An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Computer Engineering Department

Distributed and Operating Systems (1063645)

Project part #1:
“Bazar.com: A Multi-tier Online Book Store”

Students’ names:
Wa’ad Khalifa
Diala Nawwas

Instructor name:
Dr. Samer Arandi

2020/2021

Abstract:

REST stands for representational state transfer and it is an architectural principle rather than a specific technology or standard. The REST-style architecture is just like the web. There are requests and response that are the transfer of resources. Resources are any addressable object (as in something with a URI on th web).

A RESTful web service is implemented with HTTP and has the following specific requirements: All resources are identified by URIs, Only the HTTP methods are used as an interface (POST, GET, PUT, DELETE), Resources have an internet type (XML, JSON, etc.)

introduction:

In this project, we have a two-level or two-tier web design, a front-end, and a back-end.
The front-end contains a server front that performs three tasks: search by topic (GET), search by id (GET), and purchase(item_number) (POST).
The back-end contains two servers. The first server is a Catalog server that contains three tasks: query-by-subject (GET),query-by-item (GET), update operation the number of items in stock to be increased or decreased. (PUT). The second server is the Order server that contains one task a purchase(item_number) (POST).
We use Flask for Python a lightweight micro-framework to write the code, the response is returned as a JSON object. First, the Catalog Server code was written, then the Order server, which works to update th catalog server in terms of the decrease in the quantity of the book after a purchase is requested from the server front.

Scenario:
The client requests information about a book through id or topic. This request is transmitted to the front-end server. The front-end server through id or topic sends a request to the catalog server, such as a signal that it has the book's information, then the catalog server sends responses back to the front-end server, and then the client recieves it. Purchase order, the request is transmitted from the front-end server, which sends a request to the order server according to the id, and then the order server sends a request to the catalog server and searches for the book if it exists or not. It sends a message that there is no information about the request in the URL if it is not exists, otherewise it gives a success message.

Outputs:
We used three virtual machines, and each virtual had an IP address, and I traveled between three requisiti servers, via postman.

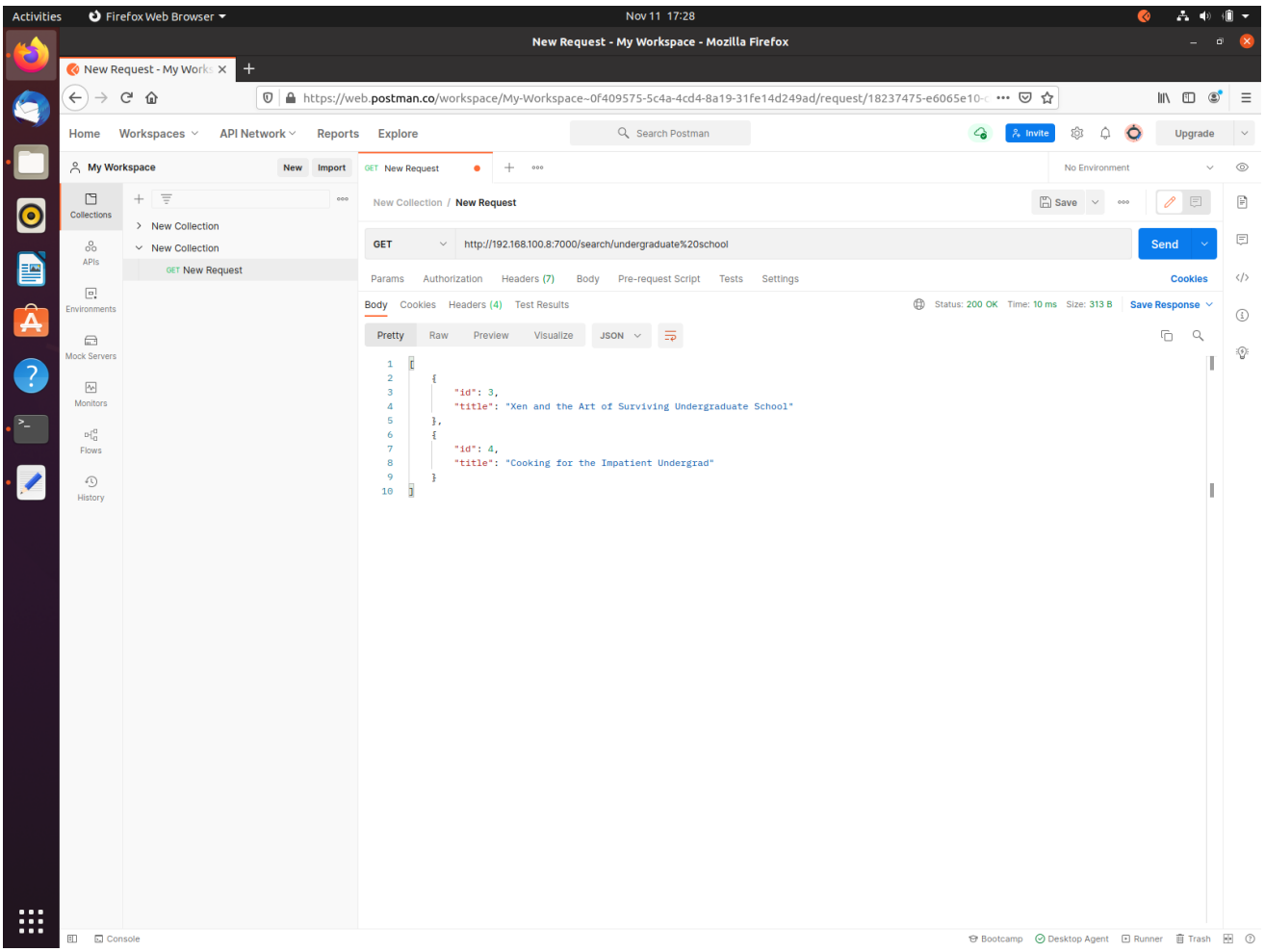
Catalog server output:

The catalog server maintains the catalog which consists of four entries. Each entry maintains the number items in stock, cost of the book, quantity, and the topic of the book.
Catalog server receive GET or search requests from front-end server, and update request from order serve

