An-Najah National University
Faculty of Engineering and IT

جامعة النجاح الوطنية
كلية الهندسة وتكنولوجيا المعلومات

Distributed operating systems Project Report #2
An-Najah National University
Computer Engineering Department

Students:

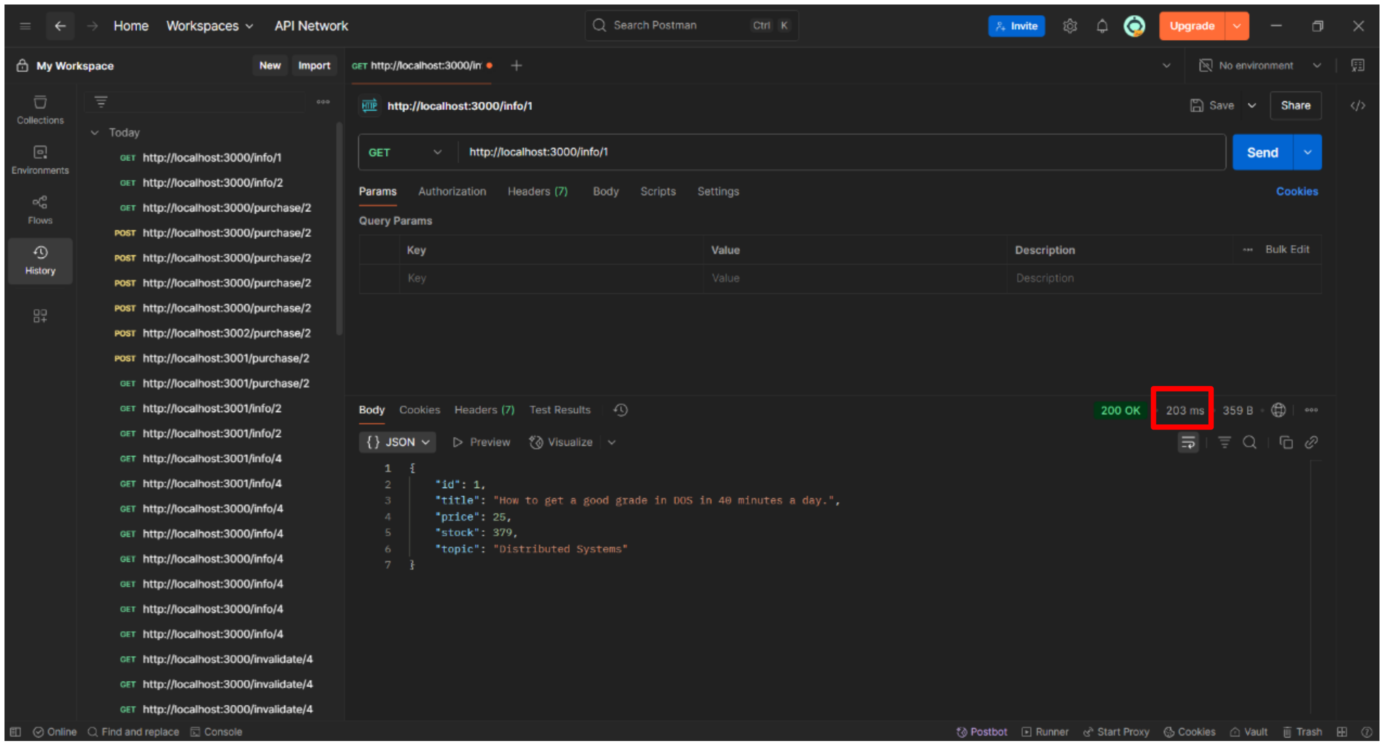
1. Ahmad Dweikat ID: 12042774
2. Abd Alhameed Mizher ID: 12029826

Submission Date: 24th May 2025

PART 1: Replication and Caching

In this part, we will add replication and caching to improve request processing latency.

- When send GET request for the first time (before caching the data)



History:

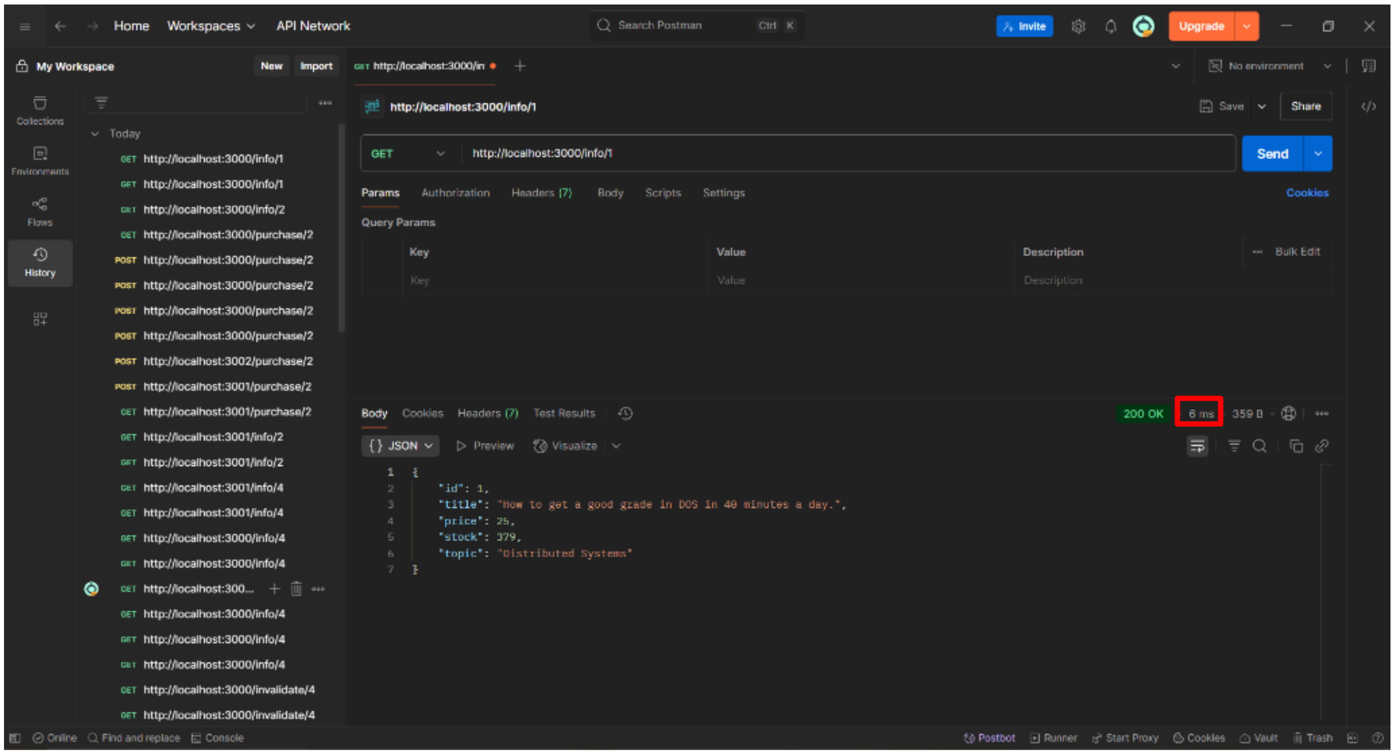
```
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/2
GET  http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3002/purchase/2
POST http://localhost:3001/purchase/2
GET  http://localhost:3001/purchase/2
GET  http://localhost:3001/info/2
GET  http://localhost:3001/info/2
GET  http://localhost:3001/info/4
GET  http://localhost:3001/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/invalidate/4
GET  http://localhost:3000/invalidate/4
GET  http://localhost:3000/invalidate/4
```

GET http://localhost:3000/info/1

200 OK 203 ms 359 B

```
{
    "id": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 25,
    "stock": 379,
    "topic": "Distributed Systems"
}
```

- When send GET request for the second time for the same id (after caching the data)



History:

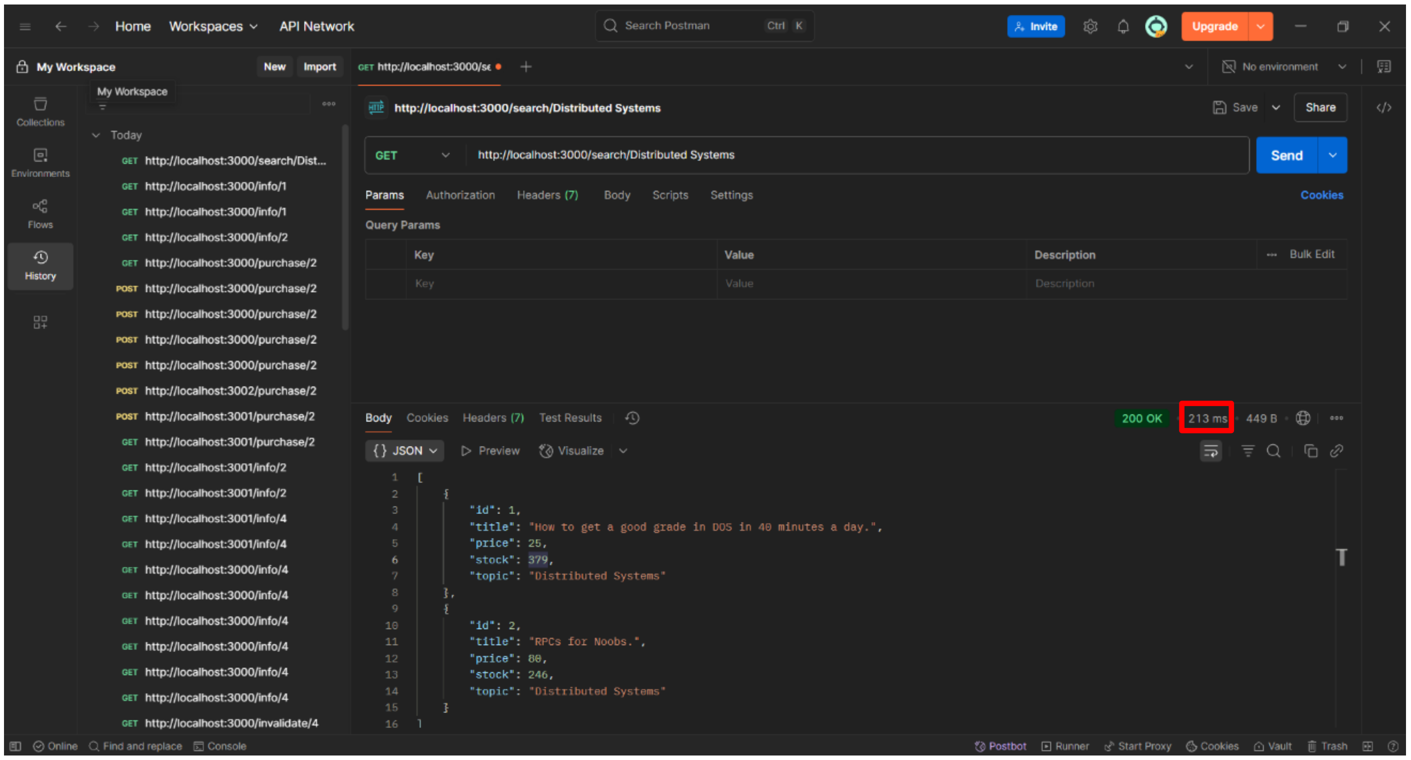
```
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/2
GET  http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3002/purchase/2
POST http://localhost:3001/purchase/2
GET  http://localhost:3001/purchase/2
GET  http://localhost:3001/info/2
GET  http://localhost:3001/info/2
GET  http://localhost:3001/info/4
GET  http://localhost:3001/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:300...
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/invalidate/4
GET  http://localhost:3000/invalidate/4
```

