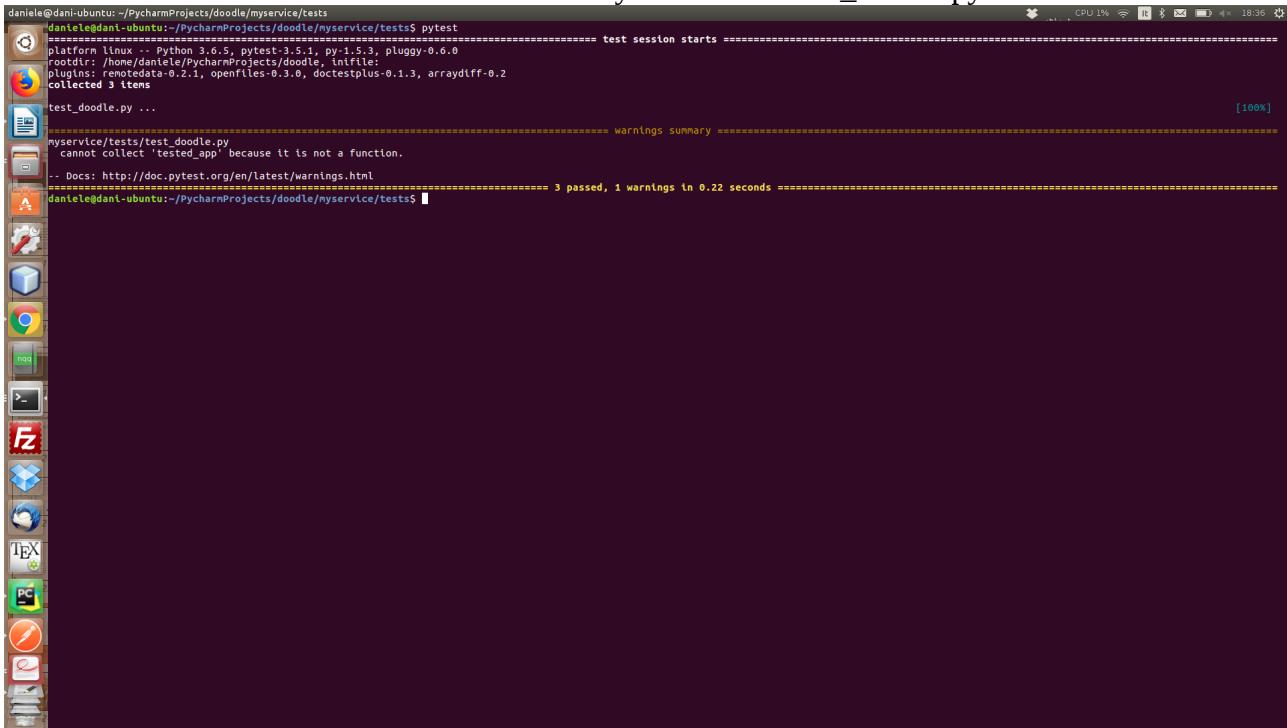


Advanced Software Engineering
University of Pisa
Homework 1
Daniele Gadler
564198

1. Link to the Github Repository:

<https://github.com/DanyEle/ase-lab-18/tree/master/hw-1/doodle>

2. Screenshot of successful execution of myservice/tests/test_doodle.py



```
daniele@dani-ubuntu:~/PycharmProjects/doodle/myservice/tests$ pytest
=====
platform linux -- Python 3.6.5, pytest-3.5.1, py-1.5.3, pluggy-0.6.0
rootdir: /home/daniele/PycharmProjects/doodle, inifile:
plugins: remotedata-0.2.1, openfiles-0.3.0, doctestplus-0.1.3, arraydiff-0.2
collected 3 items

test_doodle.py ...

myservice/tests/test_doodle.py
    cannot collect 'tested_app' because it is not a function.

-- Docs: http://doc.pytest.org/en/latest/warnings.html
=====
3 passed, 1 warnings in 0.22 seconds
daniele@dani-ubuntu:~/PycharmProjects/doodle/myservice/tests$
```

All tests have been passed, even though one warning has been generated.

