

UNIVERSITÀ DI PISA

CORSO DI LAUREA MAGISTRALE IN CYBER SECURITY

Bedrock a Party

Fabio LANUBILE

ANNO ACCADEMICO 2020-2021

Indice

1	Introduzione	2
2	Test con Postman	3
2.1	Get party vuoti	3
2.2	Creazione party	3
2.3	Get party caricati	4
2.4	Get parties	4
2.5	Creazione party senza invitati	5
2.6	Dettagli party con id 2	5
2.7	Dettagli party con id 100	6
2.8	Caricamento cibo	6
2.9	Caricamento cibo da un utente non invitato	7
2.10	Lista cibi	7
2.11	Eliminazione cibo	8
3	Test effettuato con pytest	8

1 Introduzione

Bedrock a party è un microservizio RESTful ideato per organizzare una festa e gestire quali invitati si occuperanno di portare bibite e/o cibi.

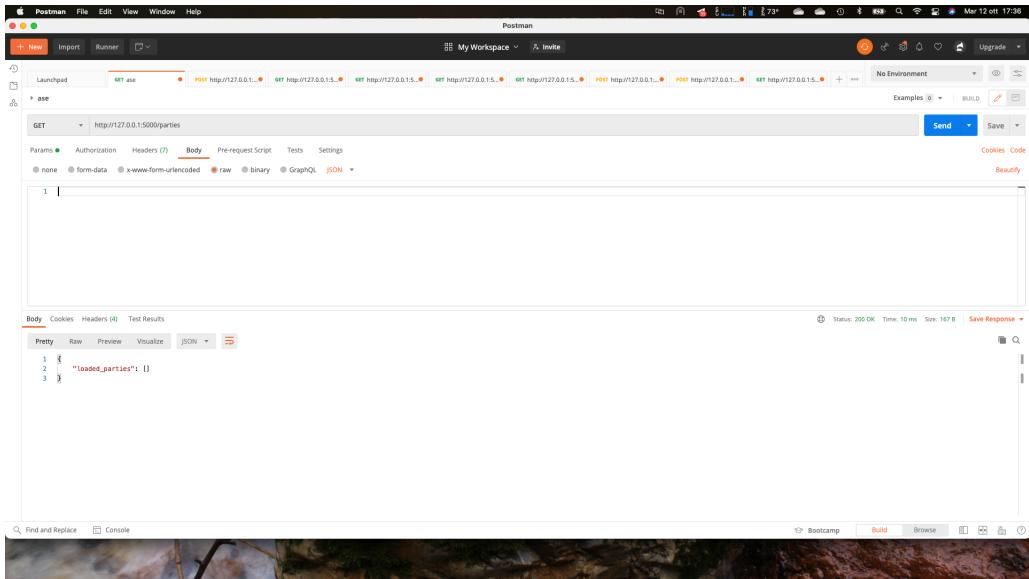
Il link alla repo github:

https://github.com/boosterino/exercise_ase_1

Il microservizio offre alcune rotte che consentono la creazione di party, l'eliminazione e l'aggiunta di invitati. Tutte queste operazioni sono possibili tramite i metodi HTTP **GET**, **POST** e **DELETE**.

