Bedrock a Party

Homework 1 – Advanced Software Engineering

Gabriele Sipione

October 10, 2021

1 Introduction

The homework, called *Bedrock a party* aims to create a RESTful service to manage the parties and the supplies for the latter in Bedrock. In particular, it consists of the calling and handling of the functions in the blueprint file *parties.py*. This implies that most of the work requested was about reading and understanding the skeleton provided. Once the methods were fully undersood, they were used following as close as possible the TODO comments wich gave direction on what was necessary to code to complete the homework. Finally the program has been tested with the help of *Postman* and *pytest*.

The zip contains this file and the *parties.py* file. If the images are not clear enough they will be in my github repository with the homework.

2 Tests

The solution has been tested with *pytest* using the *test_party.py* file. Additionally, using *Postman*, to run some more tests.

Figure 1: Pytest result.

2.1 Party-related tests

Here it's shown the test of features such as:

- Creating a party.

- Retrieving a party using the party id.

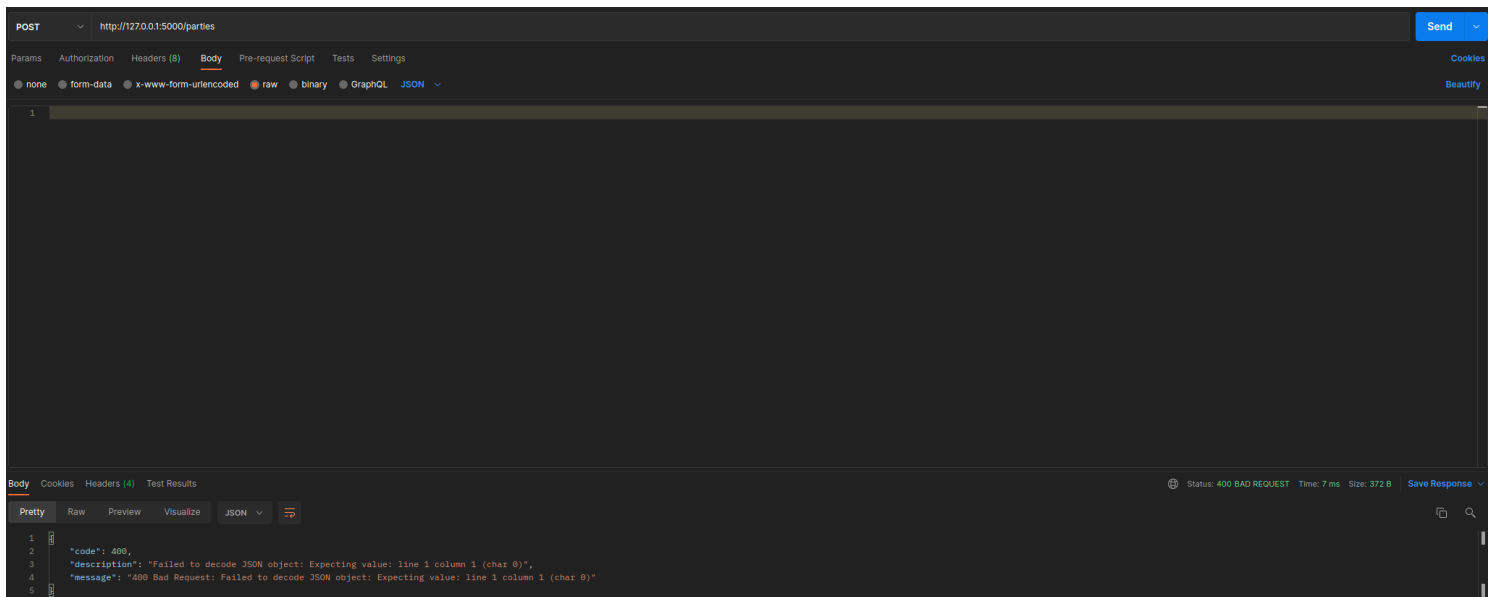- Getting all the parties loaded in the system and get the number of the latter.

