BAZAR.COM

Homework 1

Marwa banifadel & Batool jum’a

OCTOBER 27, 2024

The front-end tier:

1- Front end service

The backend

1- Order service
2- Catalog service

---

1) Front-end service:

There are three operations in this server

• search(topic): the request is sent to the front-end server, and then the catalog returns the item.

HTTP. get(http://localhost:5002/search/Undergra duate school)

• info(item_number): the request is sent to the front-end server, and then the catalog return the information.

HTTP. get(http://localhost:5002/info/1)

• purchase(item_number): The purchase order is sent to the to the front-end, and then the order server return completed.

HTTP.post(http://localhost:5002/purchase/1)

2) Order service:

• purchase(item_number): The purchase order is sent to the to the front-end, and then the order server return completed.

HTTP.post(http://localhost:5001/purchase/1)

3) Catalog service:

• search(topic): the request is sent to the catalog server, and then the catalog returns the item.

HTTP. get(http://localhost:5000/search/Distributed systems)

• info(item_number): the request is sent to the catalog server, and then the catalog return the information.

HTTP. get(http://localhost:5000/info/1)

To run this project :

in cmd in Bazar.com →

docker build -t my-app .

- The command docker build -t my-app . is used to build a new image in Docker. Run to catalog
- **t my-app**: This option is used to tag the image being built
- This dot indicates that the build context is the current directory. To (dockerfile)

docker network create projectPart1-net

- The command docker network create projectPart1-net is used to create a new network in Docker.

In cmd in catalog folder to run catoalog server →

docker run --name=catalog -p 5000:5000 --network=projectPart1-net -it -v .:/home my-app

- Is used to run a Docker container based on the image my-app.

- **docker run**: This command is used to create and start a new container from a specified image.
- **--name=catalog**: This option assigns a name (catalog) to the container. Naming your container makes it easier to reference it in future commands (e.g., stopping or removing the container).
- **-p 5000:5000**: This option maps port 5000 on the host machine to port 5000 on the container. This allows you to access your application running inside the container via http://localhost:5000 on your host machine.
- **--network=projectPart1-net**: This specifies that the container should be connected to the Docker network named projectPart1-net. This enables the container to communicate with other containers on the same network.
- **-it**: This option runs the container in interactive mode and allocates a pseudo-TTY (terminal). It allows you to interact with the container's command line directly.
- **-v .:/home**: This option mounts the current directory (denoted by .) to the /home directory in the container. This allows you to share files between your host and the container, making it easier to develop and test your application without needing to rebuild the image each time you make changes.
- **my-app**: This is the name of the image from which the container is created. In this case, it's the image you built earlier using docker build.

npm install express

- This command installs the **Express** library, which is a web application framework for Node.js. It simplifies the process of building web applications and APIs by providing features such as routing, middleware support, and more.

npm install axios

- This command installs **Axios**, a promise-based HTTP client for the browser and Node.js. Axios allows you to make HTTP requests to external APIs or services easily.

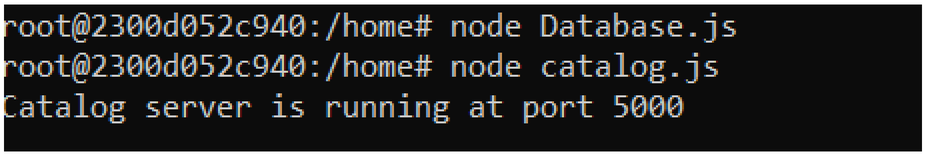
npm i sqlite3

- This command installs **SQLite3**, a library that provides a way to work with SQLite databases in Node.js applications. It allows you to create, read, update, and delete data in an SQLite database.

To run js code (catalog server):

node Database.js

node catalog.js



```
root@2300d052c940:/home# node Database.js
root@2300d052c940:/home# node catalog.js
Catalog server is running at port 5000
```

and Re-steps for the order file, front-end file:

cmd in order to order server →

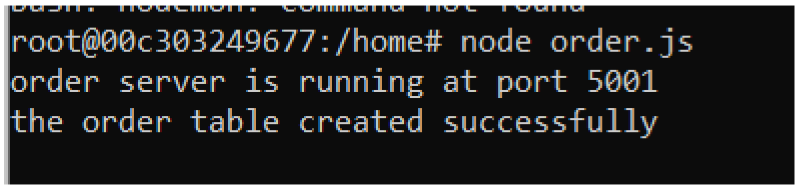
docker run --name=order -p 5001:5001 --network=projectPart1-net -it -v .:/home my-app

npm install express

npm install axios

npm i sqlite3

node order.js



```
root@00c303249677:/home# node order.js
order server is running at port 5001
the order table created successfully
```

cmd in frontend to frontend server →

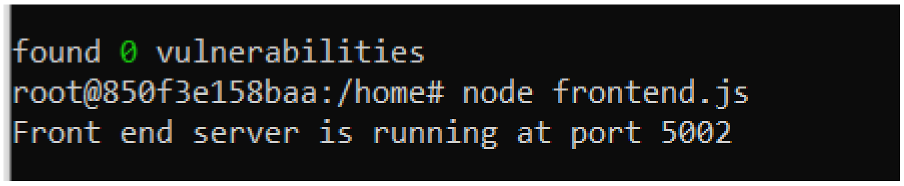
docker run --name=frontend -p 5002:5002 --network=projectPart1-net -it -v .:/home my-app

npm install express

npm install axios

npm i sqlite3

node frontend.js



```
found 0 vulnerabilities
root@850f3e158baa:/home# node frontend.js
Front end server is running at port 5002
```

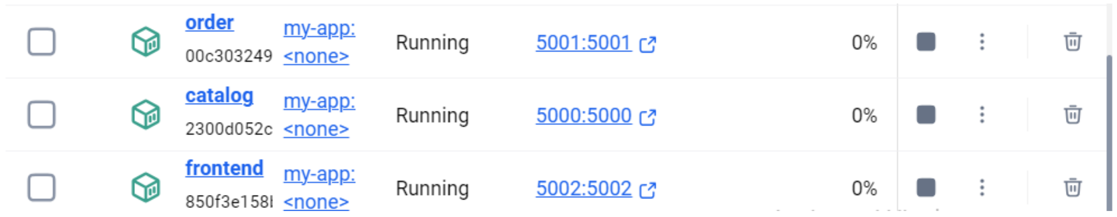
We were supposed not to do these three operations and to pull them from the docker file, but to be sure..