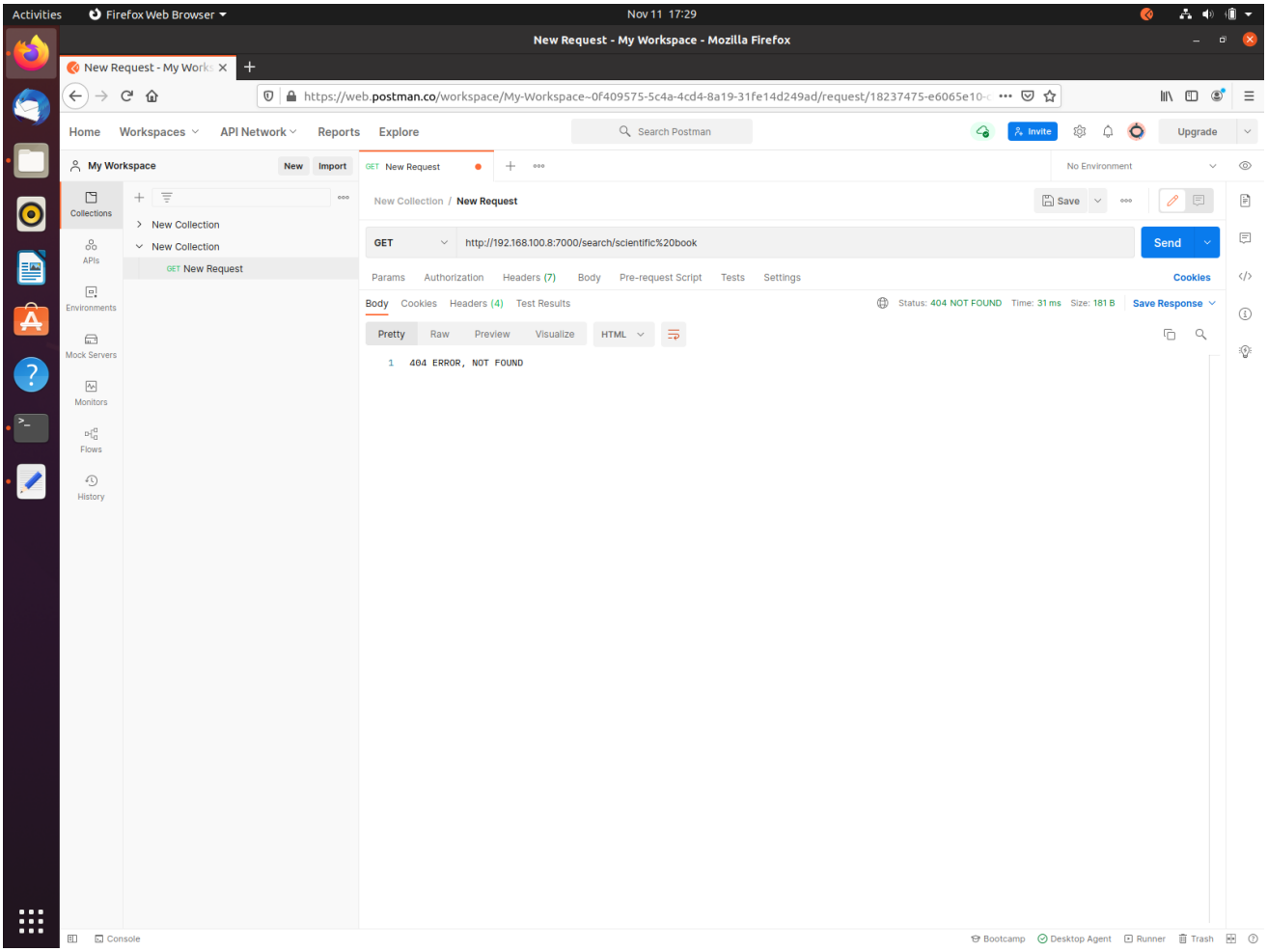
Operation:

- **Search: Search by ID and search by topic. (GET)**

Example of search by topic:

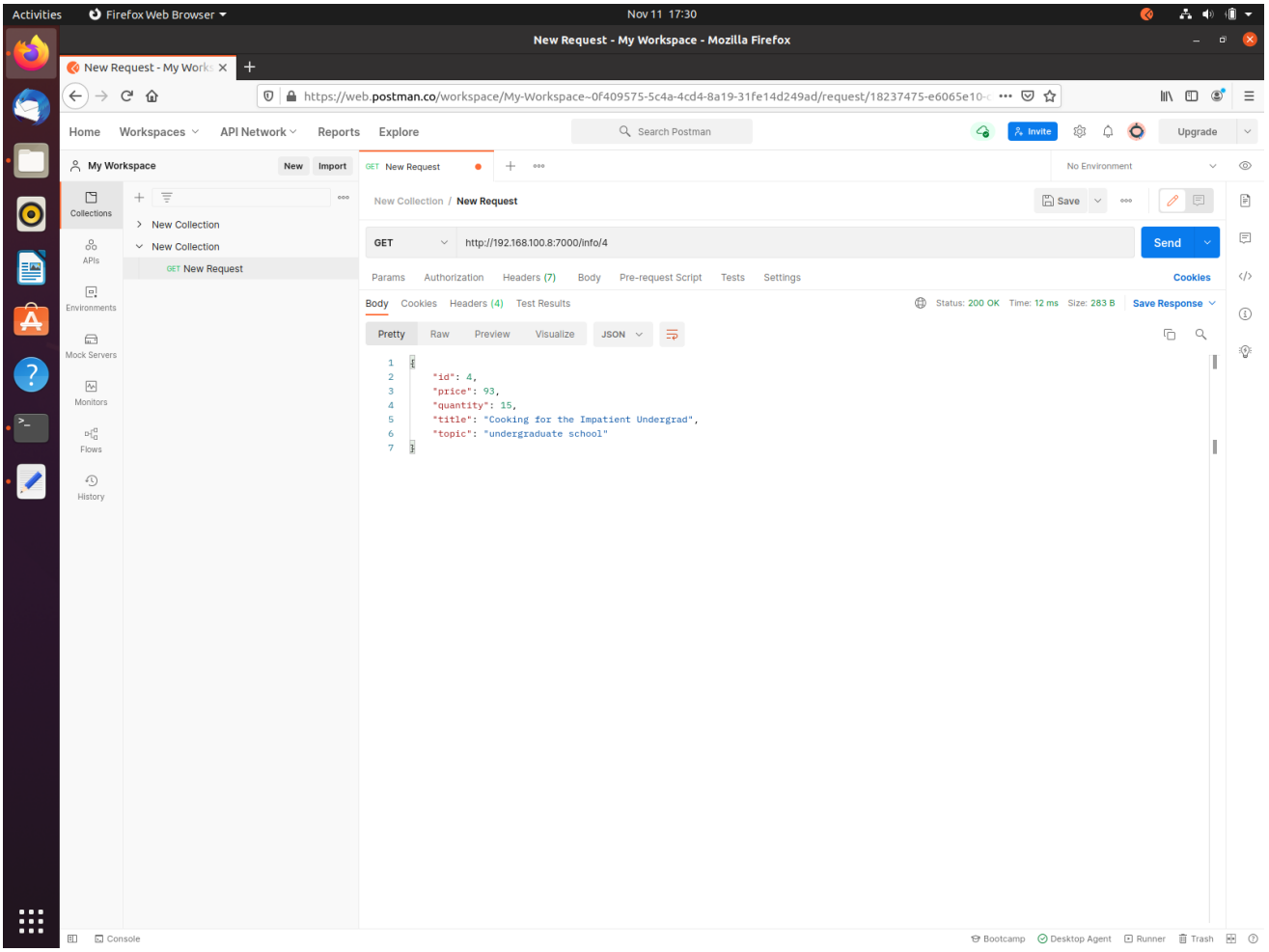


Successful request with status 200 ok.

