ADVANCED SOFTWARE ENGINEERING - HOMEWORK 1

Student: Orlando Leombruni - Matricola: 475727

1 - APPLICATION SOURCE CODE, DOCUMENTATION AND DEPLOYMENT

The source code can be found at https://github.com/leombro/ase-fall-17/tree/master/homework1. Methods are adequately commented and a simple Javadoc is available in the /doc/ subfolder, both on GitHub and in the Moodle submission. Moreover, in the main folder of the submission there's a Postman collection (in JSON format) of tests for the various functionalities of the app (see section 3).

The application has been deployed to the Heroku service at the following link: https://leombro-ase-lab-17-hw1.herokuapp.com.

2 - IMPLEMENTATION CHOICES

I decided to treat the two classes provided by the teachers (DOODLE.JAVA and VOTE.JAVA) as an "external library", without changing their source code. Through testing on a "basic" version of my implementation and on the https://toydoodle.herokuapp.com app, I noticed some incorrect behavior; here's a list of them, together with my proposed fixes.

- 1. When performing PUT /doodles/ with a JSON body request that contains only the title field, the app throws a NullPointerException instead of ignoring the request gracefully. 1
 - **Solution.** Check that the Spring-generated Doodle object has both the title and options fields non-null.
- 2. The PUT /doodles/{id}/vote operation can modify an already-existing vote. (It shouldn't.) **Solution.** Check that the name indicated in the JSON body request hasn't voted yet.
- 3. When performing PUT /doodles/{id}/vote with a JSON body request that doesn't contain the name field, the app inserts a vote anyway (with name = null).²
 - **Solution.** Check that the name field is present in the JSON body request.
- 4. The POST /doodles/{id}/vote/{name} operation can insert a new vote. (It shouldn't.)
 - **Solution.** Check that the name indicated in the JSON body request has already voted.
- 5. A "malicious" user can update another user's vote by means of an incorrect JSON body request when performing the POST /doodles/{id}/vote/{name} operation.
 - Example: user "john" modifies the vote of another user "jack" by sending the JSON request { 'name': 'jack', 'option': '...' } in a POST /doodles/{id}/vote/john operation.
 - **Solution.** Check that both the JSON's and Vote's name fields coincide.

Of course, other good programming practices (ex. testing for null before invoking an object's method) have been adopted.

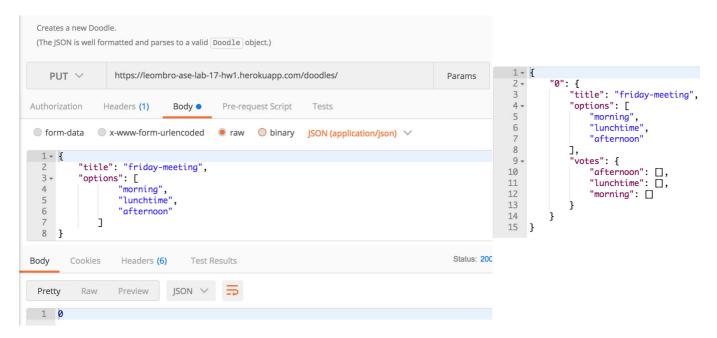
¹ This behavior is due to the fact that the constructors for the Doodle class iterate on the list of options in order to build the list of votes. The Doodle (Doodle doodle) constructor copies the list of options from doodle without testing for null, so in the abovementioned scenario it tries to iterate on a null object (DOODLE.JAVA, line 28).

² The addVote(Vote vote) method doesn't check wheter vote.getName() returns nu11 (Doodle.Java, line 49).

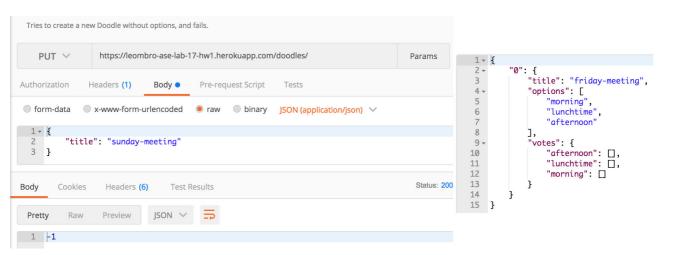
3 - TESTS

Here's a quick rundown of the app's functionalities. The screenshots include, where meaningful, the request and response bodies and, for all methods different from GET, the current status of the opened doodles.