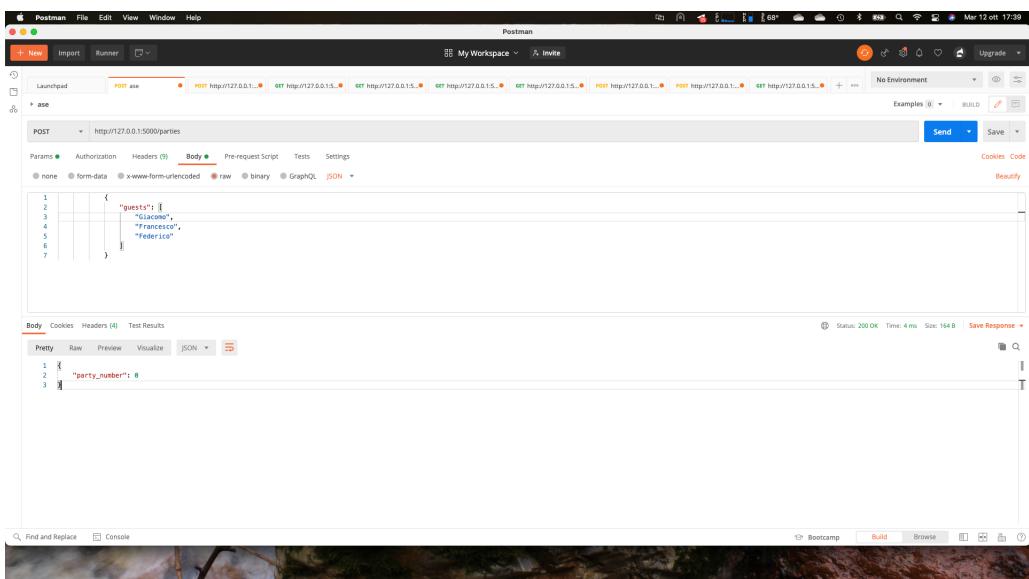
2 Test con Postman

2.1 Get party vuoti



Chiamata in **GET** alla rotta **/parties** che restituisce i party caricati, dunque restituisce un valore vuoto.

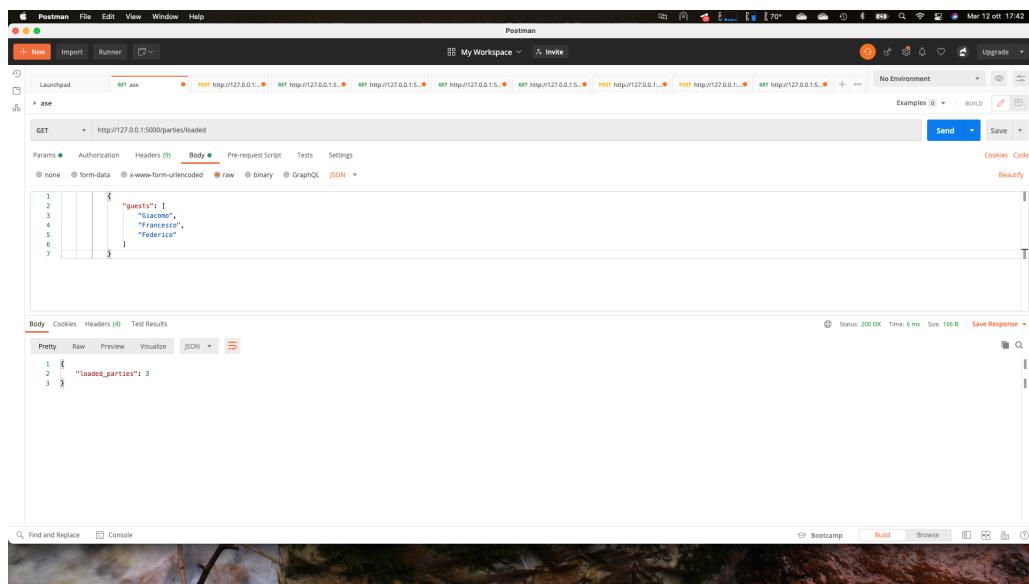
2.2 Creazione party



Chiamata in **POST** alla rotta **/parties** che crea un nuovo party con i valori passati tramite il body. Restituisce l'id del party.

Vengono caricati altri due party che avranno rispettivamente id 1-2.