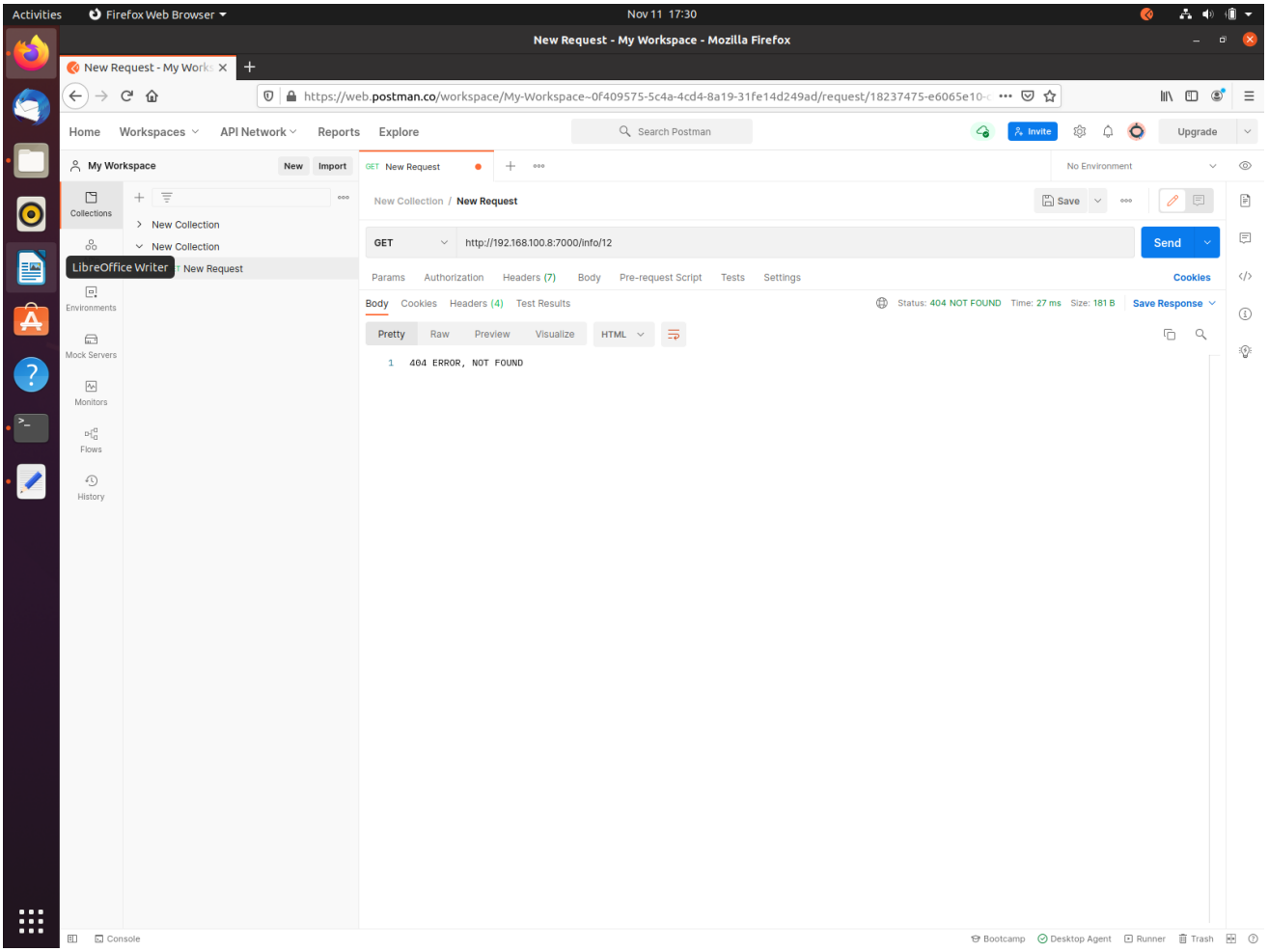


It failed because this book does not exist, status code 404.

Example of search by id:



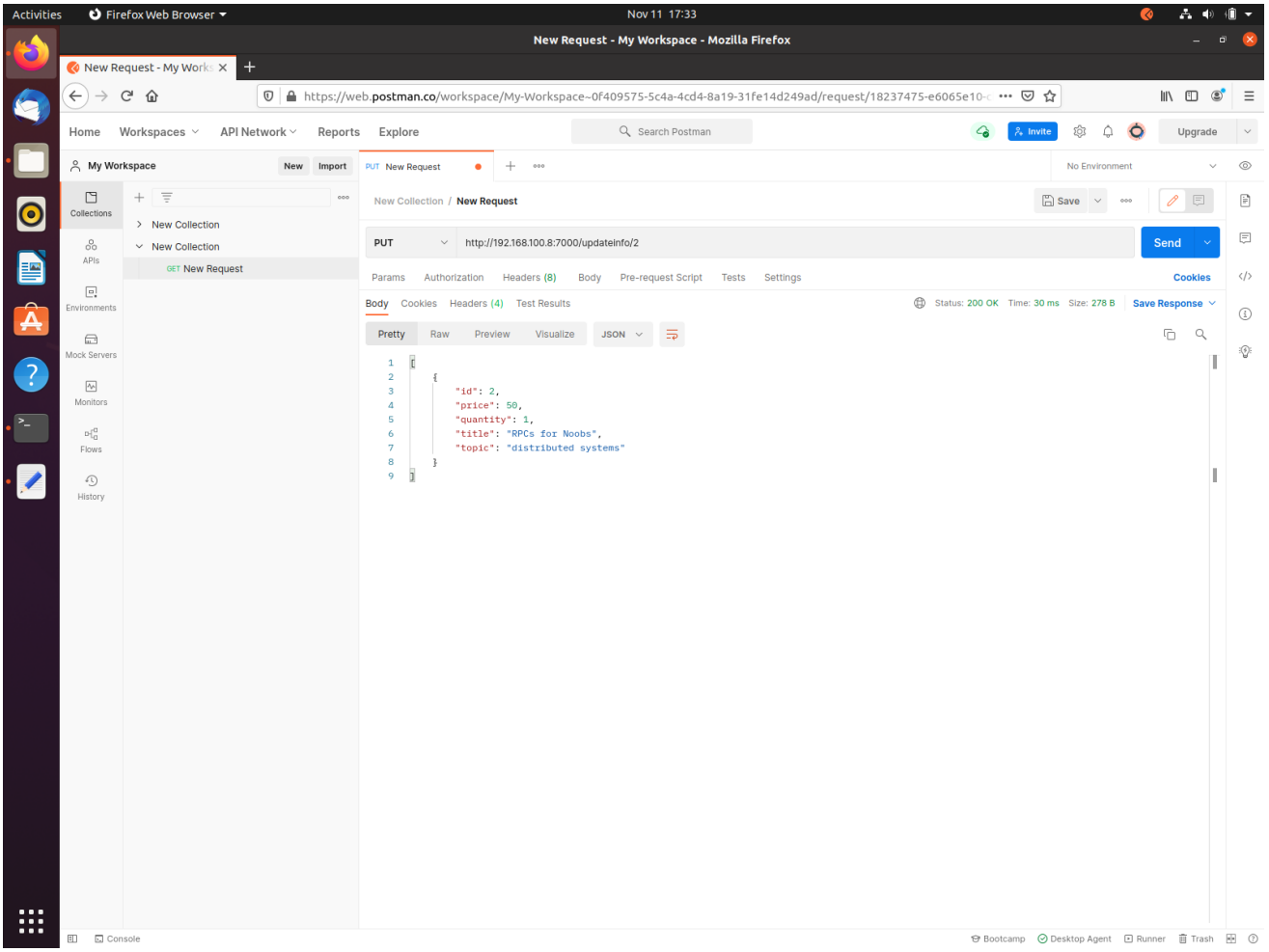
Successful request with status 200 ok.