1. npm install express
2. npm install axios
3. npm i sqlite3

I use :

1. Catalog server at port : 5000
2. order server at port : 5001
3. frontend server at port : 5002

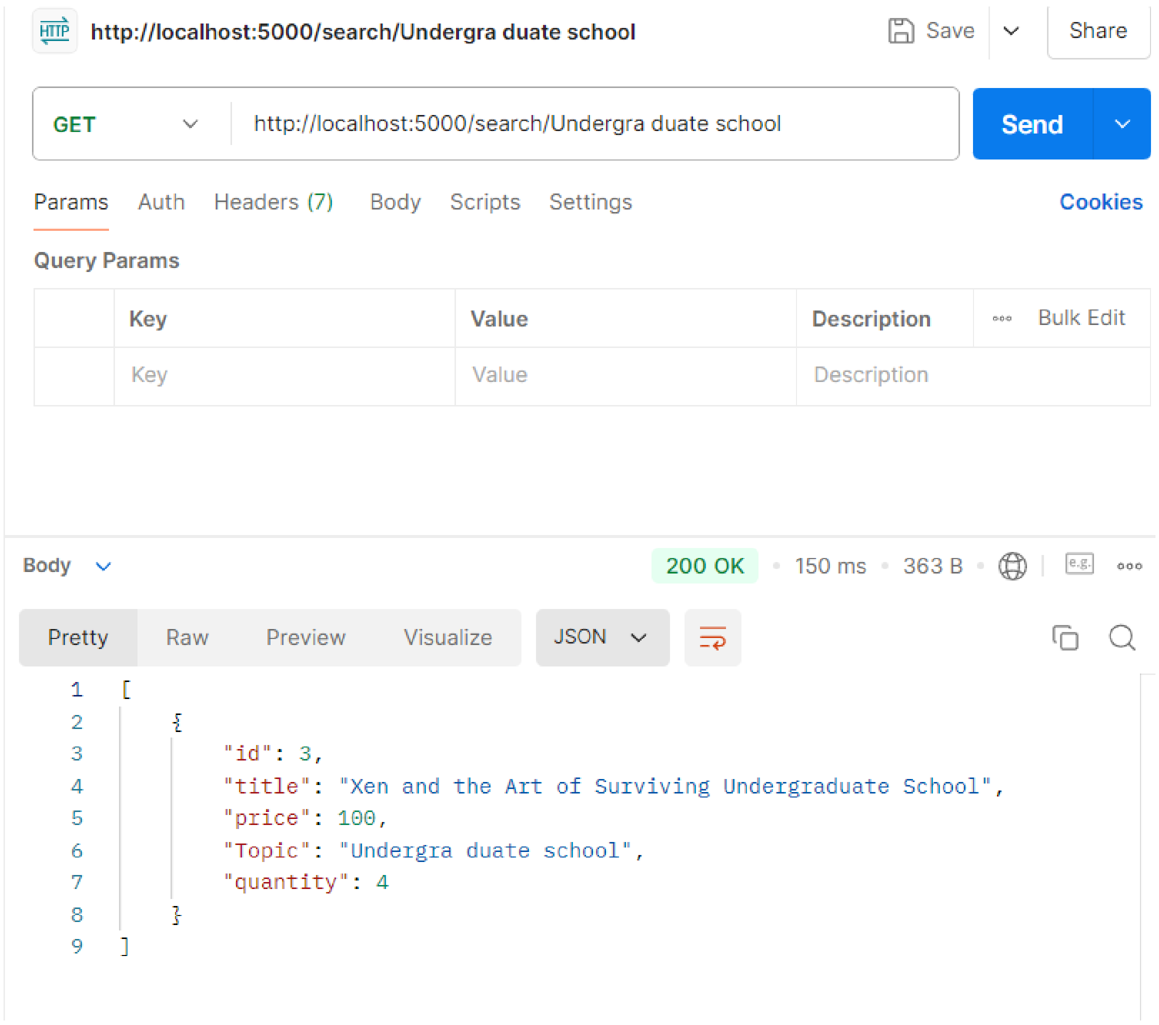
On docker desktop :



order 00c303249	my-app: <none>	Running	5001:5001	0%
catalog 2300d052c	my-app: <none>	Running	5000:5000	0%
frontend 850f3e158l	my-app: <none>	Running	5002:5002	0%

Now let's make sure the project works.

