Bazer.com
DOS project

By:
Asala Tibi (11819226)
Hala Barqawi (11821121)

Overall program design :

We design a multi-tier online book store that contains three servers .A catalog server which is connected with the database ,it stores the information about each book in the store , so I can make requests to the servers to get books by topic or id . In addition to updating the price or the number of items in stock . The second is the order server makes a purchase request so it contacts with the catalog server to make sure if the book is available and it decrements the number of book in stock if the operation is successful .Finally,frontend server which clients directly connect with it ,it calls the two backend servers .

we wrote the code in Node js ,we used axios for http request , express framework for all servers ,and sqlite database.

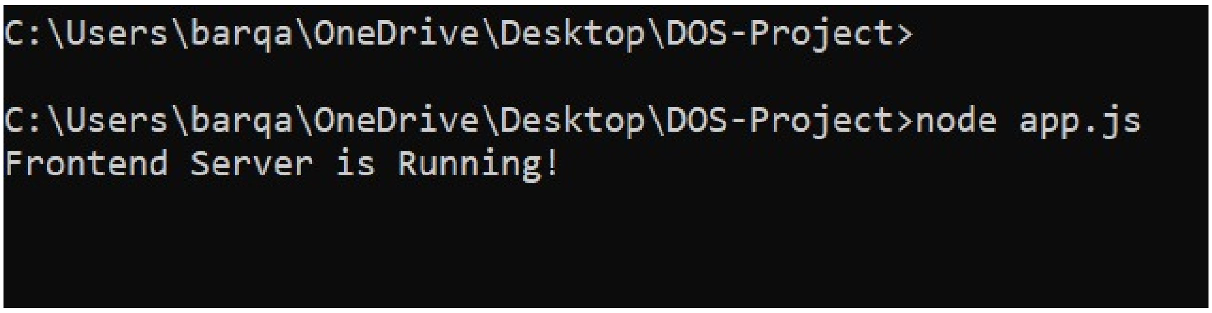
How it works:

- the client send request by browser or postman to localhost on port 3001, he only contacts with frontend server

- the frontend receives the request and calls one of backend servers according to the request type .

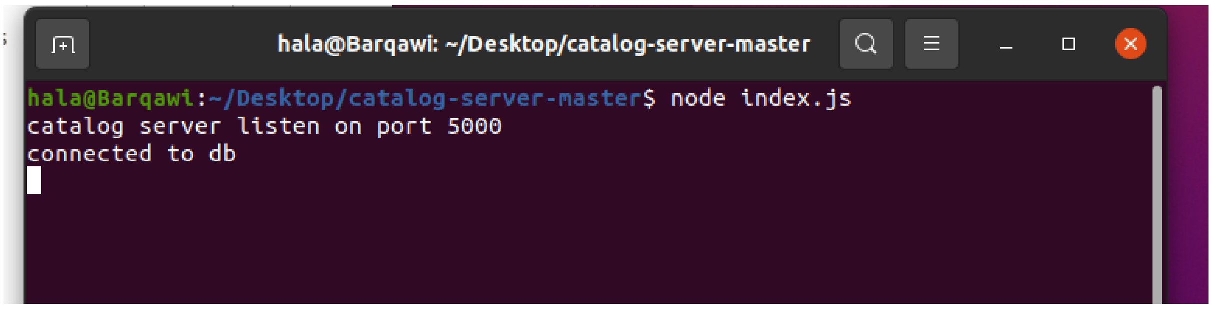
- If the request wants to get books by topic or id ,it calls the catalog server which is running on a virtual machine to handle this request.

- if the request makes purchase operations ,it calls the order server which is running on another virtual machine .