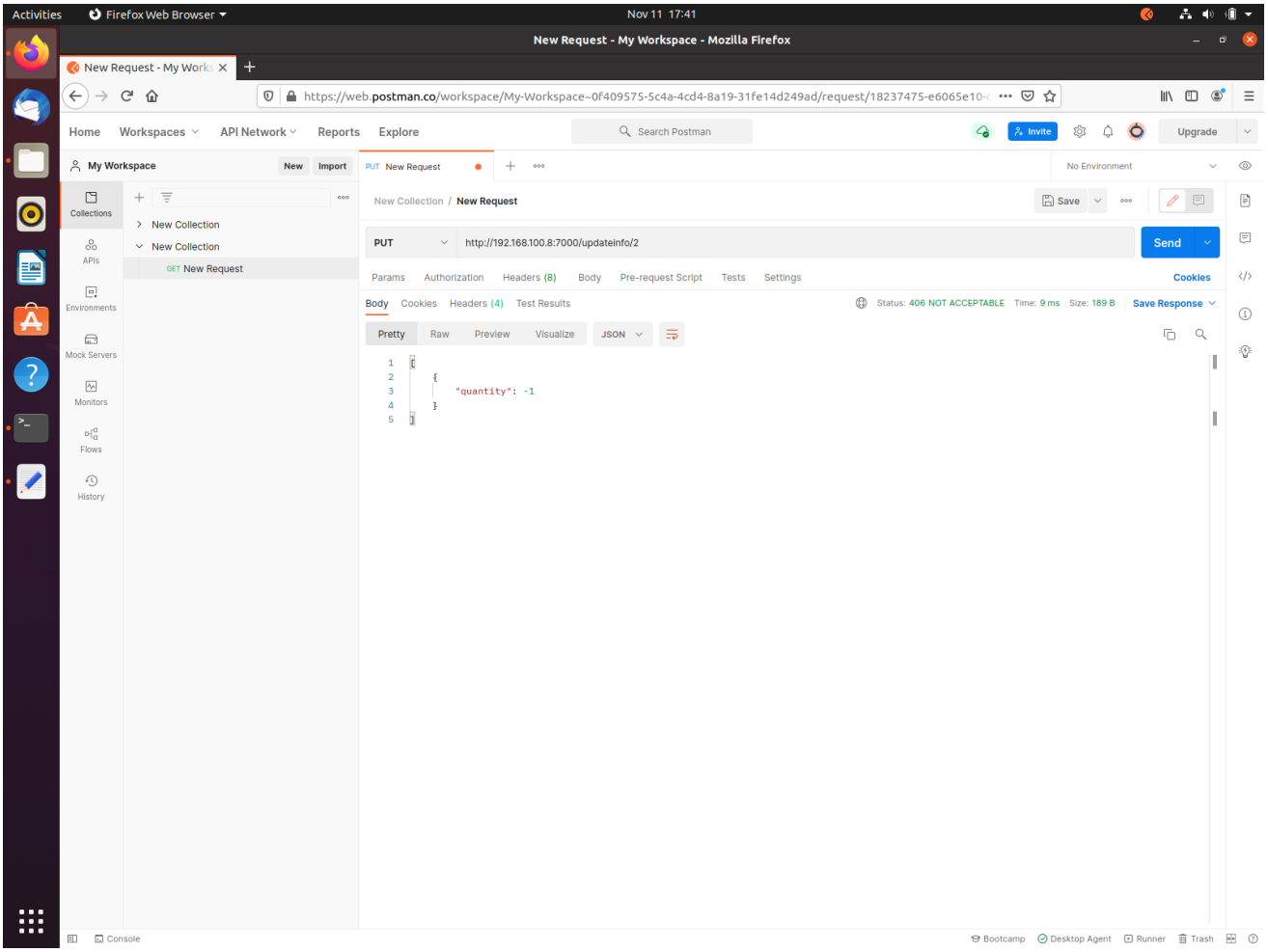
It’s failed because this book does not exist, status code 404.

- **Update (PUT)**

Update the quantity of kind of books when receiving a request from the Order Server.



id =2 completed successfully, status code 200 OK.



Request failed, because the item is out of stock, -1 is not acceptable as a value of the quantity, status code 406.

Here is the JSON file of the four entries:

main.py FourEntries.json

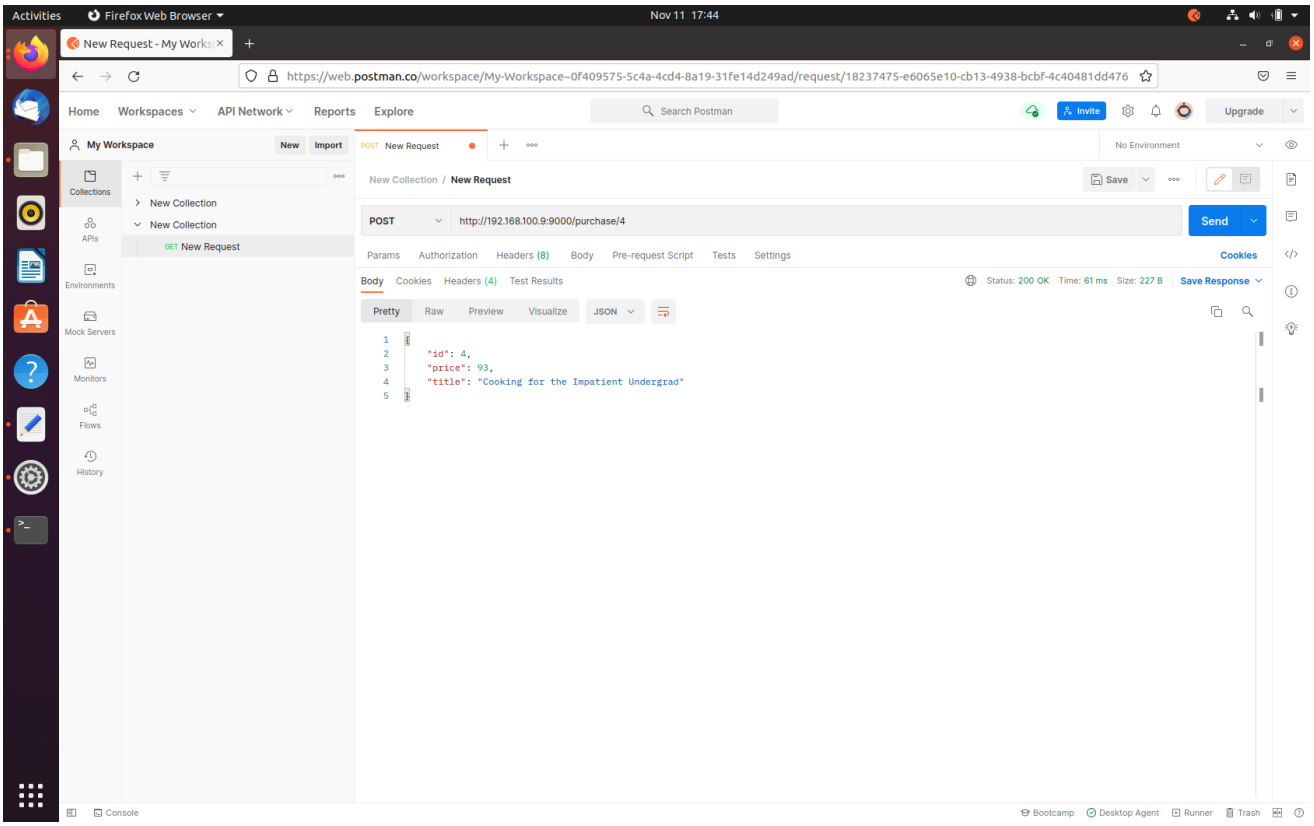
```
{"books": [{"id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day",
            "price": 50,
            "topic": "distributed systems",
            "quantity": 9
            },
{"id": 2,
 "title": "RPCs for Noobs",
 "price": 50,
 "topic": "distributed systems",
 "quantity": 1
},
{"id": 3,
 "title": "Xen and the Art of Surviving Undergraduate School",
 "price": 66,
 "topic": "undergraduate school",
   "quantity": 12
},
{"id": 4,
 "title": "Cooking for the Impatient Undergrad",
 "price": 93,
 "topic": "undergraduate school",
   "quantity": 14
}]}
```

Order Server output :

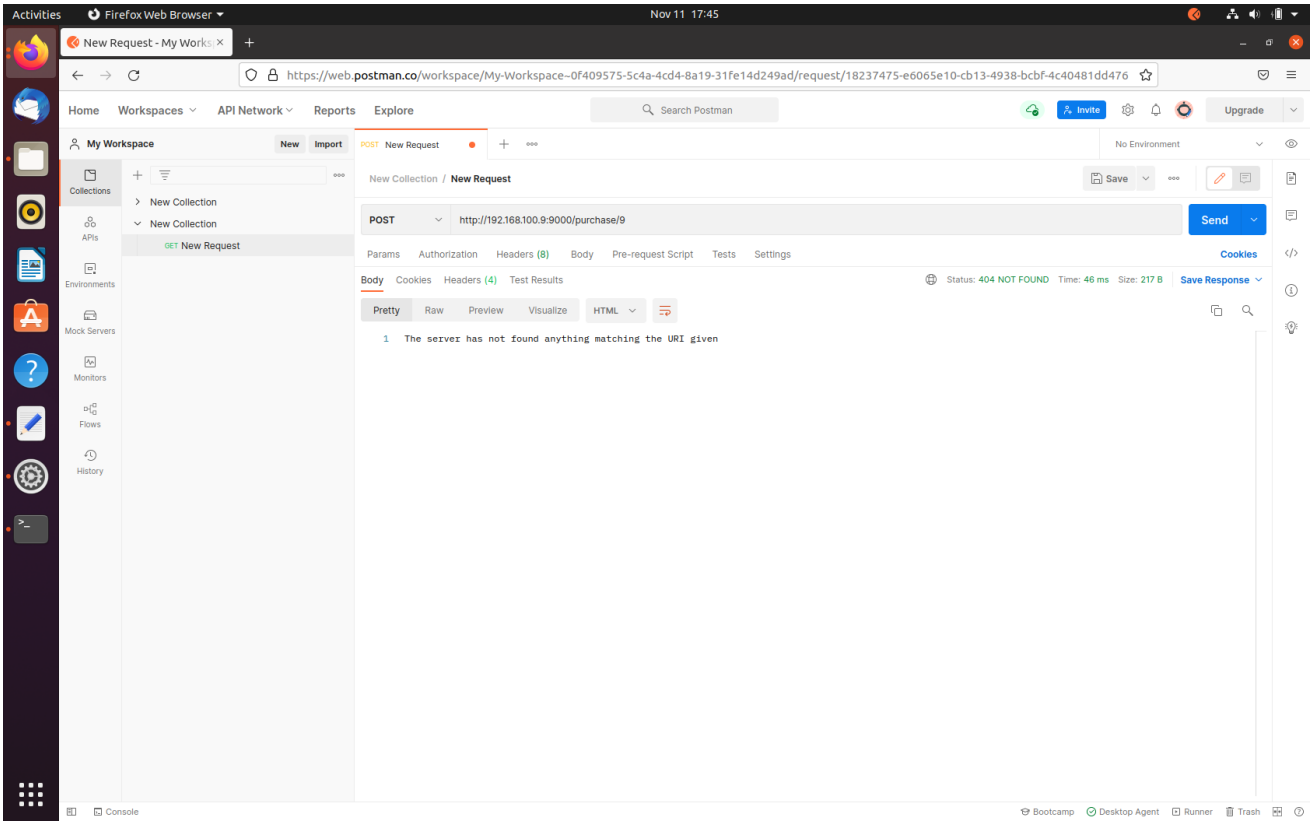
The order server maintains a list of all orders received for the books, it receives a purchase request from t front-end server. Upon receiving a purchase request, the order server must first verify that the item is in stock by querying the catalog server and then decrement the number of items in stock by one. The purcha request can fail if the item is out of stock.

Operation:

- **Purchase, add order, specifies an item number for purchase (POST).**

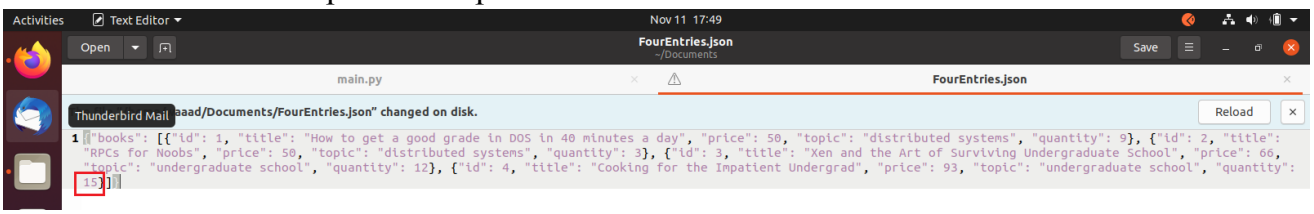


Request completed successfully, status code 200 OK.