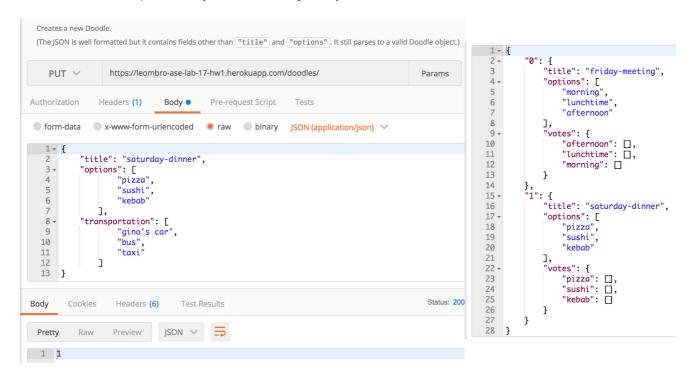
1. Correct PUT /doodles



2. Wrong PUT /doodles (no options field)



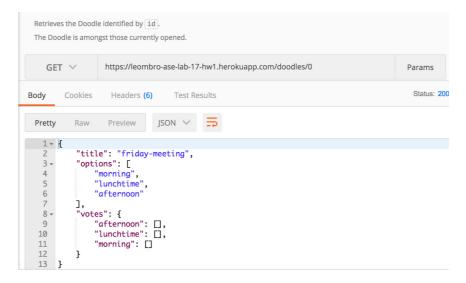
Correct PUT /doodles (with unnecessary fields)



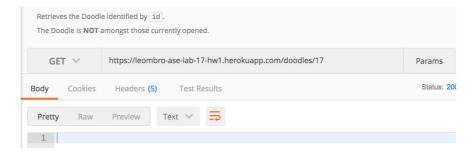
4. GET /doodles

```
Retrieve all opened Doodles.
                 https://leombro-ase-lab-17-hw1.herokuapp.com/doodles/
   GET Y
                                                                                          Params
Pretty
         Raw
                 Preview
                             JSON V
 1 - 1
  2 +
          "0": {
              "title": "friday-meeting",
 3
              "options": [
  4 +
                  "morning",
 5
                  "lunchtime",
  6
                  "afternoon"
 8
 9 +
              "votes": {
                   "afternoon": □,
10
                  "lunchtime": [],
11
                   "morning": []
12
13
14
15 -
              "title": "saturday-dinner",
16
              "options": [
17 -
                  "pizza",
18
                  "sushi",
"kebab"
19
20
21
              ],
22 -
              "votes": {
                   "pizza": □,
23
                  "sushi": [],
24
25
26
              }
27
          }
28 }
```

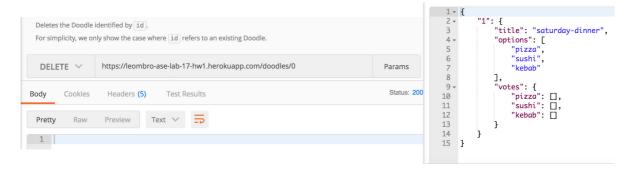
5. GET /doodles/{id} (existing Doodle)



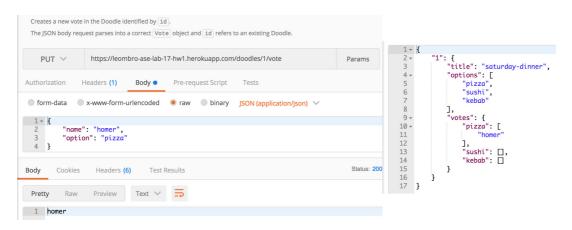
6. GET /doodles/{id} (non-existing Doodle)



