DOS - LAB1

Abdullah SaadEddin 11715207
Mohammad Shammout 11717628

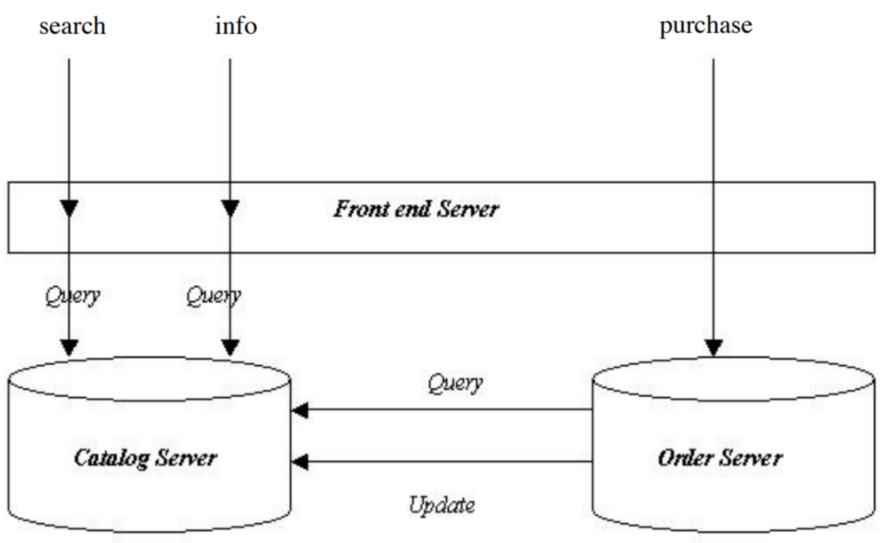
Introduction:

We use to implement (Bazar.com) servers design, express.js which is a framework in node.js used to build REST backend applications and microservices and SQLite for the database in each server.

we chose Express.js because it’s simple to setup and use,
also, it’s mange a lot of things like routes and handling requests.

also, Nodejs has a lot of libraries that simplify the operation of working with database line knex.js that we use to manage our database.

System design:



as requested we create 3 servers to implement the design in the figure, from the figure we see that the catalog server and order server communicate between each other, and the frontend server creates an interface for the 2 servers’ functionality.

1. **Front-End:**
   Our frontend design has 3 main routes:
    1. /info, which talks to the catalog server
    2. /search, which talks to the catalog server
    3. /purchase/:itemnumber, which talks to the catalog server

   all routes are sending requests to catalog and order servers and send the response of requests to the user as a response.

   we saved the IP address of the catalog and order servers as an environment variables, also we didn't do any edits on the response before sending it to the user because we do all that in the others servers.

2. **Catalog Server:**
   This server has a database that contains all books information, and the server manages this database, because of that, the other servers can't work without this server, they rely on it to git different information about the books and to edit them also.
   this server has 4 servers:

    a. /getbook/subject
       get all books with the same subject that is passed as a header in the request

    b. /getbook/itemNumber
       get all book information with the same item number that is passed as a header in the request

    c. /updatebook/stock
       update the book stock, this should have three headers (item number) which represent the book number, (amount)the amount that we need to, (operation) increase or decrease

    d. /updatebook/cost
       this will update the book cost so we should pass in the header the book number and the new cost.

3. **Order server:**
   this server has one operation which is
   (/purchase/:itemnumber)
   in this request the server will connect with the catalog server and get the information of the book to check the stock after that it will create a new record in the orders database which will represent the order

this all is the current system design we run one instance of each server at the same time, and all servers depend on each other so if one fails the whole system will fail, we can solve that and improve our design by creating more than one instance of each server by using Docker as an example.
also if we move the database to its own server and layer this will reduce the communication between the main servers and fast the process.