2.3 Get party caricati



The screenshot shows the Postman application interface. A GET request is being made to the URL `http://127.0.0.1:5000/parties/loaded`. The request body is set to JSON and contains the following data:

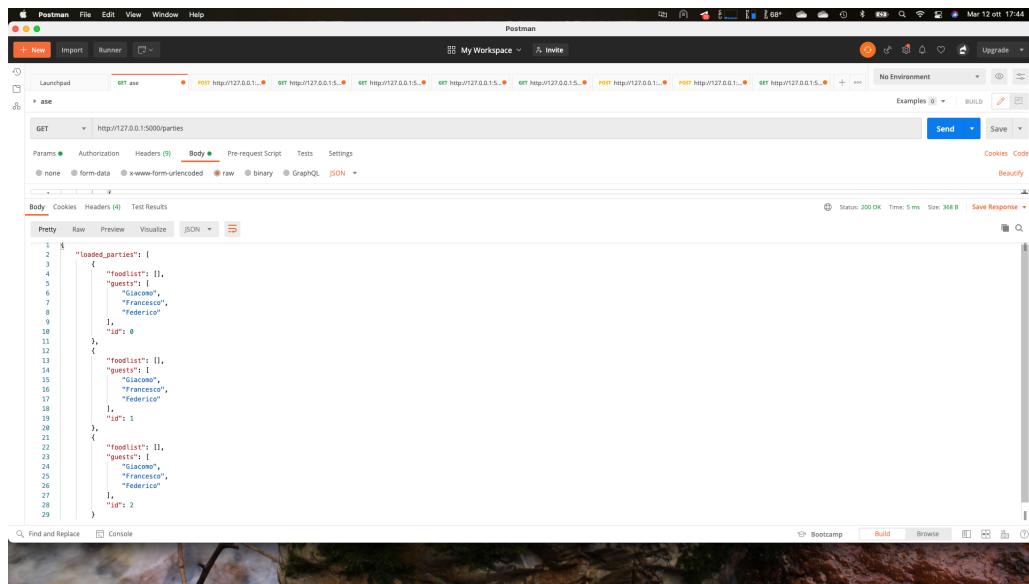
```
1  {
2     "guests": [
3         "Giacomo",
4         "Francesco",
5         "Federico"
6     ]
7 }
```

The response status is 200 OK, with a time of 6 ms and a size of 166 B. The response body is:

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

Chiamata in **GET** alla rottta `/parties/loaded` che mostra l'indice dell'ultimo party.

2.4 Get parties



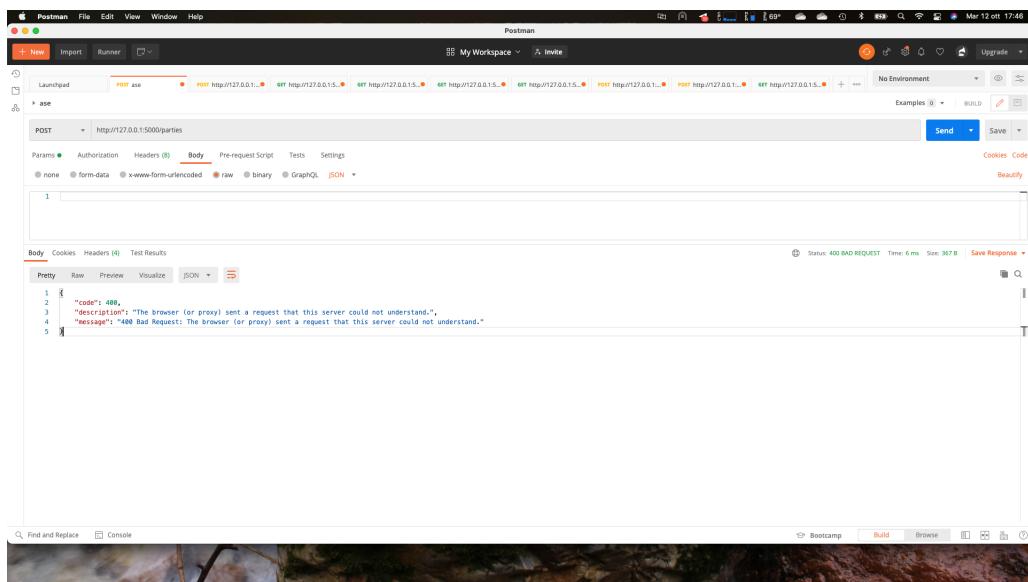
The screenshot shows the Postman application interface. A GET request is being made to the URL `http://127.0.0.1:5000/parties`. The request body is set to JSON and contains the following data:

```
1  {
2     "loaded_parties": [
3         {
4             "roolList": [],
5             "guests": [
6                 "Giacomo",
7                 "Francesco",
8                 "Federico"
9             ],
10            "id": 0
11        },
12        {
13            "roolList": [],
14            "guests": [
15                "Giacomo",
16                "Francesco",
17                "Federico"
18            ],
19            "id": 1
20        },
21        {
22            "roolList": [],
23            "guests": [
24                "Giacomo",
25                "Francesco",
26                "Federico"
27            ],
28            "id": 2
29        }
30    ]
31 }
```

