

Bazar Multi-tier Online Bookstore Part -1-
Distributed Operating Systems
2024

Siham khuffash

Farah Dilea'

Program Design and how to works:

-we have created 3 servers , two in VM using docker , (catalog server , order server) and the front server run on local machine.

first we create Dockerfile.

then we create an image called siham-ord3, and instance from it named siham-instance-ord3.

the server will on port: 4002 from VM

we use this command: (docker run -d --name siham-instance-ord3 -p 4002:4002 siham-ord3:latest) to connect the port 4002 from the server to port 4002, on local machine.

-we do these steps also to the second server.

after that we have create a compose.yml file to let the servers communicate with each other.

-now, when we run the `http://localhost:5051/info/id`.

the front-end server will send a request to catalog server run on 0.0.0.0:3000 and get the response from it.

-How to run Program:

- 1-Install node.js and Express framework on the two machines.
3. Open the frontend terminal and start the server by typing: npm start.
4. Open the catalog terminal and start the server by typing: npm start.
5. Open the order terminal and start the server by typing: npm start.
6. Start sending requests from postman/browser on localhost:5051.

-docker image 1:

df7dc9944300	image-for-catalog:latest	"docker-entrypoint.s..."	4 hours ago	Up 4 hours
0.0.0.0:3010->3000/tcp				catalog-conta1

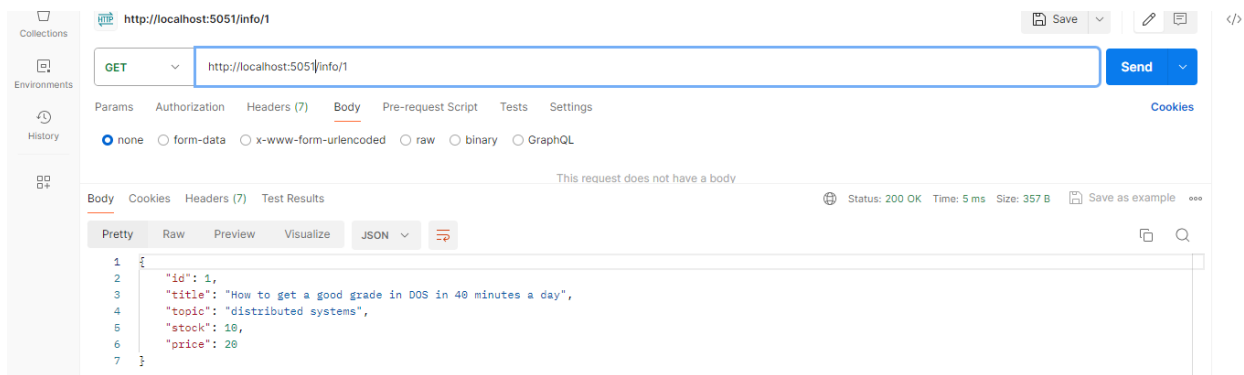
-docker image 2:

711c8873a01	image-for-catalog:latest	"docker-entrypoint.s..."	4 hours ago	
3abfc63e9d67	orderimage:latest	"docker-entrypoint.s..."	5 hours ago	
0.0.0.0:3010->3000/tcp				catalog-conta1

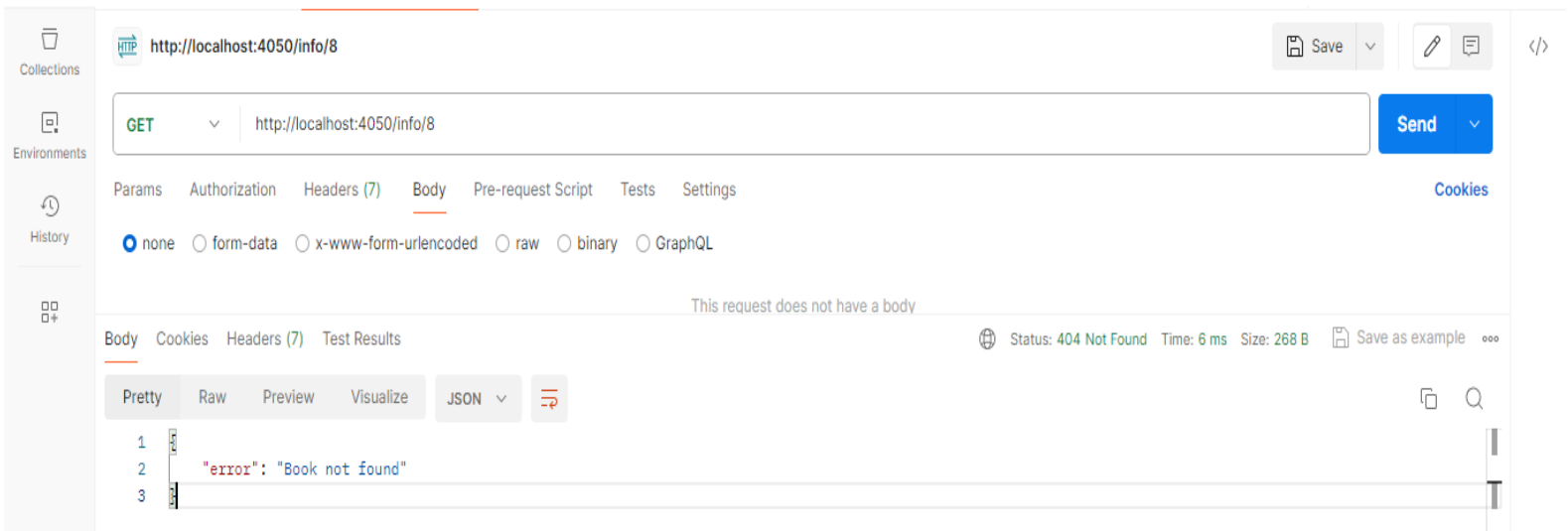
Get information by id:

Request: localhost:5051/books/info/[id]

-Requesting information of book 1

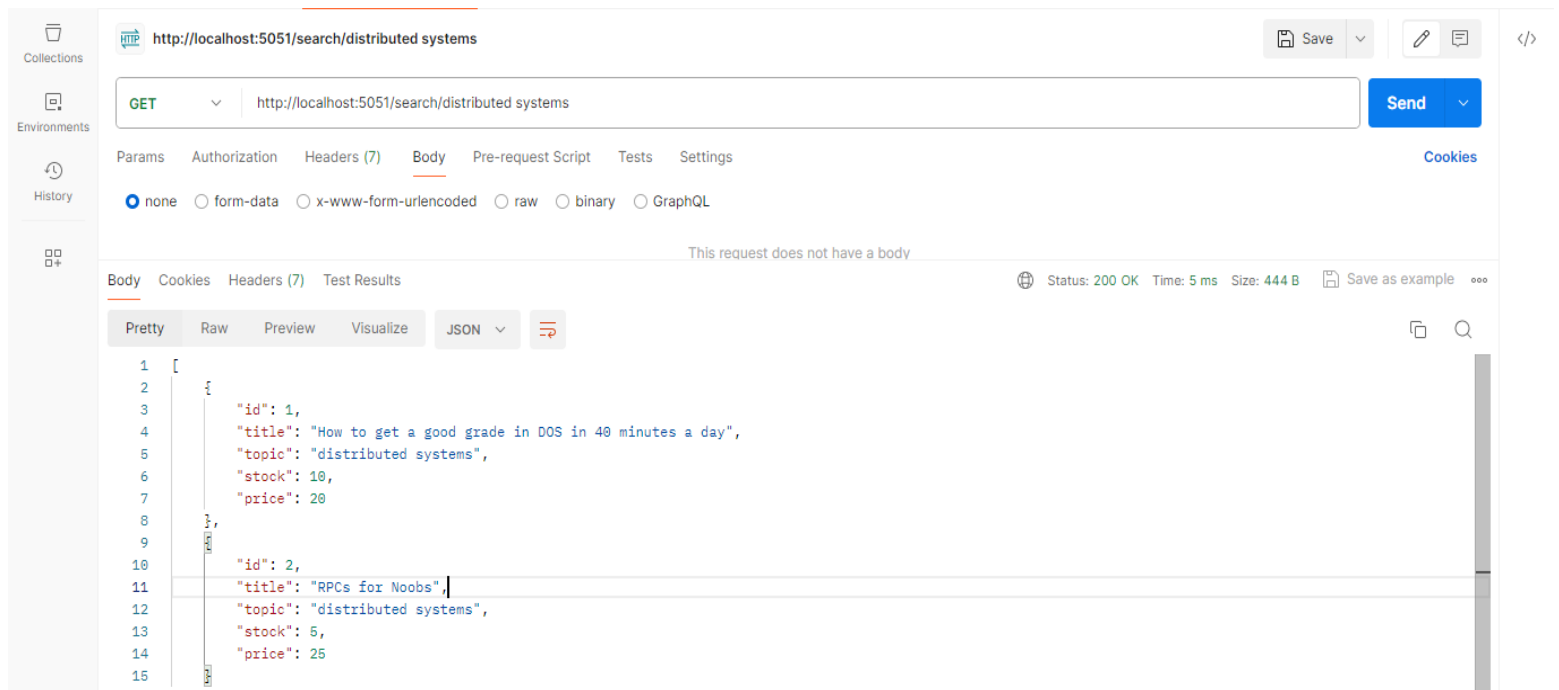


-Request a book with id=8 , which is not in the store:



Search by topic:

Searching with topic distributed systems :