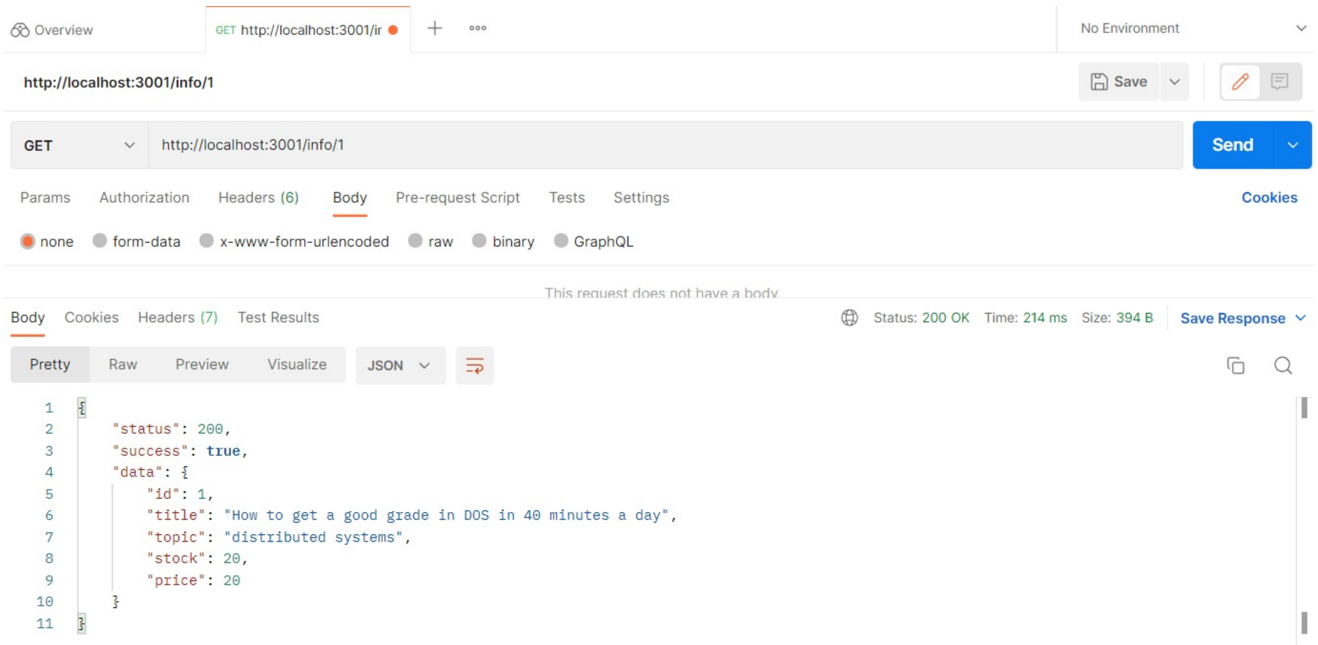
```
C:\Users\barqa\OneDrive\Desktop\DOS-Project>

C:\Users\barqa\OneDrive\Desktop\DOS-Project>node app.js
Frontend Server is Running!
```

The frontend server is running on port 3001.(Windows -the host machine-)



```
hala@Barqawi:~/Desktop/catalog-server-master$ node index.js
catalog server listen on port 5000
connected to db
```

On the first virtual machine (Ubuntu) , the catalog server is running on port 5000.



http://localhost:3001/info/1

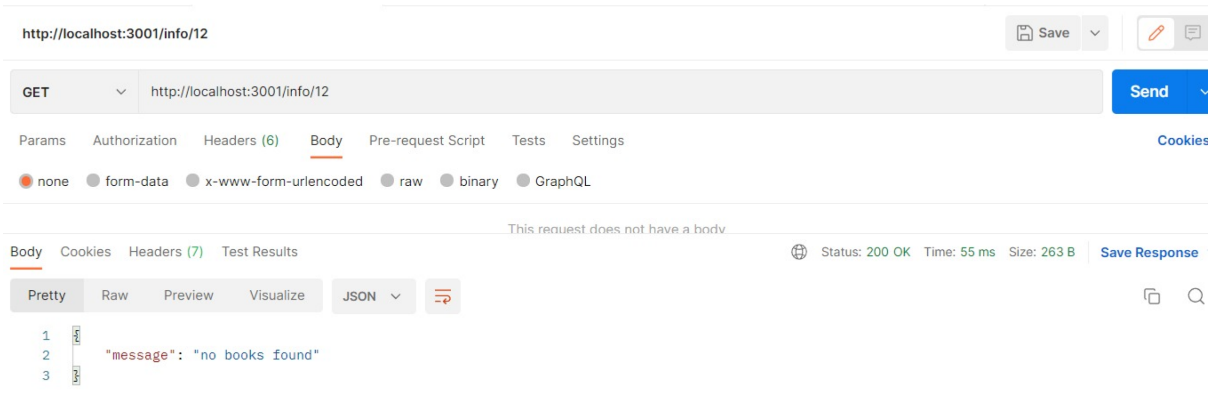
GET http://localhost:3001/info/1

This request does not have a body

Status: 200 OK Time: 214 ms Size: 394 B

```
{
    "status": 200,
    "success": true,
    "data": {
        "id": 1,
        "title": "How to get a good grade in DOS in 40 minutes a day",
        "topic": "distributed systems",
        "stock": 20,
        "price": 20
    }
}
```

Http request to the frontend to get a book by id .



http://localhost:3001/info/12
Save
GET
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Body Cookies Headers (7) Test Results
Save Response
Pretty Raw Preview Visualize JSON

