Final Project

CS493 - Fall 2019

Author: Charles Siebert
E-mail: siebertc@oregonstate.edu

December 10, 2019

https://siebertc-cs493-final.appspot.com/

Contents

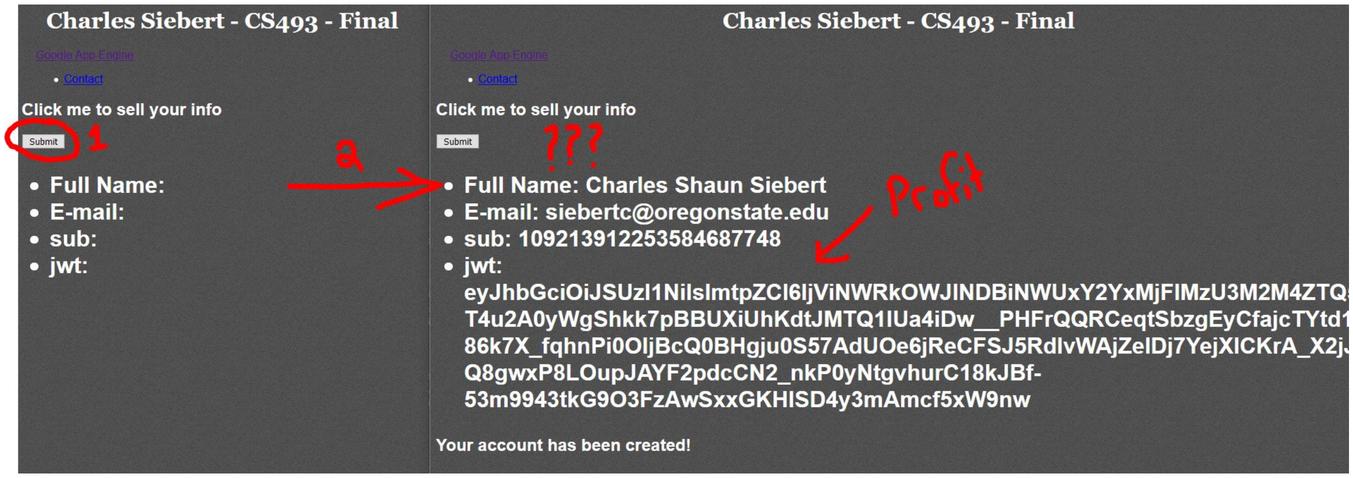
1. Introduction

This is the Final Project for CS493, where we put together all aspects of what we've learned in class, along with authorization on protected resources, and deployed on Google Cloud Platform (GCP). In this project, we're to allow the creation of accounts, and authenticate to protected resources on our RESTful API design. Access to my Project is found at https://siebertc-cs493-final.appspot.com/

The rest of this document outlines the endpoints required for Homework 4, with some tweaks to fit the requirements that were otherwise specified in the Final Project outline. The base web page is taken from my Google OAuth2.0 project, that endpoint isn't implemented and has a useless button that goes to an endpoint that doesn't exist. After signing in with Google OAuth2.0, a user entity is automatically generated.

1.1 Account Creation

To create a user account, you will go to https://siebertc-cs493-final.appspot.com/ and there will be a "submit" button that will open a third party page to authenticate with Google Auth2.0 and after successfully returning back to my webpage, it will display your **jwt** along with the **sub** property to uniquely identify you with your user account. Please see the below picture to see an example transition when using my account creation.



Charles Siebert - CS493 - Final

Click me to sell your info

- Full Name:
- E-mail:
- sub:
- jwt:

Charles Siebert - CS493 - Final

Click me to sell your info

- Full Name: Charles Shaun Siebert
- E-mail: siebertc@oregonstate.edu
- sub: 109213912253584687748
- jwt:
  eyJhbGciOiJSUzI1NiIsImtpZCI6IjViNWRkOWJlNDBiNWUxY2YxMjFlMzU3M2M4ZTQ
  T4u2A0yWgShkk7pBBUXiUhKdtJMTQ1lUa4iDw__PHFrQQRCeqtSbzgEyCfajcTYtd1
  86k7X_fqhnPi0OljBcQ0BHgju0S57AdUOe6jReCFSJ5RdlvWAjZeIDj7YejXICKrA_X2j
  Q8gwxP8LOupJAYF2pdcCN2_nkP0yNtgvhurC18kJBf-
  53m9943tkG9O3FzAwSxxGKHlSD4y3mAmcf5xW9nw

Your account has been created!