The response status is 200 OK, with a time of 5 ms and a size of 368 B. The response body is identical to the request body.

Chiamata in **GET** alla rottta `/parties` che mostra i dettagli dei party caricati.

2.5 Creazione party senza invitati

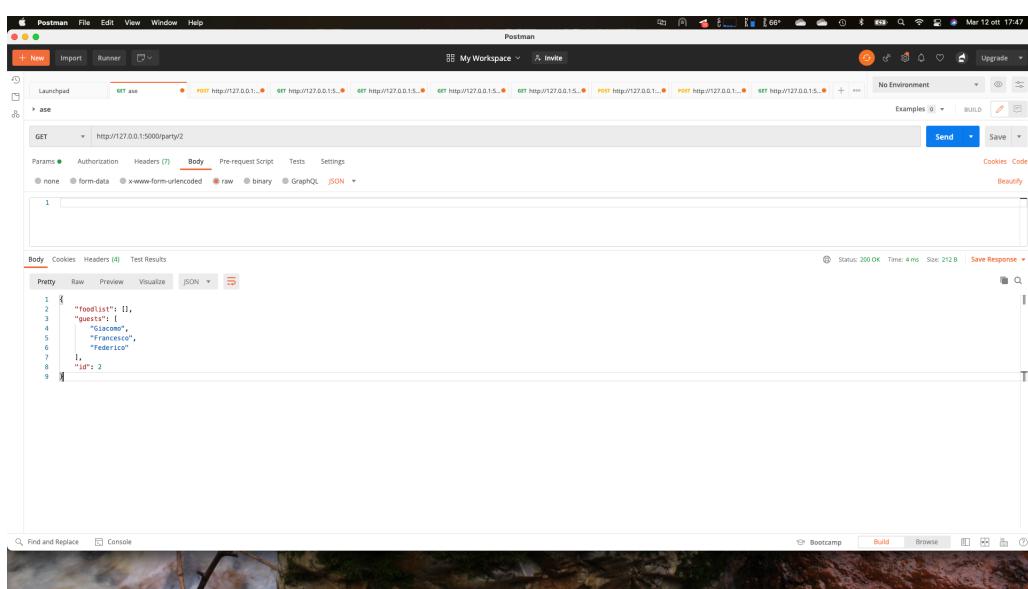


The screenshot shows the Postman application interface. A POST request is being made to `http://127.0.0.1:5000/parties`. The response status is 400 BAD REQUEST, with a message indicating an invalid request. The response body is:

```
{  
  "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."  
}
```

Chiamata in **POST** alla rotta **/parties** che prova a creare un party senza invitati dunque riceviamo un errore 400, come da specifica.