1- Search from catalog server :



http://localhost:5000/search/Undergra duate school

```
GET http://localhost:5000/search/Undergra duate school
```

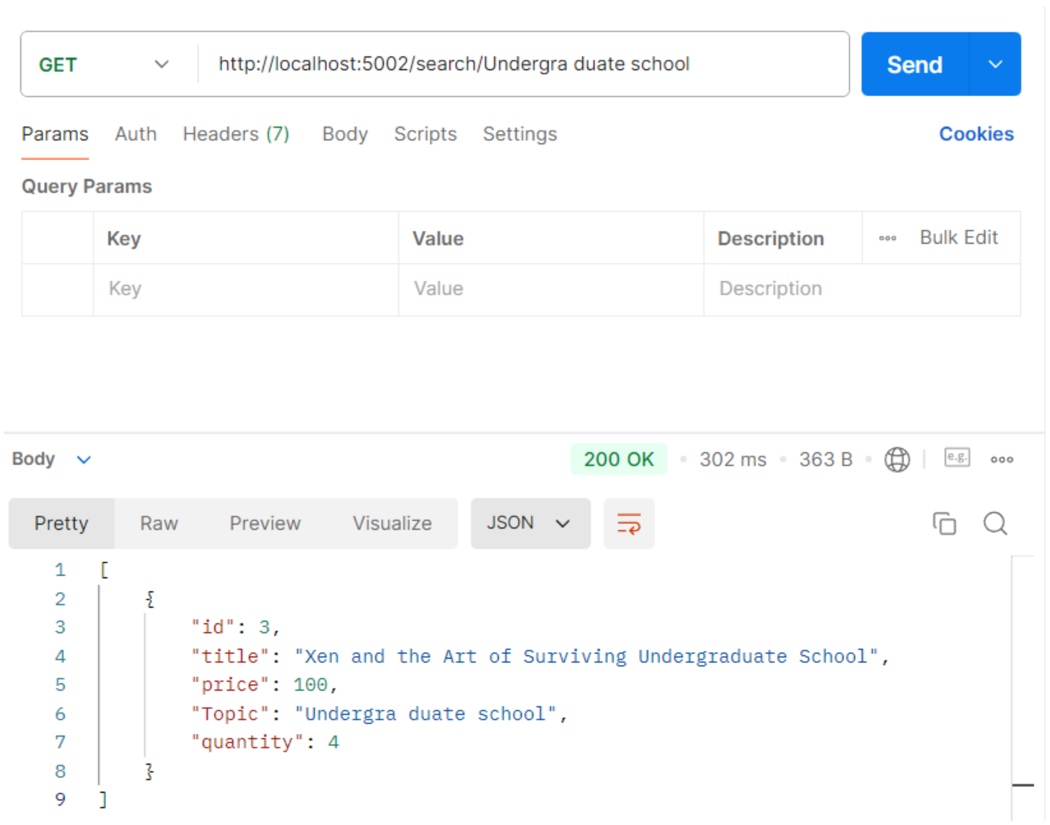
Query Params

Key	Value	Description
Key	Value	Description

Body — 200 OK • 150 ms • 363 B

```
[
    {
        "id": 3,
        "title": "Xen and the Art of Surviving Undergraduate School",
        "price": 100,
        "Topic": "Undergra duate school",
        "quantity": 4
    }
]
```

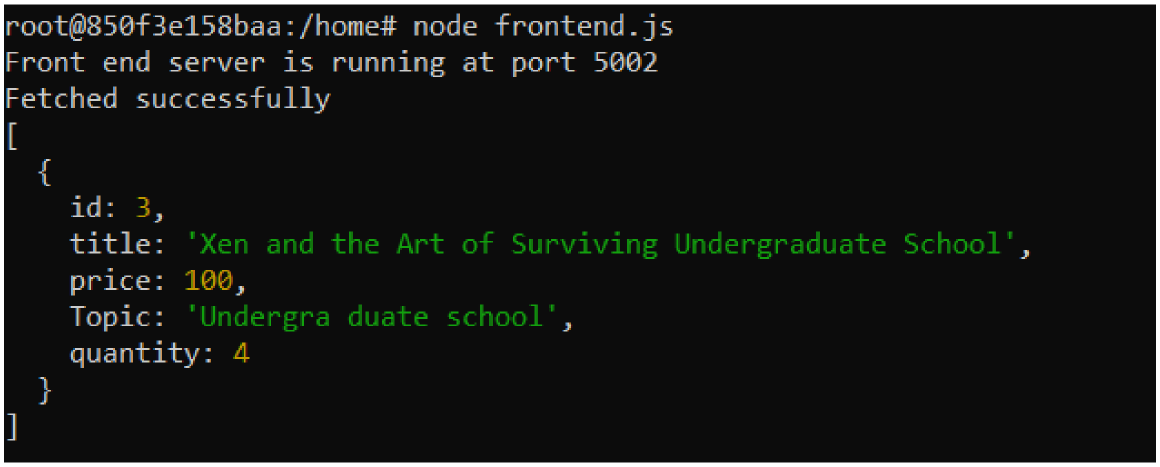
2- Search from frontend server :



```
http://localhost:5002/search/Undergra duate school
```

