JSON	Yes

Request Body
Required

Request Body Format
JSON

Request JSON Attributes

Name	Type	Description	Required?
weight	Integer	Weight of the cargo	Yes
content	String	What makes up this piece of cargo	Yes
expected_delivery	String	Expected arrival of the cargo	Yes

Request Body Example
```
{
	"weight": 200,
	"content": "Powerful PC II",
	"expected_delivery": "12/31/2020"
}
```

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 cargo will not be edited.
Failure	405 Method Not Allowed	A user is not able to edit all boats in a list. In order to receive this, you will have to run the URL PUT /boats.
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will

		return a 406 not acceptable error code.

Response Examples

- If the request is successful, they will see all the updated properties with ID and the self link staying the same.
- The response window will not show the carrier that it is on, however when the user sends another GET request, they will see the carrier property. This will be touched on base more when the request is gone over for putting cargo on a boat.

Success

Status: 200 OK


```
{
    "id": "5430613960032256",
    "weight": "200",
    "content": "Powerful PC II",
    "delivery date": "12/31/2020",
    "self": "https://wallerirfinalproject.appspot.com/cargo/5430613960032256"
}
```


Failure

Status: 400 Bad Request

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

Status: 405 Method Not Allowed

Not allowed to edit (put) all boats

Status: 406 Not Acceptable

Server only accepts application/json data.

PATCH /cargo/:cargo_id

Request

Request Parameters

Key	Value	Description	Required?
Content-Type	Application/json	Only accept type JSON	Yes

Patch allows you to be able to supply any of the arguments and have the property still be updated. I'll show below with examples.

Request Body

Required*

Request Body Format
JSON

Request JSON Attributes

Name	Type	Description	Required?
weight	Integer	Weight of the cargo	Yes/No
content	String	What makes up this piece of cargo	Yes/No
delivery_date	Integer	Expected arrival of the cargo	Yes/No

The reason why these are not all required is PATCH allows you to update parts of a entity unlike PUT where you need all the parts.

Request Body Example
```
{
        "weight": 200,
        "content": "Powerful PC II",
        "delivery_date": "12-29-2021"
}
```

Response

Response Body Format
JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	If the user supplies the request with an invalid boat_id then they will receive this error message.
Failure	406 Not Acceptable	This is when anything but application/json is sent in. For my test sent in text/html. This will return a 406 not acceptable error code.

Response Examples

- When a user updates cargo via PATCH, nothing will show up in the response window. They can get the updated boat by sending a get request to cargo and will see the updated cargo in the list that is returned.
-

Success

Status: 200 OK

{
"weight": 200,
"expected_delivery": "12/31/2020"
}

Result:


```
{
    "items": [
        {
            "expected_delivery": "12/31/2020",
            "carrier": [],
            "content": "Powerful PC",
            "weight": 200,
            "id": "5697302505193472",
            "self": "https://wallerirfinalproject.appspot.com/cargo/5697302505193472"
        }
    ]
}
```


{
"weight": 500,
"content": "Powerful PC II"
}


```
{
    "items": [
        {
            "content": "Powerful PC II",
            "weight": 500,
            "expected_delivery": "12/31/2020",
            "carrier": [],
            "id": "5697302505193472",
            "self": "https://wallerirfinalproject.appspot.com/cargo/5697302505193472"
        }
    ]
}
```


Remember, there will be nothing that shows up in the response window, you need to send another get request to cargo to see the changes.

Failure

Status 404 Not Found
Something went wrong with the data
Status: 406 Not Acceptable
Server only accepts application/json data.

Put Cargo on Boat

Allows you to put a piece of Cargo on a Boat

PUT /boats/:boat_id/cargo/:cargo_id

This will differ from everything up to this point. How this works is the cargo you want to put on a boat is identified by the cargo id and the boat will go on the boat id that is supplied. I will first supply you with pictures of the boat and cargo entity when created.

Boat


```
{
   "items": [
      {
         "type": "Yatch",
         "cargo": [],
         "length": 100,
         "owner": "112167133362268180865",
         "name": "Main Boat",
         "id": "5726607939469312",
         "self": "https://wallerirfinalproject.appspot.com/boats/5726607939469312"
      }
   ]
}
```


Cargo


```
{
   "items": [
      {
         "expected_delivery": "12/31/2019",
         "carrier": [],
         "content": "Powerful PC",
         "weight": 100,
         "id": "5163657986048000",
         "self": "https://wallerirfinalproject.appspot.com/cargo/5163657986048000"
      }
   ]
}
```


Reminder that the owner is the sub value of the token that is provided in the authentication header. Next I run the command that is in bold above. There is no header that is required such as

content type. I don’t error check for that because no content is being provided. Just id’s of the boat and cargo.

