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 #2: “Turning the Bazar into an Amazon: Replication, Caching and Consistency”

Students’ names:
Wa’ad Khalifa
Diala Nawwas

Instructor name:
Dr. Samer Arandi

2020/2021

How it works?

Based on the first part of the project, we made adjustments so that we significantly modified the front-end server. In this part, we have 5 servers, 3 servers as previous part, 2 servers are replicas, we replicated order server and catalog server, we also added 3 new books.

In the front-end server, first, we created a cache in the form of JSON file. The way this cache works, it is empty at the beginning, then if the client requests information about the book, whether by topic or id (read operations), the front-end server searches for the book in the cache, if it exists, it takes it and sends it to the client. if the requested book was not found in the cache, the front-end server sends the request to one of the replicas of catalog servers according to load balancing, and takes the information of the request and stores it in the cache, and sends it to the client. All that has been said previously is in the case of read-only.

But in the case of the request process (write operation), the front-end server goes to cache, if the book is found there, it deletes it according to invalidate operation, and then goes to one of the order servers based on load balancing, then goes to the catalog server that corresponds to it and sees whether the book is in the catalog or not, in case it exists, it changes the book’s quantity and then the catalog sends a request to the second catalog (replica) that the value of the quantity has been changed, so the quantity value in the replica will also be changed exactly as another catalog server, and the same thing in the order after it is checked and everything is complete by sending a request to the other order replica about a new order.

Improvements:

It is possible to develop the project by increasing the number of replicas and reducing pressure on the main servers, it is also possible to update the cache in case of write operation instead of deleting the value.

Outputs:

We used five virtual machines, and each virtual had an IP address, and we traveled between five servers, via postman, and we also make a client-side which sends requests and get responses.

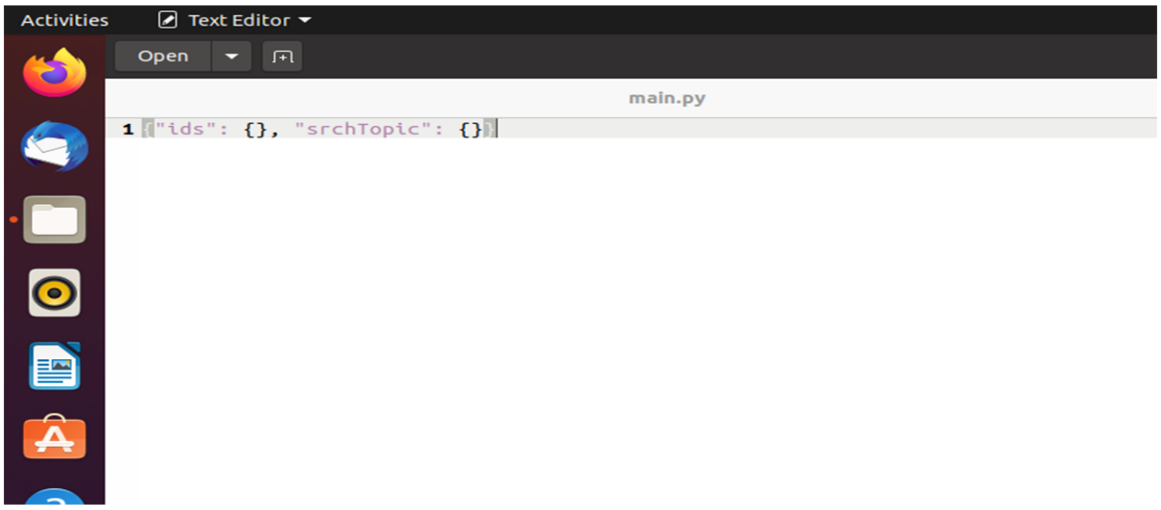
Front-end Server output:

The front-end server supports three processes. It receives requests, stores them in a cache, and then returns responses.