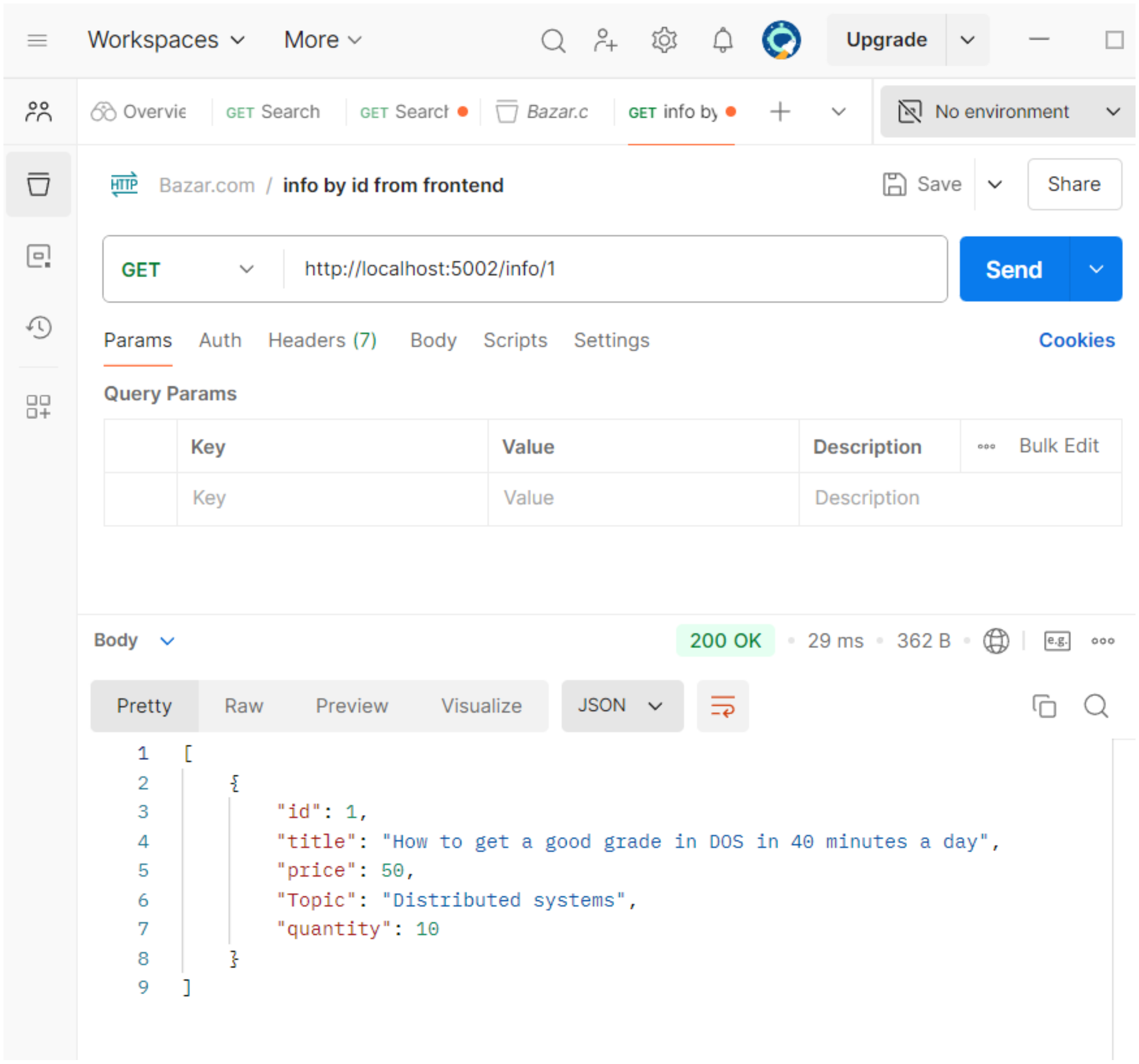
```
[
    {
        "id": 3,
        "title": "Xen and the Art of Surviving Undergraduate School",
        "price": 100,
        "Topic": "Undergra duate school",
        "quantity": 4
    }
]
```

- In docker :



```
root@850f3e158baa:/home# node frontend.js
Front end server is running at port 5002
Fetched successfully
[
  {
    id: 3,
    title: 'Xen and the Art of Surviving Undergraduate School',
    price: 100,
    Topic: 'Undergra duate school',
    quantity: 4
  }
]
```

3- Info by id from frontend server :