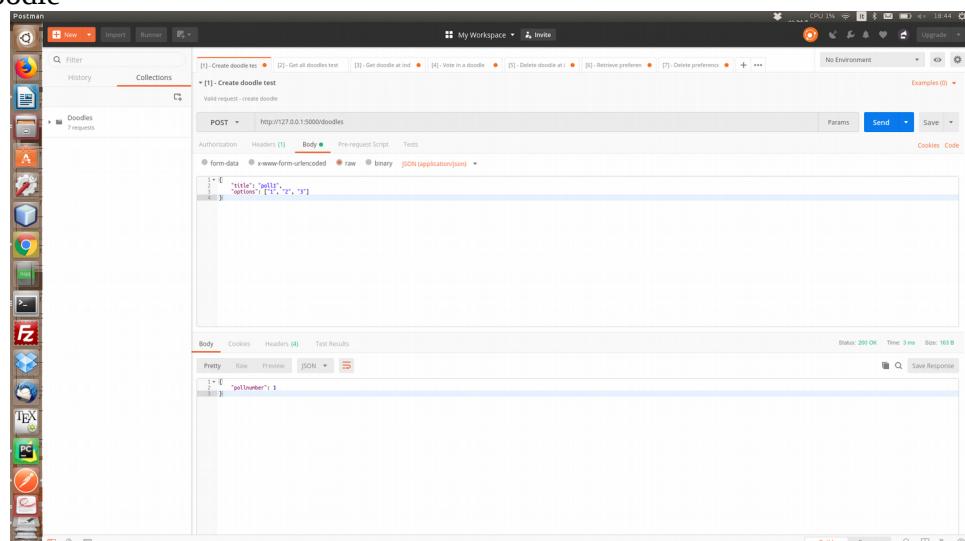
Further details:

In my code, I added a few more cases throwing (400) client errors in case the input provided does not fit the expected type (for example: if a string is expected, but a data structure that cannot be parsed as a string is passed).

3. Screenshots of the tests performed with Postman for all above operations:

[Test 1 – Create Doodle test and get doodle]

- 1) Create doodle



2)

The screenshot shows the Postman application interface. A collection named 'Doodles' is selected. A POST request is being made to `http://127.0.0.1:5000/doodles`. The request body is set to JSON and contains the following data:

```
{
  "title": "poll12",
  "options": ["1", "2"]
}
```

The response status is 200 OK, and the response body is:

```
{
  "pollnumber": 2
}
```

3)

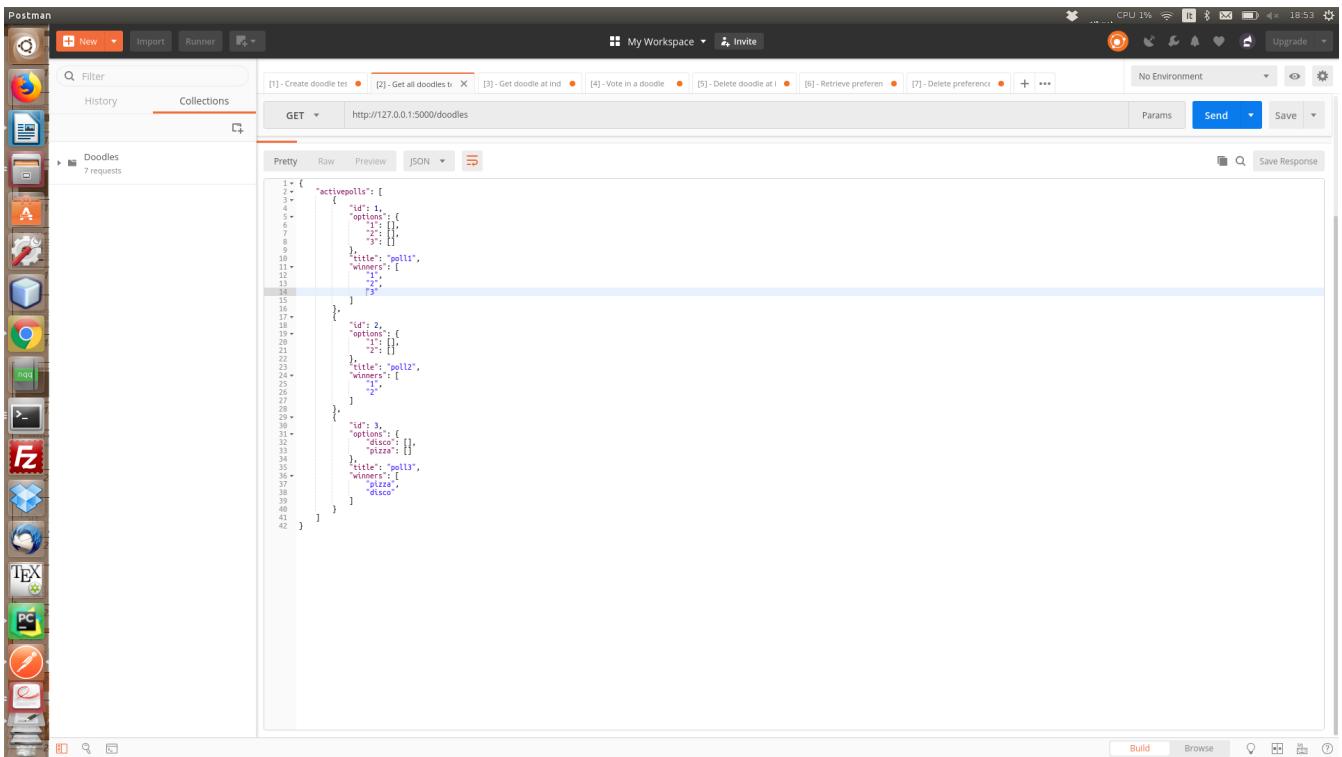
The screenshot shows the Postman application interface. A collection named 'Doodles' is selected. A POST request is being made to `http://127.0.0.1:5000/doodles`. The request body is set to JSON and contains the following data:

```
{
  "title": "poll3",
  "options": ["pizza", "disco"]
}
```

