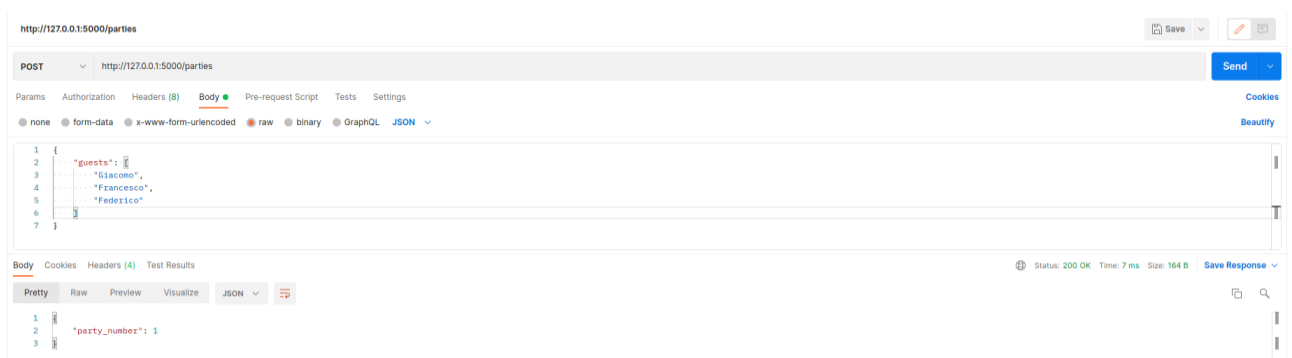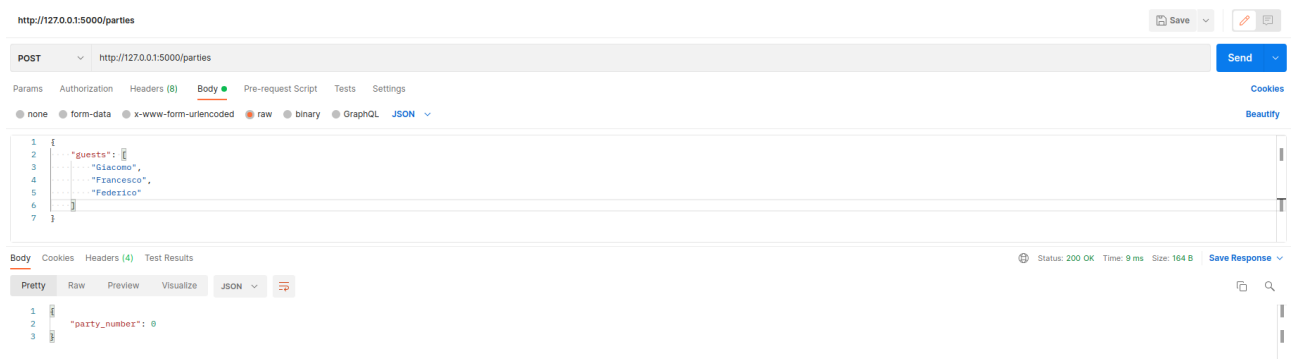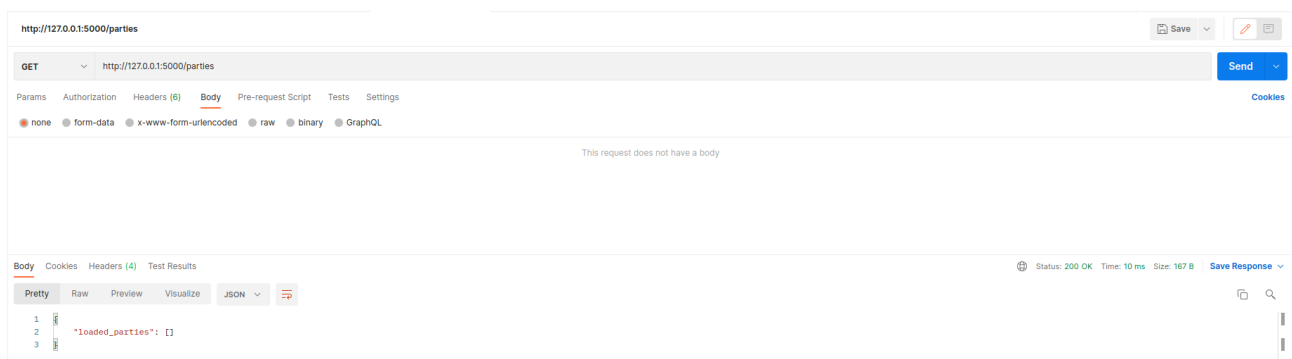I followed the instruction given creating a virtual environment and installing required modules.

After completing the decorators with the allowed HTTP methods according to the table provided, I implemented all the methods.