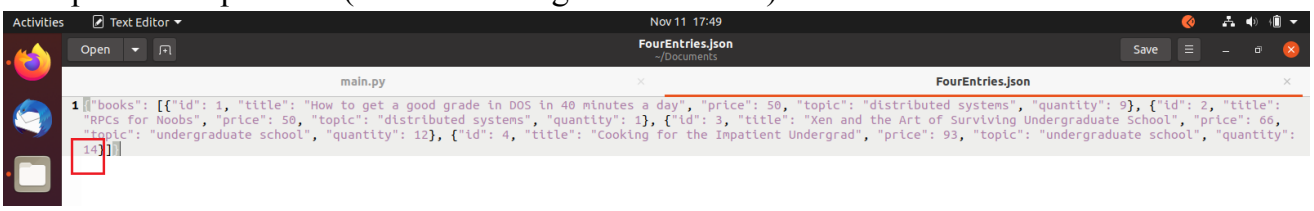


Request with id =9 has failed, because book with id 9 does not exist, status code 404.

Here is what happens when an order is done successfully, the operation will be done and the quantity of t
kind of books will decrease.
The four entries before purchase operation:



after purchase operation (when reloading the JSON file):



and here is the list of orders file, it contains information about the required book:

main.py | ListOfOrders.json

```
[{"id": 4, "price": 93, "title": "Cooking for the Impatient Undergrad"},
 {"id": 2, "price": 50, "title": "RPCs for Noobs"},
 {"id": 3, "price": 66, "title": "Xen and the Art of Surviving Undergraduate School"},
 {"id": 4, "price": 93, "title": "Cooking for the Impatient Undergrad"}]
```

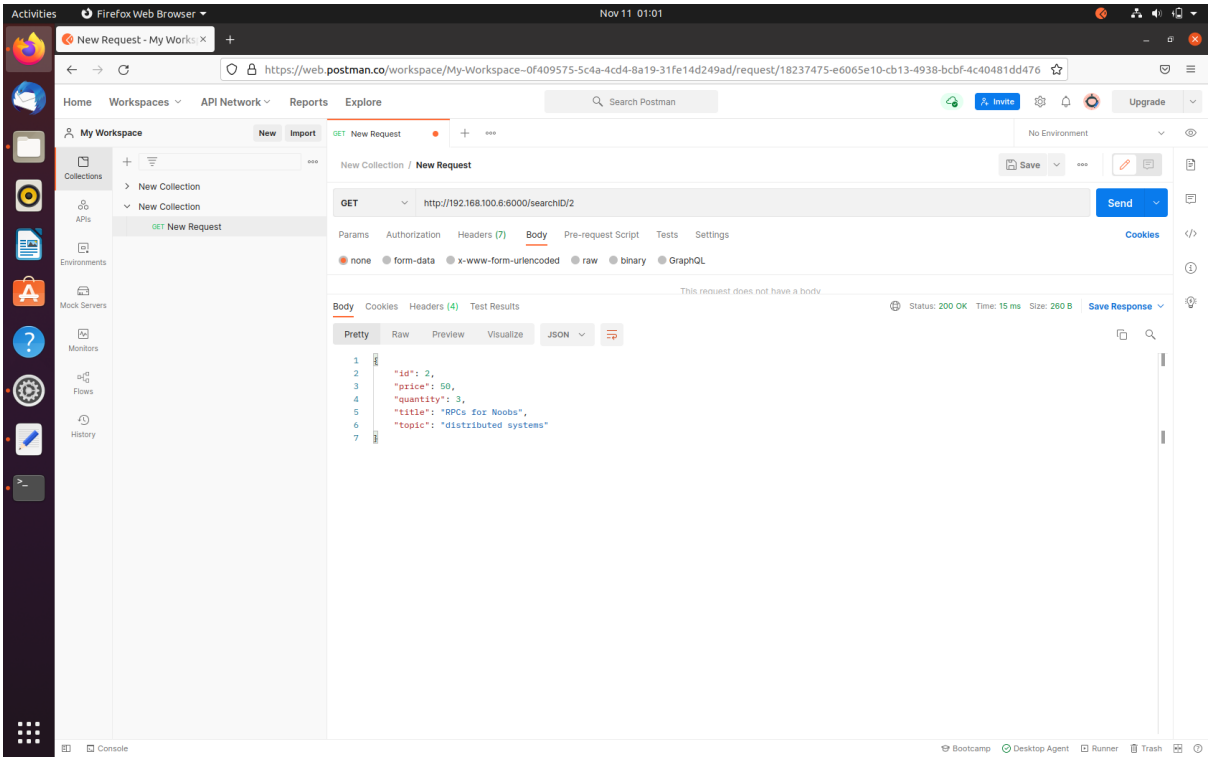
Front-end Server output:

The front-end server supports three operations. It receives requests and then gets back responses.

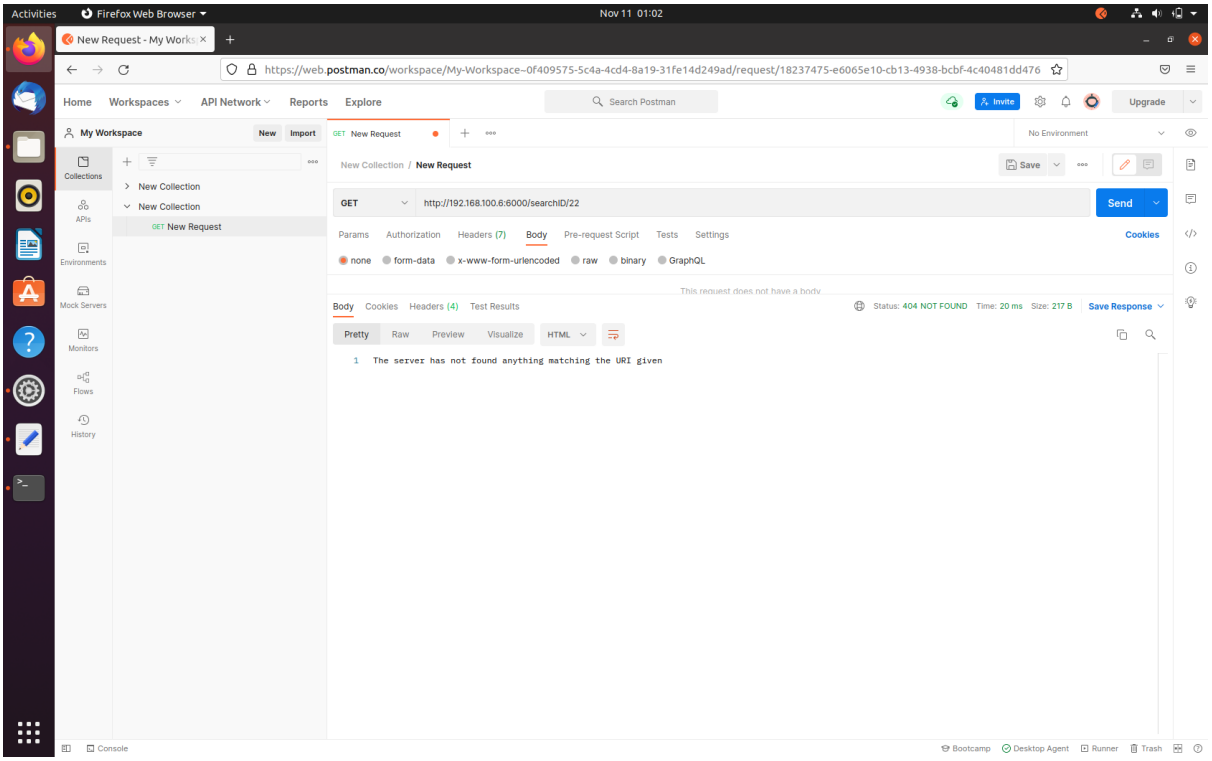
Operations:

- **Search: Search by ID and search by topic. (GET)**

Example of search by id:

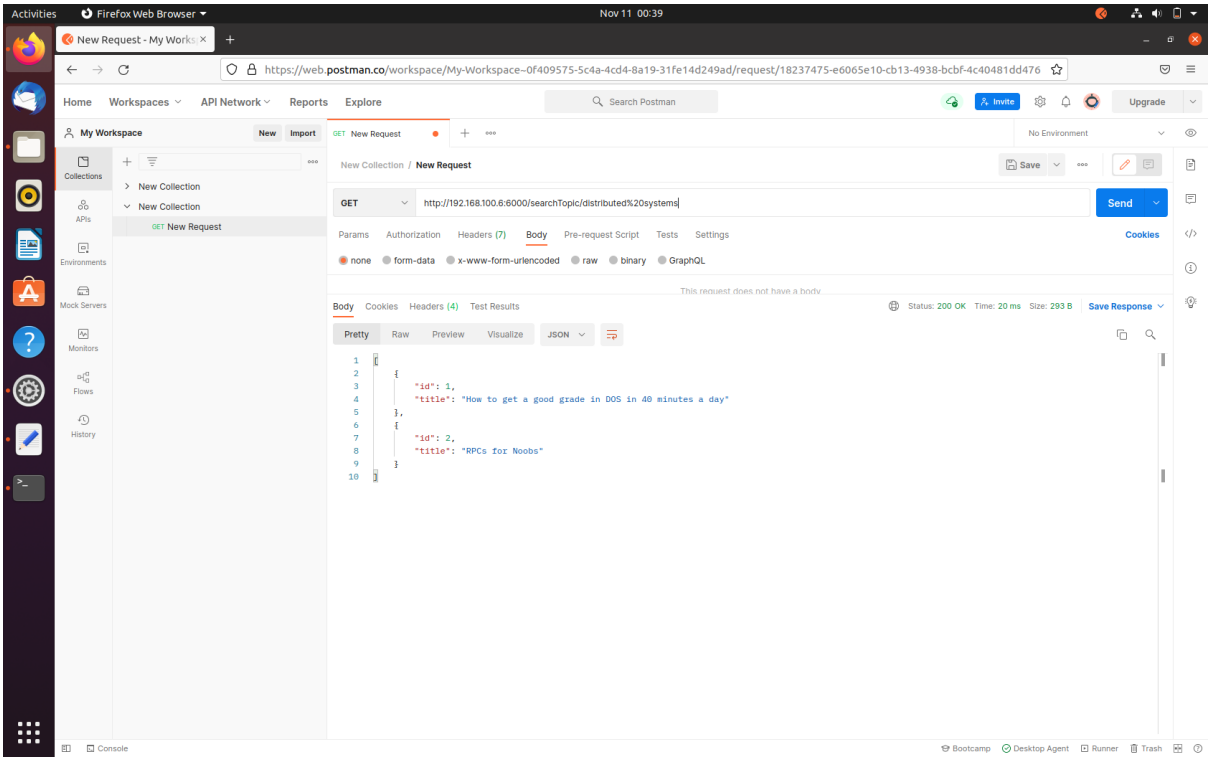


Request completed successfully, status code 200 OK.

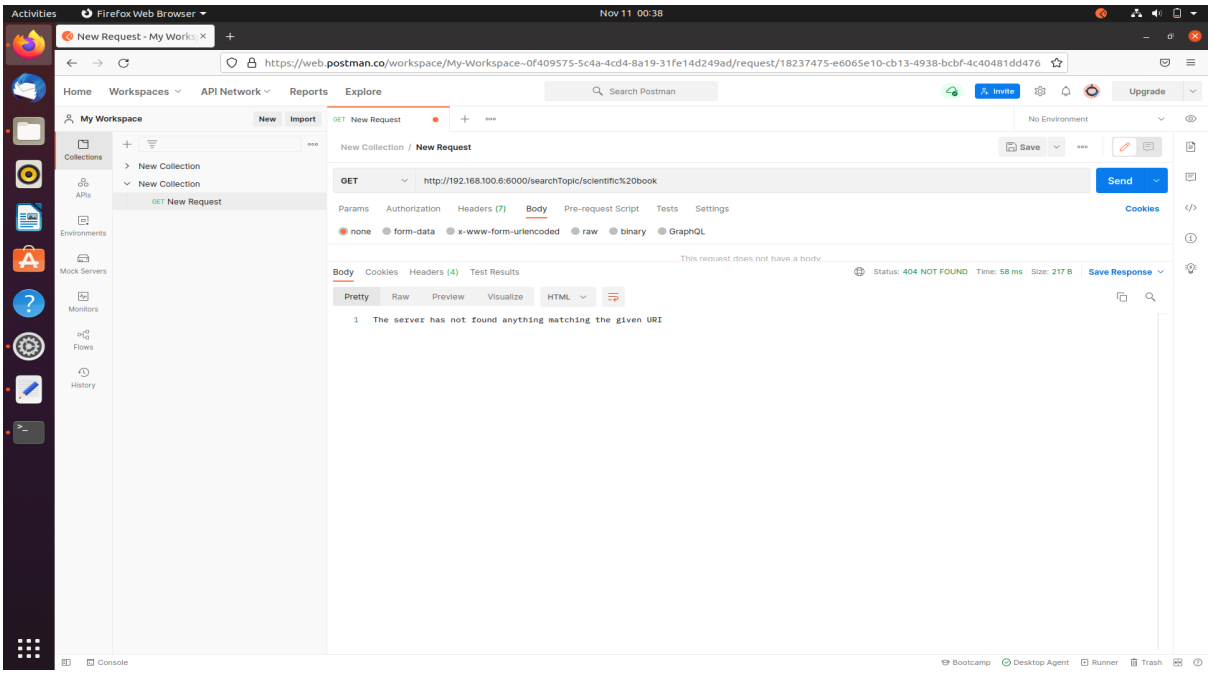


Request failed because this book doesn't exist, status code 404.