The response status is 200 OK, and the response body is:

```
{
  "pollnumber": 3
}
```

4) Get all doodles

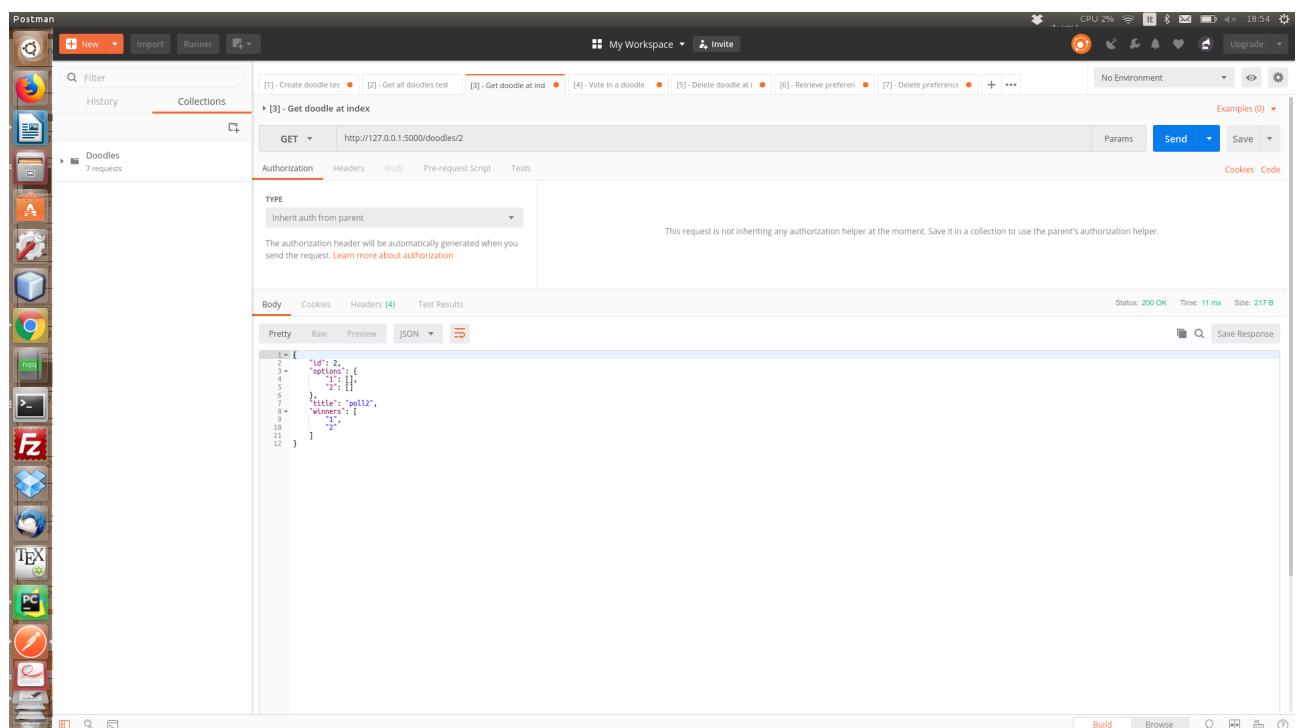


The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, there is a breadcrumb navigation: '[1] - Create doodle test' → '[2] - Get all doodles test' → '[3] - Get doodle at index' → '[4] - Vote in a doodle' → '[5] - Delete doodle at i' → '[6] - Retrieve preference' → '[7] - Delete preference'. The main workspace shows a 'Doodles' collection with 7 requests. A 'GET' request is selected, pointing to the URL `http://127.0.0.1:5000/doodles`. The response is displayed in 'Pretty' mode, showing a JSON array of three poll objects:

```
[{"id": 1, "options": [{"id": 1, "option": "1"}, {"id": 2, "option": "2"}, {"id": 3, "option": "3"}], "title": "poll1", "winners": [{"id": 1}, {"id": 2}, {"id": 3}]}, {"id": 2, "options": [{"id": 1, "option": "1"}, {"id": 2, "option": "2"}], "title": "poll2", "winners": [{"id": 1}, {"id": 2}]}, {"id": 3, "options": [{"id": 1, "option": "pizza"}, {"id": 2, "option": "bacon"}], "title": "poll3", "winners": [{"id": 1}, {"id": 2}]}]
```

[Test 2 – retrieve single doodle]

5) Get a single doodle



The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, there is a breadcrumb navigation: '[1] - Create doodle test' → '[2] - Get all doodles test' → '[3] - Get doodle at index' → '[4] - Vote in a doodle' → '[5] - Delete doodle at i' → '[6] - Retrieve preference' → '[7] - Delete preference'. The main workspace shows a 'Doodles' collection with 7 requests. A 'GET' request is selected, pointing to the URL `http://127.0.0.1:5000/doodles/2`. The response is displayed in 'Pretty' mode, showing a JSON object representing the second poll:

```
{"id": 2, "options": [{"id": 1, "option": "1"}, {"id": 2, "option": "2"}], "title": "poll2", "winners": [{"id": 1}, {"id": 2}]}
```

6)