Figure 1. Transitions on my homepage detailing account creation.

1.2 Relational Schema

Please see the attached picture of a relational schema that is used to model the data relationships that is stored in the NoSQL Datastore used in my GCP project. The user identity is used by Google OAuth2.0 in the **sub** property, so while our NoSQL Datastore creates unique IDs specifically within itself as shown by the **id** in the following schemas, the **sub** property is a unique ID that is given by Google as a unique identifier. I use this property to check for user accounts and identify who owns what boats.

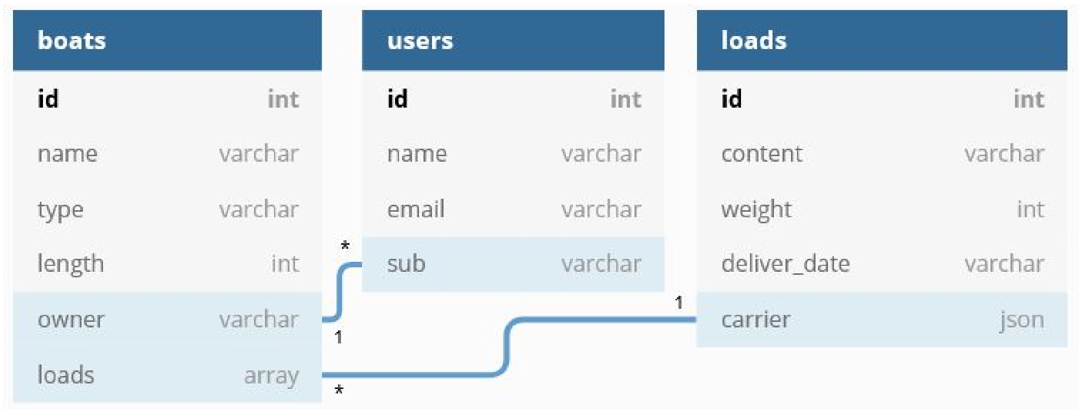


Figure 2. Diagram showing relationships between all three entities.

1.3 Authentication

There are JWT authentications at the following endpoints: Create Boat, edit boat, delete boat, delete load, add load to a boat, and remove load from a boat.

The test requires TWO JWT tokens. I've made a spare gmail account for you to use
Username: siebertccs493final@gmail.com
Password: SuperSecretPassword123

JWT tokens go into the environment variables jwt1 and jwt2

1.4 Other unfinished sections

Here's the considerations for sections I couldnt finish in time. I've worked on editing loads and boats, where specifically boats need user authentication for editing them. I've managed to get patching working, but never did a check for who was doing the edits. Along with all of the endpoints and the status codes, the documentation hasn't been fully updated. All testing should show the status codes of: 200, 201, 204, 401, 403, 405, and 406 on some endpoints. Please look at those.

2. Create a User

Allows you to create a new user

POST /boats

2.1 Request

2.1.1 Request Parameters

Authentication: 'Bearer' + JWT

2.1.2 Request Body

2.1.3 Request Body Format

JSON

2.2 Response

2.2.1 Response Body Format

JSON

2.2.2 Response Statuses

Outcome	Status Code	Notes
Success	201 Created	

2.2.3 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 `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. You must not store the self attribute in Datastore.

The `self` attribute will contain the live link to the REST resource corresponding to the loads the boat has. In other words, this is the URL to get the loads on the boat. You must not store the self attribute in Datastore.

Success

```
Status: 201 Created
{
    "id": "abc123",
    "name": "Charles Shaun Siebert",
    "email": "siebertc@oregonstate.edu",
    "sub": 1234567890,
    "self": "https://<your-app>/users/abc123"
}
```

3. View a User

Allows you to view an existing user

GET /boats/:user_id

3.1 Request

3.1.1 Request Parameters

Name	Type	Description	Required?
user_id	String	ID of the user	Yes

3.1.2 Request Body

None

3.2 Response

3.2.1 Response Body Format

JSON

3.2.2 Response Statuses

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

3.2.3 Response Examples

Success

```
Status: 200 OK
{
    "id": "abc123",
    "name": "Charles Shaun Siebert",
    "email": "siebertc@oregonstate.edu",
    "sub": 1234567890,
    "self": "https://<your-app>/users/abc123"
}
```

Failure

```
Status: 404 Failure
{
    "Error: "No user with this user_id exists"
}
```

