Computer Engineering Department

Course Name: Distributed Operating System

Instructor: : Dr. Samer Arendi	Lab2:Bazar.Com
Academic Year: 2022-2023	
Semester:1st	

Students	
1-Randa Fadi Alawneh (11819413)	2-Amal Zetawi
Performed on:	Submitted on:

Objectives:

The purpose of this lab is to teach you Replication, Caching, Consistency and to teach concepts of multi-tier web design and micro-services.

Introduction:

In this part, we will add replication and caching to improve request processing latency. While the front-end server in lab 1 was a very simple component, in this part, we will add two types of functionality to the front-end node. First, we will add an in-memory cache that caches the results of recent requests to the catalog servers. Second, assume that both the order and catalog server are replicated - their code and their database files are replicated on multiple machines.

To deal with this replication, the front end node needs to implement a load balancing algorithm that takes each incoming request and sends it to one of the replicas. We used Round Robin as load balancing algorithm.

We build Cache in-memory cache that integrated into the front-end server process, in which case, internal function calls are used to get and put items into the cache.

To ensure strong consistency guarantees, we implement a serverpush techniques where backend replicas send invalidate requests to the in-memory cache prior to making any writes to their database files. The invalidate request causes the data for that item to be removed from the cache. also we add other caching features such as a limit on the number of items in the cache, So we need a cache replacement policy ,we used LRU to replace older items with newer ones.

The replicas also use an internal protocol to ensure that any writes to their database are also performed at the other replica to keep them in sync with one another.

Procedure:

Here is an explanation of some of the steps of building the project:

- We need 3 computer ,So we use ubuntu as a front end server ,Windows Computer as as Catalog server ,another Windows Computer as Order Server , Every 3 computer connected on the same network.

- We use Node Framework, because Node provide Speed, Scalability, Productively. Also we use Visual Studio code as editor , In order to create and test the API we use POSTMAN.

- Order and Catalog Server need a database , So we use SQLITE which is a very simple database.

- We use SQLLITE Studio to browse SQLITE database.

HOW to Run the BAZAR.COM:

- Each Folder of Different Services, Put it in different computer, and open this folder by using visual code studio.

- We need to install Some libraries to each Server Such : Express.js, SuperAgent, Sqlite, body parser . For example we need Express.js to create sever running at specific port and to build route, also We need SuperAgent to make request.

- In order to install library we need to write visual code terminal in each services :npm intall Eexpress,npm install superagent, npm install sqlite, npm install bodyparser.

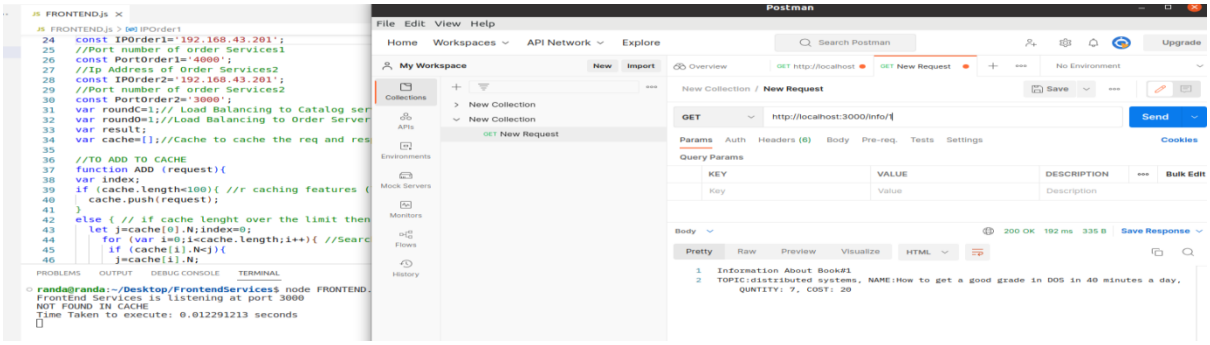
- In order to start each services , we need to write at each services terminal node filename.

- open postman in front end services and start to create API and test the result.