The screenshot shows the Postman application interface. A collection named 'Doodles' is selected. A specific test case, '3] - Get doodle at index', is highlighted. The request method is 'GET' to the URL 'http://127.0.0.1:5000/doodles/12'. The 'Authorization' tab is active, showing 'Inherit auth from parent'. The response status is '404 NOT FOUND' with a message: 'The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.' The JSON response body is: { "code": 404, "description": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.", "message": "404 Not Found: The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again." }.

7) Test vote in a doodle

The screenshot shows the Postman application interface. A collection named 'Doodles' is selected. A specific test case, '4] - Vote in a doodle', is highlighted. The request method is 'PUT' to the URL 'http://127.0.0.1:5000/doodles/2'. The 'Body' tab is active, showing the JSON payload: { "person": "fred", "option": "1" }. The response status is '200 OK' with a message: '{"winners": ["1"]}'. The JSON response body is: { "winners": ["1"] }.

8)

The screenshot shows the Postman application interface. A collection named "Doodles" is selected. A request titled "[4] - Vote in a doodle" is displayed, using the PUT method on the URL `http://127.0.0.1:5000/doodles/1`. The "Body" tab is active, showing a JSON payload:

```
1 - {  
2 -   "person": "fred",  
3 -   "option": "2"  
4 - }
```

The response status is 200 OK, with a response body showing the updated doodle details:

```
1 - {  
2 -   "winners": [  
3 -     "1",  
4 -     "2"  
5 -   ]  
6 - }
```

9)

The screenshot shows the Postman application interface. A collection named "Doodles" is selected. A request titled "[4] - Vote in a doodle" is displayed, using the PUT method on the URL `http://127.0.0.1:5000/doodles/2`. The "Body" tab is active, showing a JSON payload:

```
1 - {  
2 -   "person": "fred",  
3 -   "option": "2"  
4 - }
```

