Promineo Tech Back End Final Project

Hello and welcome to my first REST API. It was a long process but here it is. What made this really challenging is our beloved instructor George Heeres demonstrated an API without Lomboc and with Constructor Injection instead of Dr. Rob's Property Injection. There are things I would improve and things I would have done differently. I only used the Swagger UI to test this. I think I will try more traditional testing. Errors are easier to debug now. I would also implement logging of some type to help with locating errors. Lastly I will work on a get an entire column or select column method, too.

For this project I will use option 2. Starting with my beer table, then brewery table and finally the tub table.

This is an excellent point in which to paste in my Proposal Template just for some background on my project and then I will past my Swagger input and results.

Proposal Template:

Project Participants:

Julie Curran

Title:

Collectable Beer Can Seller's Inventory API

Executive Summary:

This project is being prepared for a beer lover who found that he could sell empty craft beer cans with unique labels. He didn't have a problem remembering if he had a duplicate can or where they were located when he only had 10-20 cans. It was a different story when friends and contacts started to collect for him. The goal is to create an API that will allow the seller to determine if, once sold, there is another can of that same brewery and title to sell and the location of the duplicate can. This allows him to save time looking and possibly missing out on sales if he didn't know he had a duplicate can.

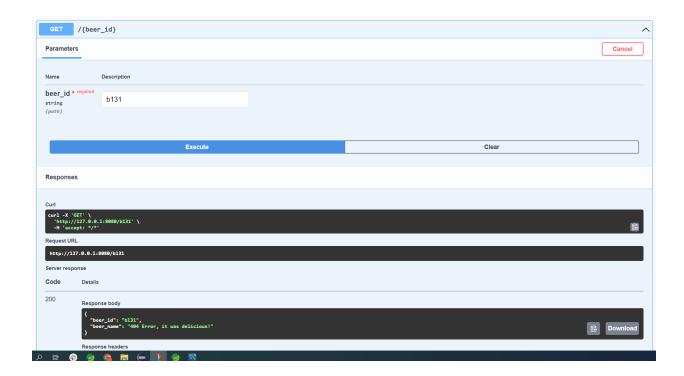
Initial Features:

- Entities: Brewery, Beer Title, and Tub
- The seller needs to perform the following operations:
 - View a can and the can id in his inventory. (GET/beer/{beer_names})
 - View how many of cans one specific beer titles (GET /tub{tub_id can_quantity}
 - Delete beer titles from inventory as they sell (DELETE/beer/{beer_id})
 - Update or Create beer can titles, brewery and location to the list (POST(post/tub/{tub_id beer_id brewer_id})
 - Add a can title, brewery or tub (POST /beer JSON Body of request {beer_id} (POST /brewery JSON Body of request {brewery_id beer_id} (POST /tub JSON Body of request {tub_id}
 - The last two additions allow the seller to enter the beer_id or other objects they want to associate with the new object.

Stretch Goals (to be completed if time allows, or after graduation):

- Program lists to display in alphabetical order.
- Print a list of cans and where they can be located so seller can easily fullfill an order.
- Create a search with a wild card and ignoring case.
- Create an auto-complete input field.

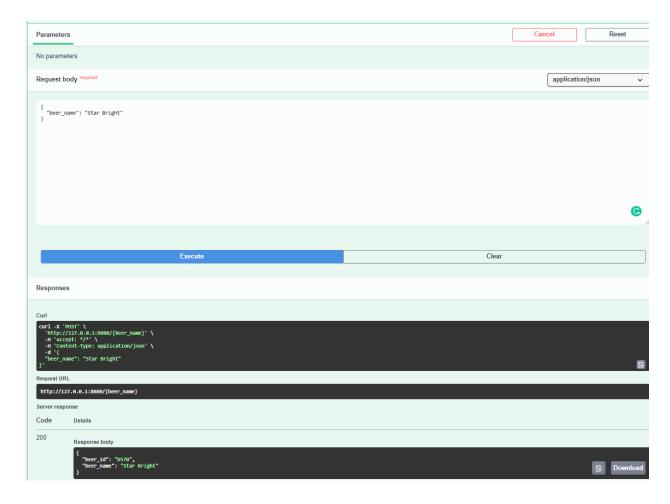
Now we will start with the GET for all three.