4. View all Users

Allows you to view all existing users

GET /users

4.1 Request

4.1.1 Request Parameters

None

4.1.2 Request Body

None

4.2 Response

4.2.1 Response Body Format

JSON

4.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	

4.2.3 Response Examples

Success

```
Status: 200 OK

{
    "items": [
        {
            "id": "abc123",
            "name": "Charles Shaun Siebert",
            "email": "siebertc@oregonstate.edu",
            "sub": 1234567890,
            "self": "https://<your-app>/users/abc123"
        },
        {
            "id": "def456",
            "name": "Shaun Siebert",
            "email": "siebertcharles@oregonstate.edu",
            "sub": 12345678902342,
            "self": "https://<your-app>/users/def456"
        }
    ],
}
```

5. Create a Boat

Allows you to create a new boat

POST /boats

5.1 Request

5.1.1 Request Parameters

None

5.1.2 Request Body

Required

5.1.3 Request Body Format

JSON

5.1.4 Request JSON Attributes

Name	**Type**	**Description**	**Required?**
name	String	The name of the boat.	Yes
type	String	The type of the boat. E.g., Sailboat, catamaran, etc.	Yes
length	Integer	Length of the boat in feet.	Yes

5.1.5 Request Body Example

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

5.2 Response

5.2.1 Response Body Format

JSON

5.2.2 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 other than the ones that are listed).
Failure	406 Bad Request	Server received incorrect content-type MIME in request body.

5.2.3 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 `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. You must not store the self attribute in Datastore.

The `self` attribute will contain the live link to the REST resource corresponding to the loads the boat has. In other words, this is the URL to get the loads on the boat. You must not store the self attribute in Datastore.

Success

```
Status: 201 Created
{
    "id": "abc123",
    "name": "Sea Witch",
    "type": "Catamaran",
    "length": 28,
    "loads": null,
    "owner": 123456790,
    "self": "https://<your-app>/boats/abc123"
}
```

Failure

```
Status: 400 Bad Request
{
    "Error: "The request object is missing at least one of the required attributes"
}
```

Failure

```
Status: 406 Bad Request
{
    "Error: "The server only accepts application/json data."
}
```

6. View a Boat

Allows you to view an existing boat

GET /boats/:boat_id

6.1 Request

6.1.1 Request Parameters

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

6.1.2 Request Body

None

6.2 Response

6.2.1 Response Body Format

JSON

6.2.2 Response Statuses

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

6.2.3 Response Examples

Success

```
Status: 200 OK
{
    "id": "abc123",
    "name": "Sea Witch",
    "type": "Catamaran",
    "length": 28,
    "owner": 123456790,
    "loads": [
        {
            "id": "123abc",
            "self": "https://<your-app>/loads/123abc"
        }
    ]
    "self": "https://<your-app>/boats/abc123"
}
```

Failure

```
Status: 404 Failure
{
    "Error: "No boat with this boat_id exists"
}
```

7. View all Boats

Allows you to view all existing boats. Implements pagination, every three boats.

GET /boats

7.1 Request

7.1.1 Request Parameters

None

7.1.2 Request Body

None

7.2 Response

7.2.1 Response Body Format

JSON

7.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	

7.2.3 Response Examples

Success

```
Status: 200 OK

{
    "items": [
        {
            "type": "Yatch",
            "loads": null,
            "length": 99,
            "name": "Odyssey",
            "id": "4787714943614976",
            "owner": 123456790,
            "self": "https://siebertc-cs493-hw4.appspot.com/boats/4787714943614976"
        },
        {
            "type": "Yatch",
            "loads": null,
            "length": 99,
            "name": "Odyssey",
            "id": "4823100818456576",
            "owner": 123456790,
            "self": "https://siebertc-cs493-hw4.appspot.com/boats/4823100818456576"
        },
        {
            "length": 99,
            "name": "Odyssey",
            "type": "Yatch",
            "loads": null,
            "id": "5172860540682240",
            "owner": 123456790,
            "self": "https://siebertc-cs493-hw4.appspot.com/boats/5172860540682240"
        }
    ],
    "next": "https://siebertc-cs493-hw4.appspot.com/boats?cursor=Ci8SKWoUbX5zaWViZXJ0Yy1jczQ5My1o
}
```

8. Delete a Boat

Allows you to delete a boat. Note that if the boat is currently holding a load, deleting the boat makes the carrier field on the load become the default null value. The load has no carrier after the boat is holding it is deleted.