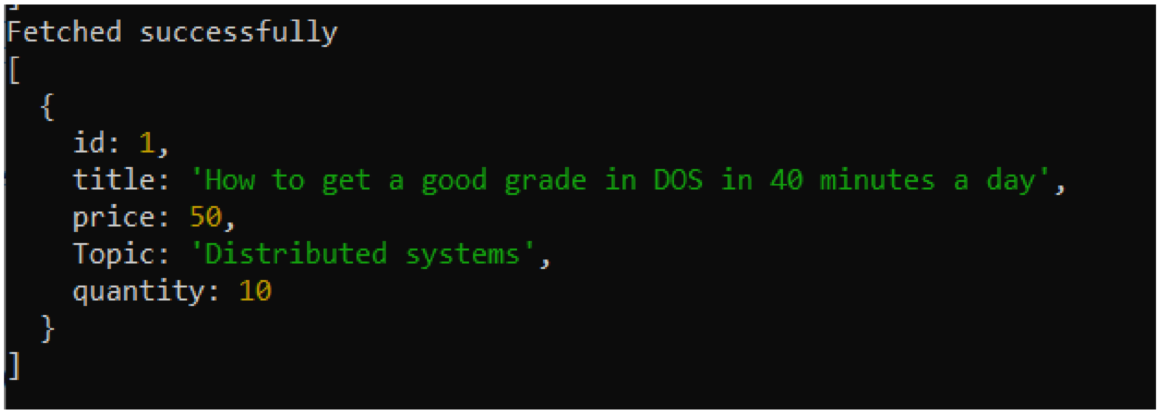


GET http://localhost:5002/info/1

Key	Value	Description
Key	Value	Description

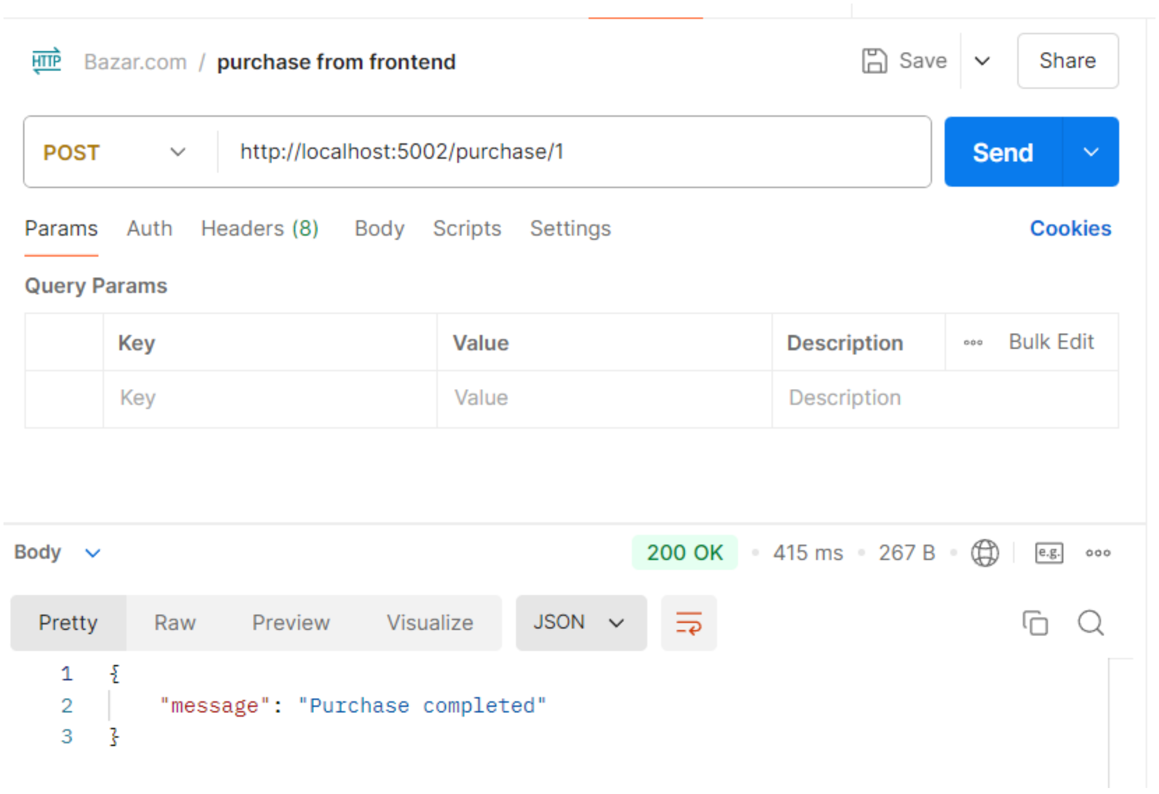
```
[
    {
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day",
        "price": 50,
        "Topic": "Distributed systems",
        "quantity": 10
    }
]
```

- In docker :



```
Fetched successfully
[
  {
    id: 1,
    title: 'How to get a good grade in DOS in 40 minutes a day',
    price: 50,
    Topic: 'Distributed systems',
    quantity: 10
  }
]
```

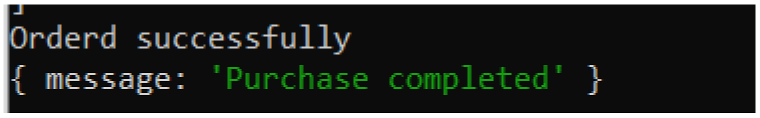
4- Purchase from frontend server :



POST http://localhost:5002/purchase/1

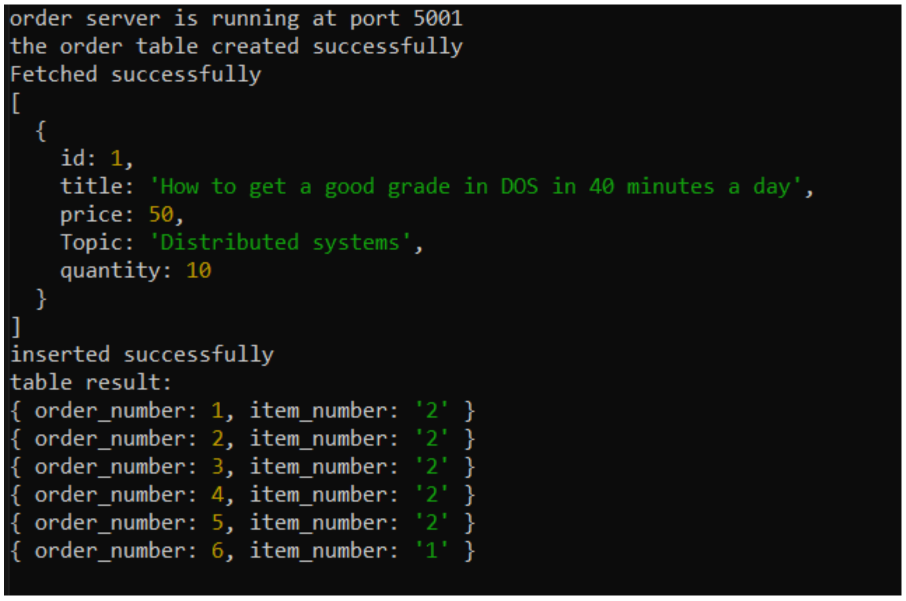
```
{
    "message": "Purchase completed"
}
```

- in docker frontend :