(Screenshots are small to catch all the information given by Postman ever for non-pytested methods)

all_parties:

- If the method is **POST** then a try-except block is opened and the provided method **create_party** passing the request.
    - If the content is bad formatted (no guests) an exception is raised and error code 400 is returned.
- If the method is **GET** the result of **get_all_parties** is returned (already jsonified by the provided function).

```json
1  {
2      "guests": [
3          "Giacomo",
4          "Francesco",
5          "Federico"
6      ]
7  }
```

```json
1  {
2      "party_number": 2
3  }
```

```json
1  {
2      "loaded_parties": [
3          {
4              "foodlist": [],
5              "guests": [
6                  "Giacomo",
7                  "Francesco",
8                  "Federico"
9              ],
10             "id": 0
11         },
12         {
13             "foodlist": [],
14             "guests": [
15                 "Giacomo",
16                 "Francesco",
17                 "Federico"
18             ],
19             "id": 1
20         },
21         {
22             "foodlist": [],
23             "guests": [
24                 "Giacomo",
25                 "Francesco",
```

```json
7                  "Francesco",
8                  "Federico"
9              ],
10             "id": 0
11         },
12         {
13             "foodlist": [],
14             "guests": [
15                 "Giacomo",
16                 "Francesco",
17                 "Federico"
18             ],
19             "id": 1
20         },
21         {
22             "foodlist": [],
23             "guests": [
24                 "Giacomo",
25                 "Francesco",
26                 "Federico"
27             ],
28             "id": 2
29         }
30     ]
31  }
```

```
http://127.0.0.1:5000/parties

POST    http://127.0.0.1:5000/parties                                          Send

Params  Authorization  Headers (8)  Body ●  Pre-request Script  Tests  Settings    Cookies

none  form-data  x-www-form-urlencoded  ● raw  binary  GraphQL  JSON ∨           Beautify

1  {
2      "guests": []
3  }

Body  Cookies  Headers (4)  Test Results    Status: 400 BAD REQUEST  Time: 21 ms  Size: 173 B   Save Response

Pretty  Raw  Preview  Visualize  HTML ∨

1  Empty party
```
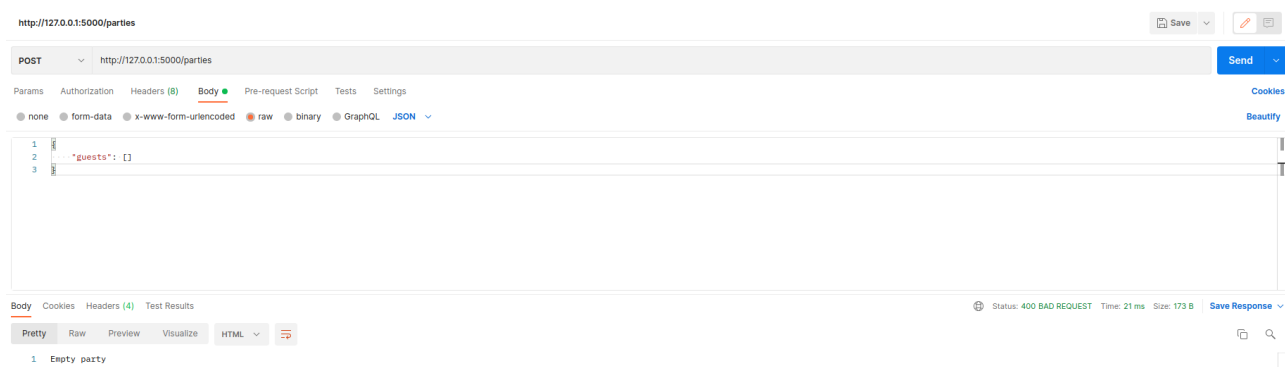
loaded_parties:

- Return the length of the dictionary containing the parties incapsulated in JSON with key **loaded_parties**.


```
http://127.0.0.1:5000/parties/loaded

GET    http://127.0.0.1:5000/parties/loaded                                     Send

Params  Authorization  Headers (6)  Body  Pre-request Script  Tests  Settings    Cookies

● none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

                        This request does not have a body

Body  Cookies  Headers (4)  Test Results    Status: 200 OK  Time: 8 ms  Size: 166 B   Save Response

Pretty  Raw  Preview  Visualize  JSON ∨

1  {
2      "loaded_parties": 3
3  }
```
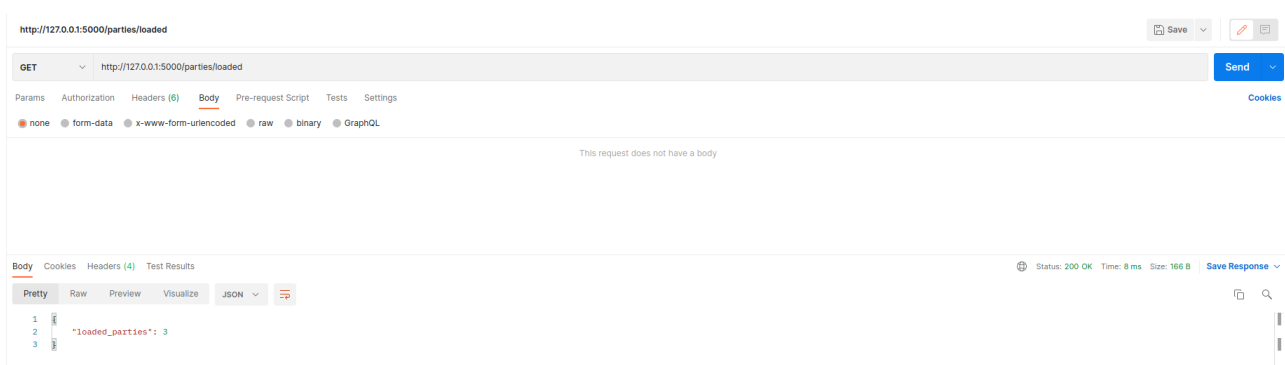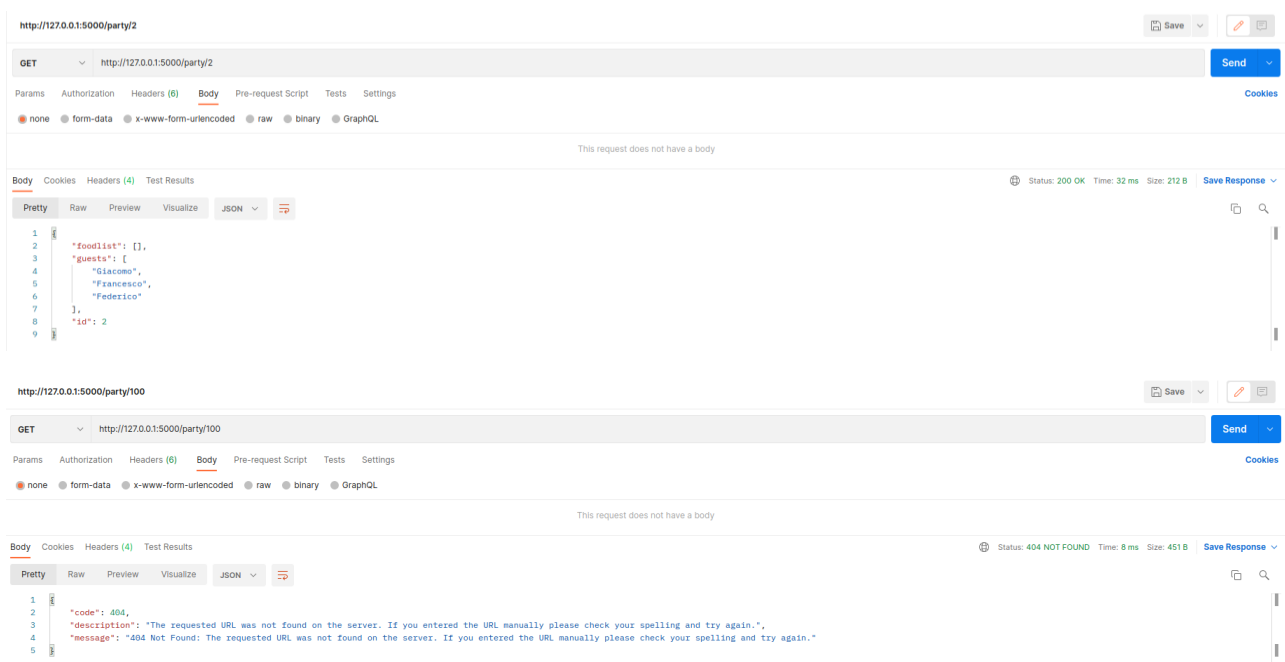
single_party:

- Check whether the party exists.
- If the method is **GET** return the jsonified serialization of the party with **id** (taken from the URL).
- If the method is **DELETE** return the result of my utility function **delete_party** (created for the modularity of the code) called with the **id** (taken from the URL).


```
http://127.0.0.1:5000/party/2

GET    http://127.0.0.1:5000/party/2                                            Send

Params  Authorization  Headers (6)  Body  Pre-request Script  Tests  Settings    Cookies

● none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

                        This request does not have a body

Body  Cookies  Headers (4)  Test Results    Status: 200 OK  Time: 32 ms  Size: 212 B   Save Response

Pretty  Raw  Preview  Visualize  JSON ∨

1  {
2      "foodlist": [],
3      "guests": [
4          "Giacomo",
5          "Francesco",
6          "Federico"
7      ],
8      "id": 2
9  }
```



```
http://127.0.0.1:5000/party/100

GET    http://127.0.0.1:5000/party/100                                          Send

Params  Authorization  Headers (6)  Body  Pre-request Script  Tests  Settings    Cookies

● none  form-data  x-www-form-urlencoded  raw  binary  GraphQL

                        This request does not have a body

Body  Cookies  Headers (4)  Test Results    Status: 404 NOT FOUND  Time: 8 ms  Size: 451 B   Save Response

Pretty  Raw  Preview  Visualize  JSON ∨

1  {
2      "code": 404,
3      "description": "The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.",
4      "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."
5  }
```