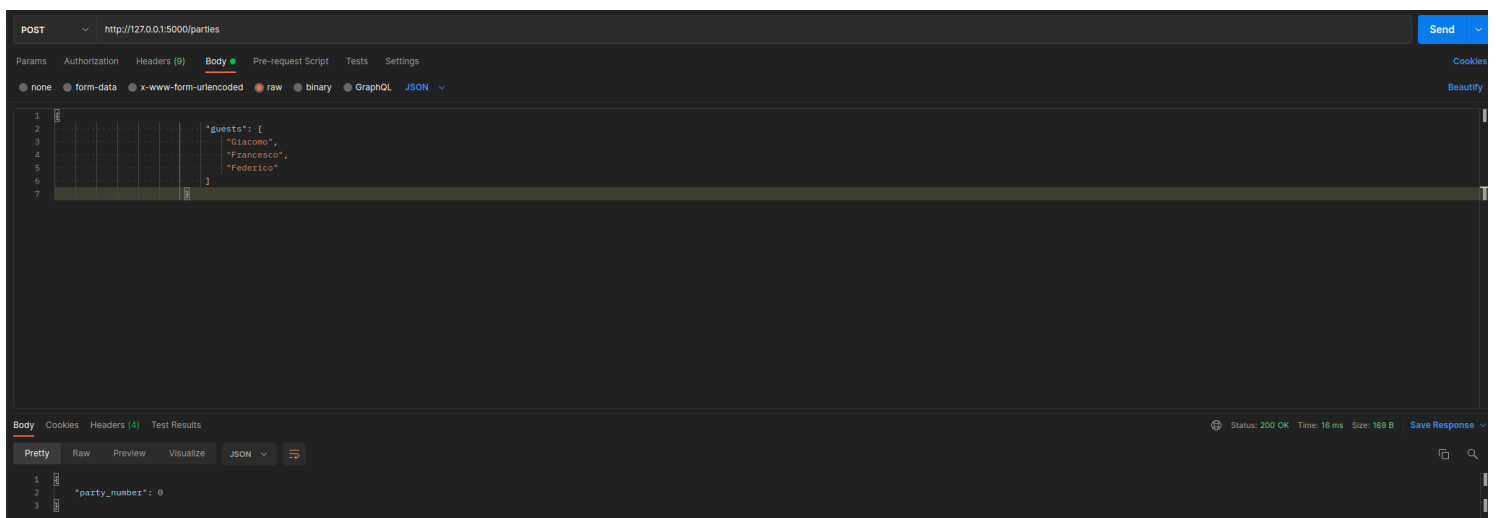Figure 2: Creating a party with no guests.
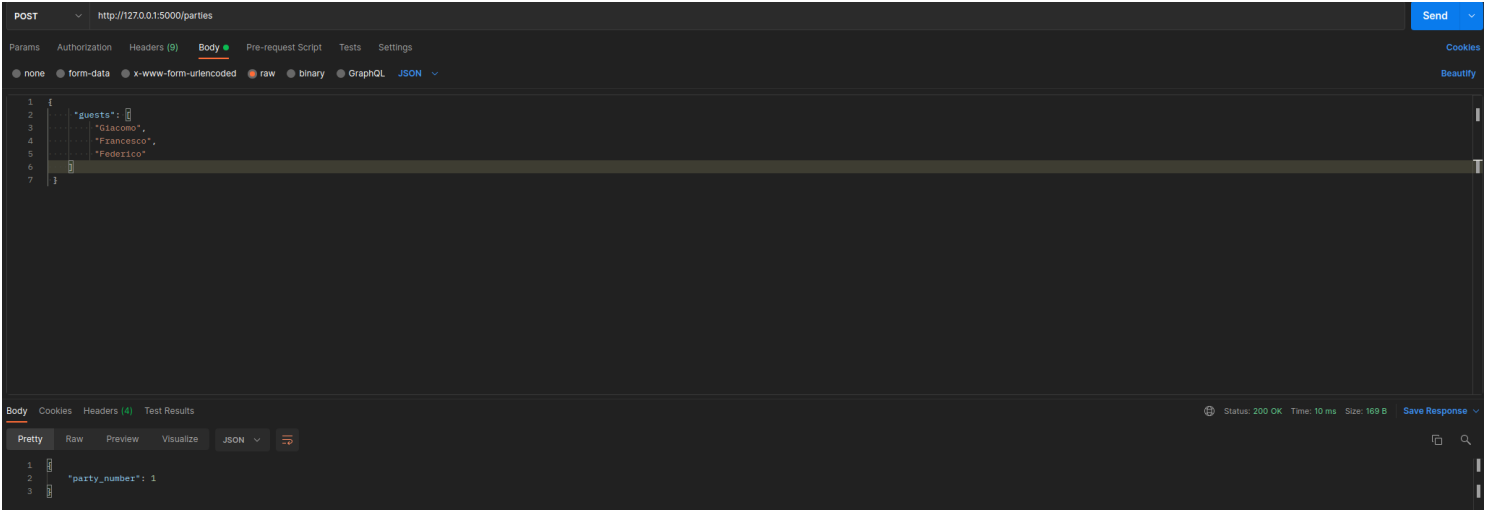