GET http://localhost:3000/info/1

200 OK 6 ms 359 B

```
{
    "id": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 25,
    "stock": 379,
    "topic": "Distributed Systems"
}
```

- When send GET request for the first time for the book topic (before caching the data)

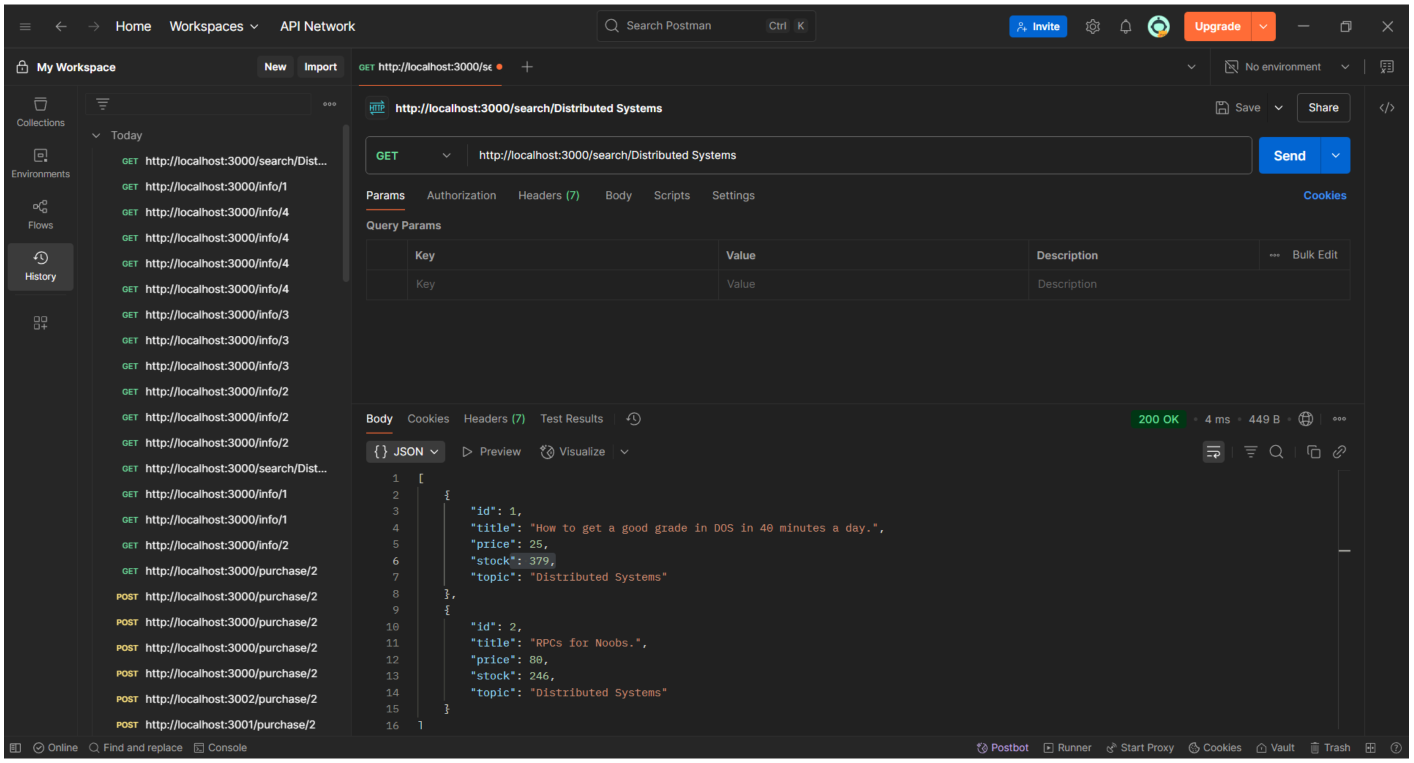


GET http://localhost:3000/search/Distributed Systems

200 OK · 213 ms · 449 B

```
[
    {
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day.",
        "price": 25,
        "stock": 379,
        "topic": "Distributed Systems"
    },
    {
        "id": 2,
        "title": "RPCs for Noobs.",
        "price": 80,
        "stock": 246,
        "topic": "Distributed Systems"
    }
]
```

- When send GET request for the second time for the same book topic (after caching the data)

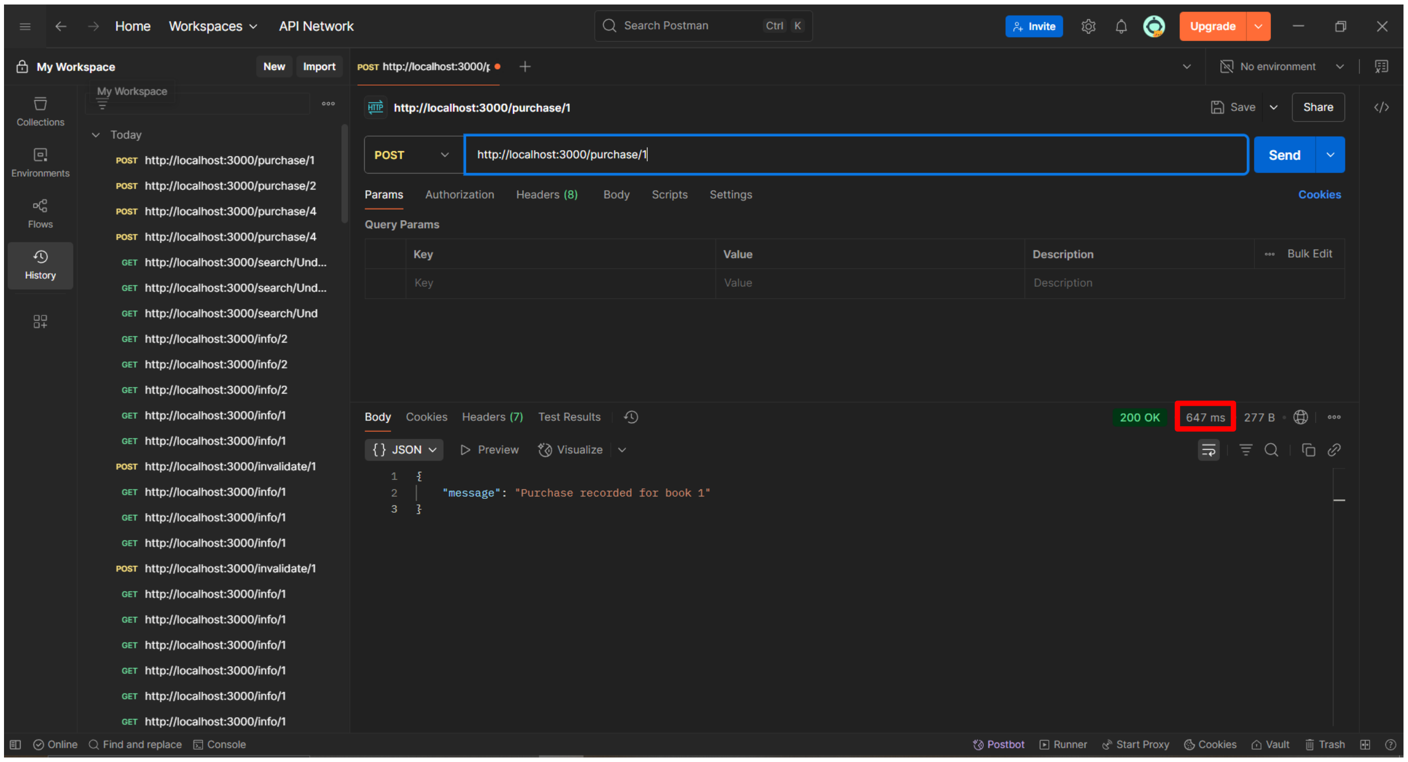


GET http://localhost:3000/search/Distributed Systems

200 OK · 4 ms · 449 B

```
[
    {
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day.",
        "price": 25,
        "stock": 379,
        "topic": "Distributed Systems"
    },
    {
        "id": 2,
        "title": "RPCs for Noobs.",
        "price": 80,
        "stock": 246,
        "topic": "Distributed Systems"
    }
]
```

- When send **POST** request for the first time for purchase a book (before caching the data)