2.6 Dettagli party con id 2

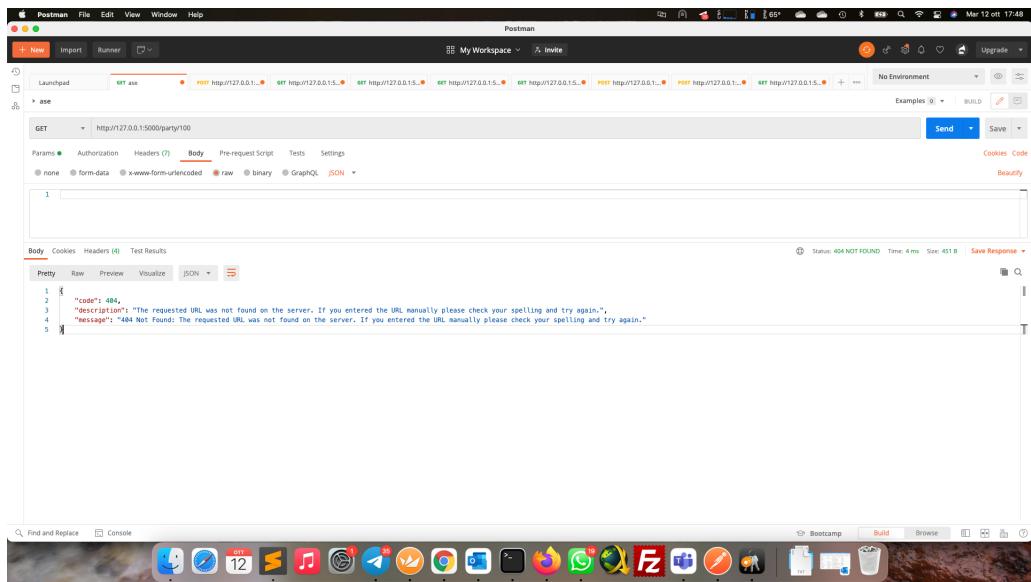


The screenshot shows the Postman application interface. A GET request is being made to `http://127.0.0.1:5000/party/2`. The response status is 200 OK, and the response body is:

```
{  
  "id": 2,  
  "invited": [],  
  "guests": [  
    "Giacomo",  
    "Francesco",  
    "Federico"  
  ]  
}
```

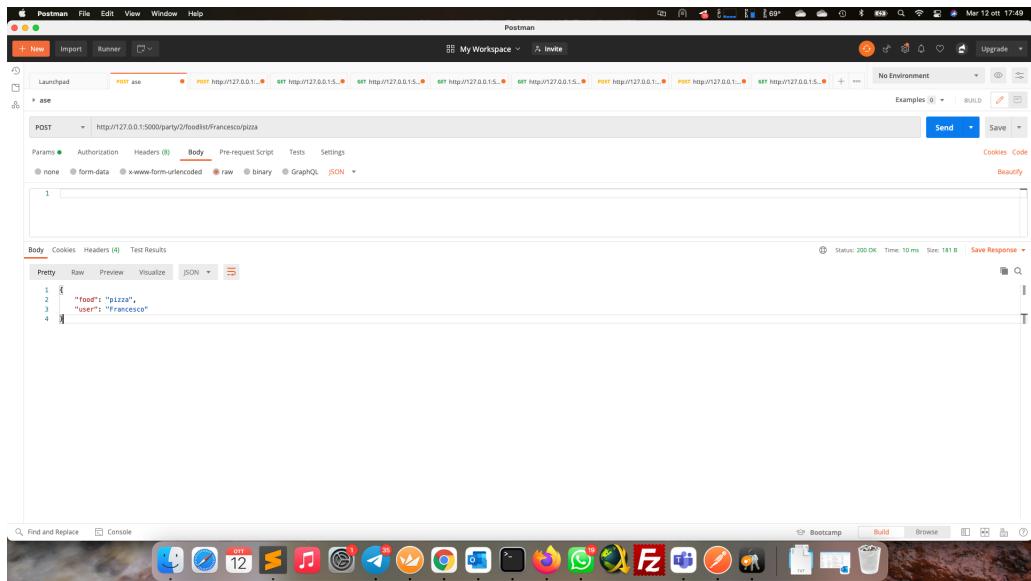
Chiamata in **GET** alla rotta **/party/2** che restituisce i dettagli del party 2.