Operations:

- **Empty Cache**:

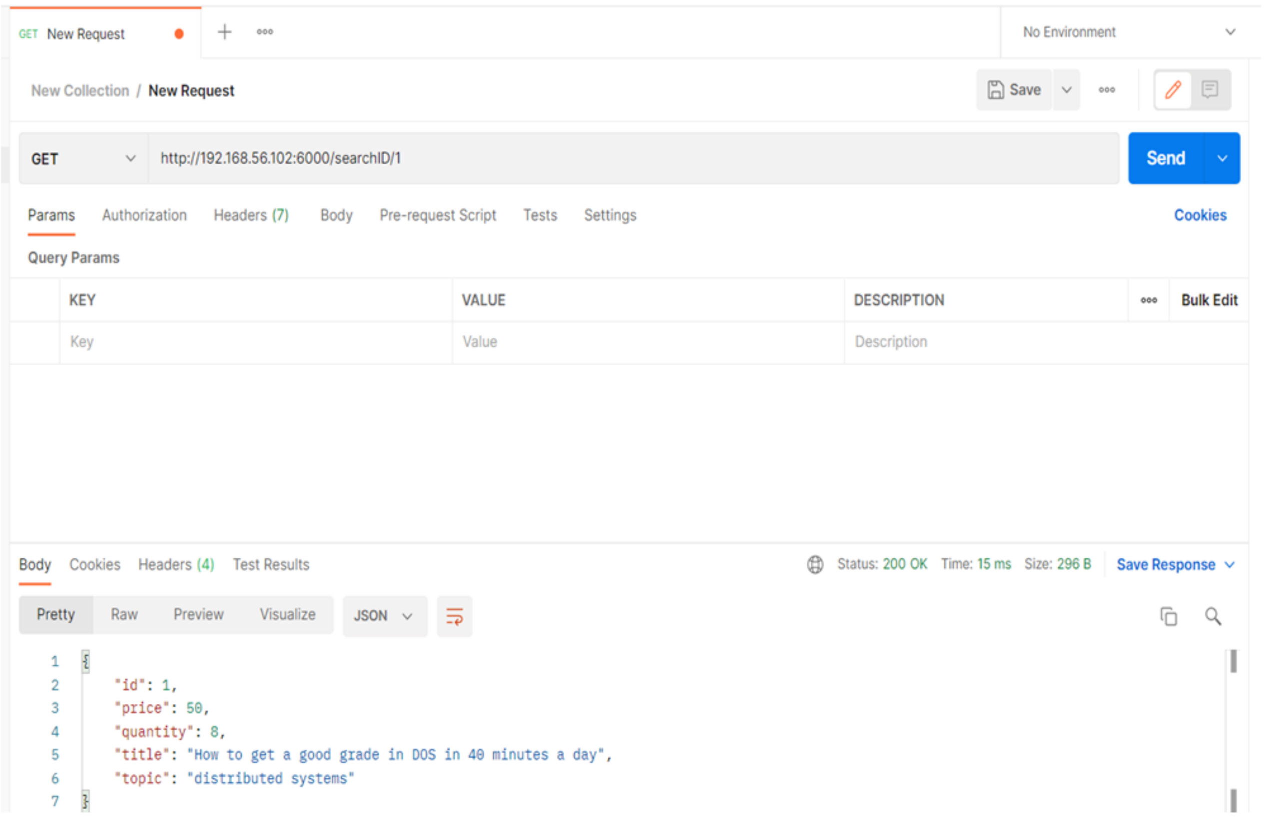
Before creating any operation, whether search or order.



```
{"ids": {}, "srchTopic": {}}
```