How to install and run:

as we mentioned at the start we use Node.js so the Prerequisite will be:

1. Nodejs
   install the LTS version of nodejs
2. npm
   make sure that npm is installed and added to the path in your device by running (npm --version) in the terminal if you get a version number, it installed and added,

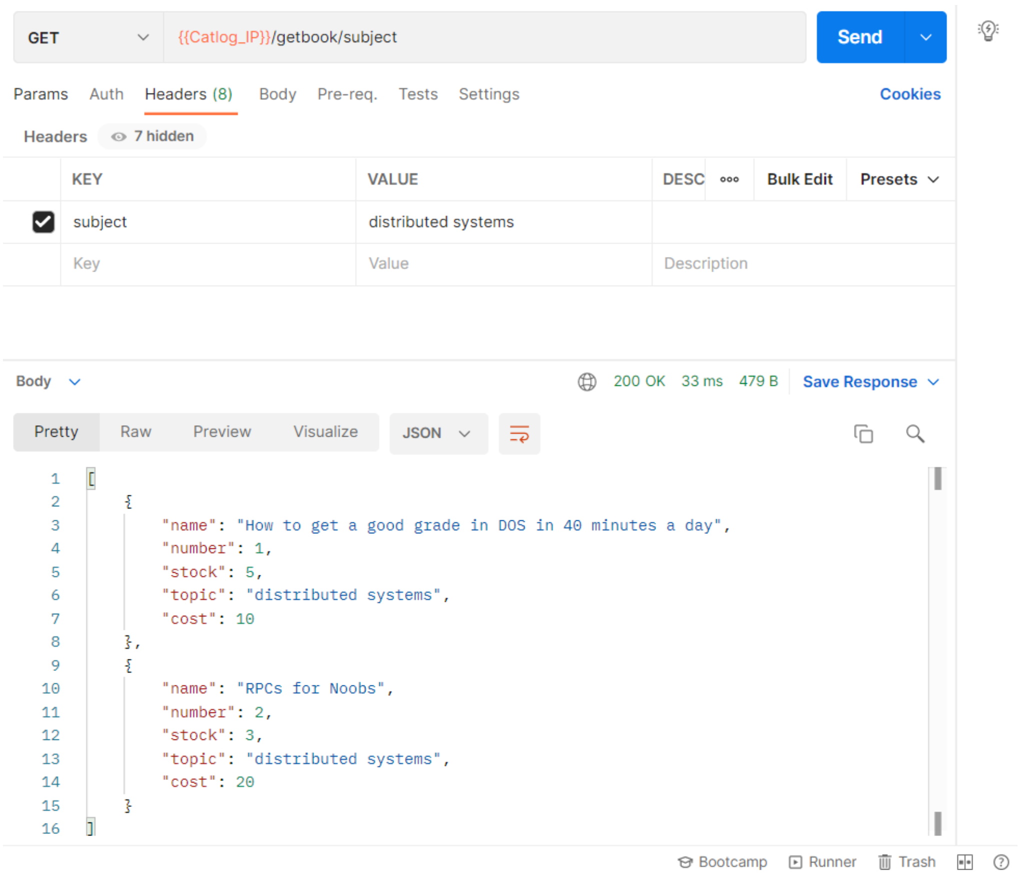
after you install both

1. clone the repository from [abdullah-saadeddin/DOSHW: this repo is for the DOS course HW2 (github.com)]
2. open terminal on each server folder directory
   and run (npm install)
   for example, open the terminal on the frontend server folder and there run the command, not in the main directory
3. open terminal on each server folder directory and run that server using the (npm start) command also this should be done on each server folder, not in the main one

when you run all 3 servers, you can make requests to all of them using Postman or any program that does the same thing.