Figure 3: Creating a party.

POST ∨ http://127.0.0.1:5000/parties Send ∨

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Cookies

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

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

Body Cookies Headers (4) Test Results Status: 200 OK Time: 10 ms Size: 169 B Save Response ∨

Pretty Raw Preview Visualize JSON ∨

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

Figure 4: Creating a party.

POST ∨ http://127.0.0.1:5000/parties Send ∨

Params Authorization Headers (9) Body ● Pre-request Script Tests Settings Cookies

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

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

Body Cookies Headers (4) Test Results Status: 200 OK Time: 9 ms Size: 169 B Save Response ∨

Pretty Raw Preview Visualize JSON ∨

```
1  {
2      "party_number": 2
3  }
```
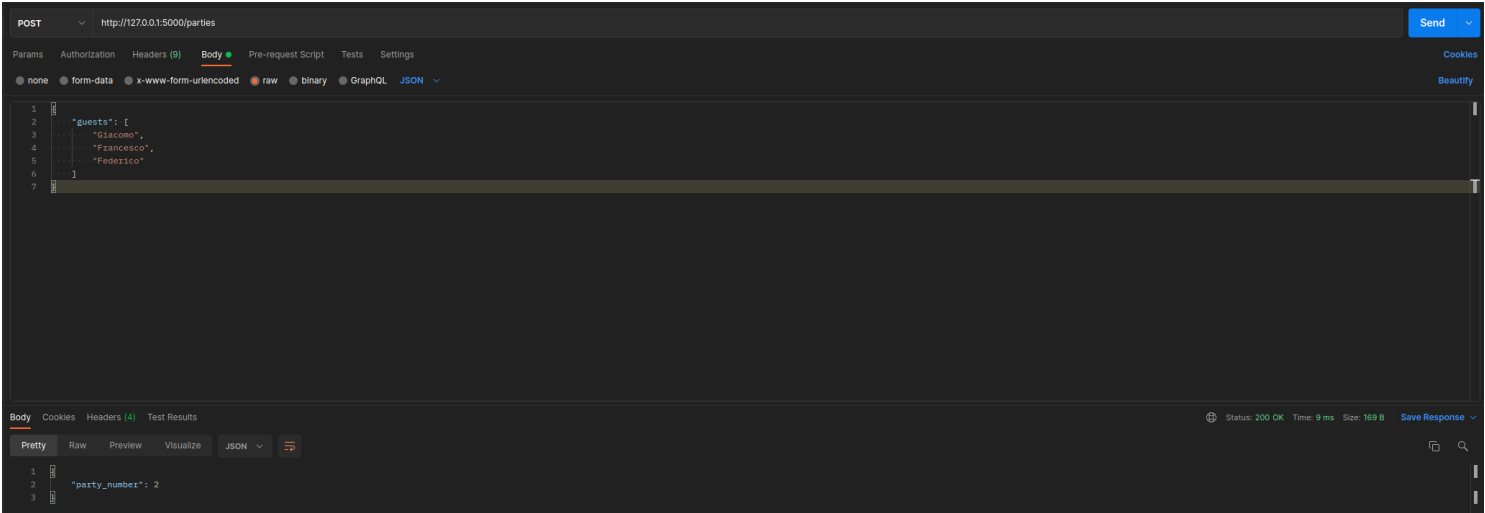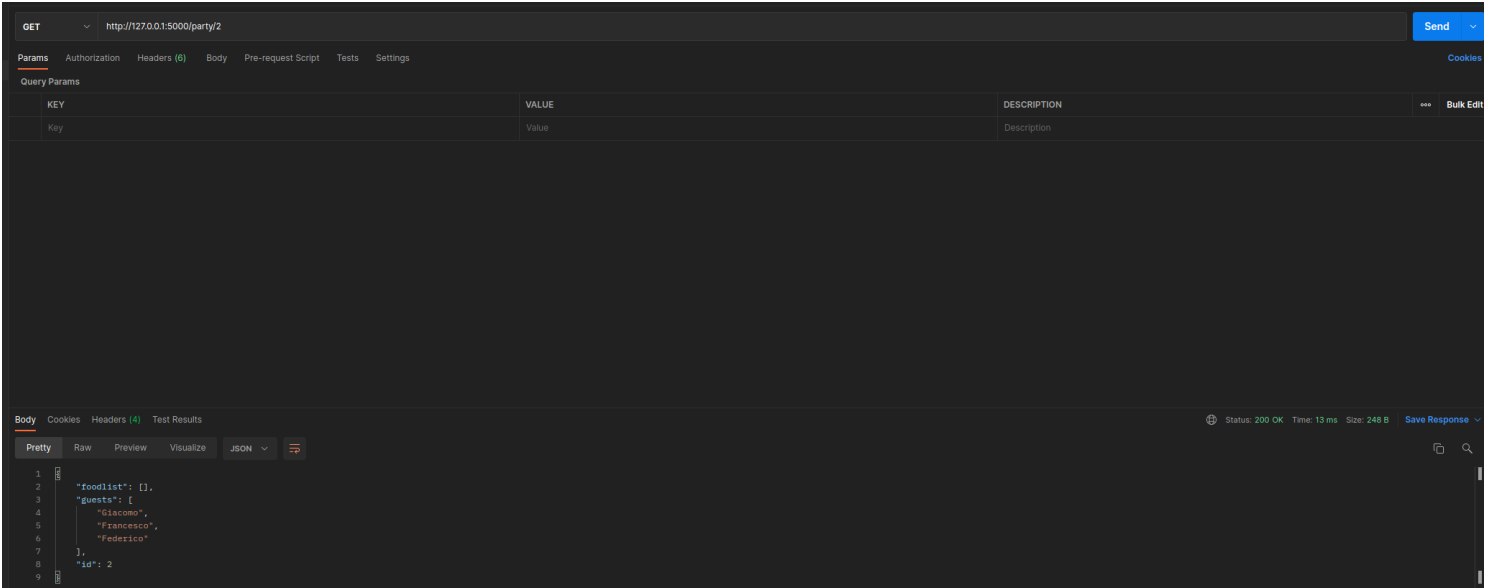
Figure 5: Creating a party.