The response status is 200 OK, with a response body showing the updated doodle details:

```
1 - {  
2 -   "winners": [  
3 -     "1",  
4 -     "2"  
5 -   ]  
6 - }
```

10)

The screenshot shows the Postman application interface. In the center, there is a request panel titled 'PUT' with the URL 'http://127.0.0.1:5000/doodles/2'. The 'Body' tab is selected, showing a JSON payload:

```
{ "person": "barney", "option": "I" }
```

Below the request, the response section displays a status of '200 OK' with a time of '10 ms' and a size of '164 B'. The response body is shown in JSON format:

```
{ "wishes": [ "I" ] }
```

11) Same person voting already voted option from themselves

The screenshot shows the Postman application interface. In the center, there is a request panel titled 'PUT' with the URL 'http://127.0.0.1:5000/doodles/2'. The 'Body' tab is selected, showing a JSON payload:

```
{ "person": "barney", "option": "I" }
```

Below the request, the response section displays a status of '400 BAD REQUEST' with a time of '8 ms' and a size of '368 B'. The response body is shown in JSON format:

```
{ "code": 400, "description": "The browser (or proxy) sent a request that this server could not understand.", "message": "400 Bad Request: The browser (or proxy) sent a request that this server could not understand." }
```

12) Vote non-existing option

The screenshot shows the Postman application interface. A PUT request is being made to `http://127.0.0.1:5000/doodles/2`. The request body is set to `JSON (application/json)` and contains the following JSON:

```
1 - {  
2 -   "person": "wimla",  
3 -   "option": "g"  
4 - }
```

The response status is `400 BAD REQUEST` with a message: `"code": 400, "description": "The browser (or proxy) sent a request that this server could not understand.", "message": "400 Bad Request: The browser (or proxy) sent a request that this server could not understand."`