2.7 Dettagli party con id 100



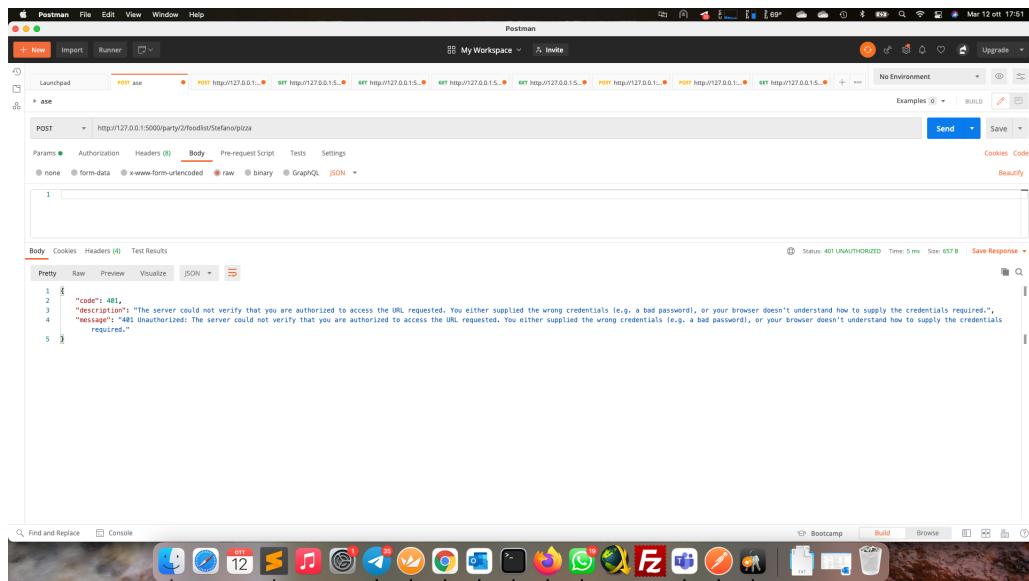
Chiamata in **GET** alla rottta **/party/100** che cerca di restituire i dettagli del party con id 100, non presente dunque riceviamo un errore 404.

2.8 Caricamento cibo



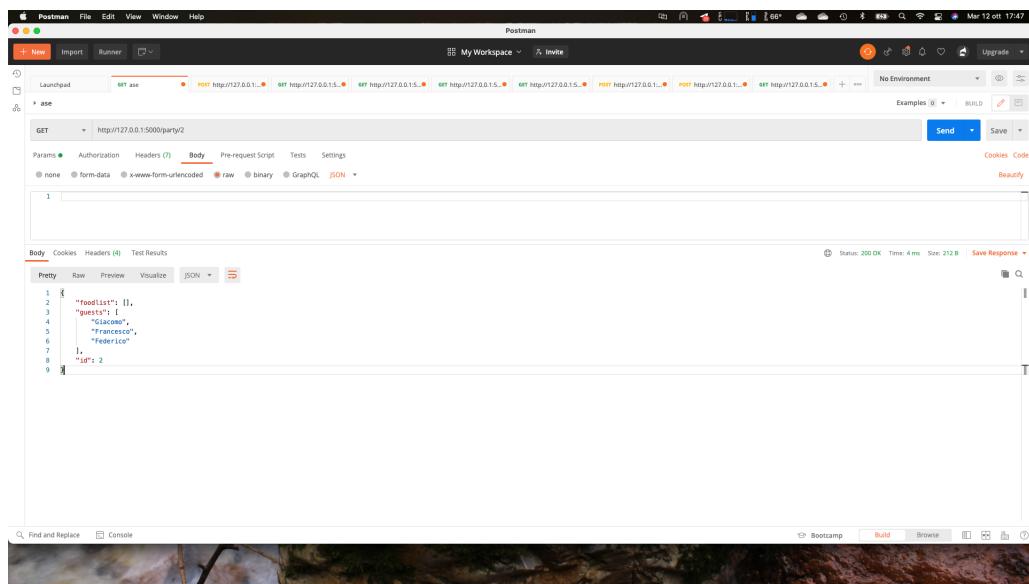
Chiamata in **POST** alla rottta **/party/2/foodlist/Francesco/pizza** che carica il cibo assegnato all'utente. Restituisce il cibo e l'utente che lo porta.