DELETE /boats/:boat_id

8.1 Request

8.1.1 Request Parameters

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

8.1.2 Request Body

None

8.2 Response

No Body

8.2.1 Response Body Format

Success: No Body
Failure: JSON

8.2.2 Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No boat with this boat_id exists

8.2.3 Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Failure
{
    "Error: "No boat with this boat_id exists"
}
```

```
Status: 403 Forbidden
{
    "Error": "You do not have access to edit this entity."
}
```

9. Delete a Boat 2

Allows you to delete a boat. Note this is on the root of ”boats”. We need to always return an error on this type of request.

DELETE /boats

9.1 Request

9.1.1 Request Body

None

9.2 Response

No Body

9.2.1 Response Body Format

No response

9.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Failure	405 Method Not Allowed	The method calling on this folder isn’t allowed, and returns an error

10. Edit a Boat

Allows you to edit a boat. The request body MUST have all three attributes.

PATCH /boats/:boat_id

10.1 MIME Type

application/json

10.2 Request

10.2.1 Request Parameters

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

10.2.2 Request Body

Required

10.2.3 Request Body Format

JSON

10.2.4 Request JSON Attributes

Name	**Type**	**Description**	**Required?**
name	String	The name of the boat.	No
type	String	The type of the boat. E.g., Sailboat, catamaran, etc.	No
length	Integer	Length of the boat in feet.	No

10.2.5 Request Body Example

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

10.3 Response

10.3.1 Response Body Format

JSON

10.3.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	201 Successful	
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.
Failure	403 Forbidden	You do not have access to edit this entity.
Failure	404 Not Found	No boat with this boat$_{exists}$
Failure	406 Not Acceptable	Content is not supported by this endpoint.

10.3.3 Response Examples

When PUT is used to edit a boat, the response should return a 303 code with the location of the updated boat in the appropriate header field.

Success

```
Status: 201 Other Method
{
    "id": "abc123",
    "name": "Sea Witch",
    "type": "Catamaran",
    "length": 28,
    "owner": 123456790,
    "self": "https://<your-app>/boats/abc123"
}
```

Failure

```
Status: 400 Bad Request
{
    "Error: "The request object is missing at least one of the required attributes"
}
```

```
Status: 403 Forbidden
{
    "Error": "You do not have access to edit this entity."
}
```

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

```
Status: 406 Not Acceptable
{
    "Error: "Server only accepts application/json data."
}
```

11. Put a Boat

Allows you to PUT a boat. Note this is on the root of ”boats”. We need to always return an error on this type of request.

PUT /boats

11.1 Request

11.1.1 Request Body

None

11.2 Response

No Body

11.2.1 Response Body Format

No response

11.2.2 Response Statuses

Outcome	Status Code	Notes
Failure	405 Method Not Allowed	The method calling on this folder isn’t allowed, and returns an error

12. Create a Load

Allows you to create a new load. A new load must not be assigned to a boat at time of creation.

POST /loads

12.1 Request

12.1.1 Request Parameters

None

12.1.2 Request Body

Required

12.1.3 Request Body Format

JSON

12.1.4 Request JSON Attributes

Name	Type	Description	Required?
weight	Integer	How much the load weighs	Yes
content	String	Name of the contents that the load is carrying	Yes
delivery_date	String	Due date of the shipment	Yes

12.1.5 Request Body Example

```
{
    "weight": 5,
    "content": "LEGO Blocks",
    "delivery_date": "1/1/2020"
}
```

12.2 Response

12.2.1 Response Body Format

JSON

12.2.2 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 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 other than the ones that are listed).
Failure	406 Bad Request	Server received incorrect content-type MIME in request body.

12.2.3 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 `self` attribute will contain the live link to the REST resource corresponding to this load. In other words, this is the URL to get this newly created load. You must not store the self attribute in Datastore.

Success

```
Status: 201 Created
{
    "id": "123abc", # automatically generated by the Datastore
    "weight": 5,     #The weight of the load
    "carrier": null,  #The boat carrying the load
    "content": "LEGO Blocks",
    "delivery_date": "1/1/2020" #The date the load is to be delivered
    "self": "https://appspot.com/loads/123abc"
}
```

Failure

```
Status: 400 Bad Request
{
    "Error: "The request object is missing at least one of the required attributes"
}
```

Failure