When you submit this request, there is nothing that shows in the response window. I then submit a GET request for both boat and cargo and this is what is shown.

Boat


```
{
   "items": [
      {
         "owner": "112167133362268180865",
         "name": "Main Boat",
         "type": "Yatch",
         "cargo": [
            "5163657986048000"
         ],
         "length": 100,
         "id": "5726607939469312",
         "self": "https://wallerirfinalproject.appspot.com/boats/5726607939469312"
      }
   ]
}
```


Cargo


```
{
   "items": [
      {
         "expected_delivery": "12/31/2019",
         "carrier": [
            "5726607939469312"
         ],
         "content": "Powerful PC",
         "weight": 100,
         "id": "5163657986048000",
         "self": "https://wallerirfinalproject.appspot.com/cargo/5163657986048000"
      }
   ]
}
```


As shown by the id’s, the cargo is properly assigned to each other. Now here is where I will explain some of the side affects that are not handled. If a piece of cargo is deleted, that cargo id still will show up under this boat’s cargo. How I could make sure that doesn’t happen is checked to see if the cargo id still exists. If it doesn’t, then update the cargo property of the boat entity. The same for the reverse in this situation. That would be if a boat is deleted, remove that boat id from the carrier. Do this for checking if that id exists in the datastore.

Additionally, when a boat is updated, if the same piece of cargo is not put back on the boat, it will not show up. I tried to keep track of the cargo entity, but it was causing me problems. So, if I updated the boat, because the ID stays the same, I was able to put the cargo back on and if I send a GET request following that, it will show back up.

When I update a piece of cargo, the functionality where it still shows the carrier that it is in seems to be working properly so I was able to figure out that end of it. The last function below will list all the users in the datastore.

View all Users

Allows you to view all users. The ID is not the sub property of the token. You will have to go to jwt.io and decode the token to double check that.

GET /users

Request

Request Parameters

None

Request Body

None

Request Body Format

JSON

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 OK


```
[
  {
    "email": "wallerir@oregonstate.edu",
    "First Name": "Ryan T",
    "Last Name": "Wallerius",
    "id": "5682274280407040"
  },
```

```
  {
    "Last Name": "Wallerius",
    "email": "ryanwallerius3@gmail.com",
    "First Name": "Ryan",
    "id": "5691127080419328"
  }
]
```


Functionality that is bugged/Not working

I labeled these tests so you know exactly which ones I'm talking about. I will first start by going through the tests that are bugged.

Get /users/:user_id/boats

The first thing with this test is you have to manually copy over the user id into the request URL. I was unable to convert that into an environment variable. Also it seems that it doesn't matter what the user_id is. It just pays attention to the token that you provide it. That is why I have this labelled as bugged, however the functionality is working. If you provide it the token that belongs to the second user, it will only show the boats created by the second user. I attached response results below.

User1Token


```
[
  {
    "cargo": [
      "5163657986048000"
    ],
    "length": 150,
    "owner": "112167133362268180865",
    "name": "New Boat",
    "type": "New Type",
    "id": "5726607939469312"
  },
  {
    "name": "Main Boat",
    "type": "Yatch",
    "cargo": [],
    "length": 100,
    "owner": "112167133362268180865",
    "id": "5994784556580864"
  }
]
```


User2Token


```
[
  {
```

```
    "name": "Main Boat Other Owner",
    "type": "Fishing Boat",
    "cargo": [],
    "length": 200,
    "owner": "111594992691666983750",
    "id": "5719238044024832"
  },
  {
    "type": "Fishing Boat",
    "cargo": [],
    "length": 200,
    "owner": "111594992691666983750",
    "name": "Main Boat Other Owner",
    "id": "5741549795147776"
  }
]
```


This matches the boats created by the users. What is bugged (works sometimes and doesn't work others) is getting a boats cargo. I followed the code that was provided in lecture back when this was a requirement, but it dealt with lodgings and guests. I'm not quite sure why this is not working. The function that I am using starts on line 144 in boats.js. The reason could be there's an issue fetching the proper boat id and therefor the request is hanging.