2.9 Caricamento cibo da un utente non invitato



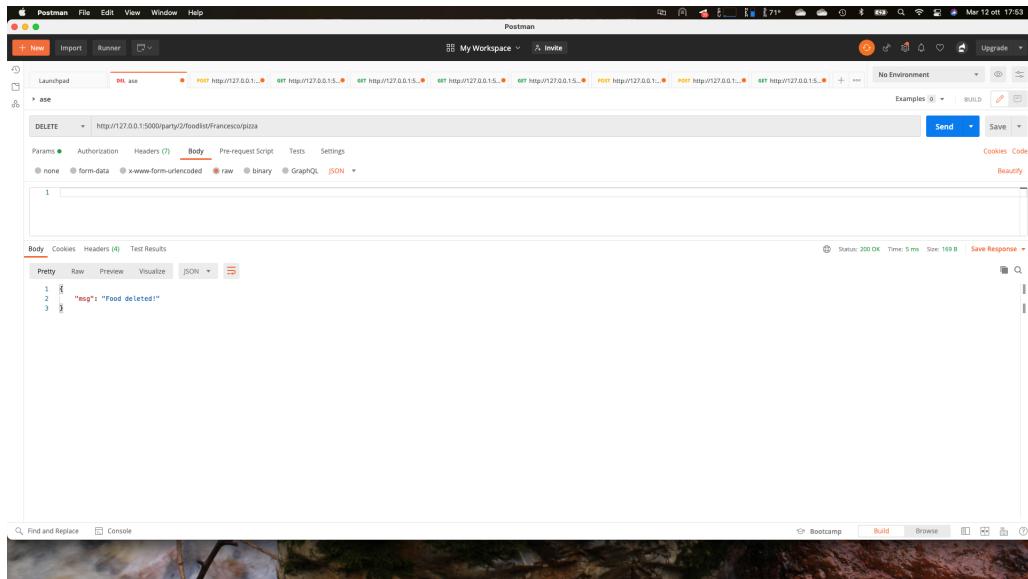
Chiamata in **POST** alla rotta `/party/2/foodlist/Stefano/pizza` che prova a caricare del cibo da un utente non invitato quindi riceviamo un errore 401.

2.10 Lista cibi



Chiamata in **GET** alla rotta `/party/2/foodlist` che mostra la lista dei cibi e gli utenti che devono portarlo.

2.11 Eliminazione cibo



Chiamata in **DELETE** alla rottta **/party/2/foodlist/Francesco/pizza** che elimina il cibo passato.

3 Test effettuato con pytest

```
boosterino:exercise_ase_1 fabiolanubile$ pwd
/Users/fabiolanubile/exercise_ase_1
boosterino:exercise_ase_1 fabiolanubile$ ls
boosterino_md           bedrock_a_party      requirements.txt      run.sh          setup.cfg      venv
boosterino:exercise_ase_1 fabiolanubile$ pytest
=====
platform darwin -- Python 3.8.5, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
rootdir: /Users/fabiolanubile/exercise_ase_1, configfile: setup.cfg, testpaths: bedrock_a_party/tests
plugins: asyncio-2.2.0
collected 2 items

bedrock_a_party/tests/test_party.py .. [100%]

=====
warnings summary
=====
./opt/anaconda3/lib/python3.8/site-packages/flask/app.py:2460
./Users/fabiolanubile/opt/anaconda3/lib/python3.8/site-packages/flask/app.py:2460: PytestCollectionWarning: cannot collect 'tested_app' because it is not a function.
  def __call__(self, environ, start_response):
-- Docs: https://docs.pytest.org/en/latest/warnings.html
=====
===== 2 passed, 1 warning in 0.39s =====
boosterino:exercise_ase_1 fabiolanubile$
```

Viene mostrato il test tramite il modulo **pytest** come richiesto da specifiche. Viene mostrato un warning derivante però, da un altro modulo installato sul mio computer quindi non rilevante ai fini del progetto.