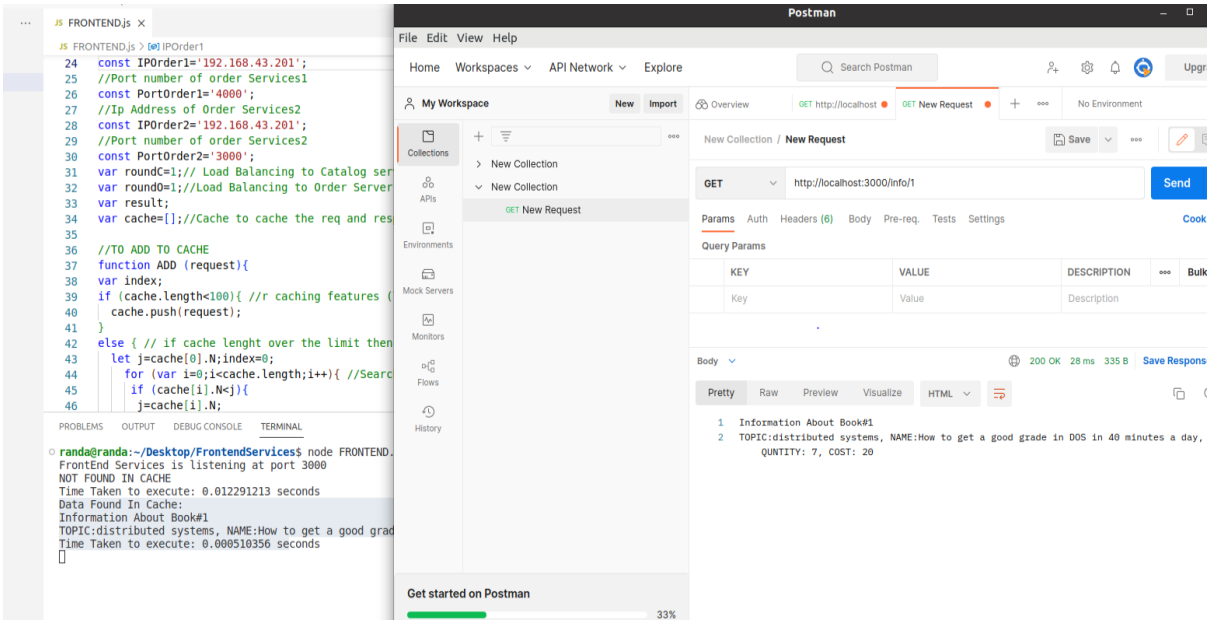
Result :

We test result by send different request to frontend server:

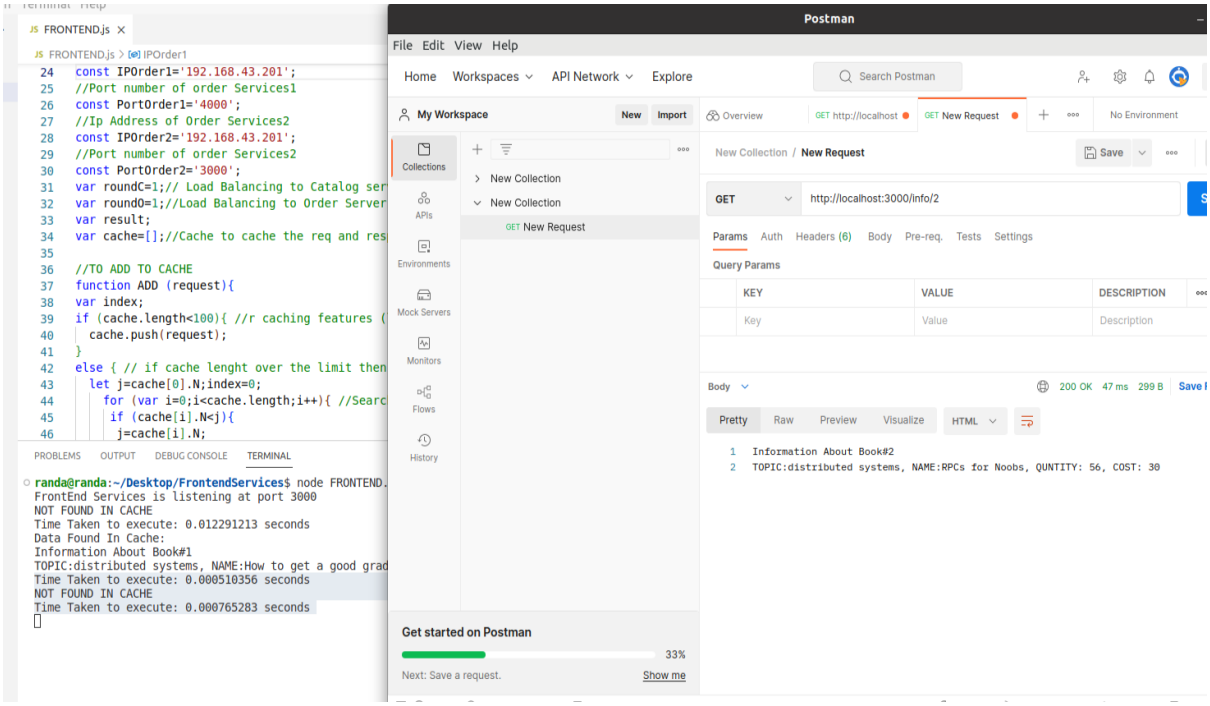
When send REQUSET first time to ask about information about specific book,(of course the information about book does not found in cache) so they send REQUEST to one catalog server from 2 replica:



When send REQUEST another time to ask about information about specific book, the req should save in cache so we don't go to catalog server to find the response:



Send another REQUEST to different book information to see the load balancing between 2 catalog server:



Load Balancing in catalog server :

```
17   console.log("Catalog Server is listening at port 5000");
18   });
19   // Connecting  SQlite Database
20   const sqlite3 = require('sqlite3');
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\AB\OneDrive\سطح المكتب\Catalog_Server1> node CatalogServer.js
Catalog Server is listening at port 5000
Catalog DataBase Connected
Information About Book#1
TOPIC:distributed systems, NAME:How to get a good grade in DOS in 40 minutes a day, QUNTITY: 7, COST: 20
```

```
17       console.log("Error Occurred - " + err.message);
18   }
19   else {
20       console.log("Catalog DataBase Connected");
```

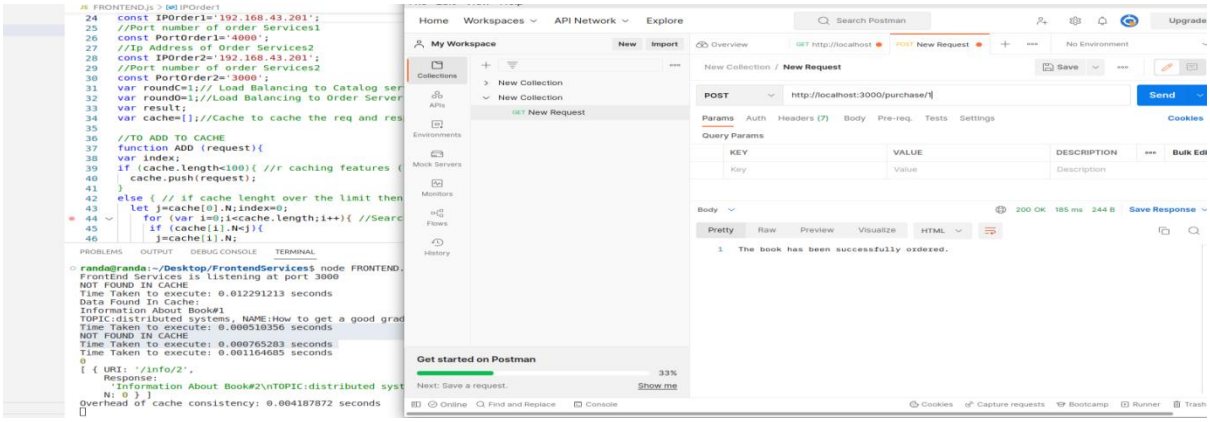
```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\AB\OneDrive\سطح المكتب\Catalog_Server2> node CatalogServer.js
Catalog Server is listening at port 6000
Catalog DataBase Connected
Information About Book#2
TOPIC:distributed systems, NAME:RPCs for Noobs, QUNTITY: 56, COST: 30
```