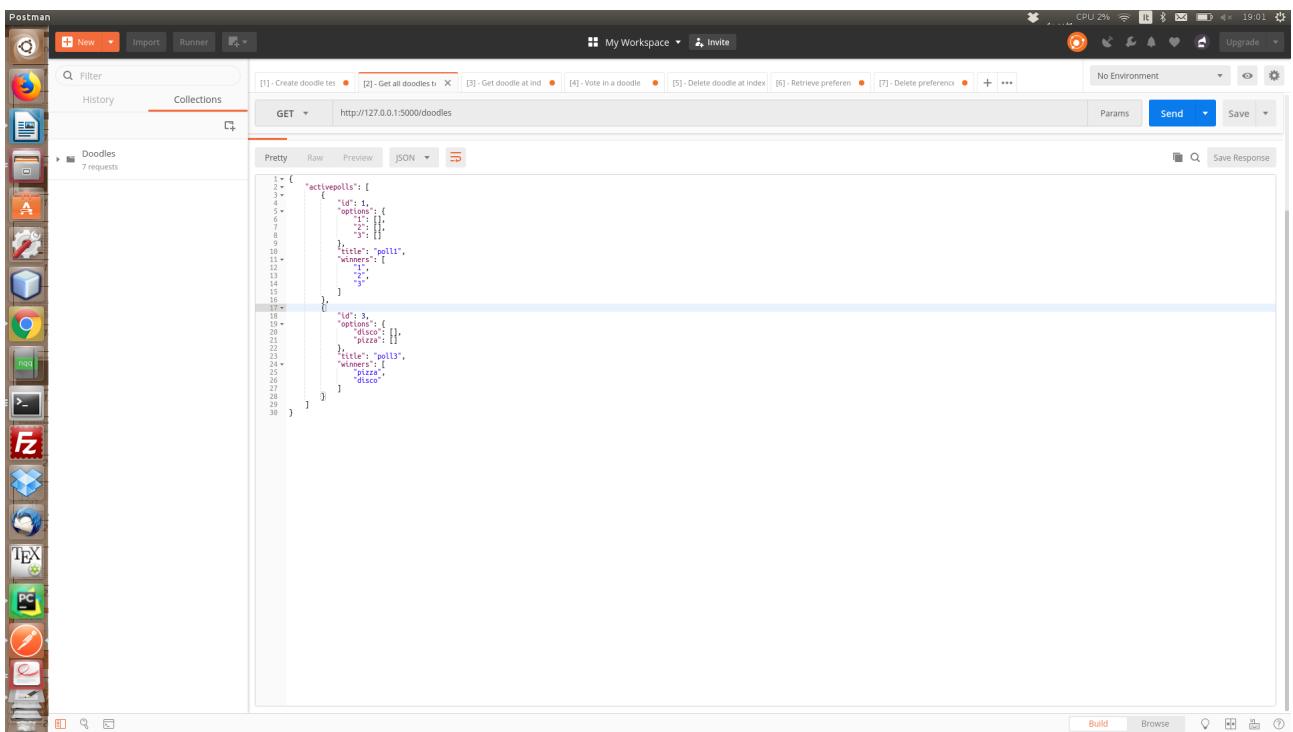
13) Delete a doodle

The screenshot shows the Postman application interface. A DELETE request is being made to `http://127.0.0.1:5000/doodles/2`. The request body is set to `JSON (application/json)` and contains the following JSON:

```
1 - {  
2 -   "wimla": [  
3 -     "1"  
4 -   ]  
5 - }
```

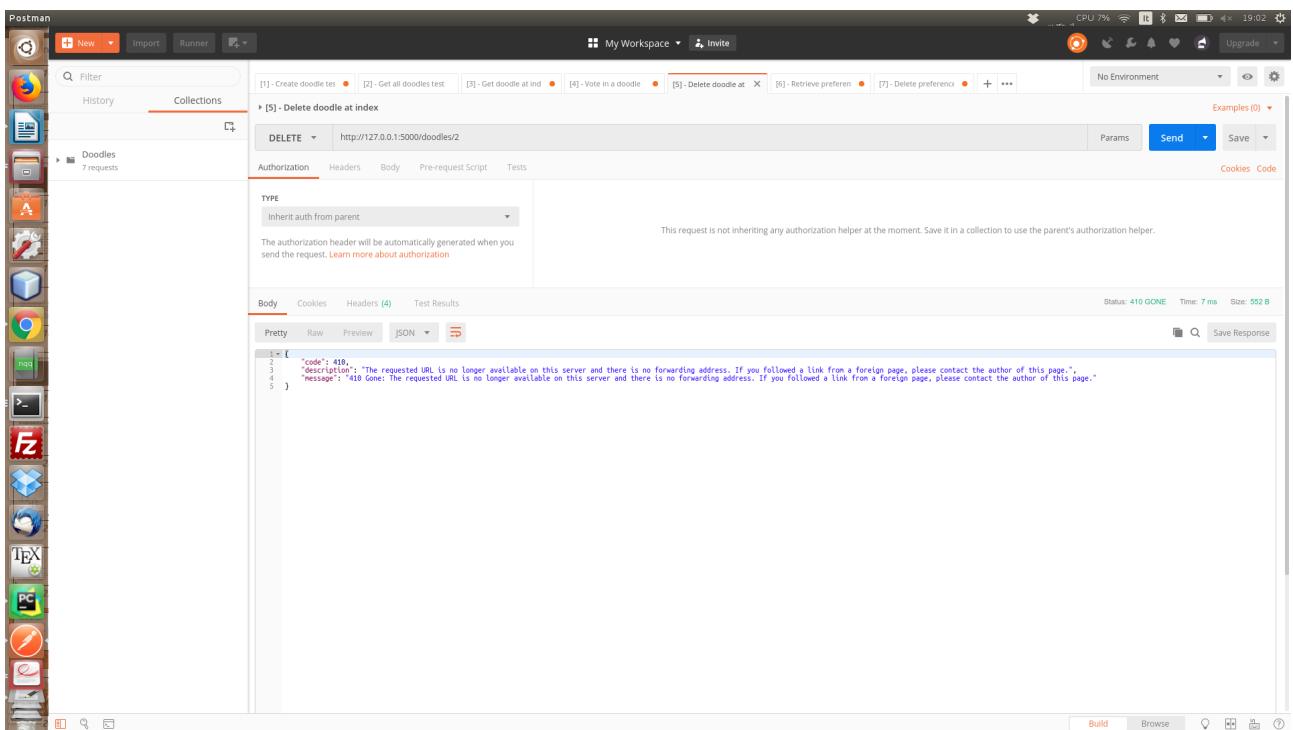
The response status is `200 OK`.