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

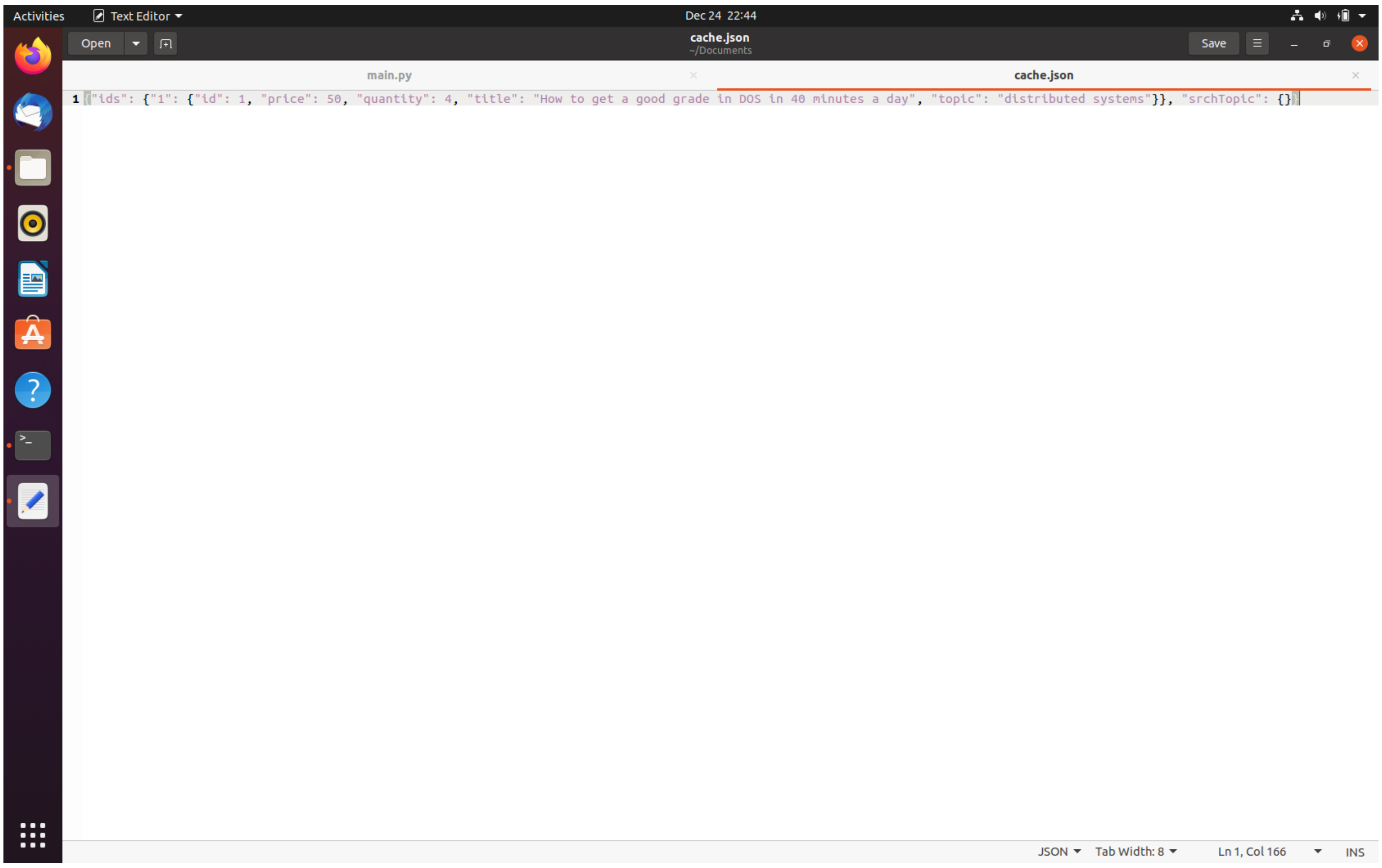
Search by ID, id=1. It stores book information id =1 in the cache, in the place where the read operations of search by ID are stored. ("ids")