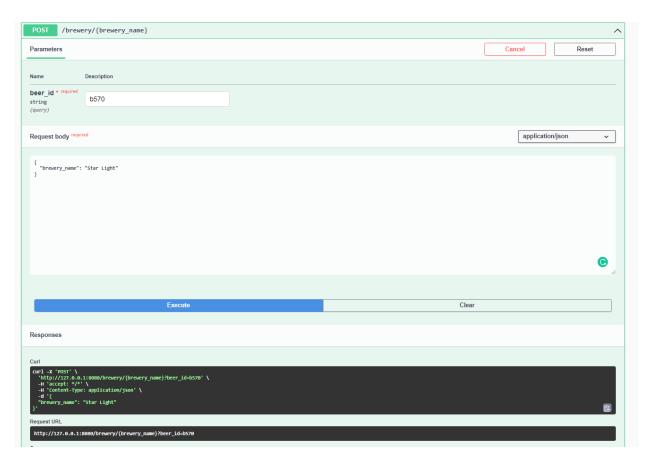
```
brewery_id * required
                           br502
string
(path)
                                                   Execute
Responses
Curl
 curl -X 'GET' \
   'http://127.0.0.1:8080/brewery/br502' \
   -H 'accept: */*'
Request URL
 http://127.0.0.1:8080/brewery/br502
Server response
Code
             Details
200
             Response body
                "brewery_id": "b526",
"brewery_name": "!null"
             Response headers
                connection: keep-alive
                content-type: application/json
                date: Sun,04 Sep 2022 06:14:12 GMT
                keep-alive: timeout=60
                transfer-encoding: chunked
```

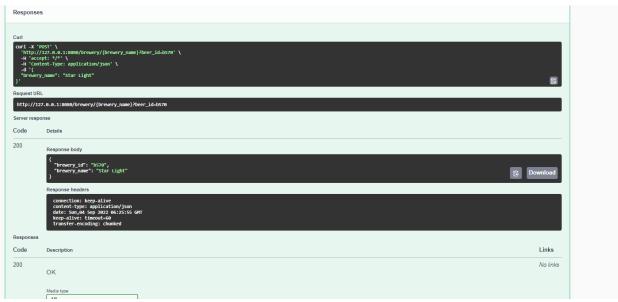
```
/tub/{tub_id}
  GET
Parameters
                  Description
Name
tub_id * required
                   t364
string
(path)
                                                Execute
Responses
Curl
 curl -X 'GET' \
   'http://127.0.0.1:8080/tub/t364' \
   -H 'accept: */*'
Request URL
 http://127.0.0.1:8080/tub/t364
Server response
Code
            Details
200
            Response body
               "tub_id": "t364",
               "tub_name": "8"
            Response headers
```

And now the post:



Star bright beer is being inserted into Star Light brewery.