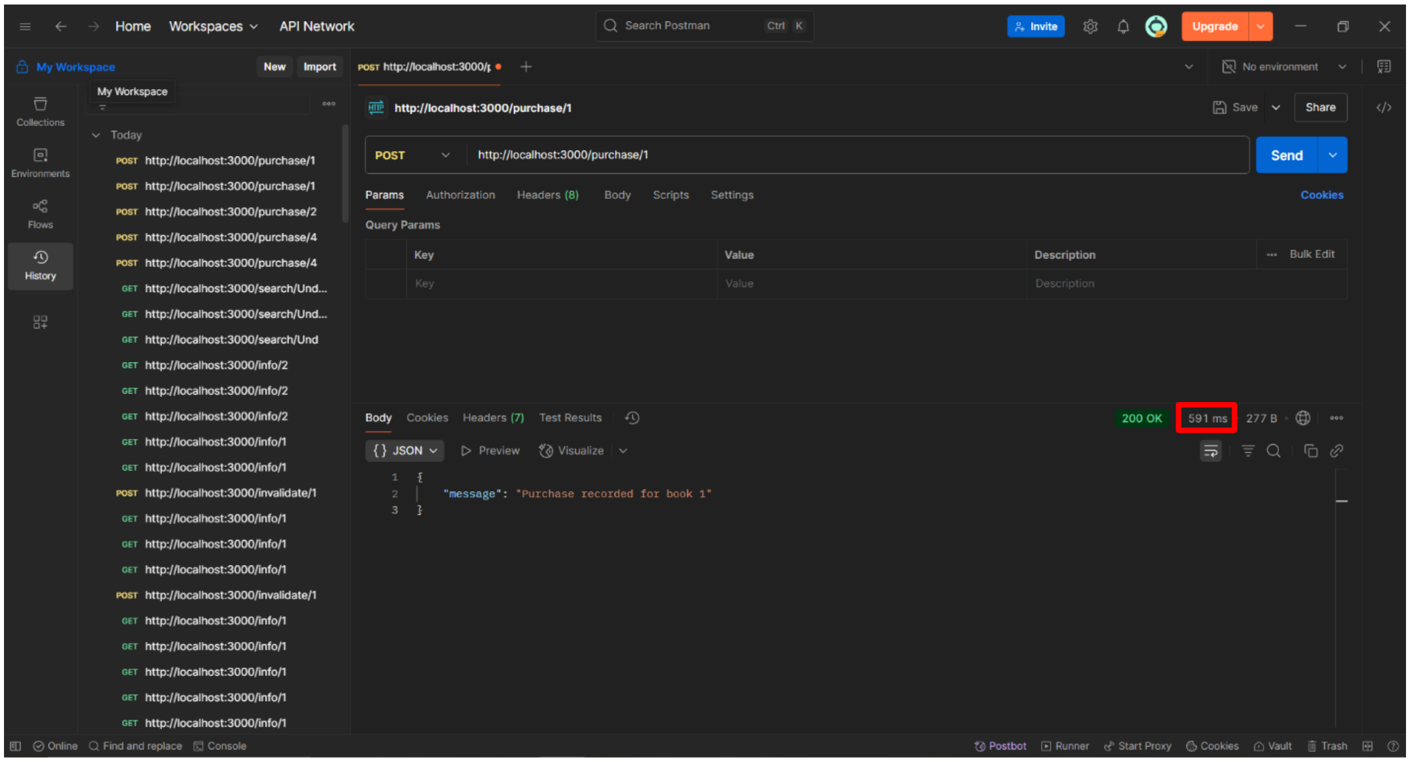


POST http://localhost:3000/purchase/1

200 OK 647 ms 277 B

```
{
    "message": "Purchase recorded for book 1"
}
```

- When send **POST** request for the second time for purchase a book (after caching the data)



POST http://localhost:3000/purchase/1

200 OK 591 ms 277 B

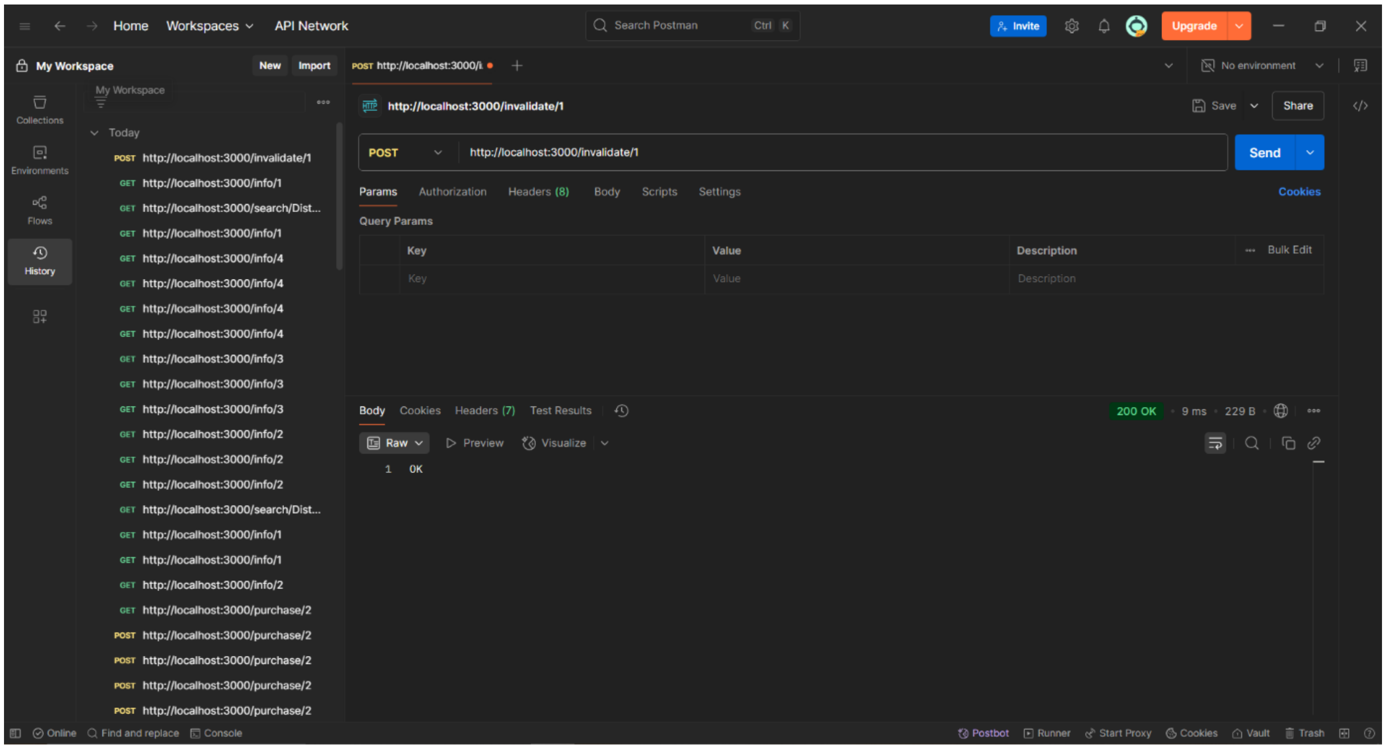
```
{
    "message": "Purchase recorded for book 1"
}
```

1- **Compute the average response time (query/buy) of your new systems. What is the response time with and without caching? How much does caching help?**

Type/Command	info	search	purchase
Without caching	203	213	647
With caching	6	4	591
Compare	203/6 = 33.83 faster than without caching	231/4 = 57.75 faster than without caching	647/591 = 1.09 faster than without caching

Invalidate Message

- When send **POST** request for invalidate the id book that saved in the cache it will be removed from the cache.



```
POST http://localhost:3000/invalidate/1
```

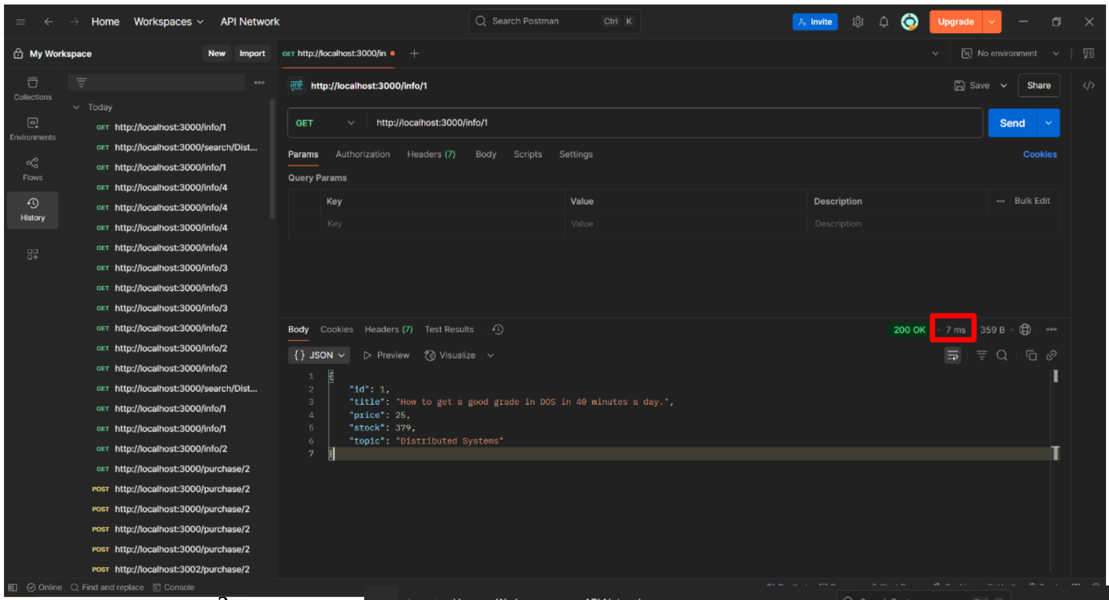
200 OK · 9 ms · 229 B

```
1  OK
```

History (Today):

```
POST http://localhost:3000/invalidate/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/search/Dist...
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/2
GET  http://localhost:3000/info/2
GET  http://localhost:3000/info/2
GET  http://localhost:3000/search/Dist...
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/2
GET  http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
```

- Here we have 2 picture once before invalidate and one after invlaidate



GET http://localhost:3000/info/1

History (Today):

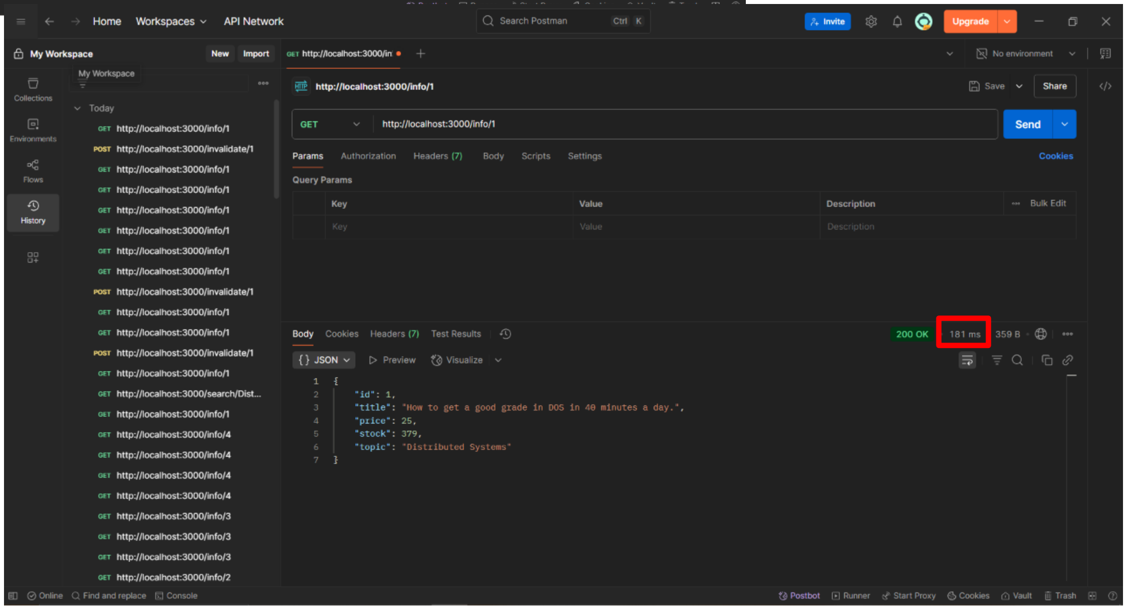
```
GET  http://localhost:3000/info/1
GET  http://localhost:3000/search/Dist...
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/2
GET  http://localhost:3000/info/2
GET  http://localhost:3000/info/2
GET  http://localhost:3000/search/Dist...
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/2
GET  http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3000/purchase/2
POST http://localhost:3002/purchase/2
```