Send order REQUEST, when send a request to order server , then order server check no of items then if no>0 , we add the order to 2 server to achieve consistency between two replica , and we edit the amount value in two catalog server , and finally we send invalid REQ to front to remove the item information from cache:



CatalogServer1:

```
18      }
19      else {
20          console.log("Catalog DataBase Connected");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\AB\OneDrive\سطح المكتب\Catalog_Server2> node CatalogServer.js
Catalog Server is listening at port 6000
Catalog DataBase Connected
Information About Book#2
TOPIC:distributed systems, NAME:RPCs for Noobs, QUNTITY: 56, COST: 30
Update Done
```

CatalogServer2:

```
15  let db = new sqlite3.Database('./Catalog2.db' , (err) => {
16      if(err) {
17          console.log("Error Occurred - " + err.message);
18      }
19      else {
20          console.log("Catalog DataBase Connected");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\AB\OneDrive\سطح المكتب\Catalog_Server2> node CatalogServer.js
Catalog Server is listening at port 6000
Catalog DataBase Connected
Information About Book#2
TOPIC:distributed systems, NAME:RPCs for Noobs, QUNTITY: 56, COST: 30
Update Done
```

DB in 2 catalog server:

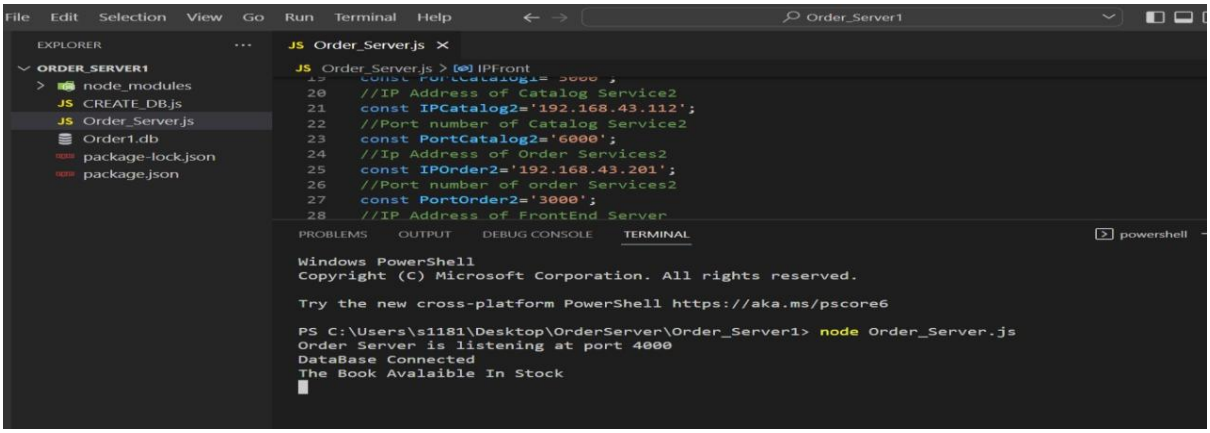
	ID	Topic	Name	Amount	Cost
1	1	distributed systems	How to get a good grade in DOS in 40 minutes a day	6	20
2	2	distributed systems	RPCs for Noobs	56	30
3	3	undergraduate school	Xen and the Art of Surviving Undergraduate School	63	50
4	4	undergraduate school	Cooking for the Impatient Undergrad	56	100