http://127.0.0.1:5000/party/0

Save

DELETE   http://127.0.0.1:5000/party/0   Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings   Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (4) Test Results   Status: 200 OK Time: 18 ms Size: 156 B Save Response

Pretty Raw Preview Visualize   HTML

1   Done

http://127.0.0.1:5000/party/0

Save

GET   http://127.0.0.1:5000/party/0   Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings   Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (4) Test Results   Status: 410 GONE Time: 24 ms Size: 551 B Save Response

Pretty Raw Preview Visualize   JSON

1 {
2   "code": 410,
3   "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.",
4   "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."
5 }

get_foodlist:

- Check whether the party exists.
- Return the serialization jsonified of the food list of the party **id** (taken from the URL).

http://127.0.0.1:5000/party/2/foodlist

Save

GET   http://127.0.0.1:5000/party/2/foodlist   Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings   Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON   Beautify
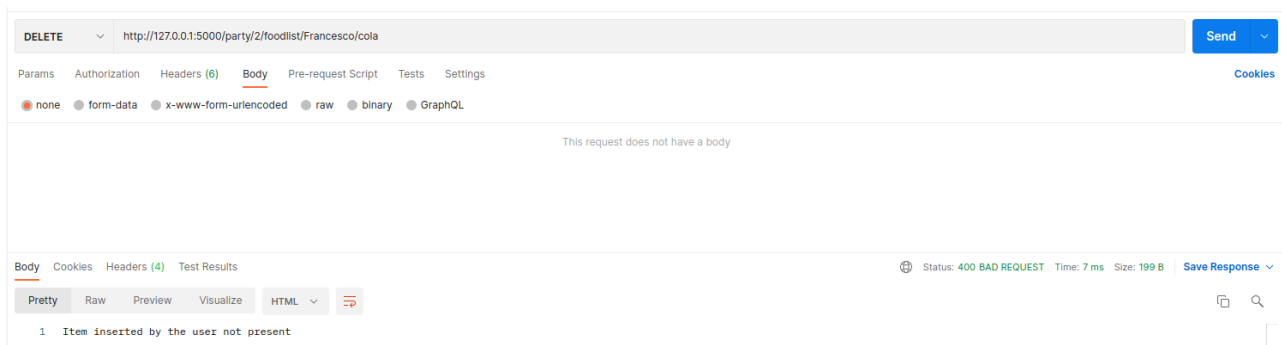
1

Body Cookies Headers (4) Test Results   Status: 200 OK Time: 8 ms Size: 196 B Save Response

Pretty Raw Preview Visualize   JSON

1 {
2   "foodlist": [
3   {
4   "food": "pizza",
5   "user": "Francesco"
6   }
7   ]
8 }

edit_foodlist:

- Check whether the party exists.
- If the method is POST:
    - In a try-except block **add_to_food_list** is called with item and user taken from the URL, then the jsonification of them is returned.
    - If the method calls raise an exception an according error code is returned instead.
- If the method is DELETE:

- In a try-except block **remove_from_food_list** is called on the party **id** (taken from the URL) and return the jsonified message.
- If the food is not existing an error code (400) is returned instead.

http://127.0.0.1:5000/party/2/foodlist/Francesco/pizza

```
1  {
2      "food": "pizza",
3      "user": "Francesco"
4  }
```

Status: 200 OK   Time: 6 ms   Size: 181 B

POST   http://127.0.0.1:5000/party/2/foodlist/Francesco/pizza

This request does not have a body

Status: 400 BAD REQUEST   Time: 22 ms   Size: 195 B

```
1  Item already inserted by the user
```


http://127.0.0.1:5000/party/2/foodlist/Stefano/pizza

POST   http://127.0.0.1:5000/party/2/foodlist/Stefano/pizza

Status: 401 UNAUTHORIZED   Time: 6 ms   Size: 180 B

```
1  Not invited guest
```


http://127.0.0.1:5000/party/2/foodlist/Francesco/pizza

DELETE   http://127.0.0.1:5000/party/2/foodlist/Francesco/pizza

Status: 200 OK   Time: 15 ms   Size: 169 B

```
1  {
2      "msg": "Food deleted!"
3  }
```

Both tests passed.

https://github.com/TommasoLencioni/ase-21/tree/main/Assignments/ase-assignment1-master