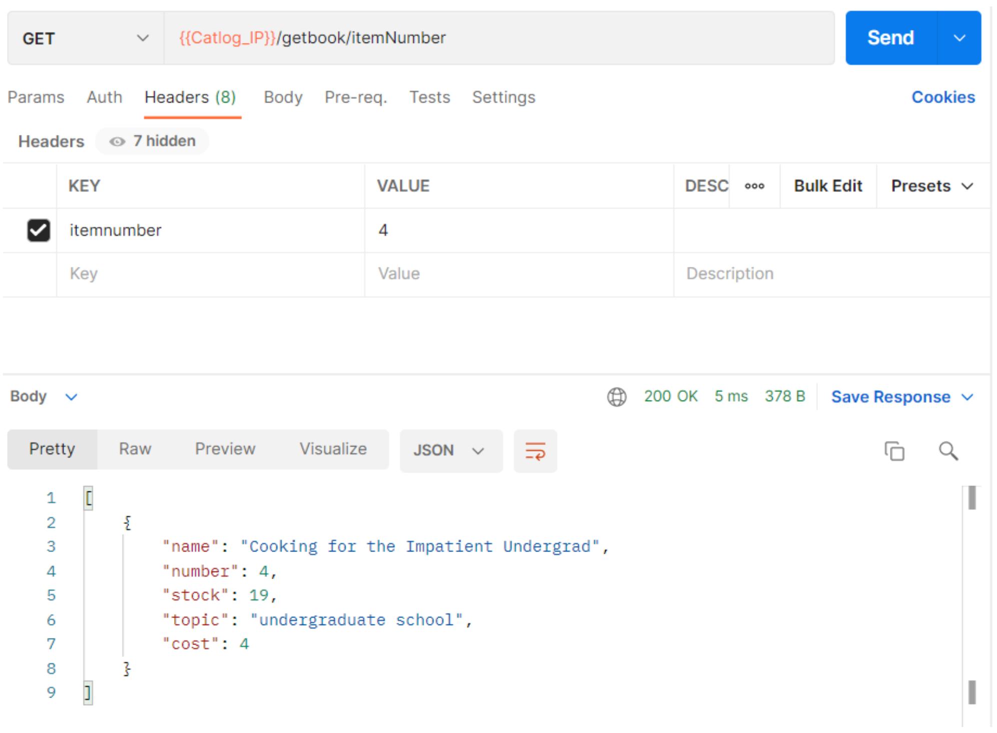
Running Examples:

we attached to the repository some logs for the application running for each server and the following screenshots are from the Postman app to test all servers REST services



KEY	VALUE	DESCRIPTION
subject	distributed systems	
Key	Value	Description

```
[
    {
        "name": "How to get a good grade in DOS in 40 minutes a day",
        "number": 1,
        "stock": 5,
        "topic": "distributed systems",
        "cost": 10
    },
    {
        "name": "RPCs for Noobs",
        "number": 2,
        "stock": 3,
        "topic": "distributed systems",
        "cost": 20
    }
]
```



GET {{Catlog_IP}}/getbook/itemNumber

Headers (8)