200 OK · 7 ms · 359 B

```json
{
    "id": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 25,
    "stock": 379,
    "topic": "Distributed Systems"
}
```

Before Invalidate

a



GET http://localhost:3000/info/1

History (Today):

```
GET  http://localhost:3000/info/1
POST http://localhost:3000/invalidate/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
POST http://localhost:3000/invalidate/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/1
POST http://localhost:3000/invalidate/1
GET  http://localhost:3000/info/1
GET  http://localhost:3000/search/Dist...
GET  http://localhost:3000/info/1
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/4
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/3
GET  http://localhost:3000/info/2
```

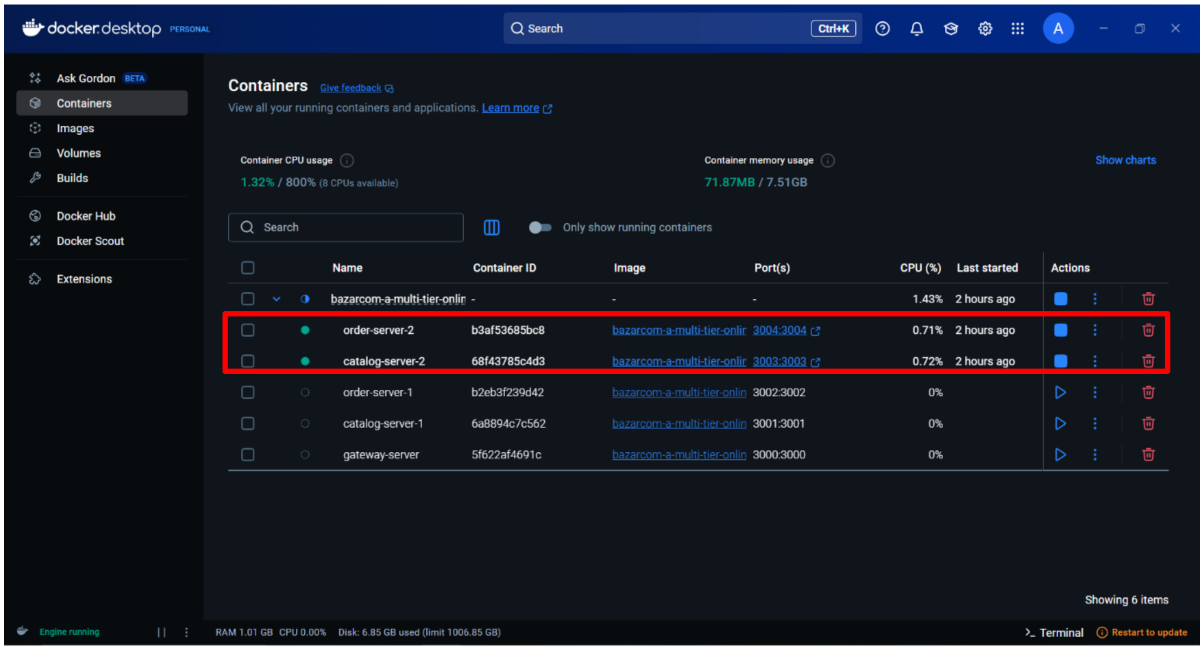
200 OK · 181 ms · 359 B

```json
{
    "id": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 25,
    "stock": 379,
    "topic": "Distributed Systems"
}
```

After Invalidate

Replica Servers

- Now here I made and replica server for catalog and for order that mean when the main or first server down the replica will run.



Container CPU usage 1.32% / 800% (8 CPUs available)

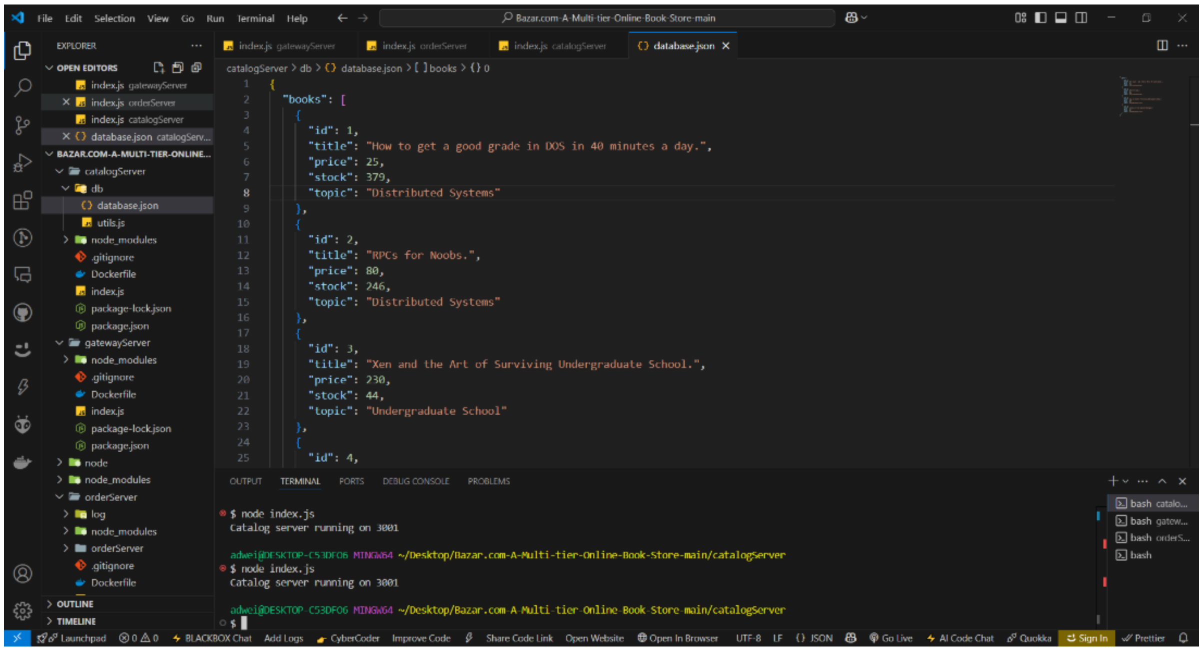
Container memory usage 71.87MB / 7.51GB

	Name	Container ID	Image	Port(s)	CPU (%)	Last started
◑	bazarcom-a-multi-tier-onlin	-	-	-	1.43%	2 hours ago
●	order-server-2	b3af53685bc8	bazarcom-a-multi-tier-onlir	3004:3004	0.71%	2 hours ago
●	catalog-server-2	68f43785c4d3	bazarcom-a-multi-tier-onlir	3003:3003	0.72%	2 hours ago
○	order-server-1	b2eb3f239d42	bazarcom-a-multi-tier-onlin	3002:3002	0%	
○	catalog-server-1	6a8894c7c562	bazarcom-a-multi-tier-onlin	3001:3001	0%	
○	gateway-server	5f622af4691c	bazarcom-a-multi-tier-onlin	3000:3000	0%	

Showing 6 items

My replica servers on the Docker while the main or first servers on the host

- Now I will turn off for example the main catalog server (by pressing ctrl + c since I using this at vs code)

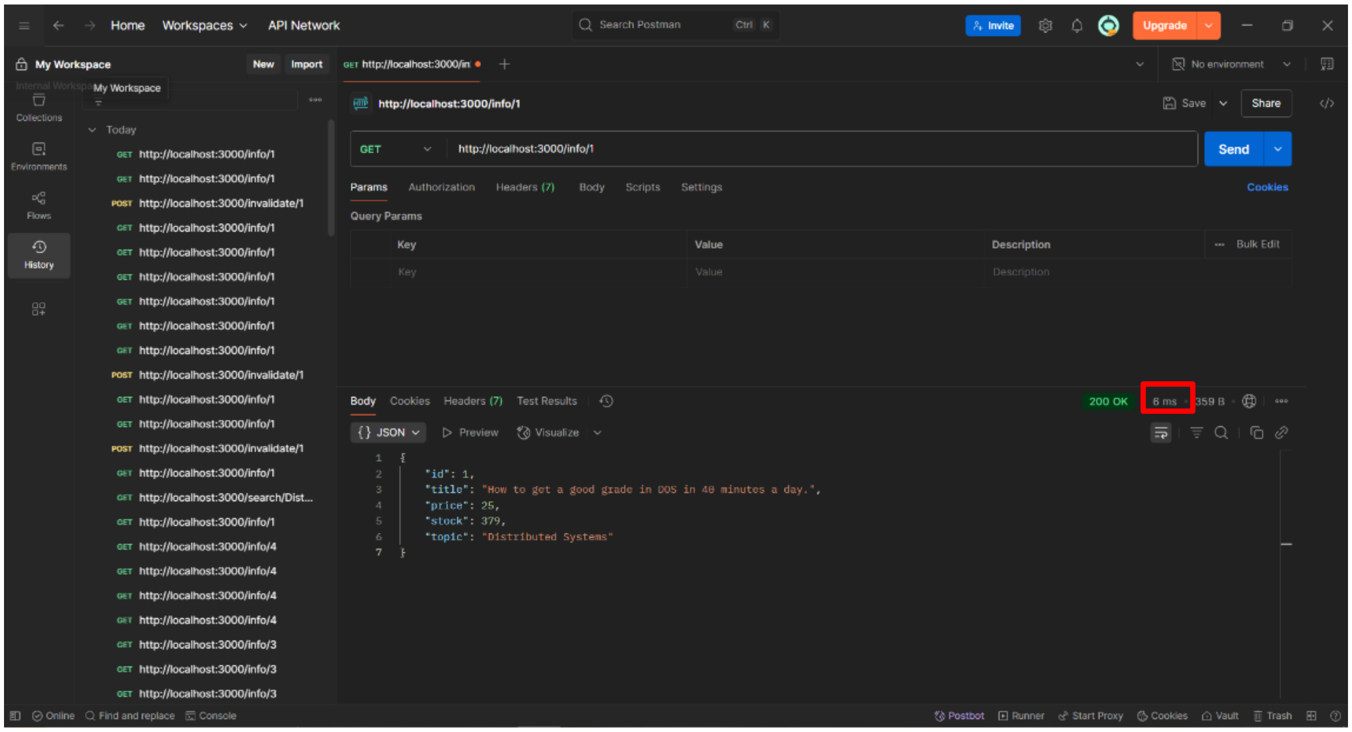


```json
{
  "books": [
    {
      "id": 1,
      "title": "How to get a good grade in DOS in 40 minutes a day.",
      "price": 25,
      "stock": 379,
      "topic": "Distributed Systems"
    },
    {
      "id": 2,
      "title": "RPCs for Noobs.",
      "price": 80,
      "stock": 246,
      "topic": "Distributed Systems"
    },
    {
      "id": 3,
      "title": "Xen and the Art of Surviving Undergraduate School.",
      "price": 230,
      "stock": 44,
      "topic": "Undergraduate School"
    },
    {
      "id": 4,
```

```
$ node index.js
Catalog server running on 3001

adwei@DESKTOP-C53DF06 MINGW64 ~/Desktop/Bazar.com-A-Multi-tier-Online-Book-Store-main/catalogServer
$ node index.js
Catalog server running on 3001

adwei@DESKTOP-C53DF06 MINGW64 ~/Desktop/Bazar.com-A-Multi-tier-Online-Book-Store-main/catalogServer
$
```