Grid view Form view

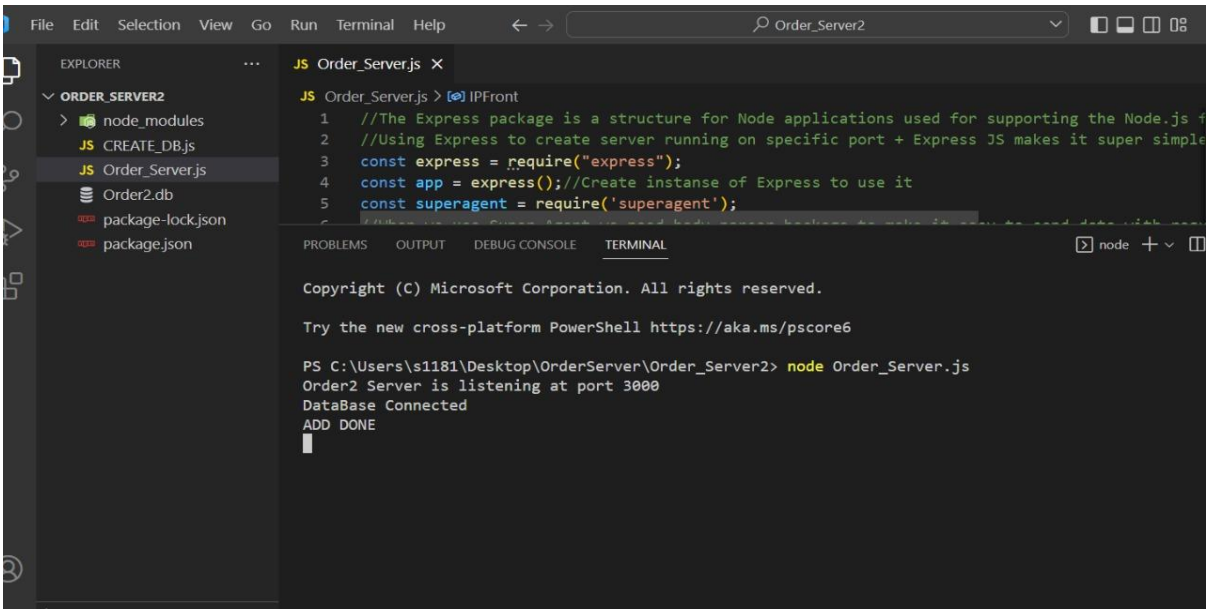
Filter data Total rows loaded: 4

	ID	Topic	Name	Amount	Cost
1	1	distributed systems	How to get a good grade in DOS in 40 minutes a day	6	20
2	2	distributed systems	RPCs for Noobs	56	30
3	3	undergraduate school	Xen and the Art of Surviving Undergraduate School	63	50
4	4	undergraduate school	Cooking for the Impatient Undergrad	56	100

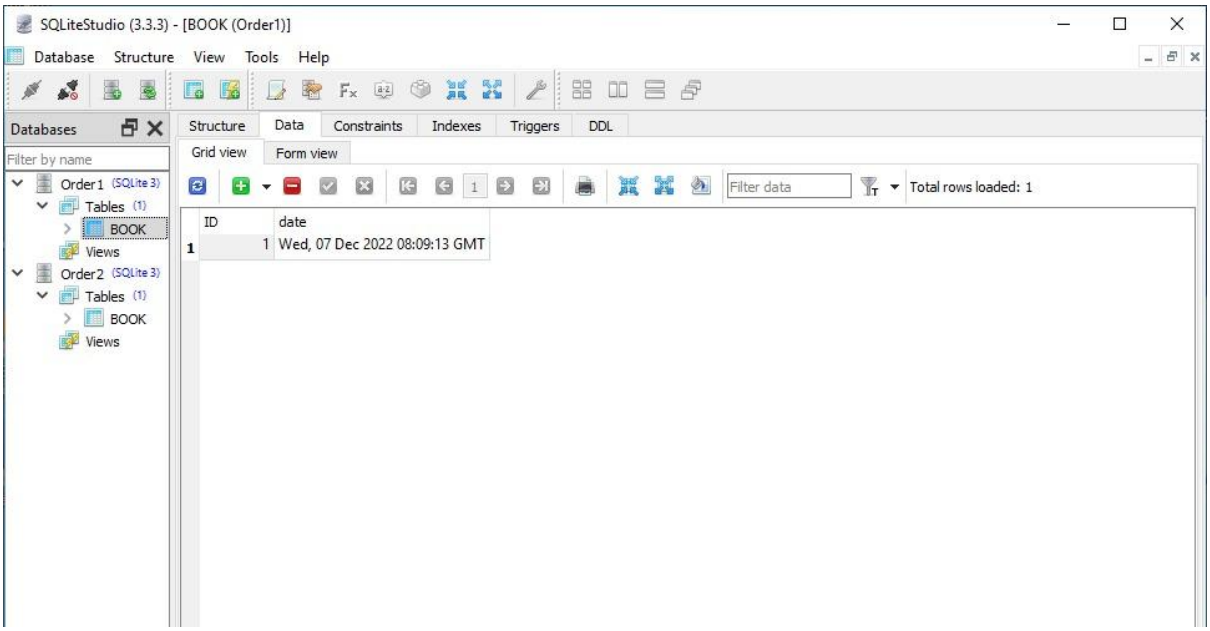
OrderServer1:



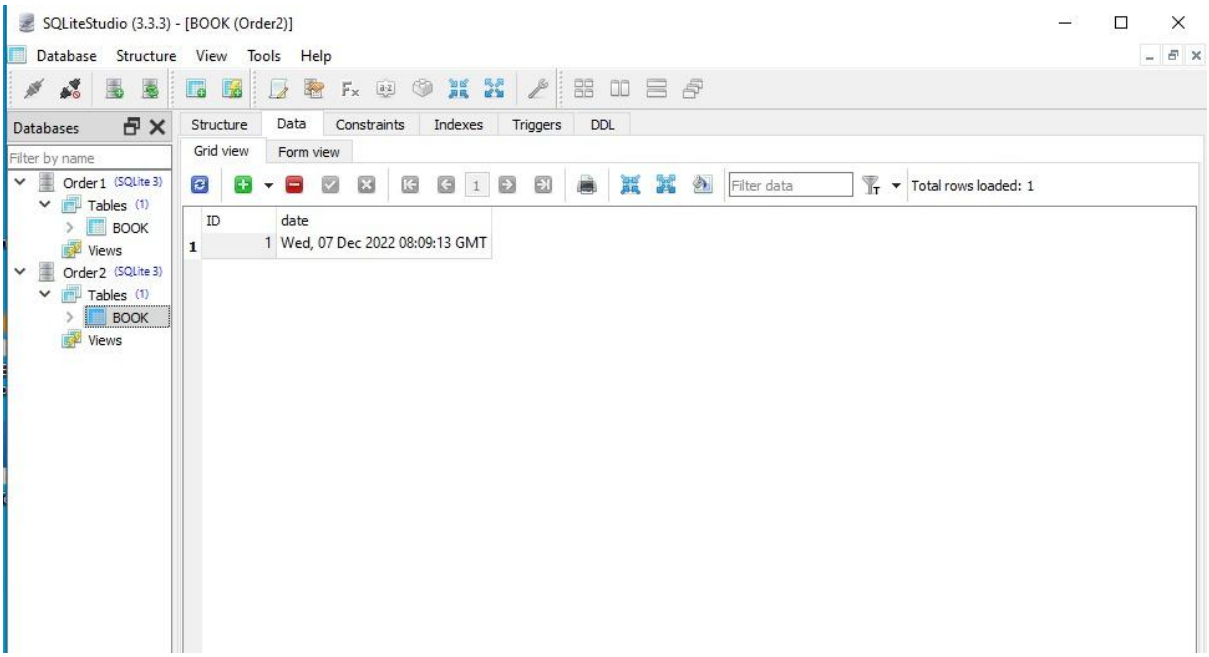
OrderServer2:



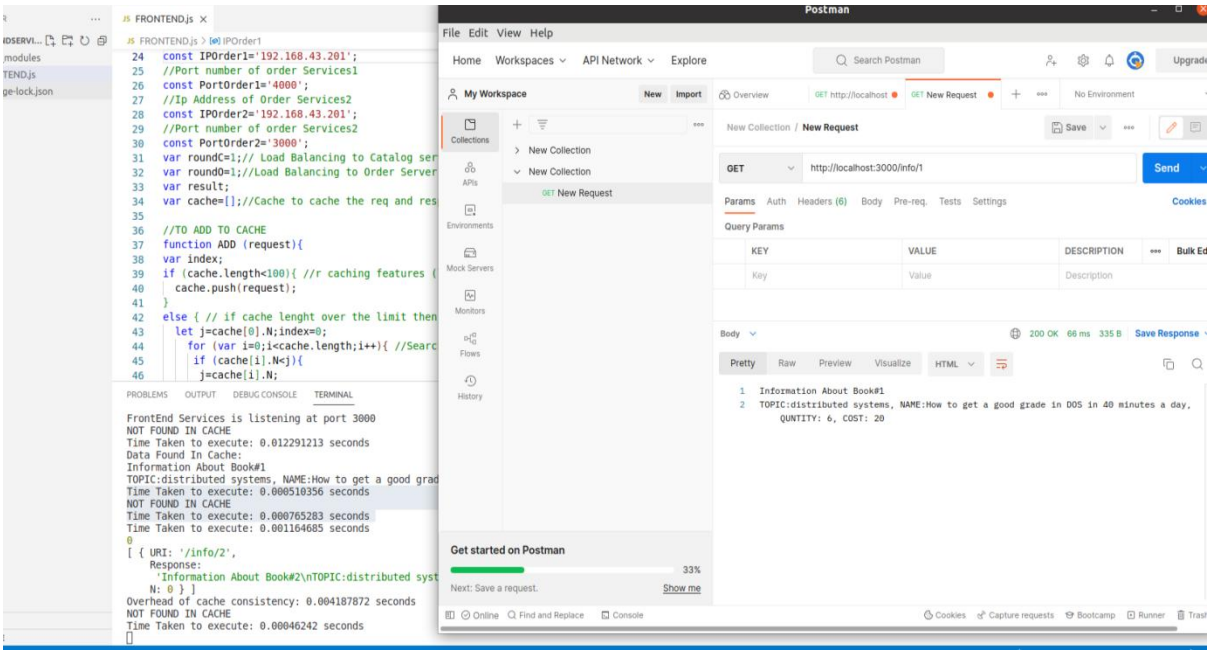
DB in orderServer1:



DB in orderServer2:



After purchase send info to Frontend to see that we don't find the value in cache and go to catalog server.



Experimental Evaluation and Measurements

Performance measurements or experiments to show/evaluate specific functionality:

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?

Request Type	Response time with cache	Response time without cache
Info/1	0.000486421 seconds	0.000515828 seconds
Search/distributed systems	0.000360265 seconds	0.000486423 seconds
Info/2	0.000465991 seconds	0.000445246 seconds
Search/undergraduate school	0.000323171 seconds	0.00046743 seconds
Info/9	0.000373168 seconds	0.000453741 seconds

2- We Construct a simple experiment that issues orders or catalog updates (i.e., database writes) to invalidate the cache and maintain cache consistency. What are the overhead of cache consistency operations?

Request Type	Overhead of cache consistency
Purchase/3	0.000653568 seconds
Purchase/1	0.00827149s
Purchase/2	0.000803418 seconds
Purchase/4	0.000752267 seconds

As we noticed in general, the use of cache reduces the response time to the request. It is true that sometimes the data is not found in the cache and sometimes is wasted time for searching in the cache(cache miss), but this time is not large compared by the time taken if we sending the request directly to the catalog server.

Cache must be handled with caution, as some data becomes invalid in the cache and we need to delete it when a request arrives to the frontend server to delete it and this waste some time by maintaining the consistency of the information in the cache.

However, despite what was previously mentioned, cache has a very important benefit in reducing response time and maintaining performance.

CONCLUSIN:

We Learn from this Project : Replication, Caching, Consistency, and how load balancing between replication.