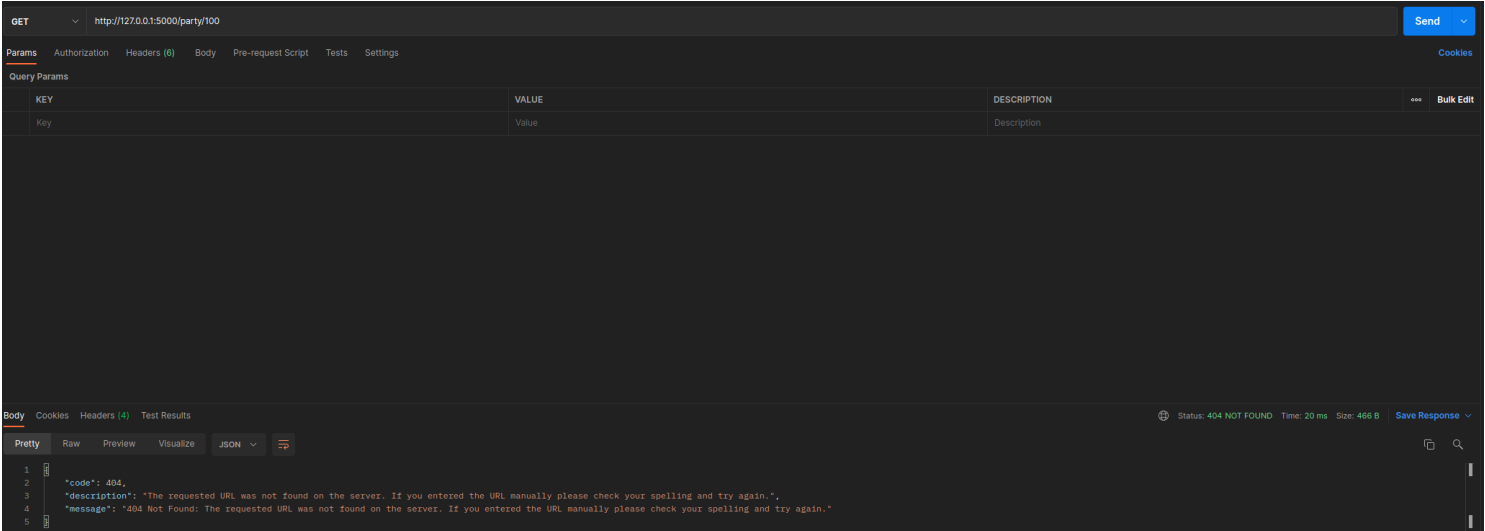Figure 6: Getting a party using the party's id.

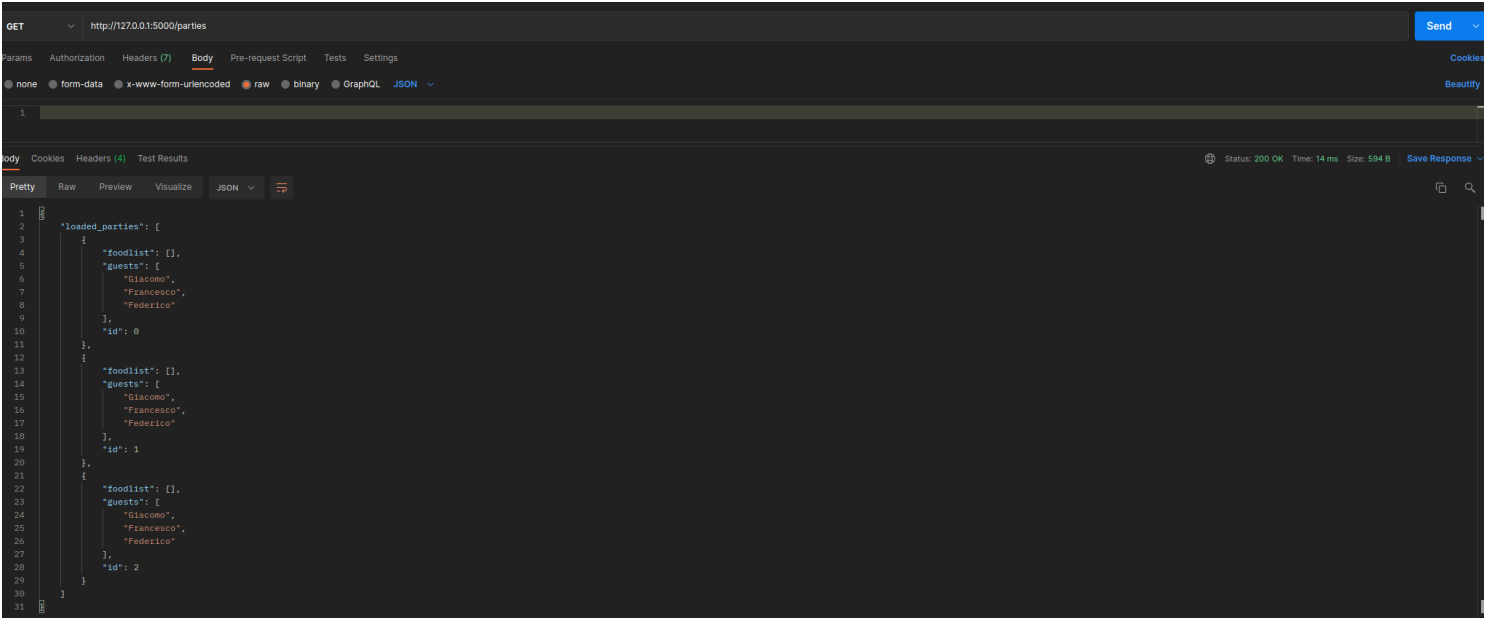Figure 7: searching for a party that doesn't exists.