GET http://192.168.56.102:6000/searchID/1

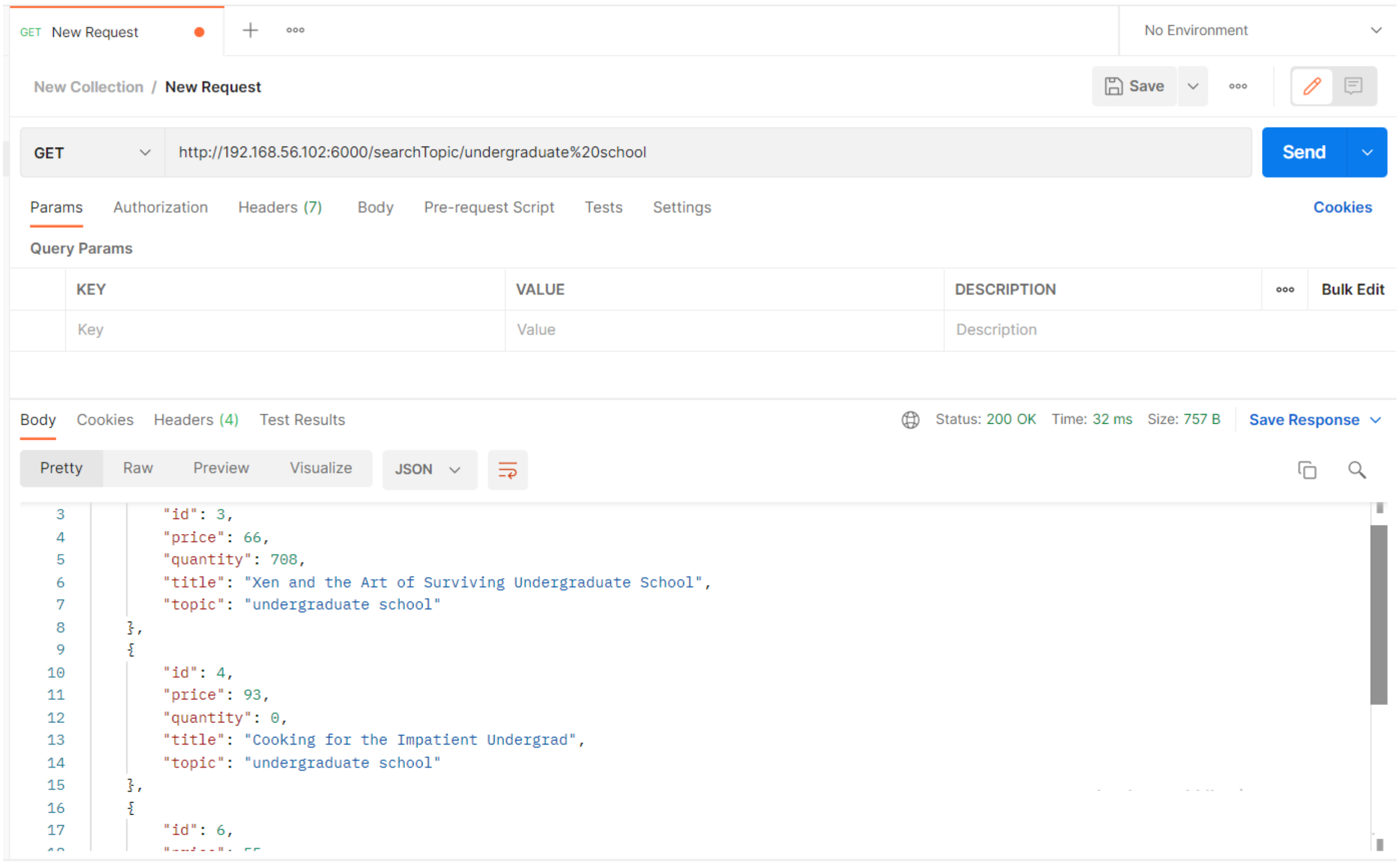
Status: 200 OK Time: 15 ms Size: 296 B

```
{
    "id": 1,
    "price": 50,
    "quantity": 8,
    "title": "How to get a good grade in DOS in 40 minutes a day",
    "topic": "distributed systems"
}
```



```
{"ids": {"1": {"id": 1, "price": 50, "quantity": 4, "title": "How to get a good grade in DOS in 40 minutes a day", "topic": "distributed systems"}}, "srchTopic": {}}
```