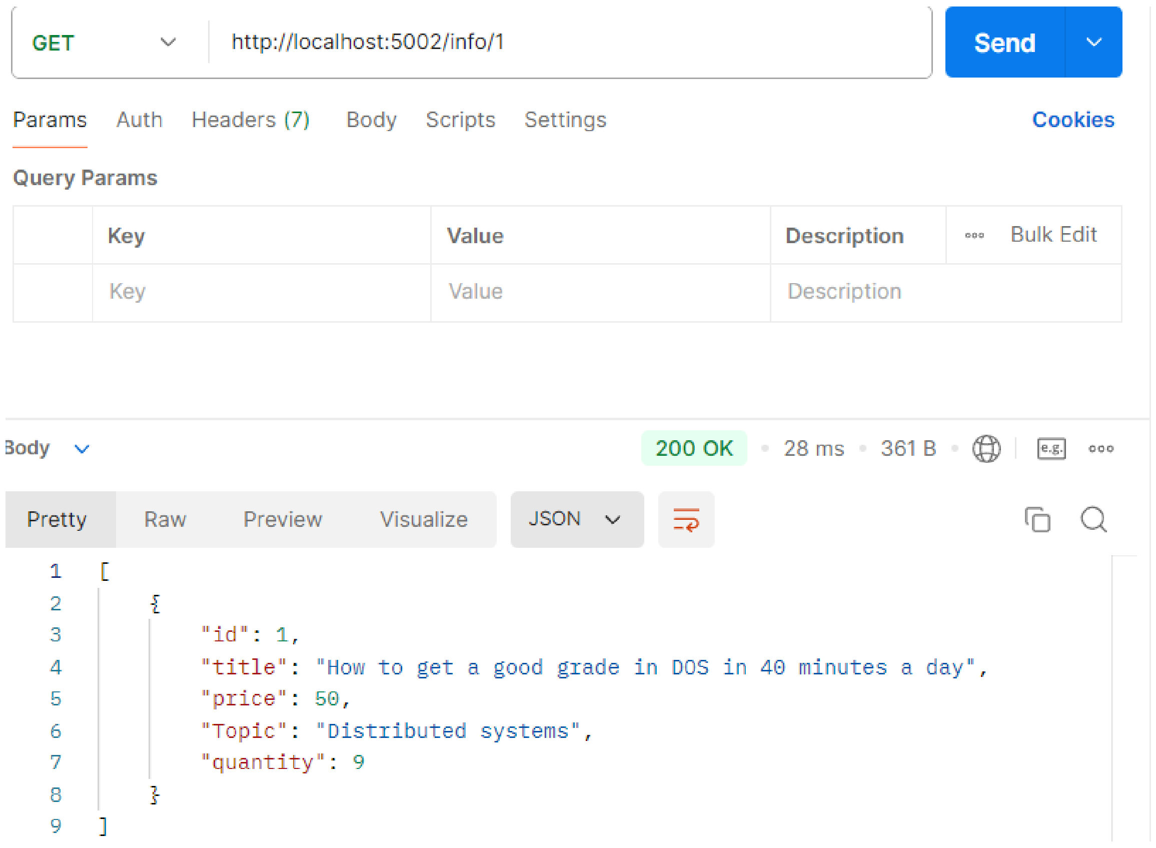
```
Orderd successfully
{ message: 'Purchase completed' }
```

- In order docker :



```
order server is running at port 5001
the order table created successfully
Fetched successfully
[
  {
    id: 1,
    title: 'How to get a good grade in DOS in 40 minutes a day',
    price: 50,
    Topic: 'Distributed systems',
    quantity: 10
  }
]
inserted successfully
table result:
{ order_number: 1, item_number: '2' }
{ order_number: 2, item_number: '2' }
{ order_number: 3, item_number: '2' }
{ order_number: 4, item_number: '2' }
{ order_number: 5, item_number: '2' }
{ order_number: 6, item_number: '1' }
```

5- Let's see that after the purchase process, the quantity decreased by 1.

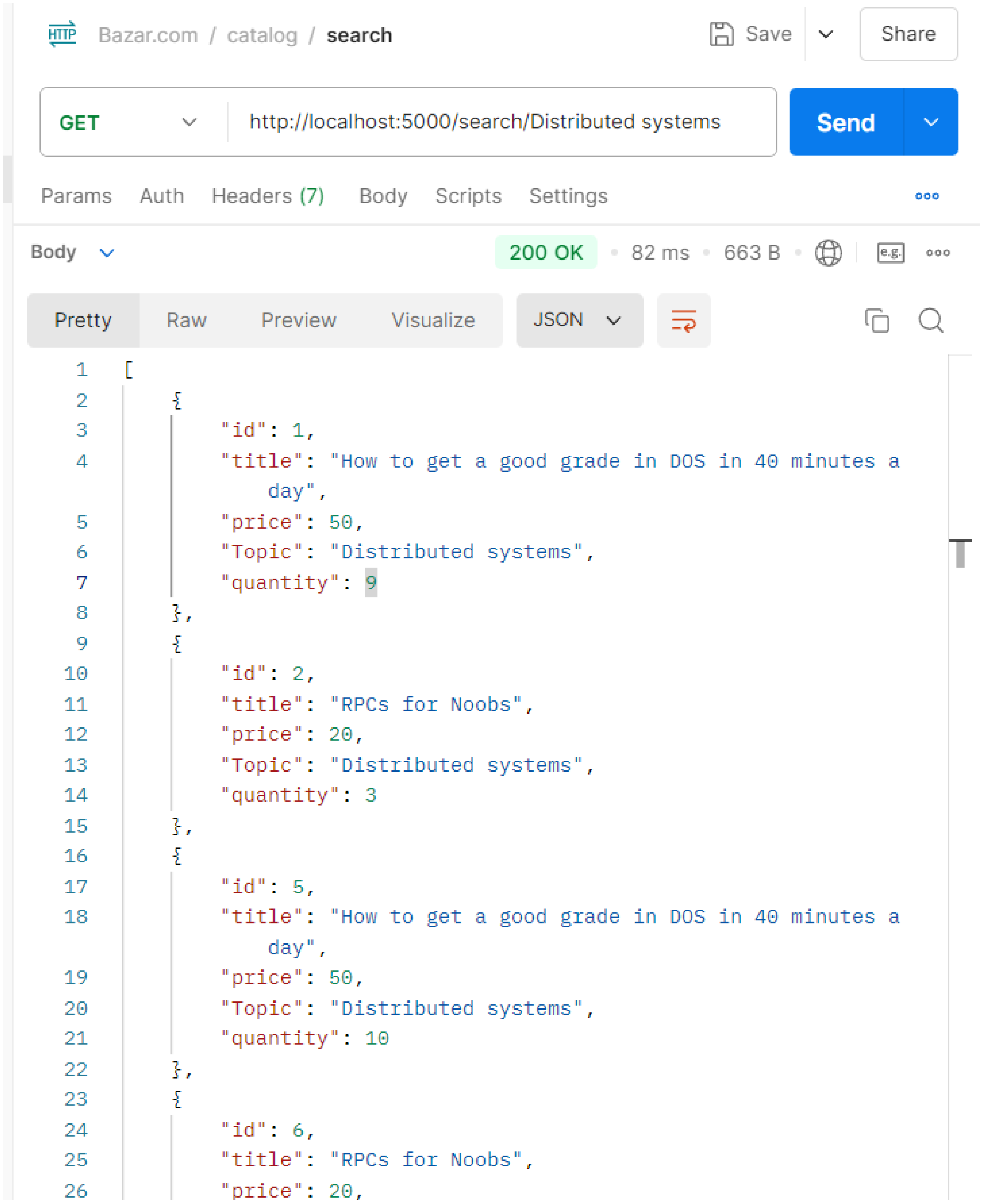


```
http://localhost:5002/info/1
```

```
200 OK • 28 ms • 361 B
```

```
[
    {
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day",
        "price": 50,
        "Topic": "Distributed systems",
        "quantity": 9
    }
]
```