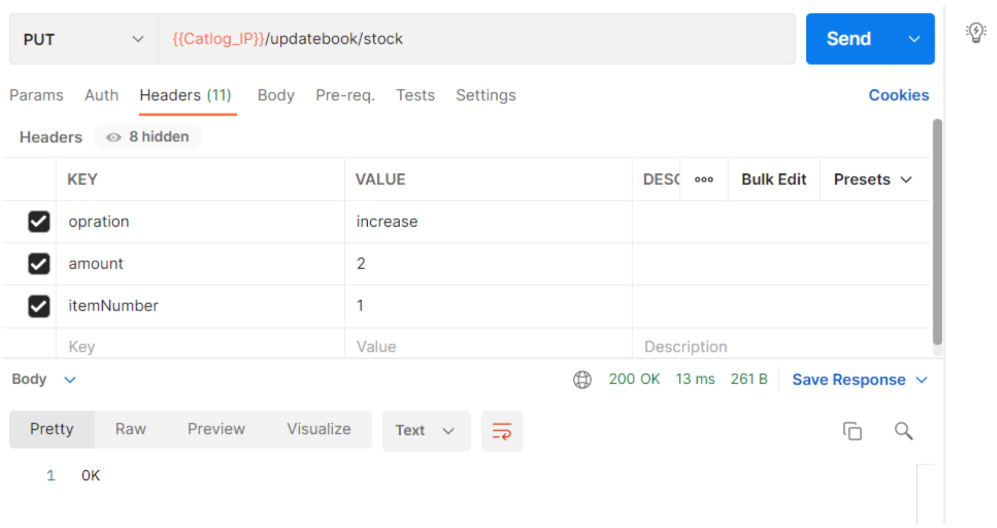
Headers 7 hidden

	KEY	VALUE	DESC
✓	itemnumber	4	
	Key	Value	Description

Body 200 OK 5 ms 378 B

JSON

```
[
    {
        "name": "Cooking for the Impatient Undergrad",
        "number": 4,
        "stock": 19,
        "topic": "undergraduate school",
        "cost": 4
    }
]
```



PUT {{Catlog_IP}}/updatebook/stock

Headers (11)

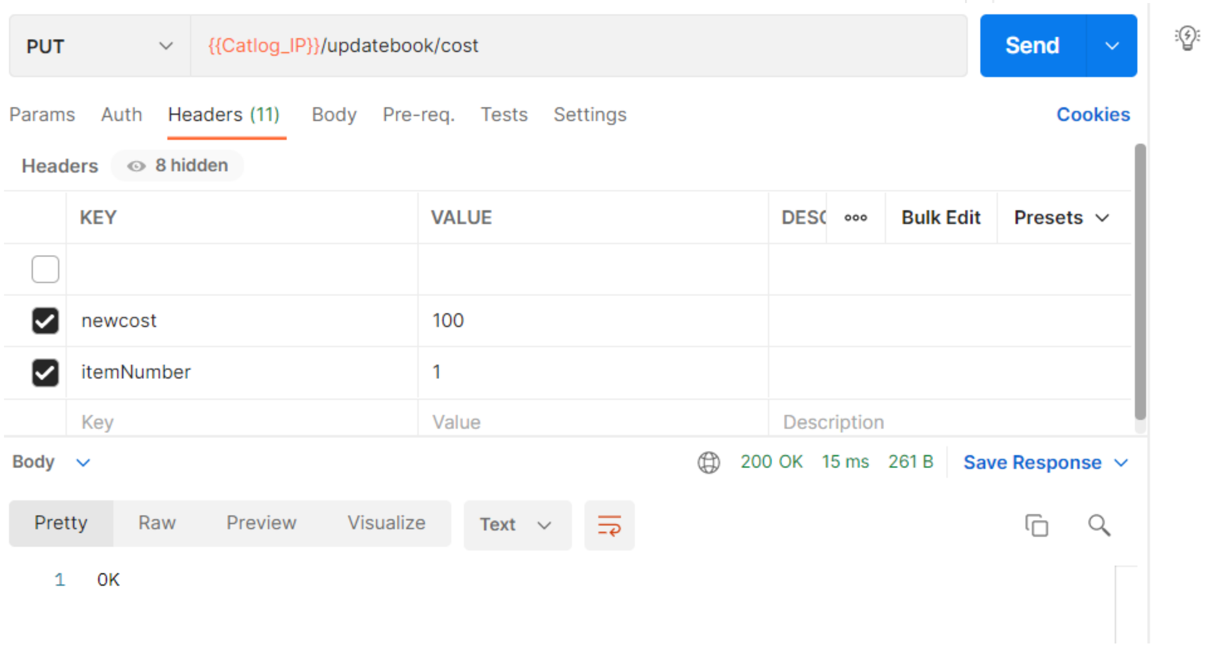
Headers 8 hidden

	KEY	VALUE	DESC
✓	opration	increase	
✓	amount	2	
✓	itemNumber	1	
	Key	Value	Description