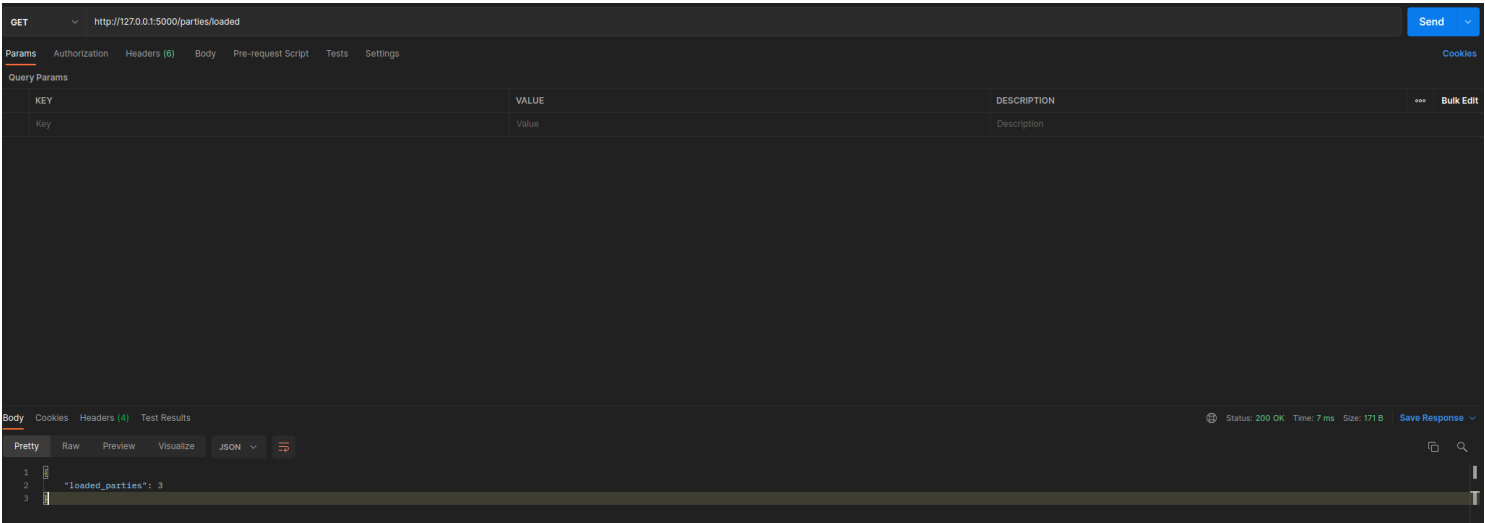Figure 8: Retrieving all parties that are loaded in the system.

Figure 9: Retrieving the number of parties loaded in the system.

Figure 10: Trying to get all the parties when the list is empty.