14) Get all existing doodles



```
[{"id": 1, "title": "polli", "winners": ["polli"], "options": [{"id": 1, "text": "polli"}, {"id": 2, "text": "disco"}]}, {"id": 3, "title": "polli", "winners": ["disco"], "options": [{"id": 1, "text": "polli"}, {"id": 2, "text": "disco"}]}]
```

15) Delete previous doodle



```
{"code": 410, "description": "The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page.", "message": "410 Gone: The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page."}
```

16) Delete non-existing doodle

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, there is a search bar labeled 'Filter' and a dropdown menu for 'History' and 'Collections'. A sidebar on the left contains icons for various tools and services, including a browser icon, a file icon, and a database icon. The main workspace shows a collection named 'Doodles' with 7 requests. A specific request is selected for deletion, indicated by a red border around the 'DELETE' button and the URL 'http://127.0.0.1:5000/doodles/12'. The 'Authorization' tab is active, showing 'Inherit auth from parent'. The response body is displayed in JSON format, showing an error message: { "code": 404, "description": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.", "message": "404 Not Found: The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again." }. The status bar at the bottom indicates 'Status: 404 NOT FOUND'.

17) Get previously existing doodle

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, there is a search bar labeled 'Filter' and a dropdown menu for 'History' and 'Collections'. A sidebar on the left contains icons for various tools and services, including a browser icon, a file icon, and a database icon. The main workspace shows a collection named 'Doodles' with 7 requests. A specific request is selected for retrieval, indicated by a red border around the 'GET' button and the URL 'http://127.0.0.1:5000/doodles/2'. The 'Authorization' tab is active, showing 'Inherit auth from parent'. The response body is displayed in JSON format, showing an error message: { "code": 410, "description": "The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page.", "message": "410 Gone: The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page." }. The status bar at the bottom indicates 'Status: 410 GONE'.

[Test 3]

18) Vote in a doodle

The screenshot shows the Postman application interface. A PUT request is being made to `http://127.0.0.1:5000/doodles/1`. The request body is set to `JSON (application/json)` and contains the following JSON:

```
1 - { "person": "fred", "option": "1" }
```

The response status is `200 OK`, time `8 ms`, and size `164 B`. The response body is:

```
1 - { "winners": [ "1" ] }
```

19)

The screenshot shows the Postman application interface. A PUT request is being made to `http://127.0.0.1:5000/doodles/1`. The request body is set to `JSON (application/json)` and contains the following JSON:

```
1 - { "person": "fred", "option": "2" }
```

The response status is `200 OK`, time `14 ms`, and size `168 B`. The response body is:

```
1 - { "winners": [ "2" ] }
```

20)

The screenshot shows the Postman application interface. A PUT request is being made to `http://127.0.0.1:5000/doodles/1`. The request body is set to `JSON (application/json)` and contains the following JSON payload:

```
1: { "person": "barney",
2:   "option": "1"
3: }
```

The response status is `200 OK`, time `10 ms`, and size `164 B`. The response body is:

```
1: {
2:   "winners": [
3:     "1"
4:   ]
5: }
```

21)Get votes from a person that voted

The screenshot shows the Postman application interface. A GET request is being made to `http://127.0.0.1:5000/doodles/1/fred`. The request has an `Authorization` header set to `Inherit auth from parent`. A note states: `This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper.` The response status is `200 OK`, time `7 ms`, and size `173 B`. The response body is:

```
1: {
2:   "votedoptions": [
3:     "1"
4:   ]
5: }
```

22) Get votes from a person that has not yet voted

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a 'Doodles' folder containing 7 requests. The main workspace displays a request for '6 - Retrieve preferences of a person in poll id'. The method is set to 'GET' and the URL is 'http://127.0.0.1:5000/doodles/1/wilma'. The 'Authorization' tab shows 'Inherit auth from parent'. The response body is shown in JSON format:

```
{ "votedoptions": [] }
```

. The status bar at the bottom indicates 'Status: 200 OK'.