Body 200 OK 13 ms 261 B

Text

```
OK
```



PUT {{Catlog_IP}}/updatebook/cost

Headers (11)

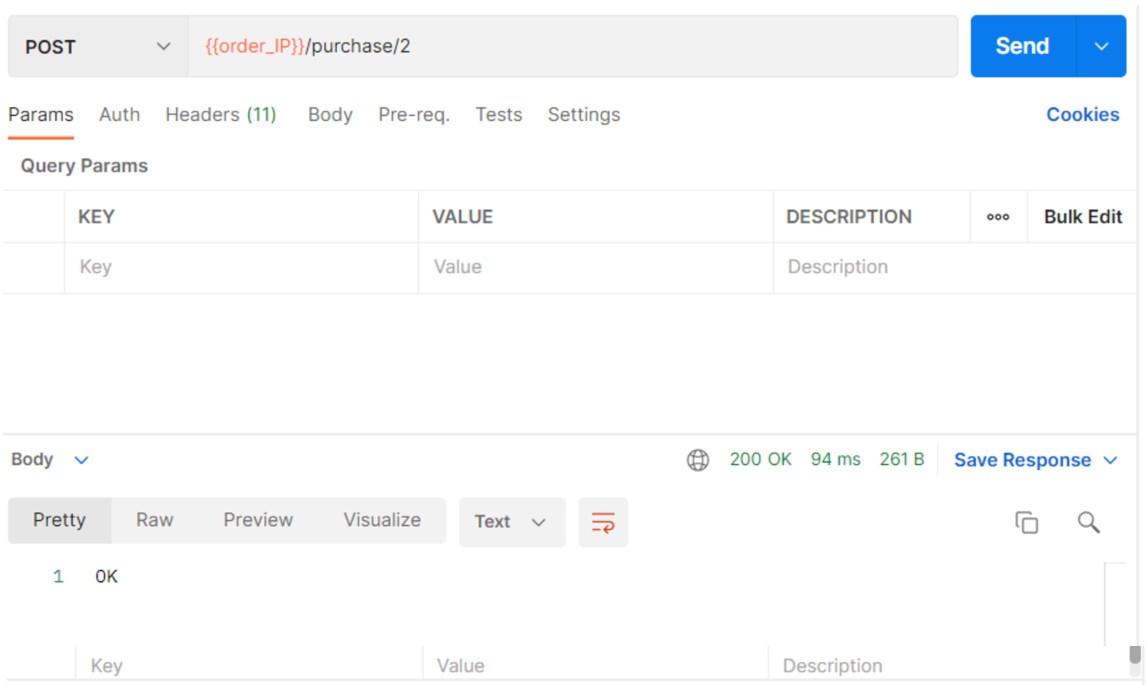
Headers 8 hidden

	KEY	VALUE	DESC
☐			
✓	newcost	100	
✓	itemNumber	1	
	Key	Value	Description

Body 200 OK 15 ms 261 B

Text

```
OK
```

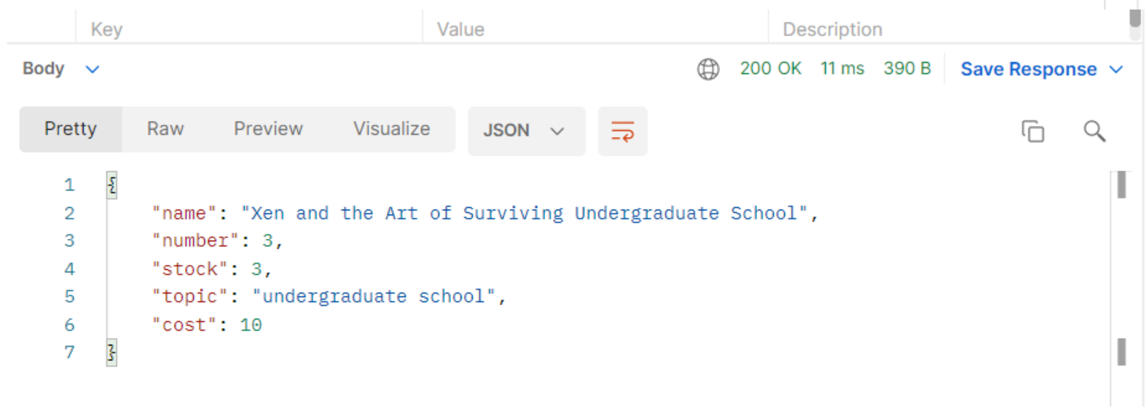


POST {{order_IP}}/purchase/2

KEY	VALUE	DESCRIPTION
Key	Value	Description