6- From catalog server(search) :

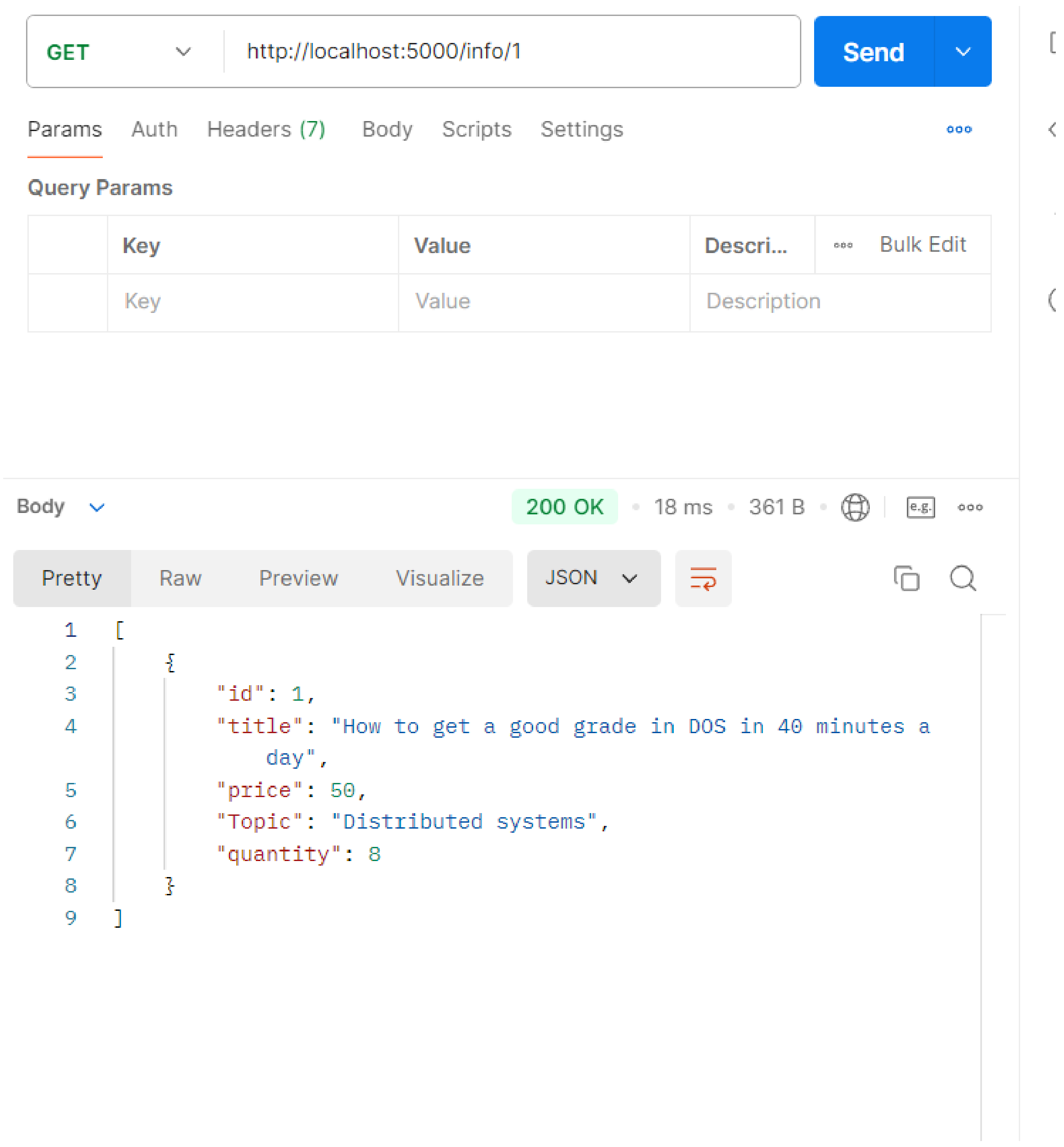


GET http://localhost:5000/search/Distributed systems

200 OK • 82 ms • 663 B

```
[
    {
        "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": 20,
        "Topic": "Distributed systems",
        "quantity": 3
    },
    {
        "id": 5,
        "title": "How to get a good grade in DOS in 40 minutes a
            day",
        "price": 50,
        "Topic": "Distributed systems",
        "quantity": 10
    },
    {
        "id": 6,
        "title": "RPCs for Noobs",
        "price": 20,
```