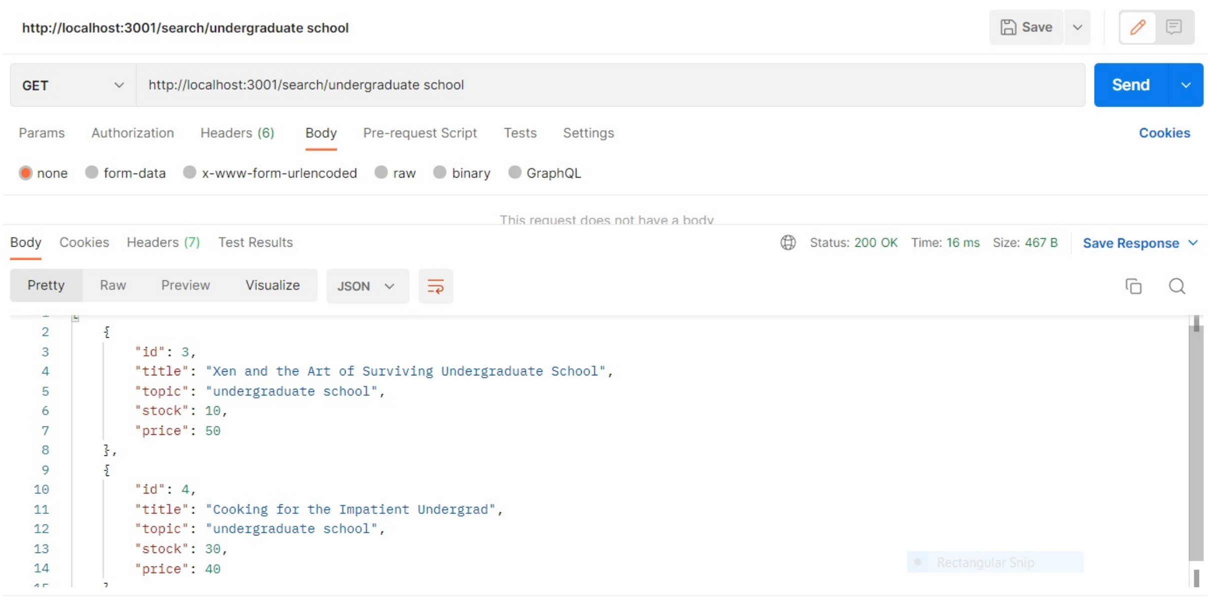
http://localhost:3001/info/12

This request does not have a body

Status: 200 OK Time: 55 ms Size: 263 B

```
{
    "message": "no books found"
}
```

In case the Id in the request was invalid



http://localhost:3001/search/undergraduate school
Save
GET
Send
Params Authorization Headers (6) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Body Cookies Headers (7) Test Results
Save Response
Pretty Raw Preview Visualize JSON

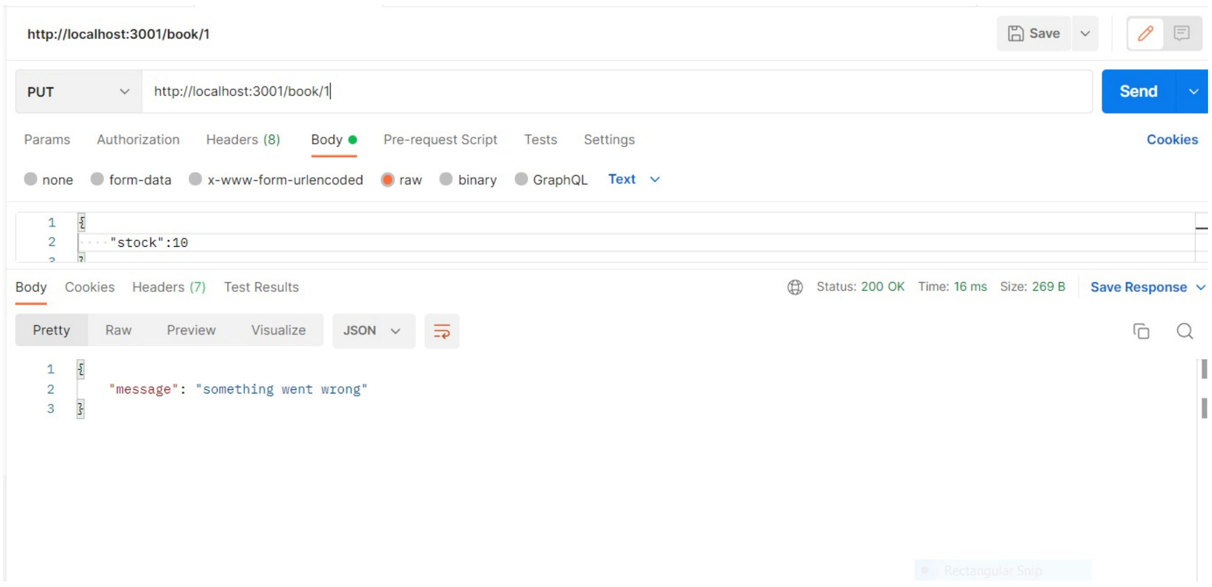
http://localhost:3001/search/undergraduate school