```
Status: 406 Bad Request
{
    "Error: "The server only accepts application/json data."
}
```

13. View a Load

Allows you to view an existing load

GET /loads/:load_id

13.1 Request

13.1.1 Request Parameters

Name	Type	Description	Required?
load_id	String	ID of the load	Yes

13.1.2 Request Body

None

13.2 Response

13.2.1 Response Body Format

JSON

13.2.2 Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No load with this load_id exists

13.2.3 Response Examples

Success

```
Status: 200 OK
{
    "id": "123abc",
    "weight": 5,
    "carrier": {
        "id": "abc123",
        "name": "Sea Witch"
        "self": https://<your-app>/boats/abc123
    },
    "content": "LEGO Blocks",
    "delivery_date": "1/1/2020"
    "self": "https://appspot.com/loads/123abc"
}
```

Failure

```
Status: 404 Failure
{
    "Error: "No load with this load_id exists"
}
```

14. View all Loads

Allows you to view all existing loads. Implements pagination, every three boats.

GET /loads

14.1 Request

14.1.1 Request Parameters

None

14.1.2 Request Body

None

14.2 Response

14.2.1 Response Body Format

JSON

14.2.2 Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

14.2.3 Response Examples

Success

```
Status: 200 OK
{
    "items": [
        {
            "content": "LEGO Blocks",
            "weight": 1,
            "delivery_date": "1/1/2021",
            "carrier": {
                "name": "Odyssey",
                "id": "5740877414662144"
            },
            "id": "5112644562321408",
            "self": "https://siebertc-cs493-hw4.appspot.com/loads/5112644562321408"
        },
        {
            "carrier": {
                "name": "Odyssey",
                "id": "5951867817295872"
            },
            "content": "LEGO Blocks",
            "weight": 1,
            "delivery_date": "1/1/2021",
            "id": "5146163359514624",
            "self": "https://siebertc-cs493-hw4.appspot.com/loads/5146163359514624"
        },
        {
            "carrier": {
                "id": "5951867817295872",
                "name": "Odyssey"
            },
            "content": "LEGO Blocks",
            "weight": 1,
            "delivery_date": "1/1/2021",
            "id": "5424333895761920",
            "self": "https://siebertc-cs493-hw4.appspot.com/loads/5424333895761920"
        }
    ],
    "next": "https://siebertc-cs493-hw4.appspot.com/loads?cursor=Ci8SKWoUbX5zaWViZXJ0Yy1jczQ5My1c
}
```

15. Delete a Load

Allows you to delete a load. Note that if the load is currently assigned to a boat, deleting the load makes the ”loads” field on the boat become removed from the array. The boat no longer has the load information specific to the one that was just deleted.

DELETE /loads/:load_id

15.1 Request

15.1.1 Request Parameters

Name	**Type**	**Description**	**Required?**
load_id	String	ID of the load	Yes

15.1.2 Request Body

None

15.2 Response

No Body

15.2.1 Response Body Format

Success: No Body
Failure: JSON

15.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	
Failure	403 Forbidden	You do not have access to edit this entity.
Failure	404 Not Found	No boat with this boat_id exists

15.2.3 Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Failure
{
    "Error: "No load with this load_id exists"
}
```

```
Status: 403 Forbidden
{
    "Error": "You do not have access to edit this entity."
}
```

16. Delete a load 2

Allows you to delete a load. Note this is on the root of ”loadss”. We need to always return an error on this type of request.

DELETE /loads

16.1 Request

16.1.1 Request Body

None

16.2 Response

No Body

16.2.1 Response Body Format

No response

16.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Failure	405 Method Not Allowed	The method calling on this folder isn’t allowed, and returns an error

17. Edit a Load

Allows you to edit a load. The request body MUST have all three attributes.

PATCH /loads/:load_id

17.1 MIME Type

application/json

17.2 Request

17.2.1 Request Parameters

Name	**Type**	**Description**	**Required?**
boat_id	String	ID of the load	Yes

17.2.2 Request Body

Required

17.2.3 Request Body Format

JSON

17.2.4 Request JSON Attributes

Name	**Type**	**Description**	**Required?**
weight	Integer	How much the load weighs	Yes
content	String	Name of the contents that the load is carrying	Yes
delivery_date	String	Due date of the shipment	Yes

17.2.5 Request Body Example

```
{
    "weight": 5,
    "content": "LEGO Blocks",
    "delivery_date": "1/1/2020"
}
```

17.3 Response

17.3.1 Response Body Format

JSON

17.3.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	201 Successful	
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.
Failure	404 Not Found	No load with this load$_{exists}$
Failure	406 Not Acceptable	Content is not supported by this endpoint.