7- From catalog server(info) :



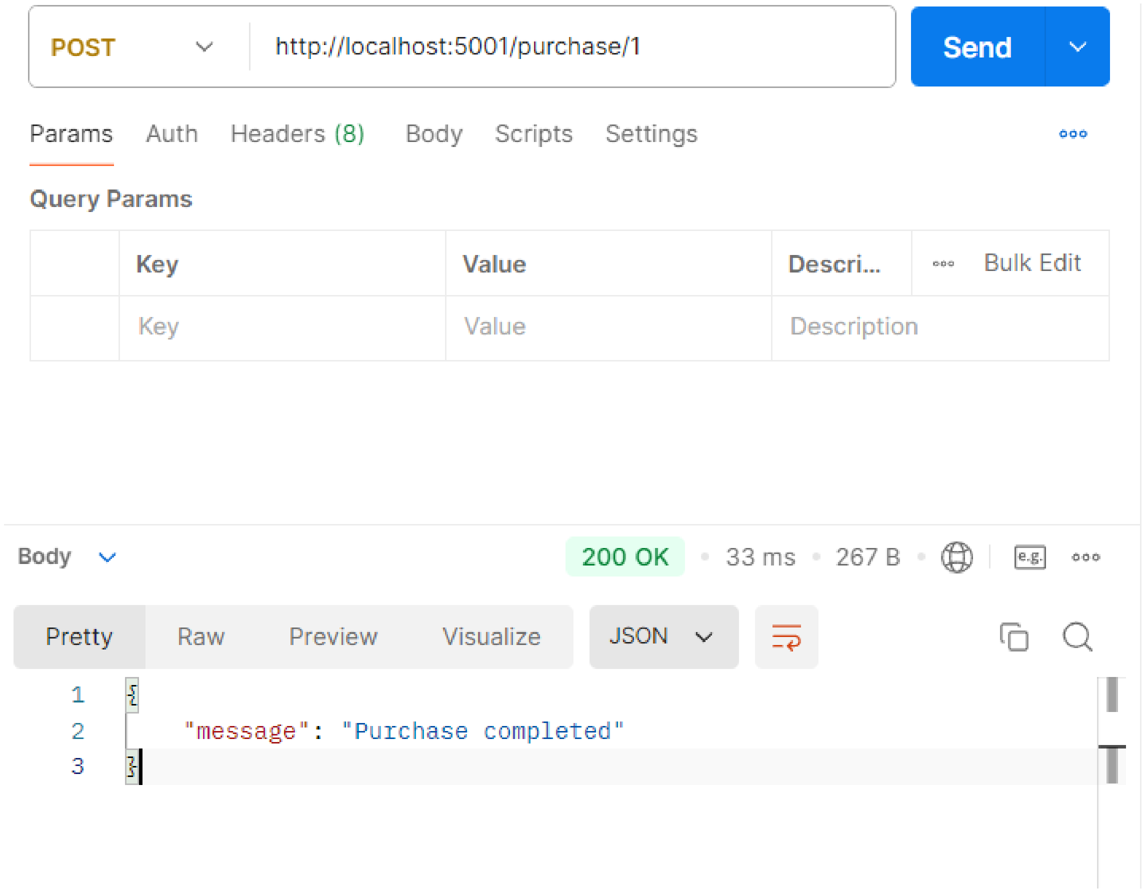
GET http://localhost:5000/info/1

Key	Value	Descri...
Key	Value	Description

200 OK • 18 ms • 361 B

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

8- From order server(purchase) :


POST http://localhost:5001/purchase/1

	Key	Value	Descri...	Bulk Edit
	Key	Value	Description	

```
{
    "message": "Purchase completed"
}
```

We also notice that the number of books before the purchase was 10, and after the purchase it became 9 → 8 .

Ps: Noticing that sometimes containers don't see the libraries (sqlite3, express, axios) even they are installed when the image was built, so we reinstalled them inside the containers. In next part of the project, we will use docker compose to fix this problem

Additional information :

To download sqlite :

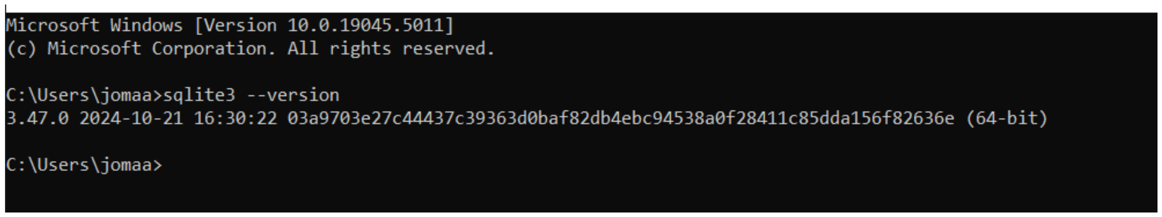
https://www.sqlite.org/download.html

from : sqlite-tools-win-x64-3470000.zip (6.04 MiB)

to create tables in sqlite :

in sqlite3.exe → sqlite3 data.db

➔ CREATE TABLE catalog (id INTEGER PRIMARY KEY, title TEXT, price REAL, Topic TEXT, quantity INTEGER);



```
Microsoft Windows [Version 10.0.19045.5011]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jomaa>sqlite3 --version
3.47.0 2024-10-21 16:30:22 03a9703e27c44437c39363d0baf82db4ebc94538a0f28411c85dda156f82636e (64-bit)

C:\Users\jomaa>
```