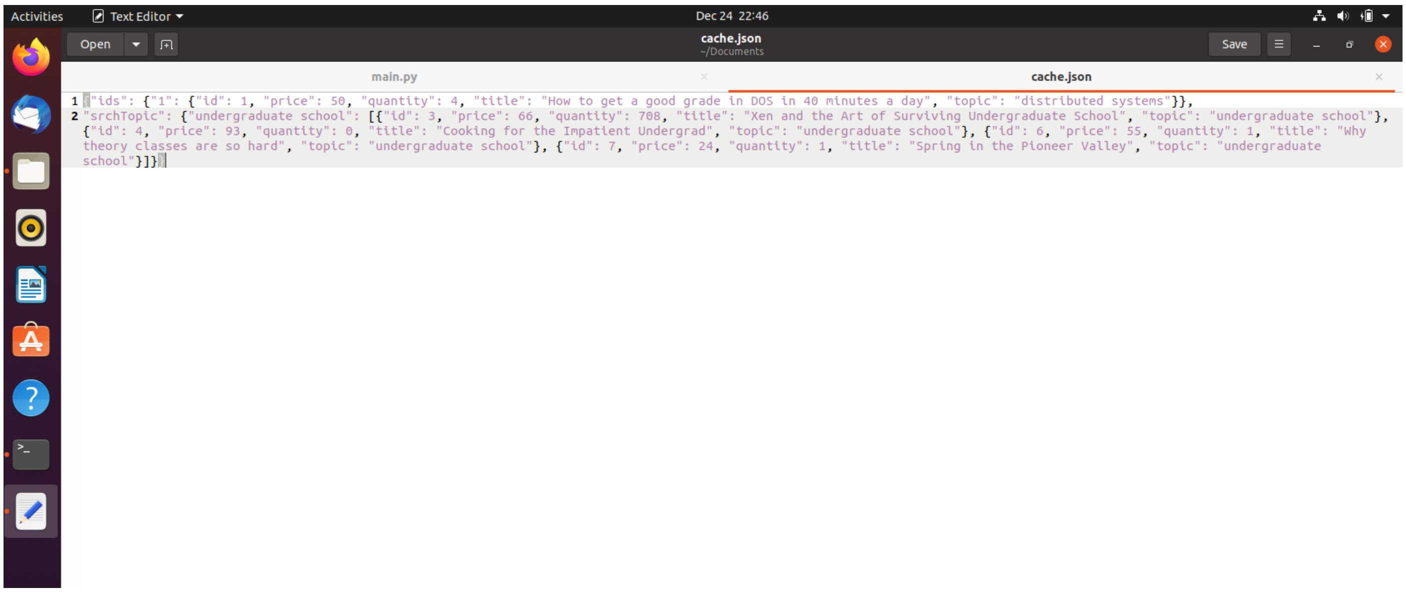
Search by topic, topic=undergraduate school, It stores books information in the cache, in the place where the read operations of search by topic are stored. (“srchTopic”)