This request does not have a body

Status: 200 OK Time: 16 ms Size: 467 B

```
    {
        "id": 3,
        "title": "Xen and the Art of Surviving Undergraduate School",
        "topic": "undergraduate school",
        "stock": 10,
        "price": 50
    },
    {
        "id": 4,
        "title": "Cooking for the Impatient Undergrad",
        "topic": "undergraduate school",
        "stock": 30,
        "price": 40
```

Http request to get book by topic



PUT http://localhost:3001/book/1

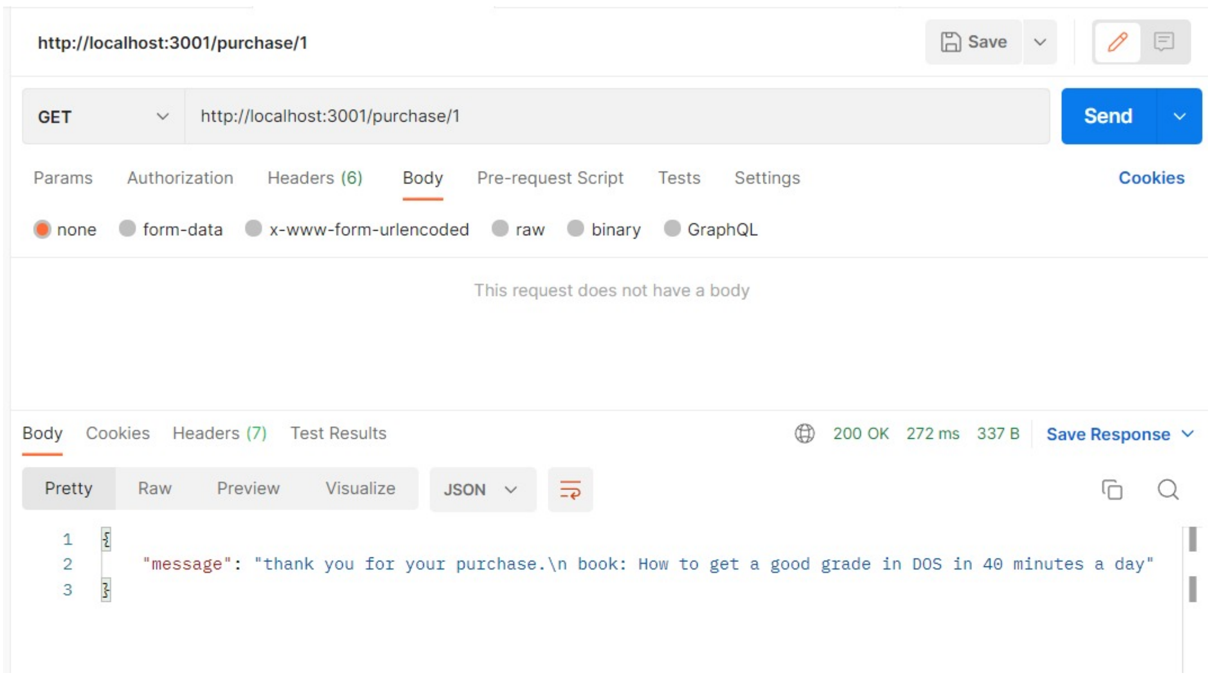
Body (raw, Text):

```
{
    "stock":10
```

Status: 200 OK Time: 16 ms Size: 269 B

```
{
    "message": "something went wrong"
}
```

http request to update price and stock,but we face a problem that the update not successful



GET http://localhost:3001/purchase/1

This request does not have a body

200 OK 272 ms 337 B

```
{
    "message": "thank you for your purchase.\n book: How to get a good grade in DOS in 40 minutes a day"
}
```

http request to make purchase operation by id

Design tradeoffs:

the clients will only contact with frontend server so this could make a bottleneck because of overhead requests on it .

Future improvements:

make replication of database and catalog server to separate server to reduce the overhead , add authentication to make more security systems.

cases in which don’t work correctly:

In case two purchase orders happen at the same time and for the same book ,this makes race conditions. I can solve it by making atomic transactions.

How can i run the program :

- on the host machine i run the frontend server on port 3001 , so clients can send requests on it by using browser or postman
- on the first virtual machine i run catalog server on port 5000 ,and IP=192.168.1.167
- on the second virtual machine i run order server on port 3003 ,and IP=192.168.1.136

Github :

❖ catalog server: https://github.com/AsalaTibi/catalog-server.git
❖ order server: https://github.com/HalaBarqawi/DOS-order.git
❖ frontend server: https://github.com/HalaBarqawi/DOS_Fronted.git