Example of search by topic:

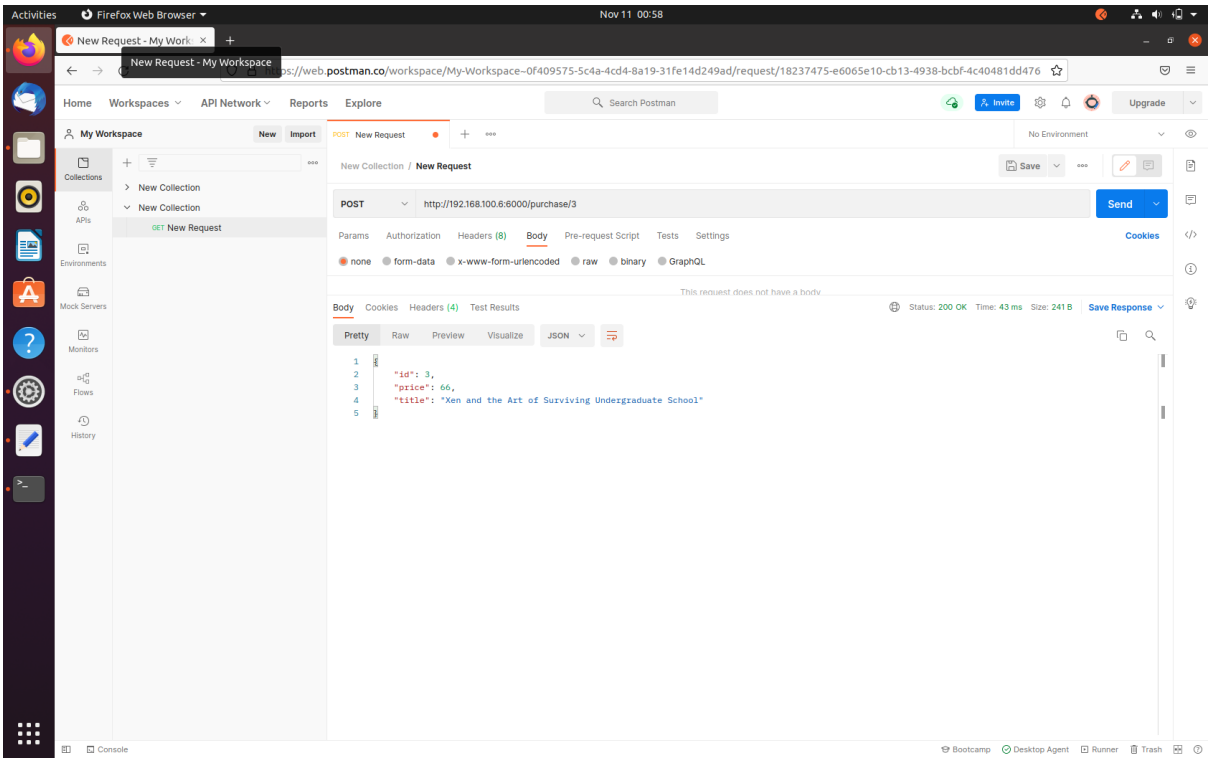


Request completed successfully, status code 200 OK.

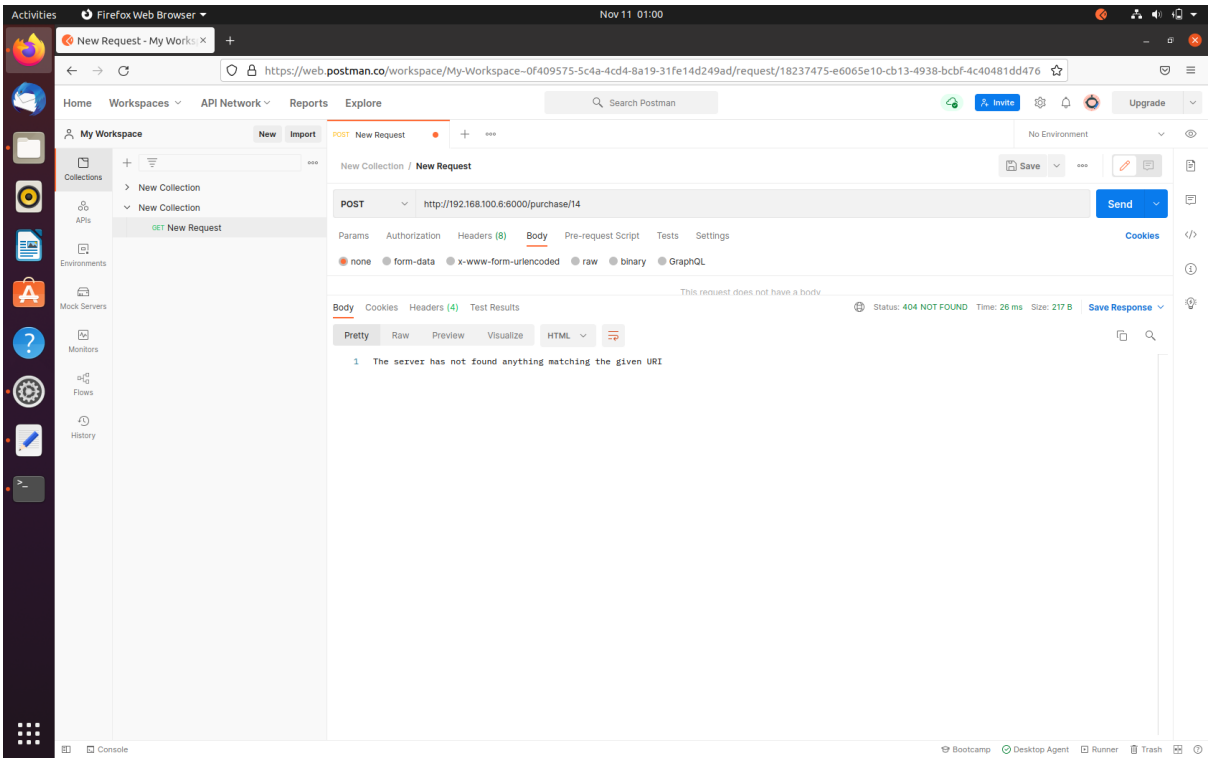


Request failed because this book doesn't exist, status code 404.

- **Purchase: add order, specify an item number for purchase (POST)**



Request completed successfully, status code 200 OK.



Request failed, there is no book with id 14, status code 404.