Assignment 4: API Spec

CS 493: Cloud Application Development

Fall 2020

Oregon State University

Last Update: Oct 12, 2020. 6:00 am Pacific

Change log

Page	**Change**	**Date**
	Initial version.	M, Oct 12

Create a Boat

Allows you to create a new boat.

POST /boats/

Request

Request Parameters

None

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., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat in feet.	Yes
Loads	list	List of loads carried by the boat	Yes

Request Body Example

```
{
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28
  “loads”: []
}
```

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 4 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).

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

Success

```
Status: 201 Created

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

Failure

```
Status: 400 Bad Request

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

Get a Boat

Allows you to get an existing boat

GET /boats/:boat_id

Request

Request Parameters

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

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

Response Examples

Success

```
Status: 200 OK

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

Failure

```
Status: 404 Not Found

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

List all Boats

List all the boats.

```
GET /boats
```

Request

Request 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

[
{
  "id": "abc123",
  "name": "Sea Witch",
   "type": "Catamaran",
  "length": 28,
  “loads”: [],
  "self": "https://<your-app>/boats/abc123"
},
{
  "id": "def456",
  "name": "Adventure",
   "type": "Sailboat",
  "length": 50,
  “loads”: [],
  "self": "https://<your-app>/boats/def456"
},
{
  "id": "xyz123",
  "name": "Hocus Pocus",
   "type": "Sailboat",
  "length": 100,
  “loads”: [],
  "self": "https://<your-app>/boats/xyz123"
}
]
```

Edit a Boat

Allows you to edit a boat.

PUT /boats/:boat_id

Request

Request Parameters

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

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., Sailboat, Catamaran, etc.	Yes
length	Integer	Length of the boat in feet.	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. You don't need to validate the values of the attributes and can assume that if the request contains any of the listed attributes, then that attribute's value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than the ones that are listed).

Failure	404 Not Found	No boat with this boat_id exists

Response Examples

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

Success

Status: 200 OK

```
{
  "id": "abc123",
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99,
  “loads”: [],
  "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: 404 Not Found

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

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

Request Parameters

None

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

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

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

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

Create a Load

Allows you to create a new slip.

```
POST /loads
```

Request

Request Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Type	Description	Required?
Weight	Integer	The weight of the load	Yes
Content	String	The content of the load	Yes
Delivery_Date	String	The delivery date of the load	Yes
Carrier	Integer	The id of the boat carrying this load	No

Request Body Example

```
{
  “Weight”: 20,
  “Content”: “Legos”,
  “Delivery_date”: “10/10/2010”,
  “carrier”: null
}
```

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 slip must not be created, and 400 status code must be returned. You don’t need to validate the values of this attribute and can assume that if the number attribute is specified, then its value is valid. You can also assume that the request will not contain any extraneous attribute (i.e., the request JSON will never contain any attribute other than number).

Response Examples

- 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 `carrier` is the ID of the boat currently carrying this load. If there is no boat carrying this load, the value of `carrier` should be null
- The value of the attribute `self` is a live link to the REST resource corresponding to this slip. In other words, this is the URL to get this newly created slip. You must not store this attribute in Datastore

Success

```
Status: 201 Created

{
  "id": "123abc",
  "Weight": 20,
  “Content”: “Legos”,
  “Delivery_date”: “10/10/2010”,
  "Carrier": null,
  "self": "https://<your-app>/slips/123abc"
}
```

Failure

```
Status: 400 Bad Request

{
"Error":  "The request object is missing the required number"
}
```

Get a Load

Allows you to get an existing slip.

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

Request

Request Parameters

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

Request Body

None

Response

Response Body Format

JSON

Response Statuses

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

Response Examples

Success

```
Status: 200 OK

{
  "id": "123abc",
  "weight”: 20,
  “content”: “Legos”,
  “delivery_date”: “10/10/2010”,
  "carrier": "abc123",
  "self": "https://<your-app>/slips/123abc"
}
```

Failure

```
Status: 404 Not Found

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

List all Loads

List all the loads.

```
GET /loads
```

Request

Request 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

[
{
  "id": "123abc",
  "weight”: 20,
  “content”: “Legos”,
  “delivery_date”: “10/10/2010”,
  "carrier": "abc123",
  "self": "https://<your-app>/slips/123abc"
},
{
  "id": "456def",
  "weight”: 200,
  “content”: “Hover bike”,
  “Delivery_date”: “10/25/2015”,
  "carrier": null,
  "self": "https://<your-app>/slips/456def"
}
]
```

Delete a Load

Allows you to delete a load. If the load being deleted is on a boat, the load is also “unloaded” from the boat.

```
DELETE /loads/:load_id
```

Request

Request Parameters

None

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	404 Not Found	No load with this load_id exists

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

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

Load put on boat

A load is put on a boat.

```
PUT /boats/:boat_id/loads/:load_id
```

Request

Request Parameters

None

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

Failure: JSON

Response Statuses

Outcome	**Status Code**	**Notes**
Success	201 Created	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and the load is not already on the boat.
Failure	403 Forbidden	There is already a load on this boat.
Failure	403 Forbidden	The load is already on another boat
Failure	404 Not Found	No boat with this boat_id exists, and/or no load with this load_id exits.

Response Examples

Success

```
Status: 201 Created
```

Failure

```
Status: 403 Forbidden

{
"Error":  "The load is already on the boat"
}
```

```
Status: 404 Not Found

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

Comment

- A load cannot be assigned to multiple boats.

Load removed from Boat

Load has been removed from the boat

```
DELETE /boats/:boat_id/loads/:load_id
```

Request

Request Parameters

None

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a load exists with this load_id and this load is on this boat.
Failure	404 Not Found	No load with this load_id is on the boat with this boat_id.

Response Examples

Success

```
Status: 204 No Content
```

Failure

```
Status: 404 Not Found

{
"Error":  "No load with this load_id is on the boat with this boat_id"
}
```

View Loads for a Boat

Load has been removed from the boat

```
GET /boats/:boat_id/loads/:load_id
```

Request

Request Parameters

None

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	Succeeds only if a boat exists with this boat_id, and a load exists with this load_id.
Failure	404 Not Found	No boat with this boat_id exists.

Response Examples

Success

```
Status: 200 OK
```

Failure

```
Status: 404 Not Found

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