23) Delete votes from a person that voted

The screenshot shows the Postman application interface. In the left sidebar, under 'Collections', there is a 'Doodles' folder containing 7 requests. The main workspace displays a request for '7 - Delete preferences expressed by a person'. The method is set to 'DELETE' and the URL is 'http://127.0.0.1:5000/doodles/1/fred'. The 'Authorization' tab shows 'Inherit auth from parent'. The response body is shown in JSON format:

```
{ "removed": true }
```

. The status bar at the bottom indicates 'Status: 200 OK'.

24) Get votes from a person that was deleted

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and a search bar. Below the navigation is a toolbar with icons for 'Filter', 'History', 'Collections' (which is selected), 'Doodles' (with 7 requests), and other tools like 'Auth', 'Headers', 'Body', 'Pre-request Script', and 'Tests'. The main workspace shows a list of items: [1] - Create doodle test, [2] - Get all doodles test, [3] - Get doodle at index, [4] - Vote in a doodle, [5] - Delete doodle at index, [6] - Retrieve preference, [7] - Delete preference, and a '+' button. A specific item '[6] - Retrieve preferences of a person in poll id' is selected. The request details section shows a 'GET' method and the URL 'http://127.0.0.1:5000/doodles/1/fred'. The 'Authorization' tab is selected, showing 'Inherit auth from parent'. The 'Body' tab shows the response body as an empty array:

```
[{"votedoptions": []}]
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 3 ms', and 'Size: 166 B'. There are also 'Send' and 'Save' buttons.

25) Delete votes from a person that did not vote

This screenshot shows a similar setup in Postman. The top navigation bar and toolbar are identical. The main workspace shows the same list of items. A specific item '[7] - Delete preferences expressed by a person' is selected. The request details section shows a 'DELETE' method and the URL 'http://127.0.0.1:5000/doodles/1/wilma'. The 'Authorization' tab is selected, showing 'Inherit auth from parent'. The 'Body' tab shows the response body as a single key-value pair:

```
{"removed": false}
```

. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 11 ms', and 'Size: 164 B'. There are also 'Send' and 'Save' buttons.

26) Retrieve votes from a non-existing poll

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, a list of requests is shown: [1] - Create doodle test, [2] - Get all doodles test, [3] - Get doodle at id, [4] - Vote in a doodle, [5] - Delete doodle at, [6] - Retrieve preference, [7] - Delete preference, and [8] - ...

The main area displays a single request: **GET** http://127.0.0.1:5000/doodles/13/fred. The 'Authorization' tab is selected. A note states: "This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper." The response status is 404 NOT FOUND, with a time of 11 ms and a size of 454 B.

The response body is JSON:

```
[{"code": 404, "description": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.", "message": "404 Not Found: The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again."}]
```

27) Get votes from a previously existing poll

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for 'New', 'Import', 'Runner', and 'Collections'. The 'Collections' tab is selected. Below the tabs, a list of requests is shown: [1] - Create doodle test, [2] - Get all doodles test, [3] - Get doodle at id, [4] - Vote in a doodle, [5] - Delete doodle at, [6] - Retrieve preference, [7] - Delete preference, and [8] - ...

The main area displays a single request: **GET** http://127.0.0.1:5000/doodles/2/fred. The 'Authorization' tab is selected. A note states: "This request is not inheriting any authorization helper at the moment. Save it in a collection to use the parent's authorization helper." The response status is 410 GONE, with a time of 8 ms and a size of 502 B.

The response body is JSON:

```
[{"code": 410, "description": "The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page.", "message": "410 Gone: The requested URL is no longer available on this server and there is no forwarding address. If you followed a link from a foreign page, please contact the author of this page."}]
```

[4 – Extra tests]

28) Invalid data structure provided as 'person'

The screenshot shows the Postman application interface. A PUT request is being made to `http://127.0.0.1:5000/doodles/1`. The request body is set to `JSON (application/json)` and contains the following invalid JSON:`1 - {
2 | "person": "[\"test\"]",
3 | "option": "1"
4 }`The response status is `400 BAD REQUEST`, with a message indicating the browser sent a request that the server could not understand.

29) Invalid data structure provided while creating the doodle:

The screenshot shows the Postman application interface. A POST request is being made to `http://127.0.0.1:5000/doodles`. The request body is set to `JSON (application/json)` and contains the following invalid JSON:`1 - {
2 | "title": "polls",
3 | "options": "pizza"
4 }`The response status is `400 BAD REQUEST`, with a message indicating the browser sent a request that the server could not understand.