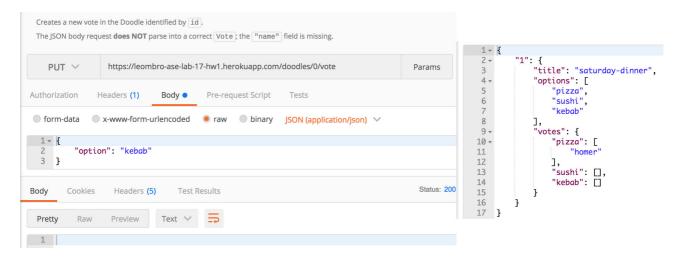
7. DELETE /doodles/{id}



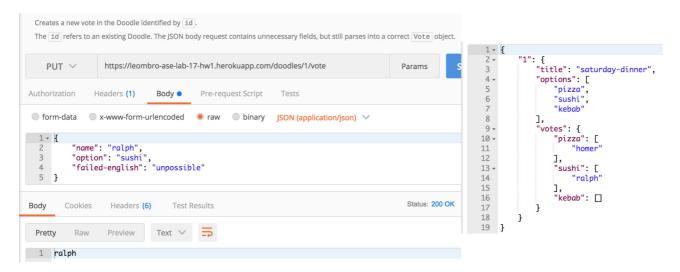
8. PUT /doodles/{id}/vote(new vote)



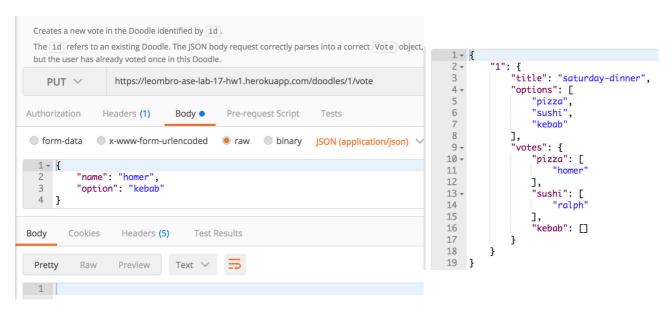
9. PUT /doodles/{id}/vote (invalid vote, "name" field missing)



10. PUT /doodles/{id}/vote (correct new vote with unnecessary fields)



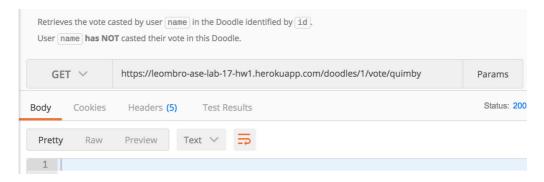
11. PUT /doodles/{id}/vote (user has already voted in that Doodle)



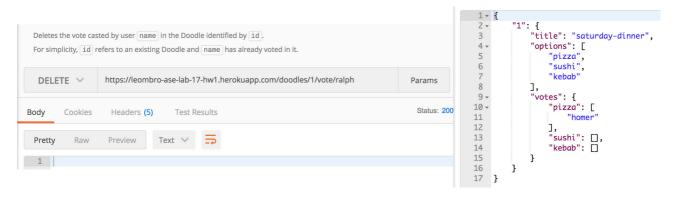
12. GET /doodles/{id}/vote/{name} (existing vote)



13. GET /doodles/{id}/vote/{name} (non-existing vote)



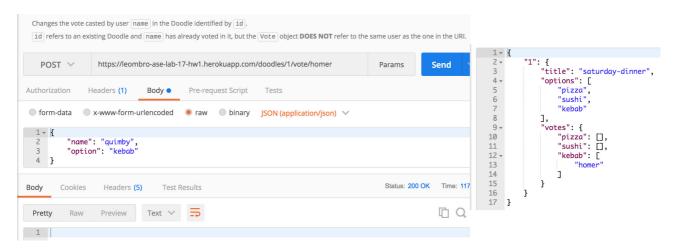
14. DELETE /doodles/{id}/vote/{name}



15. POST /doodles/{id}/vote/{name} (valid update)



16. POST /doodles/{id}/vote/{name} (wrong user)



17. POST /doodles/{id}/vote/{name} (new vote)

