Assignment 3: API Spec

CS 493: Cloud Application Development

Fall 2021

Oregon State University

Last Update: Oct 13, 2021. 11:15 am Pacific

Change log

Version	Change	Date
2.0	Fixed error messages for the endpoints for Boat Arrives at a Slip & Boat Departs a Slip which return 404 status code. The error message are now consistent with the Postman Collection.	Oct 13, 2021
1.0	Initial version.	Oct 11, 2021

Data Model

The app stores two kinds of entities in Datastore, Boats and Slips.

Boats

Property	Data Type	Notes
id	Integer	The id of the boat. Datastore automatically generates it. Don't add it yourself as a property of the entity.
name	String	Name of the boat.
type	String	Type of the boat. E.g., Sailboat, Catamaran, etc.
length	Integer	The length of the boat in feet.

Slips

Property	Data Type	Notes
id	Integer	The id of the slip. Datastore automatically generates it. Don't add it yourself as a property of the entity.
number	Integer	The number of the slip.
current_boat	Integer	If there is a boat at the slip, then id of that boat. Otherwise null.

Create a Boat

Allows you to create a new boat.

POST /boats

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	Length of the boat in feet.	Yes

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 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 that is not listed above).

Response Examples

- Datastore will automatically generate an ID and store it with the entity being created. Your app must send back this value 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": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "self": "https://<your-app>/boats/123"
}
```

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

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

Response Body Format

JSON

Response Statuses

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

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "self": "https://<your-app>/boats/123"
}
```

Failure

Status: 404 Not Found

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

List all Boats

List all the boats.

GET /boats

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 OK


```
[
{
  "id": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 28,
  "self": "https://<your-app>/boats/123"
},
{
  "id": 456,
  "name": "Adventure",
  "type": "Sailboat",
  "length": 50,
  "self": "https://<your-app>/boats/456"
},
{
  "id": 789,
  "name": "Hocus Pocus",
  "type": "Sailboat",
  "length": 100,
  "self": "https://<your-app>/boats/789"
}
]
```

Edit a Boat

Allows you to edit a boat.

PATCH /boats/:boat_id

Request

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
name	The name of the boat.	Yes
type	The type of the boat. E.g., Sailboat, Catamaran, etc.	Yes
length	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": 123,
  "name": "Sea Witch",
  "type": "Catamaran",
  "length": 99,
  "self": "https://<your-app>/boats/123"
}
```

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

Path Parameters

Name	Description
boat_id	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	
Failure	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 Slip

Allows you to create a new slip.

POST /slips

Request

Path Parameters

None

Request Body

Required

Request Body Format

JSON

Request JSON Attributes

Name	Description	Required?
number	The number of the slip.	Yes

Request Body Example

```
{
  "number": 1
}
```

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 `current_boat` is the ID of the boat currently at this slip. If there is no boat at this slip, the value of `current_boat` 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": 123,
  "number": 1,
  "current_boat": null,
  "self": "https://<your-app>/slips/123"
}
```

Failure

Status: 400 Bad Request

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

Get a Slip

Allows you to get an existing slip.

GET /slips/:slip_id

Request

Path Parameters

Name	Description
slip_id	ID of the slip

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	
Failure	404 Not Found	No slip with this slip_id exists

Response Examples

Success

Status: 200 OK

```
{
  "id": 123,
  "number": 1
  "current_boat": 345,
  "self": "https://<your-app>/slips/123"
}
```

Failure

Status: 404 Not Found

```
{
"Error":  "No slip with this slip_id exists"
}
```

List all Slips

List all the slips.

GET /slips

Request

Path Parameters

None

Request Body

None

Response

Response Body Format

JSON

Response Statuses

Outcome	Status Code	Notes
Success	200 OK	

Response Examples

Success

Status: 200 OK

```
[
{
  "id": 123,
  "number": 1,
  "current_boat": 345,
  "self": "https://<your-app>/slips/123"
},
{
  "id": 456,
  "number": 2,
  "current_boat": null,
  "self": "https://<your-app>/slips/456"
}
]
```

Delete a Slip

Allows you to delete a slip. If the slip being deleted has a boat, the boat is now considered “at sea.”

DELETE /slips/:slip_id

Request

Path Parameters

Name	Description
slip_id	ID of the slip

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 slip with this slip_id exists

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

```
{
"Error":  "No slip with this slip_id exists"
}
```

Boat Arrives at a Slip

Boat has arrived at a slip.

PUT /slips/:slip_id/:boat_id

Request

Path Parameters

Name	Description
slip_id	ID of the slip
boat_id	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

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a slip exists with this slip_id and the slip is empty.
Failure	403 Forbidden	There is already a boat at this slip.
Failure	404 Not Found	~~No boat with this boat_id exists, and/or no slip with this slip_id exits.~~ The specified boat and/or slip does not exist

Response Examples

Success

Status: 204 No Content

Failure

Status: 403 Forbidden

```
{
"Error":  "The slip is not empty"
}
```

Status: 404 Not Found

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

Comment

- A boat cannot be assigned to multiple slips.
- However, in grading the assignment, we will not test this scenario, i.e., if a boat is already assigned to a slip, we will not send requests that this boat be assigned to another slip
- Implementing a check to prevent such multiple assignments for a boat is optional.

Boat Departs a Slip

Boat has left the slip to go to sea. The slip is now empty.

DELETE /slips/:slip_id/:boat_id

Request

Path Parameters

Name	Description
slip_id	ID of the slip
boat_id	ID of the boat

Request Body

None

Response

No body

Response Body Format

Success: No body

Failure: JSON

Response Statuses

Outcome	Status Code	Notes
Success	204 No Content	Succeeds only if a boat exists with this boat_id, a slip exists with this slip_id and this boat is at this slip.
Failure	404 Not Found	~~The specified boat and/or slip does not exist.~~ No boat with this boat_id is at the slip with this slip_id

Response Examples

Success

Status: 204 No Content

Failure

Status: 404 Not Found

~~{~~
~~"Error": "The specified boat and/or slip does not exist"~~
~~}~~

```
{
"Error":  " No boat with this boat_id is at the slip with this slip_id"
}
```