Here I turned off the catalog server now it will find book using replica server

- Here the result of using replica server

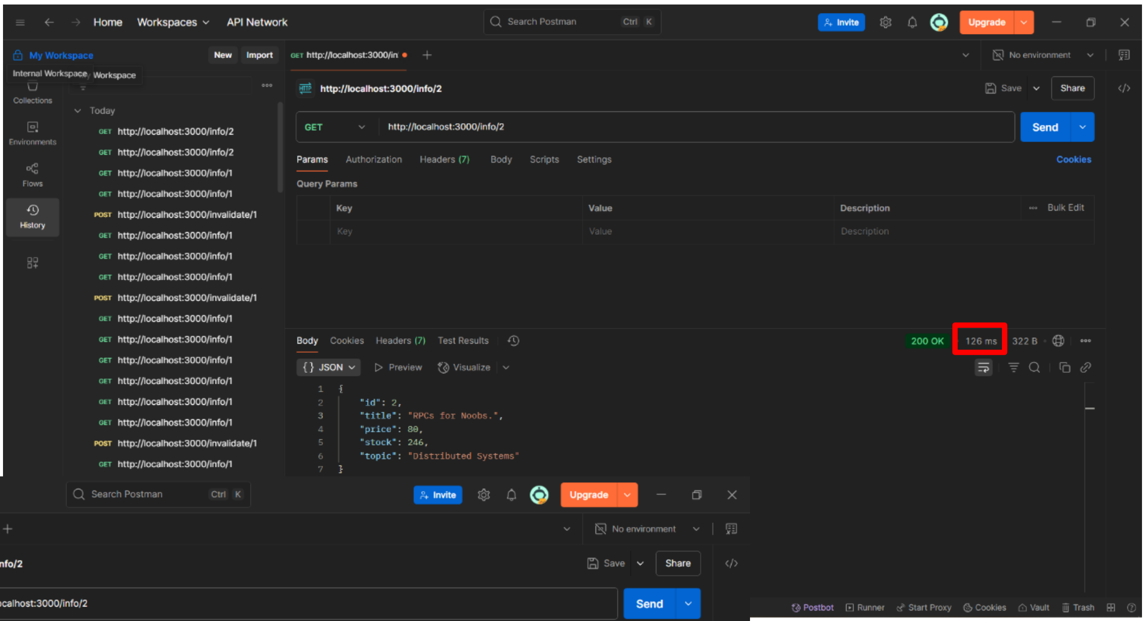


GET http://localhost:3000/info/1

200 OK 6 ms 359 B

```
{
    "id": 1,
    "title": "How to get a good grade in DOS in 40 minutes a day.",
    "price": 25,
    "stock": 379,
    "topic": "Distributed Systems"
}
```

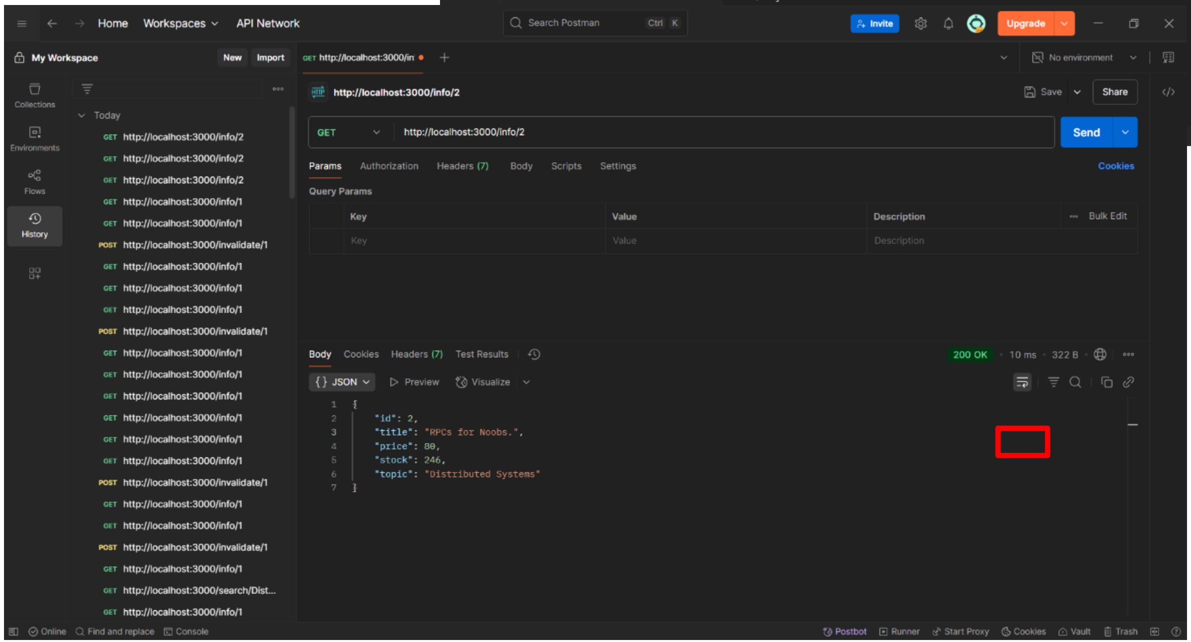
- Make an GET request to info using replica server



GET http://localhost:3000/info/2

200 OK 126 ms 322 B

```
{
    "id": 2,
    "title": "RPCs for Noobs.",
    "price": 80,
    "stock": 246,
    "topic": "Distributed Systems"
}
```

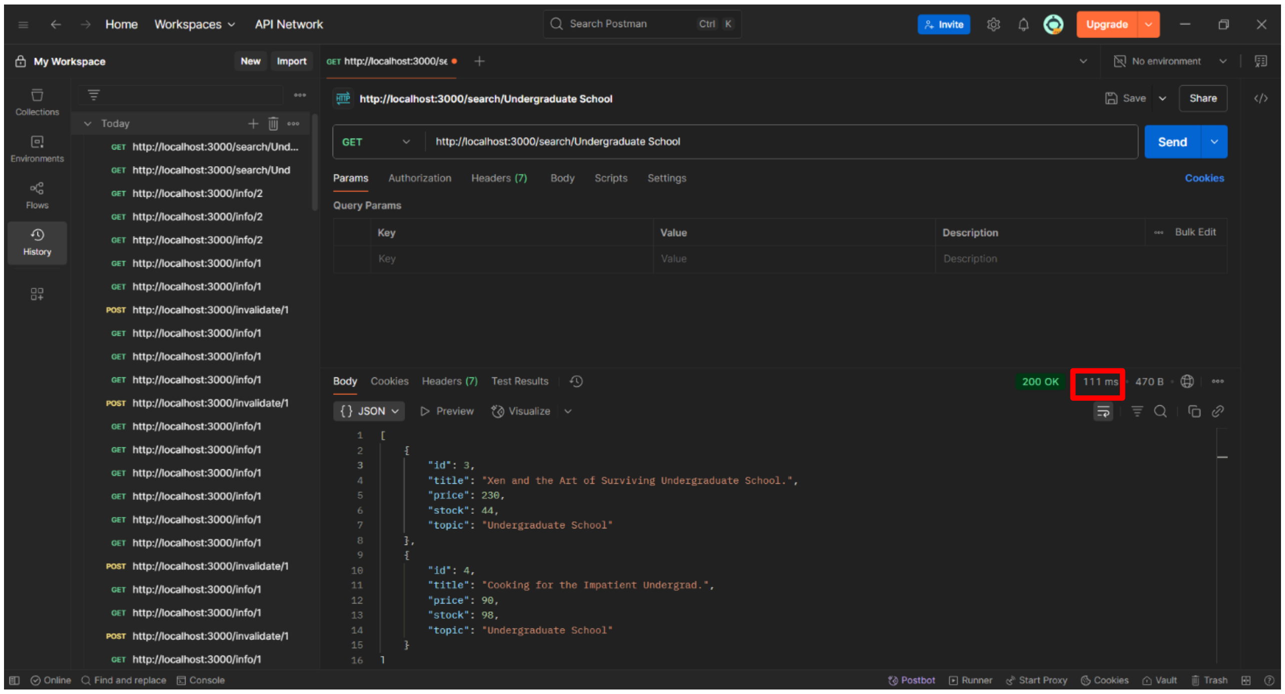


GET http://localhost:3000/info/2

200 OK 10 ms 322 B

```
{
    "id": 2,
    "title": "RPCs for Noobs.",
    "price": 80,
    "stock": 246,
    "topic": "Distributed Systems"
}
```