HTTP `http://localhost:5051/search/distributed systems` Save

GET `http://localhost:5051/search/distributed systems` Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

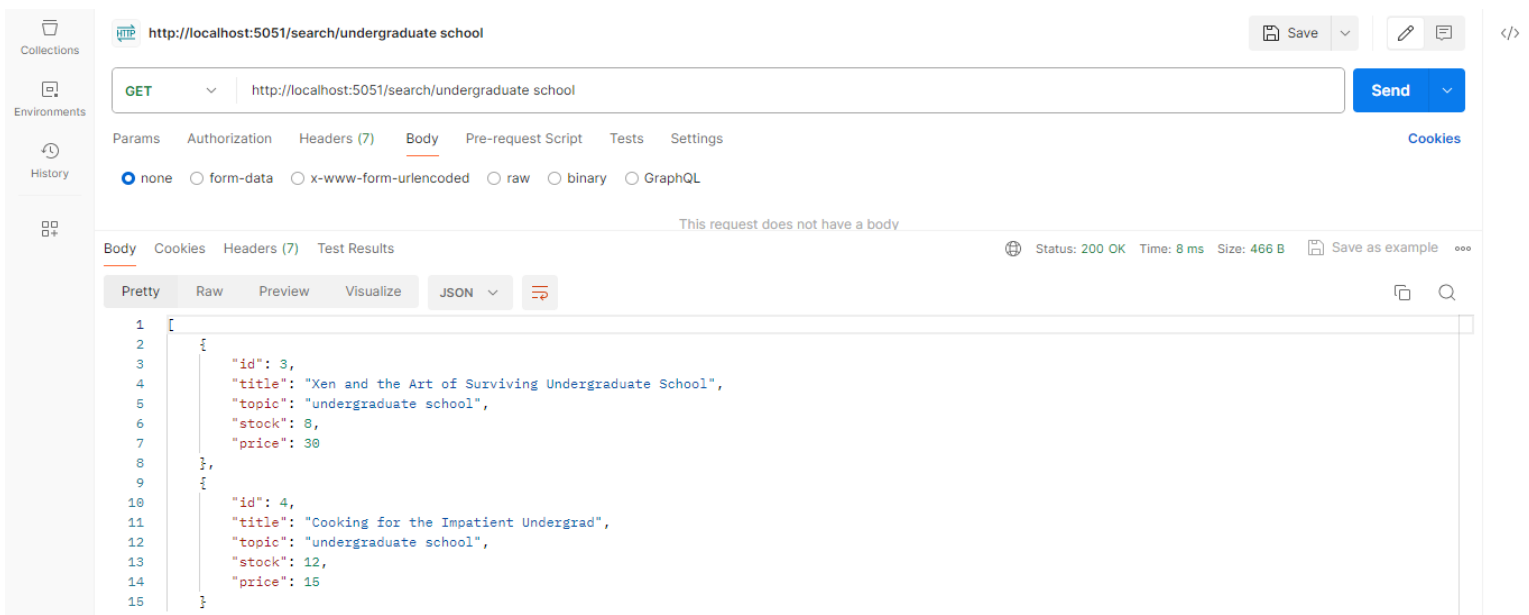
This request does not have a body

Body Cookies Headers (7) Test Results Status: 200 OK Time: 5 ms Size: 444 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "title": "How to get a good grade in DOS in 40 minutes a day",
5     "topic": "distributed systems",
6     "stock": 10,
7     "price": 20
8   },
9   {
10    "id": 2,
11    "title": "RPCs for Noobs",
12    "topic": "distributed systems",
13    "stock": 6,
14    "price": 25
15  }
16 ]
```

Searching with topic undergraduate school



HTTP `http://localhost:5051/search/undergraduate school` Save

GET `http://localhost:5051/search/undergraduate school` Send

Params Authorization Headers (7) **Body** Pre-request Script Tests Settings Cookies

☒ none ☐ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

This request does not have a body

Body Cookies Headers (7) Test Results Status: 200 OK Time: 8 ms Size: 466 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 3,
4     "title": "Xen and the Art of Surviving Undergraduate School",
5     "topic": "undergraduate school",
6     "stock": 8,
7     "price": 30
8   },
9   {
10    "id": 4,
11    "title": "Cooking for the Impatient Undergrad",
12    "topic": "undergraduate school",
13    "stock": 12,
14    "price": 15
15  }
16 ]
```

Update cost:

update cost of book id=2 to be 50.

The screenshot shows a REST client interface with the following components:

- URL Bar:** `http://localhost:5051/edit/2`
- Method:** `PUT`
- Body:** A JSON object:

```
{  "id": 2,  "title": "RPCs for Noobs",  "topic": "distributed systems",  "stock": 5,  "price": 25}
```
- Status:** `200 OK`, `Time: 7 ms`, `Size: 319 B`
- Response:** The same JSON object as the request body.