17.3.3 Response Examples

When PUT is used to edit a load, the response should return a 303 code with the location of the updated boat in the appropriate header field.

Success

```
Status: 201 Other Method
{
    "id": "123abc", # automatically generated by the Datastore
    "weight": 5,     #The weight of the load
    "carrier": null,  #The boat carrying the load
    "content": "LEGO Blocks",
    "delivery_date": "1/1/2020" #The date the load is to be delivered
    "self": "https://appspot.com/loads/123abc"
}
```

Failure

```
Status: 400 Bad Request
{
    "Error: "The request object is missing at least one of the required attributes"
}
```

```
Status: 404 Not Found
{
    "Error: "No load with this load_id exists"
}
```

```
Status: 406 Not Acceptable
{
    "Error: "Server only accepts application/json data."
}
```

18. PUT a load

Allows you to PUT a load. Note this is on the root of ”loads”. We need to always return an error on this type of request.

PUT /loads

18.1 Request

18.1.1 Request Body

None

18.2 Response

No Body

18.2.1 Response Body Format

No response

18.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Failure	405 Method Not Allowed	The method calling on this folder isn’t allowed, and returns an error

19. Put a Load on a boat

Allows you to put a load on a boat. Will not put a load if the carrier attribute already exists on a load. Makes sure that the load is a valid load ID.

PUT /boats/:boat_id/loads/:load_id

19.1 Request

19.1.1 Request Parameters

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

19.1.2 Request Body

None

19.2 Response

No Body

19.2.1 Response Body Format

Success: No Body
Failure: JSON

19.2.2 Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No boat with this boat_id exists
Failure	404 Not Found	No load with this load_id exists
Failure	403 Forbidden	The load is already assigned to another boat.
Failure	403 Forbidden	You do not have access to edit this entity.

19.2.3 Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Failure
{
    "Error: "No load with this load_id exists"
}
```

```
Status: 404 Failure
{
    "Error: "No boat with this boat_id exists"
}
```

```
Status: 403 Forbidden
{
    "Error": "The load is already assigned to another boat."
}
```

```
Status: 403 Forbidden
{
    "Error": "You do not have access to edit this entity."
}
```

20. Remove a Load from a boat

Allows you to put a load on a boat. Will not put a load if the carrier attribute already exists on a load. Makes sure that the load is a valid load ID.

PUT /loads/:load_id

20.1 Request

20.1.1 Request Parameters

Name	**Type**	**Description**	**Required?**
load_id	String	ID of the load	Yes

20.1.2 Request Body

None

20.2 Response

No Body

20.2.1 Response Body Format

Success: No Body
Failure: JSON

20.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	204 No Content	
Failure	404 Not Found	No load with this load_id exists
Failure	403 Forbidden	The load is already unassigned from any boat.
Failure	403 Forbidden	You do not have access to edit this entity.

20.2.3 Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Failure
{
    "Error: "No load with this load_id exists"
}
```

```
Status: 403 Forbidden
{
    "Error": "The load is already unassigned from any boat."
}
```

```
Status: 403 Failure
{
    "Error: "You do not have access to edit this entity."
}
```

21. View all loads of a Boat

Allows you to view all loads that live on a given boat

GET /loads/boats/:boat_id

21.1 Request

21.1.1 Request Parameters

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

21.1.2 Request Body

None

21.2 Response

21.2.1 Response Body Format

JSON

21.2.2 Response Statuses

Outcome	**Status Code**	**Notes**
Success	200 OK	
Failure	404 Not Found	No boat with this boat_id exists

21.2.3 Response Examples

Success

```
Status: 200 OK
[
    {
        "content": "LEGO Blocks",
        "weight": 1,
        "delivery_date": "1/1/2021",
        "carrier": {
            "id": "5676228493180928",
            "name": "Odyssey"
        },
        "id": "5086255679275008",
        "self": "https://siebertc-cs493-hw4.appspot.com/loads/5086255679275008"
    },
    {
        "content": "LEGO Blocks",
        "weight": 1,
        "delivery_date": "1/1/2021",
        "carrier": {
            "name": "Odyssey",
            "id": "5426176570949632"
        },
        "id": "5110792424783872",
        "self": "https://siebertc-cs493-hw4.appspot.com/loads/5110792424783872"
    }
]
```

Failure

```
Status: 404 Failure
{
    "Error: "No boat with this boat_id exists"
}
```