- Make an GET request to search using replica server

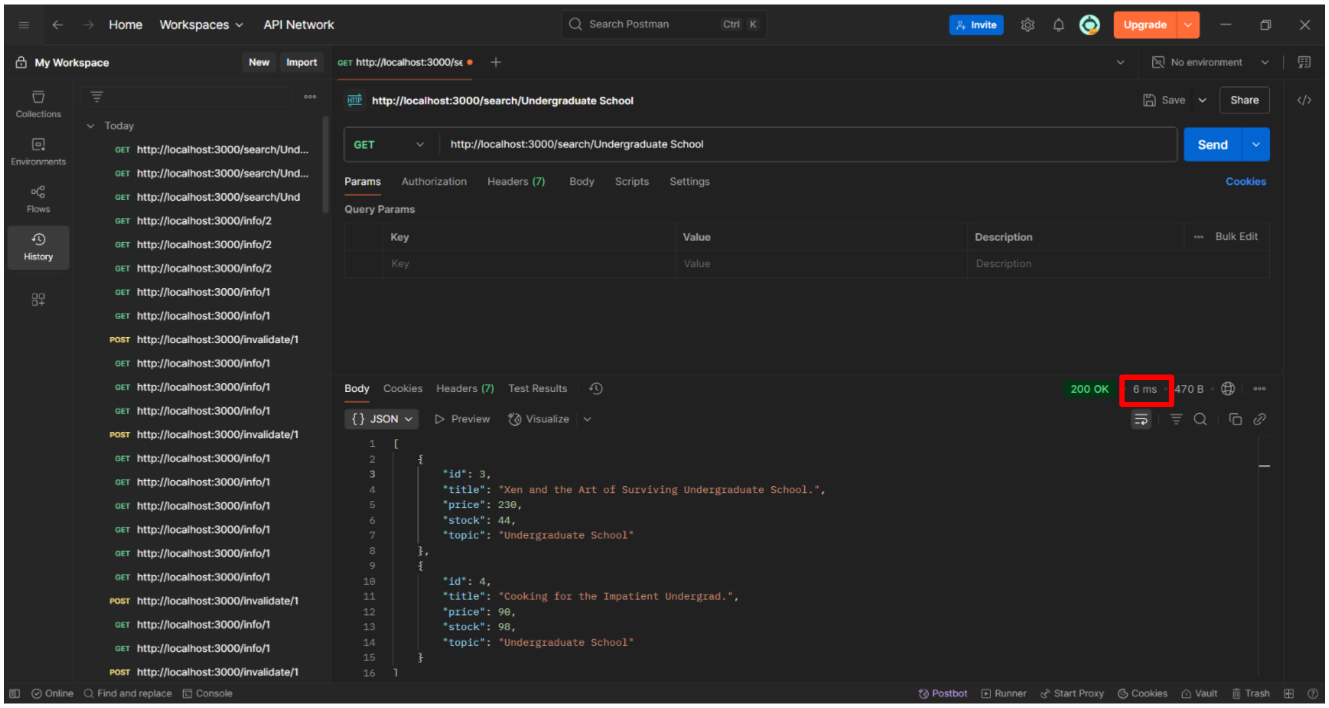


GET http://localhost:3000/search/Undergraduate School

200 OK · 111 ms · 470 B

```
[
    {
        "id": 3,
        "title": "Xen and the Art of Surviving Undergraduate School.",
        "price": 230,
        "stock": 44,
        "topic": "Undergraduate School"
    },
    {
        "id": 4,
        "title": "Cooking for the Impatient Undergrad.",
        "price": 90,
        "stock": 98,
        "topic": "Undergraduate School"
    }
]
```

- Make an GET request to search using replica server (second time)

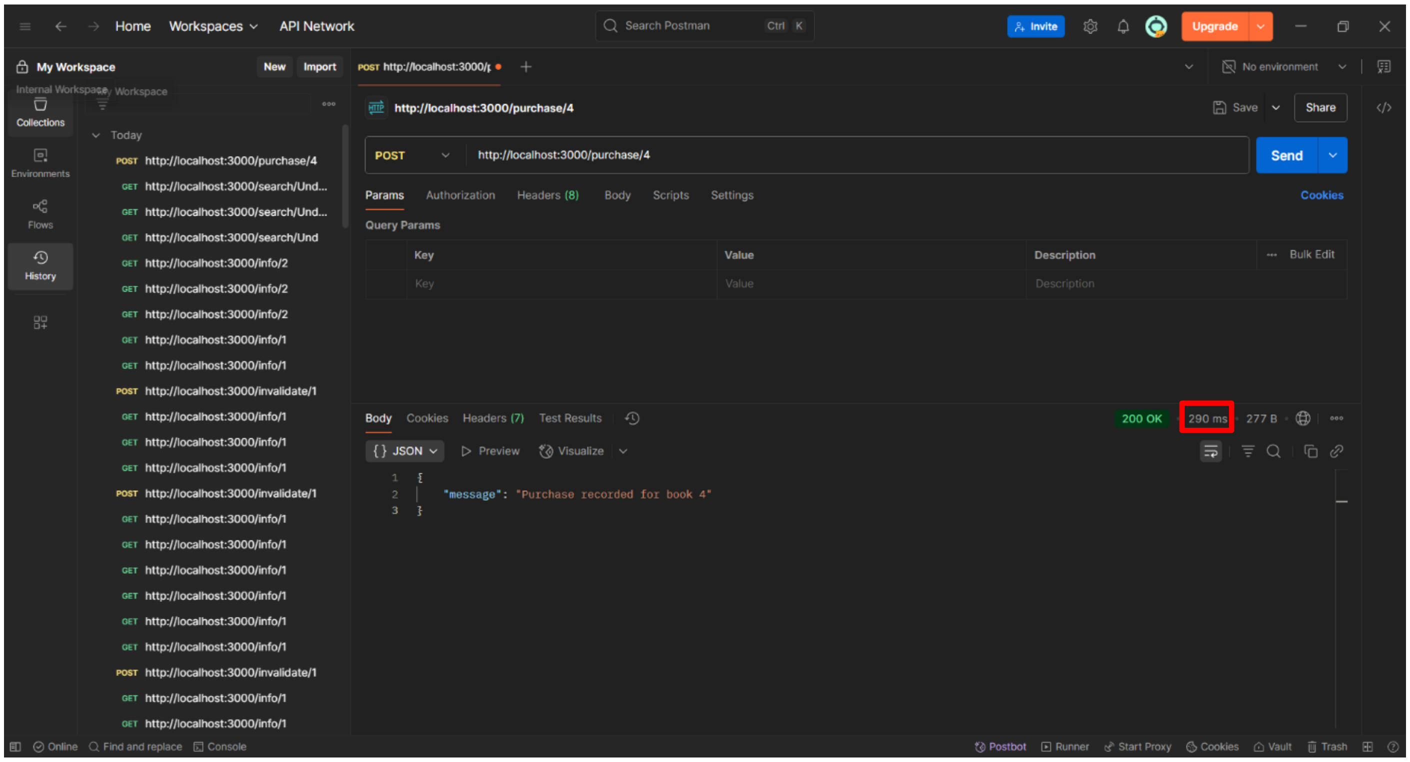


GET http://localhost:3000/search/Undergraduate School

200 OK · 6 ms · 470 B

```
[
    {
        "id": 3,
        "title": "Xen and the Art of Surviving Undergraduate School.",
        "price": 230,
        "stock": 44,
        "topic": "Undergraduate School"
    },
    {
        "id": 4,
        "title": "Cooking for the Impatient Undergrad.",
        "price": 90,
        "stock": 98,
        "topic": "Undergraduate School"
    }
]
```