GET http://192.168.56.102:6000/searchTopic/undergraduate%20school

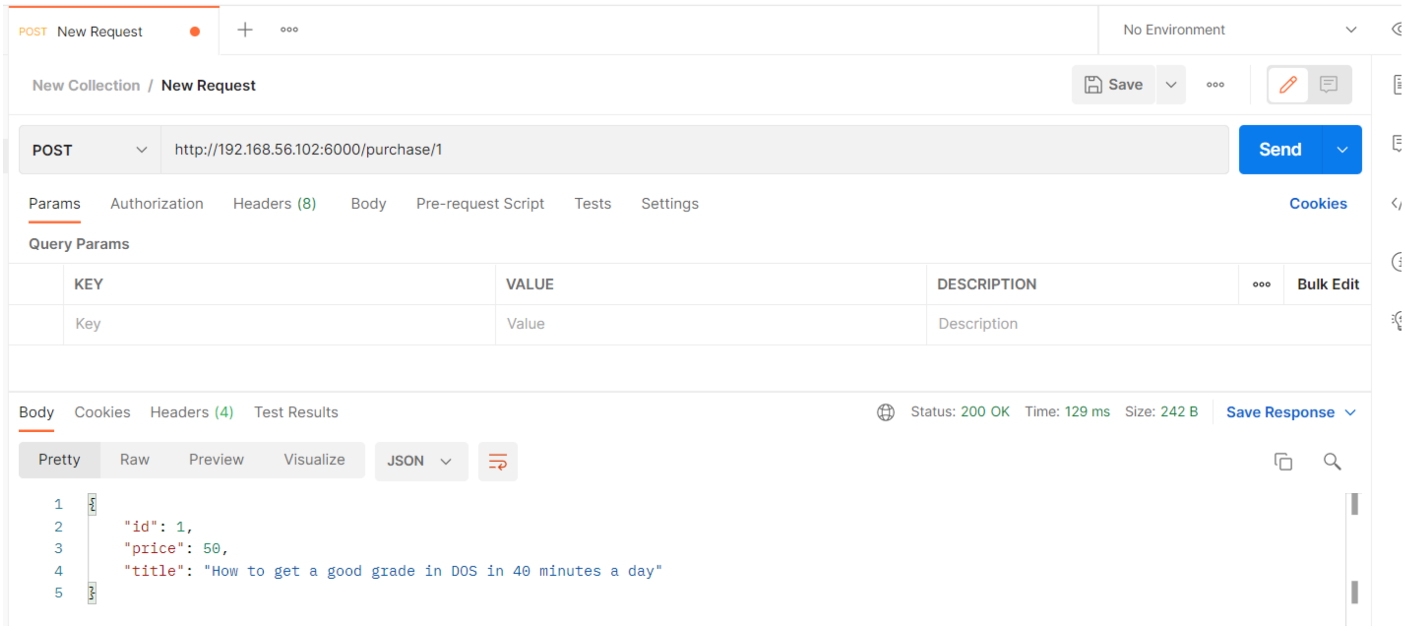
Status: 200 OK Time: 32 ms Size: 757 B

```
        "id": 3,
        "price": 66,
        "quantity": 708,
        "title": "Xen and the Art of Surviving Undergraduate School",
        "topic": "undergraduate school"
    },
    {
        "id": 4,
        "price": 93,
        "quantity": 0,
        "title": "Cooking for the Impatient Undergrad",
        "topic": "undergraduate school"
    },
    {
        "id": 6,
```



```
{"ids": {"1": {"id": 1, "price": 50, "quantity": 4, "title": "How to get a good grade in DOS in 40 minutes a day", "topic": "distributed systems"}},
"srchTopic": {"undergraduate school": [{"id": 3, "price": 66, "quantity": 708, "title": "Xen and the Art of Surviving Undergraduate School", "topic": "undergraduate school"},
{"id": 4, "price": 93, "quantity": 0, "title": "Cooking for the Impatient Undergrad", "topic": "undergraduate school"}, {"id": 6, "price": 55, "quantity": 1, "title": "Why
theory classes are so hard", "topic": "undergraduate school"}, {"id": 7, "price": 24, "quantity": 1, "title": "Spring in the Pioneer Valley", "topic": "undergraduate
school"}]}}
```

- **Purchase (write operation): add order, specify an item number for purchase (POST):** order id = 3, it will be deleted from the cache.