200 OK 94 ms 261 B

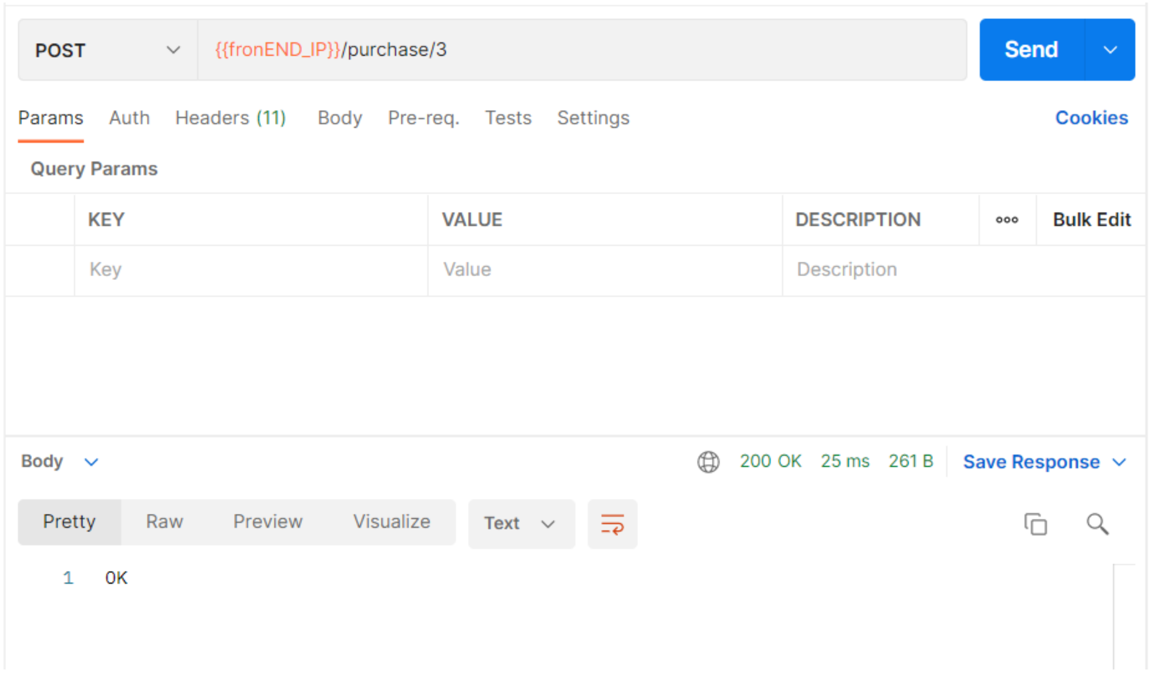
```
OK
```



Key	Value	Description

200 OK 11 ms 390 B

```
{
    "name": "Xen and the Art of Surviving Undergraduate School",
    "number": 3,
    "stock": 3,
    "topic": "undergraduate school",
    "cost": 10
}
```



POST {{fronEND_IP}}/purchase/3

KEY	VALUE	DESCRIPTION
Key	Value	Description

200 OK 25 ms 261 B

```
OK
```

Thanks