- Make an POST request to purchase for first time

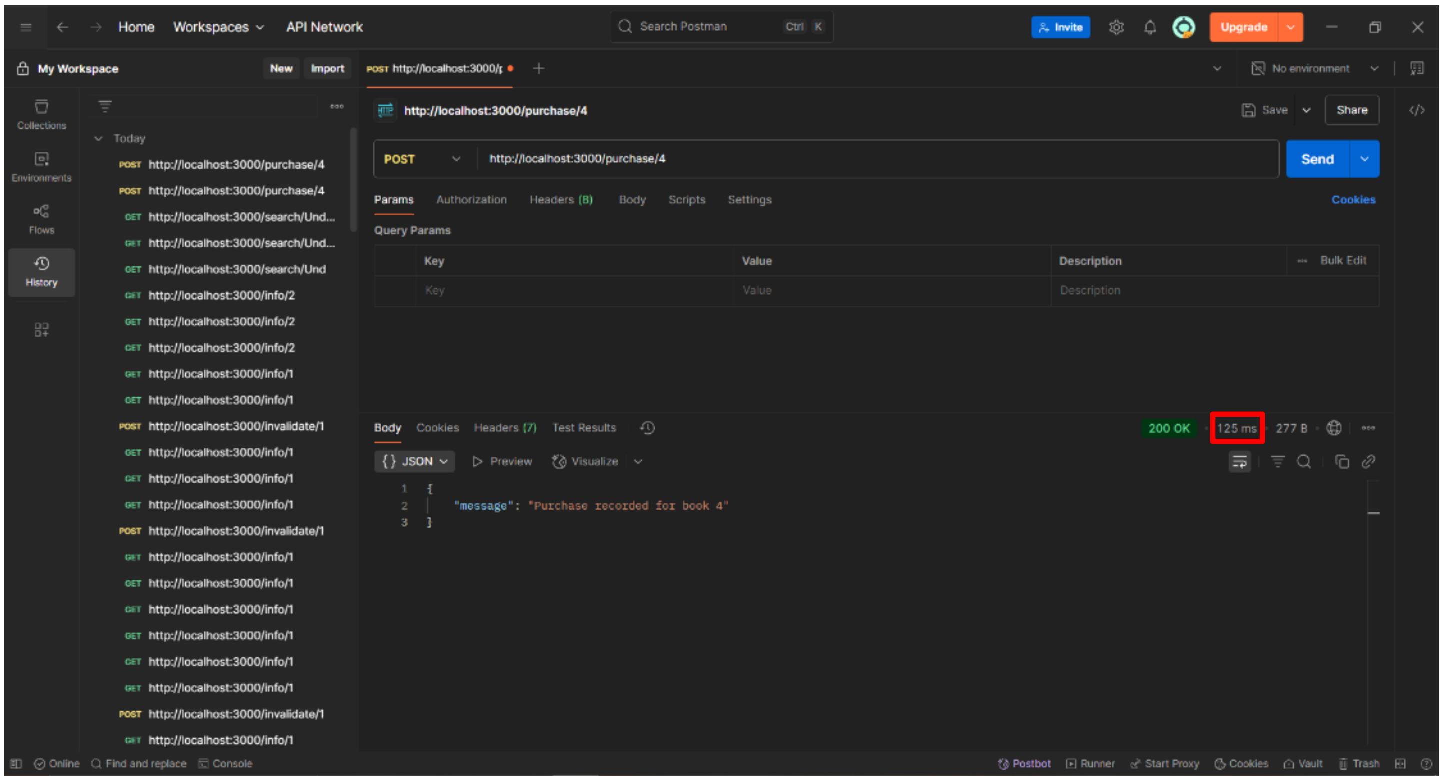


POST http://localhost:3000/purchase/4

200 OK 290 ms 277 B

```
{
    "message": "Purchase recorded for book 4"
}
```

- Send an POST request to purchse an book for second time



POST http://localhost:3000/purchase/4

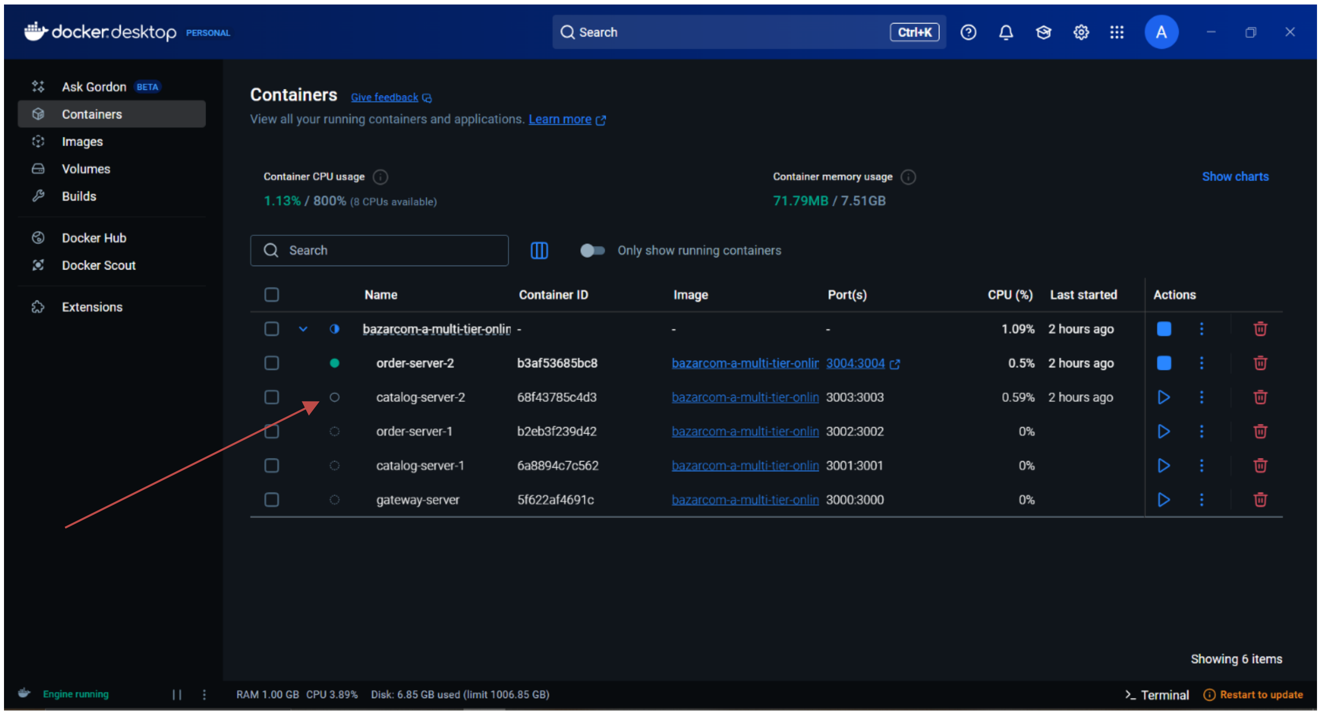
200 OK 125 ms 277 B

```
{
    "message": "Purchase recorded for book 4"
}
```

- **The Result**

Type/Command	info	search	purchase
Without caching	126	111	290
With caching	10	6	125
Compare	126/10 = 12.6 faster than without caching	111/6 = 18.5 faster than without caching	290/125 = 2.32 faster than without caching

- Now I will turn off the replica server



Containers

View all your running containers and applications. Learn more

Container CPU usage
1.13% / 800% (8 CPUs available)

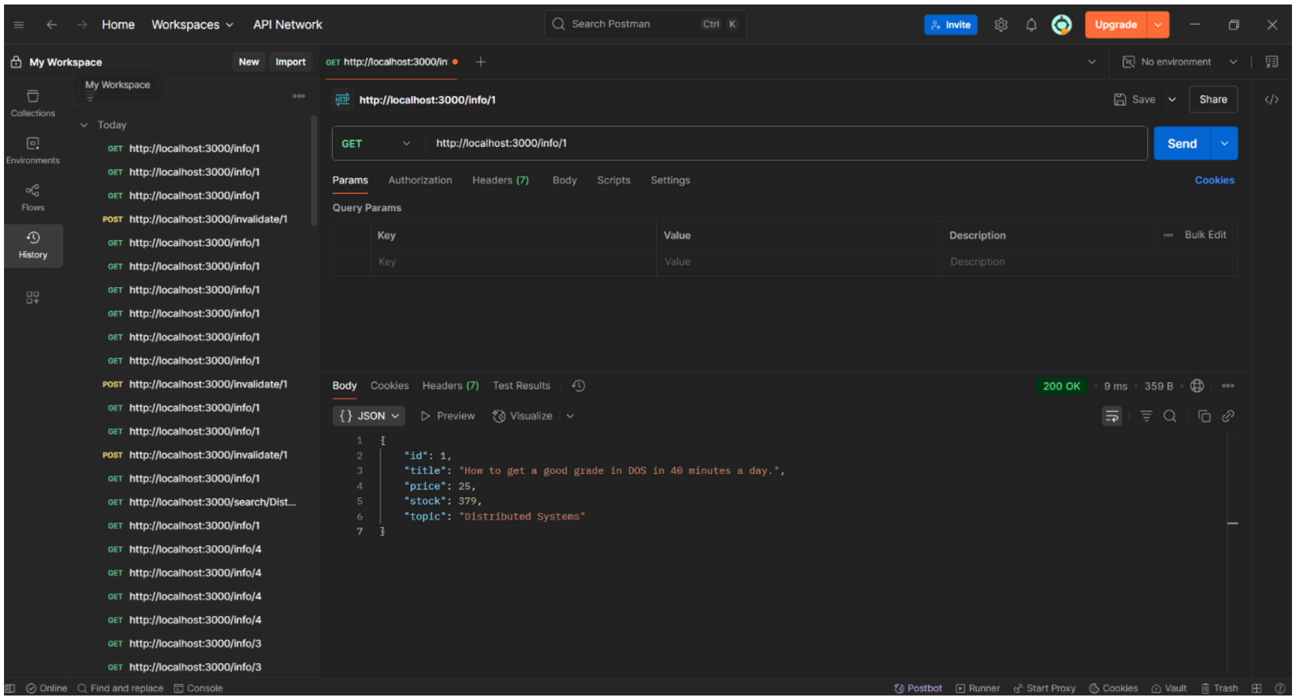
Container memory usage
71.79MB / 7.51GB

Name	Container ID	Image	Port(s)	CPU (%)	Last started
bazarcom-a-multi-tier-onlin	-	-	-	1.09%	2 hours ago
order-server-2	b3af53685bc8	bazarcom-a-multi-tier-onlin	3004:3004	0.5%	2 hours ago
catalog-server-2	68f43785c4d3	bazarcom-a-multi-tier-onlin	3003:3003	0.59%	2 hours ago
order-server-1	b2eb3f239d42	bazarcom-a-multi-tier-onlin	3002:3002	0%	
catalog-server-1	6a8894c7c562	bazarcom-a-multi-tier-onlin	3001:3001	0%	
gateway-server	5f622af4691c	bazarcom-a-multi-tier-onlin	3000:3000	0%	

Showing 6 items

RAM 1.00 GB CPU 3.89% Disk: 6.85 GB used (limit 1006.85 GB)

- Now if I send GET request the result will apear since it stored in the cache



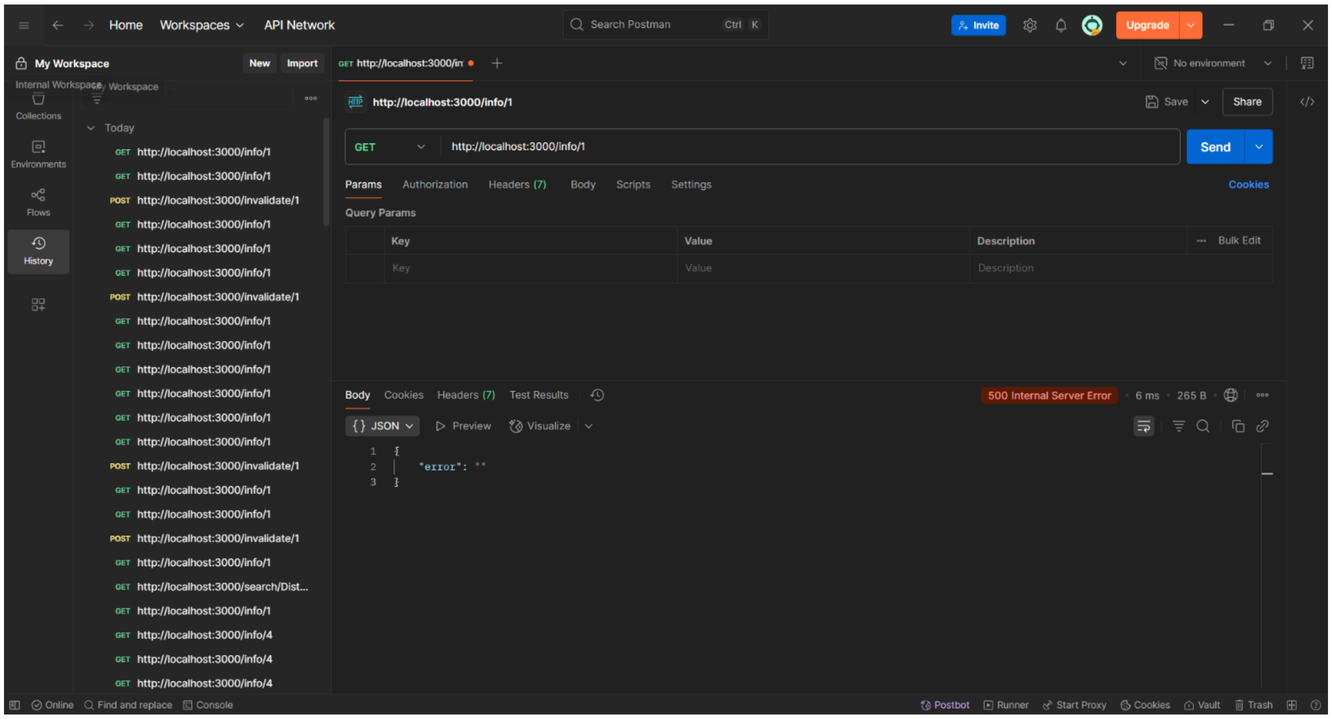
GET http://localhost:3000/info/1

200 OK · 9 ms · 359 B

```
{
  "id": 1,
  "title": "How to get a good grade in DOS in 40 minutes a day.",
  "price": 25,
  "stock": 379,
  "topic": "Distributed Systems"
}
```