POST New Request

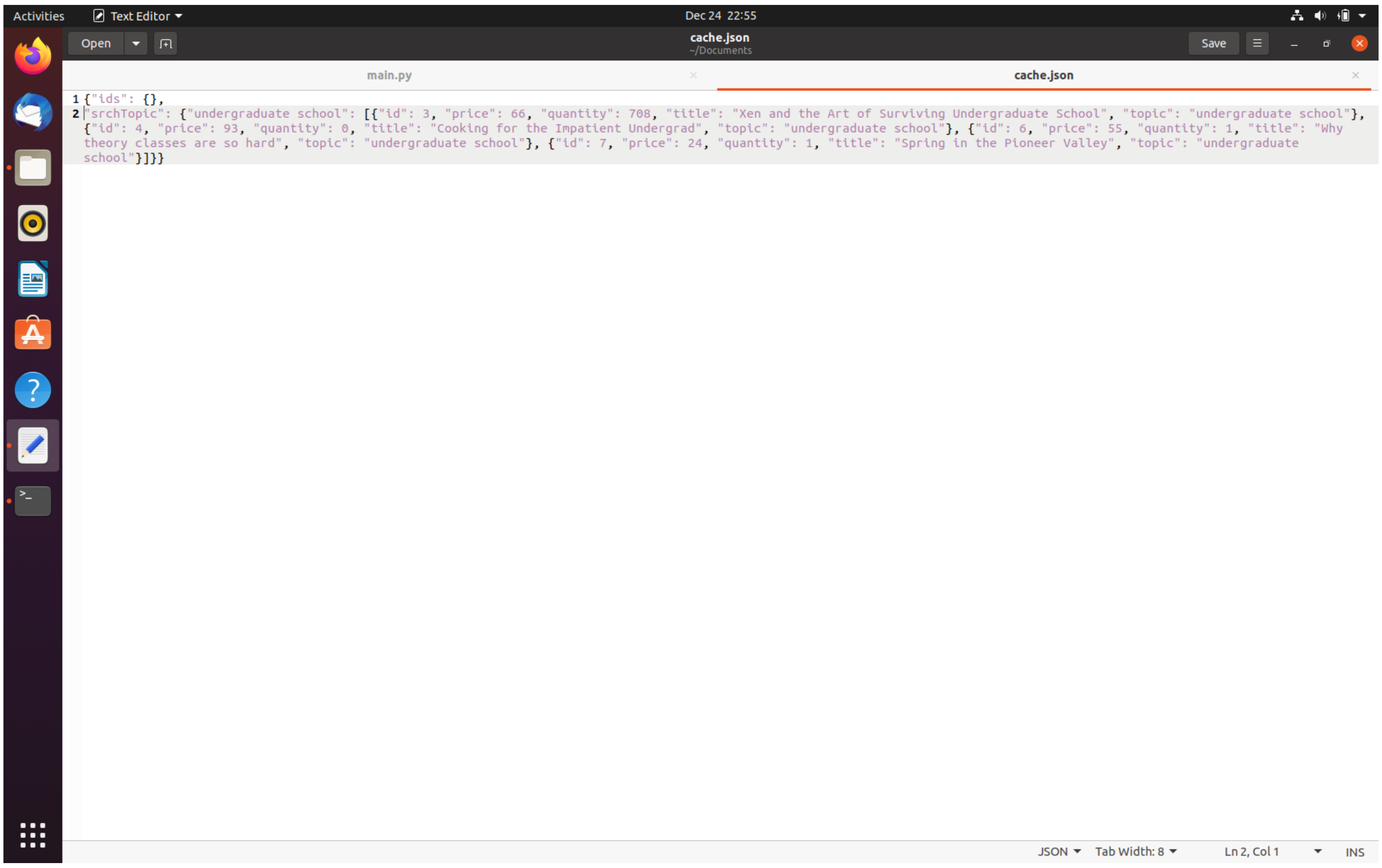
New Collection / New Request

POST http://192.168.56.102:6000/purchase/1

Status: 200 OK Time: 129 ms Size: 242 B

```
{
    "id": 1,
    "price": 50,
    "title": "How to get a good grade in DOS in 40 minutes a day"
}
```

as the picture below, updated the value will lead to removing the book information from the cache:



```
{"ids": {},
"srchTopic": {"undergraduate school": [{"id": 3, "price": 66, "quantity": 708, "title": "Xen and the Art of Surviving Undergraduate School", "topic": "undergraduate school"},
{"id": 4, "price": 93, "quantity": 0, "title": "Cooking for the Impatient Undergrad", "topic": "undergraduate school"}, {"id": 6, "price": 55, "quantity": 1, "title": "Why
theory classes are so hard", "topic": "undergraduate school"}, {"id": 7, "price": 24, "quantity": 1, "title": "Spring in the Pioneer Valley", "topic": "undergraduate
school"}]}}
```

Calculations:

We used time functions inside the client to calculate response time and latency:

Operation/Response time	**With cache**	**Without cache**
Search by id	RT=0.005361	RT=0.026059
Search by topic	RT=0.00457	RT=0.051302
Purchase	RT=0.121101	RT=0.129192

We issue orders (catalog updates/write operation) to invalidate the cache and maintain cache consistency:

What are the overhead of cache consistency operations?
0.02634215354

What is the latency of a subsequent request that sees a cache miss?
0.04684567451477051