2.2 Foodlist-related tests

Here it's shown the test of features such as:

- Adding/Removing a meal from a guest.
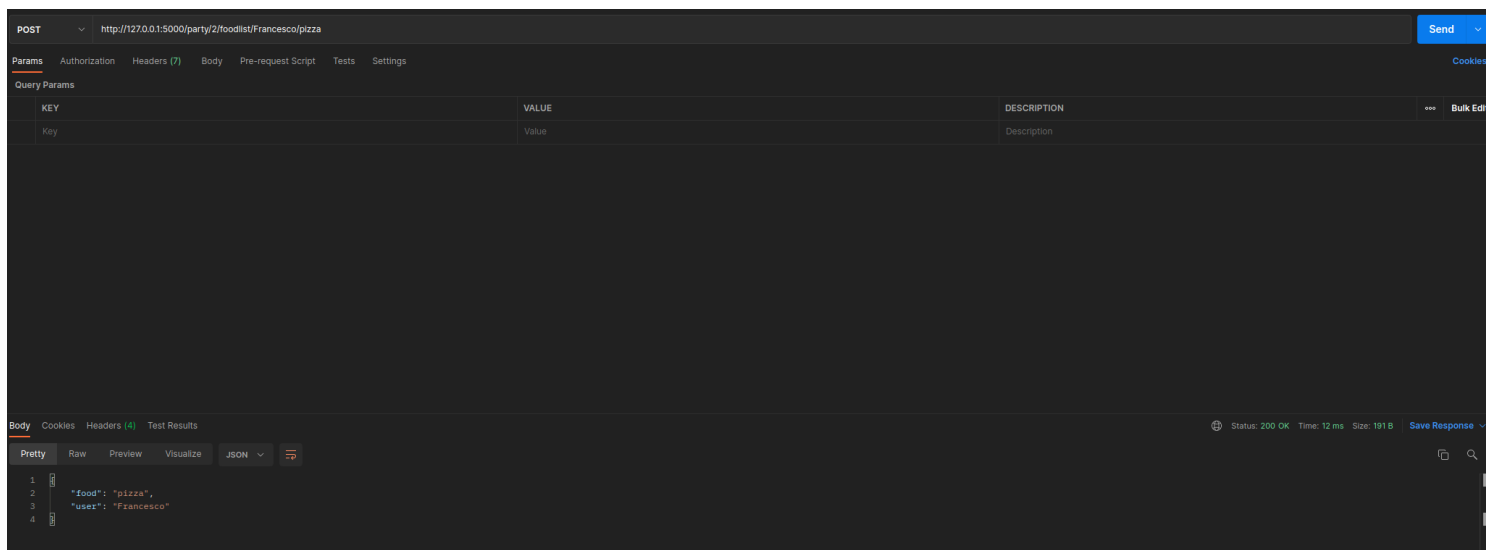- Retrieving the food list of a specific party.
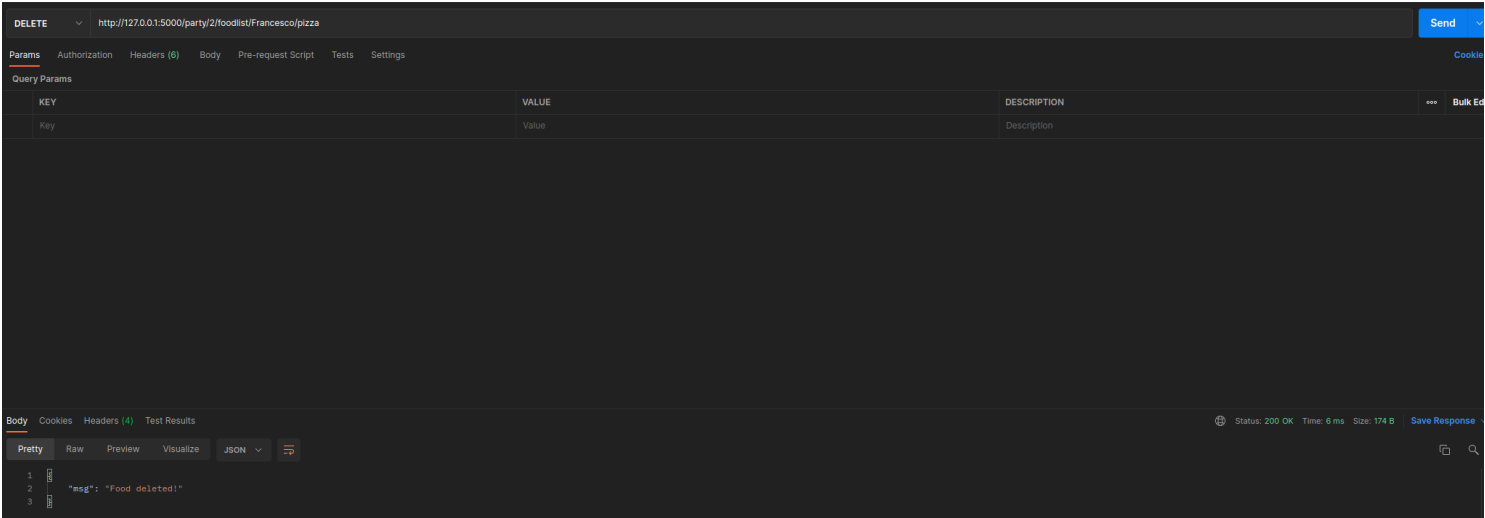