Invalidate Message

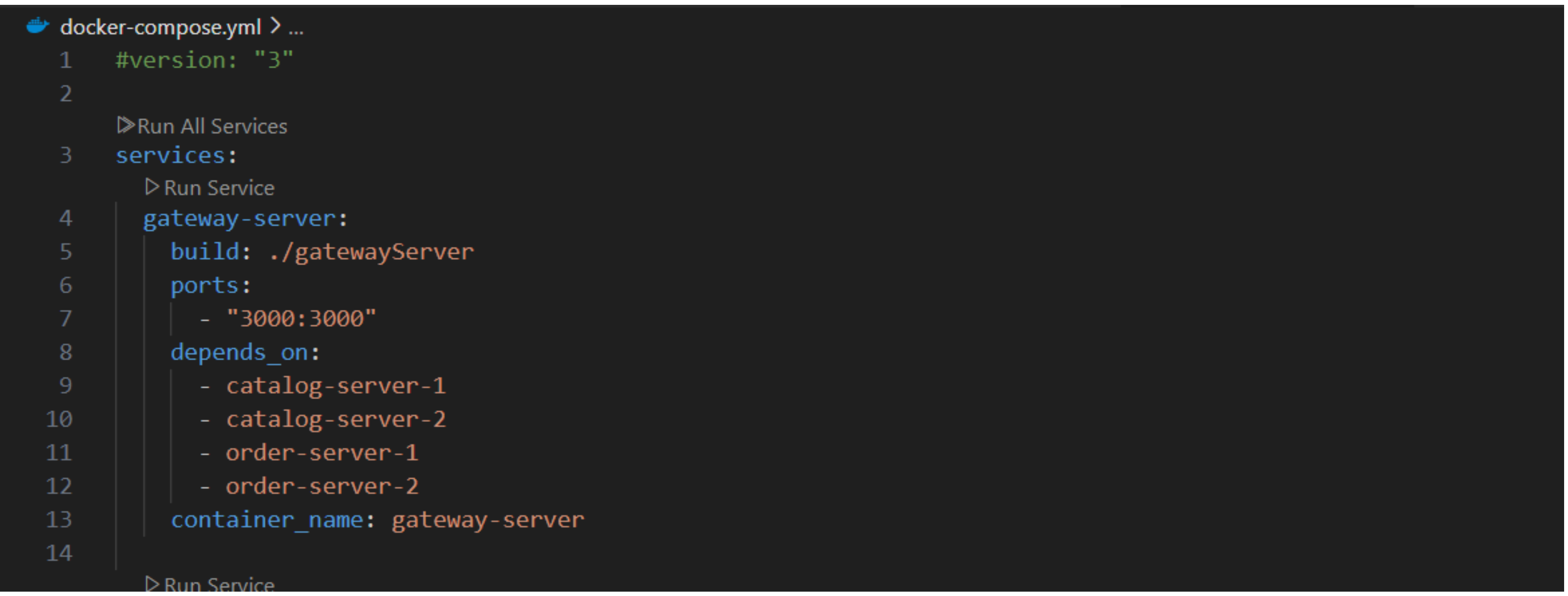
- After send an invalidate POST request then try to GET request it will display an error: “”



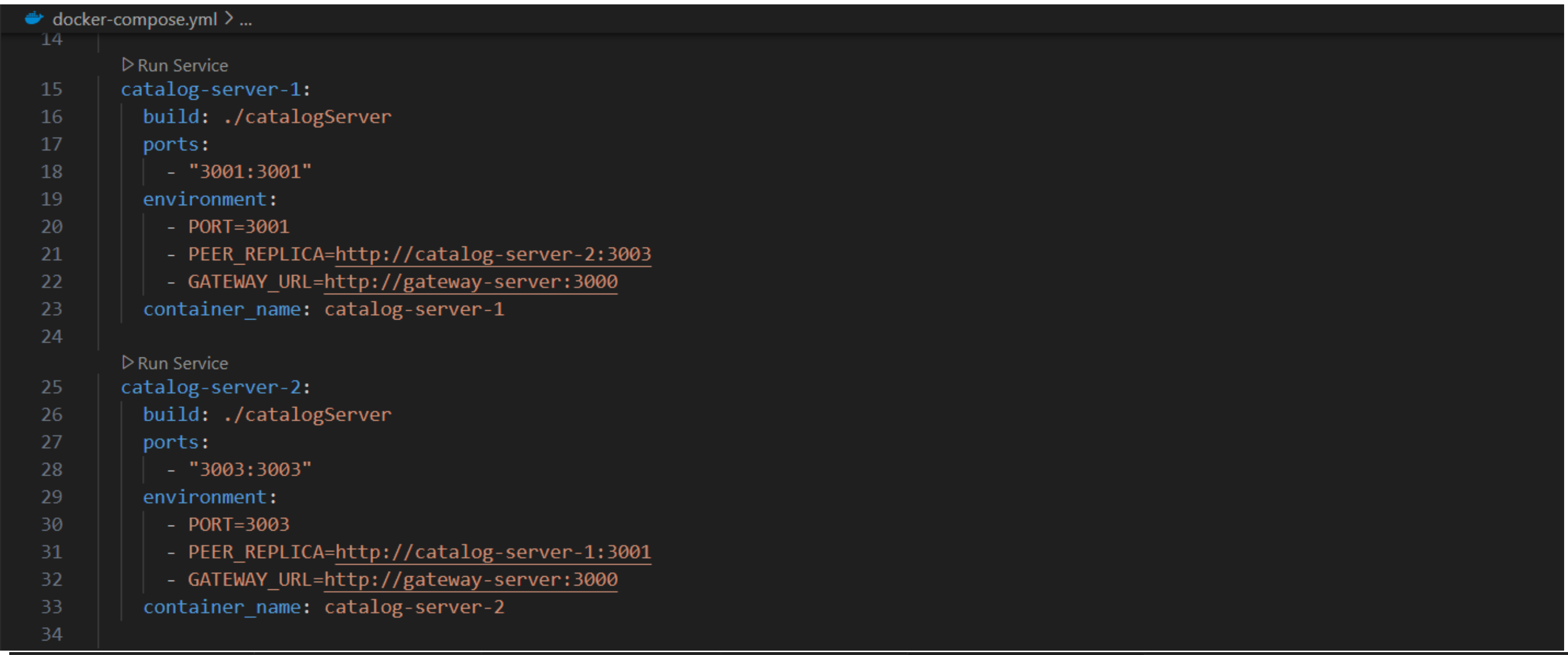
GET http://localhost:3000/info/1

500 Internal Server Error · 6 ms · 265 B

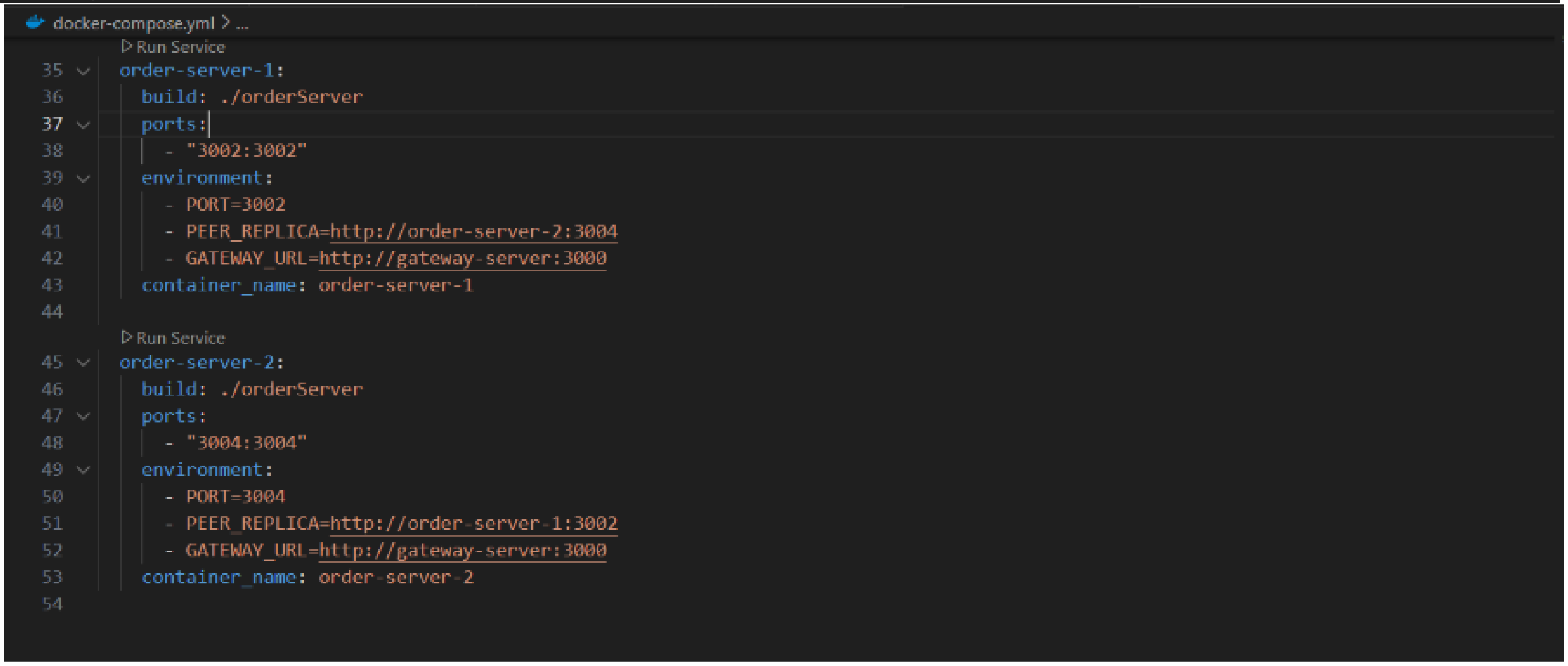
```
{
  "error": ""
}
```



```yaml
#version: "3"

services:
  gateway-server:
    build: ./gatewayServer
    ports:
      - "3000:3000"
    depends_on:
      - catalog-server-1
      - catalog-server-2
      - order-server-1
      - order-server-2
    container_name: gateway-server

```



```yaml
  catalog-server-1:
    build: ./catalogServer
    ports:
      - "3001:3001"
    environment:
      - PORT=3001
      - PEER_REPLICA=http://catalog-server-2:3003
      - GATEWAY_URL=http://gateway-server:3000
    container_name: catalog-server-1

  catalog-server-2:
    build: ./catalogServer
    ports:
      - "3003:3003"
    environment:
      - PORT=3003
      - PEER_REPLICA=http://catalog-server-1:3001
      - GATEWAY_URL=http://gateway-server:3000
    container_name: catalog-server-2

```



```yaml
  order-server-1:
    build: ./orderServer
    ports:
      - "3002:3002"
    environment:
      - PORT=3002
      - PEER_REPLICA=http://order-server-2:3004
      - GATEWAY_URL=http://gateway-server:3000
    container_name: order-server-1

  order-server-2:
    build: ./orderServer
    ports:
      - "3004:3004"
    environment:
      - PORT=3004
      - PEER_REPLICA=http://order-server-1:3002
      - GATEWAY_URL=http://gateway-server:3000
    container_name: order-server-2

```

Run the project

Run each node file using
node index.js after going to its directory (3 terminals : 1 for client , 1 for catalog and 1 for order).

Then using the Docker for the replica servers

Commands:

- **Docker-compose build**
- **Docker-compose up**