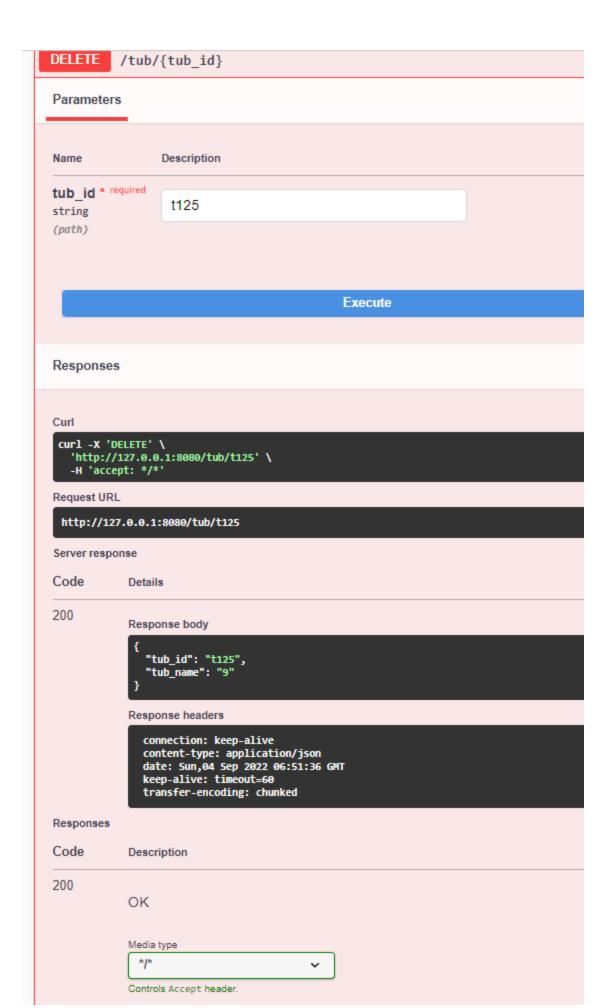
POST /tub/{tub_id}		
Parameters		Cancel Reset
Name	Description	
can_quantity * required string (query) brewery_id * required string (query) beer_id * required string (query)	2	
	br503	
	b570	
Request body required		application/json v
{ "tub_name": "9"		
}		

```
Curl
curl -X 'POST' \
   'http://127.0.0.1:8080/tub/{tub_id}?can_quantity=2&brewery_id=br503&beer_id=b570' \
-H 'accept: */*' \
   -H 'Content-Type: application/json' \
   -d '{
   "tub_name": "9"
Request URL
 http://127.0.0.1:8080/tub/{tub_id}?can_quantity=2&brewery_id=br503&beer_id=b570
Server response
Code
              Details
200
              Response body
                  "tub_id": "t125",
                 "tub_name": "9"
              Response headers
                 connection: keep-alive
                 content-type: application/json
                 date: Sun,04 Sep 2022 06:38:33 GMT keep-alive: timeout=60
                 transfer-encoding: chunked
Responses
Code
              Description
200
              OK
              Media type
                */*
              Controls Accept header.
              Example Value | Schema
                 "tub_id": "string",
"tub_name": "string"
```

Here is delete:

```
DELETE
              /{beer_id}
Parameters
Name
                         Description
beer_id * required
                           b570
string
(path)
                                                              Execute
Responses
Curl
 curl -X 'DELETE' \
    'http://127.0.0.1:8080/b570' \
    -H 'accept: */*'
Request URL
  http://127.0.0.1:8080/b570
Server response
Code
                Details
200
                Response body
                   "beer_id": "b570",
"beer_name": "Star Bright"
                Response headers
                   connection: keep-alive
                   content-type: application/json
date: Sun,04 Sep 2022 06:40:48 GMT
keep-alive: timeout=60
                   transfer-encoding: chunked
Responses
Code
                Description
200
                OK
```

```
DELETE
            /brewery/{brewery_id}
Parameters
                         Description
Name
brewery id * required
                           br91
string
(path)
                                                    Execute
Responses
Curl
curl -X 'DELETE' \
   'http://127.0.0.1:8080/brewery/br91' \
   -H 'accept: */*'
Request URL
 http://127.0.0.1:8080/brewery/br91
Server response
Code
             Details
200
              Response body
                "brewery_id": "b570",
"brewery_name": "Star Light"
              Response headers
                connection: keep-alive
                content-type: application/json
                date: Sun,04 Sep 2022 06:49:32 GMT
                keep-alive: timeout=60
transfer-encoding: chunked
```



Finally put.

```
PUT
             /tub/{tub_id}
Parameters
                              Description
Name
tub_id * required
                                t989
string
(path)
can_quantity * required
                                4
string
(query)
                                                         Execute
Responses
Curl
 curl -X 'PUT' \
   'http://127.0.0.1:8080/tub/t989?can_quantity=4' \
-H 'accept: */*'
Request URL
 http://127.0.0.1:8080/tub/t989?can_quantity=4
Server response
Code
               Details
200
               Response body
                  "tub_id": "t989",
"tub_name": "t404"
               Response headers
                  connection: keep-alive
                 content-type: application/json
date: Sun,04 Sep 2022 06:56:13 GMT
keep-alive: timeout=60
                  transfer-encoding: chunked
Responses
Code
               Description
200
               OK
```

GitHub link: https://github.com/juliecurran3/Beer-Can-Inventory-Final-Project.git