Figure 11: Adding a meal brought by a guest.
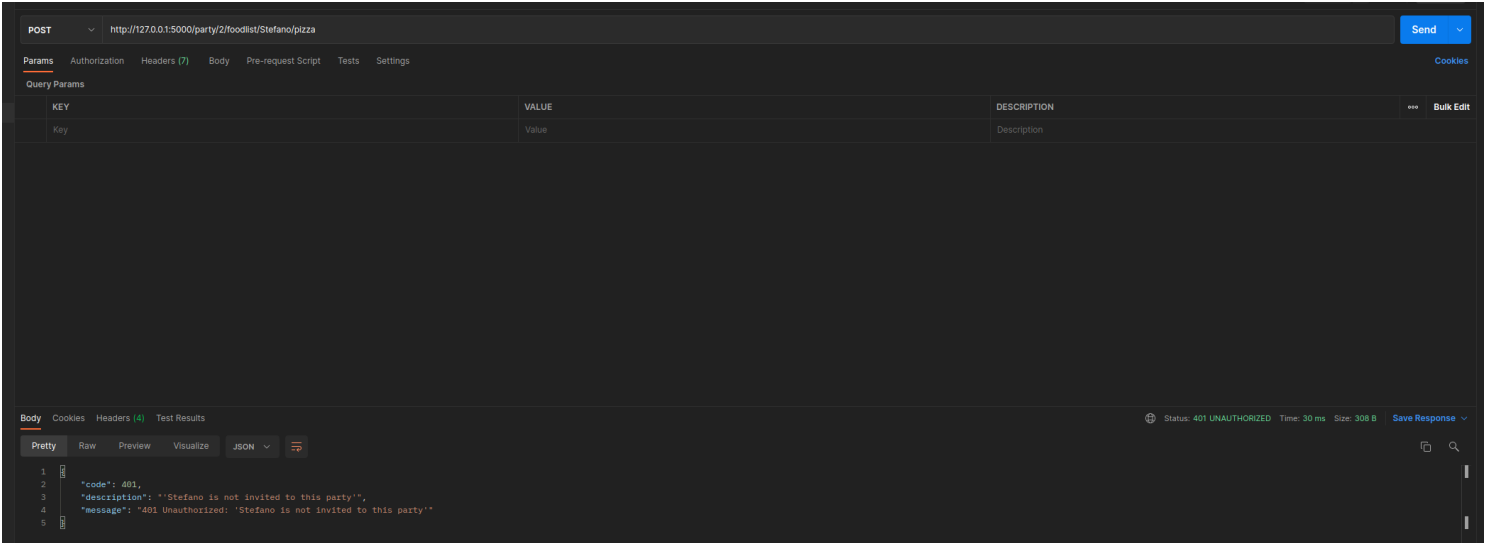
Figure 12: Removing a meal.

Figure 13: Trying to add a meal brought by an uninvited guest

Figure 14: Retrieving the party food list.