

Bazar.com

Introduction

This simple app consists of 3 parts: front – end, order server, and catalog server:

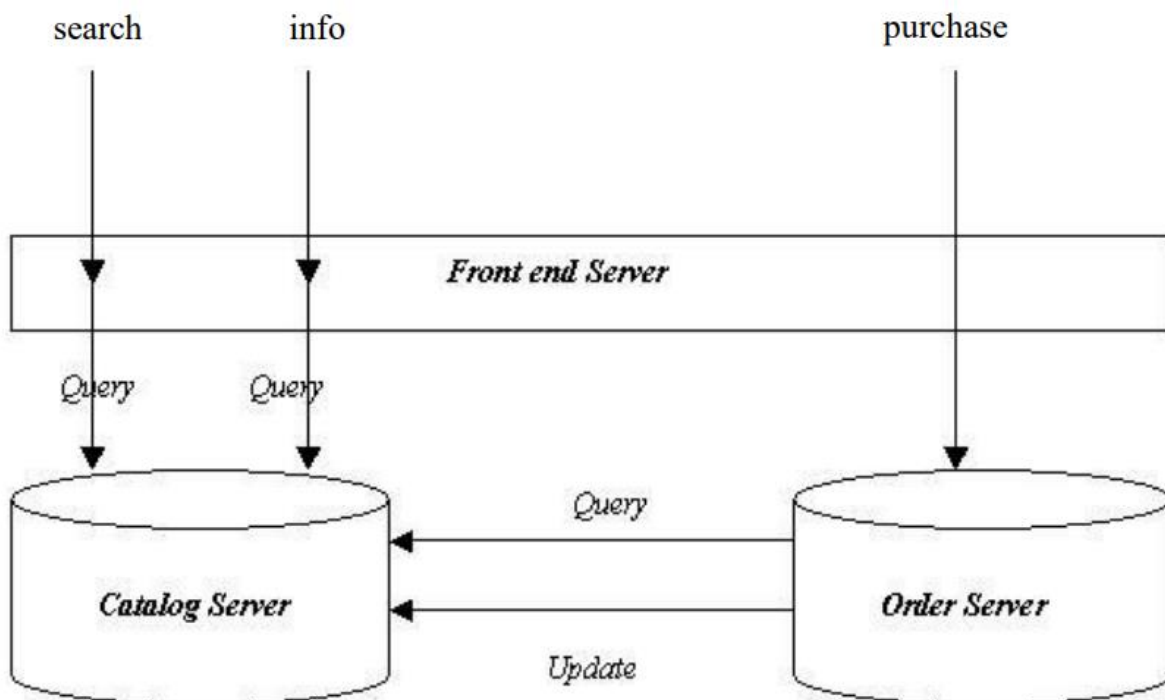
As a front end part, we used postman app to represent our front end side. By using postman app

We can make http request to other parts of app: order and catalog.

Catalog server is a server manages requests to books database.

Order server is a server handle purchase requests.

Here is a diagram explains the 3 parts:



We will explain all details later ...

Tools used to build this app:

- 1- Postman as Front-end side to make http requests.
- 2- Using Node.js to build Catalog and Order servers.
- 3- Using Docker to apply distribution concepts using light weight tools.

Procedure:

1)Front end server: we used postman as a front end to send http requests.
Here is a list of all request we will be sent:

request	goal
localhost:3007/books?id=d725	search by id
localhost:3007/books?topic=distributed systems	Search by topic
localhost:3007/books?title=Xen and the Art of Surviving Undergraduate School	Search by title
localhost:3008/books/purchase/80fc	Purchase a book
localhost:3007/books/incCount/80fc	Increase count of given book
localhost:3007/books/decCount/80fc	Decrease count of given book
localhost:3007/books/incCost/80fc	Increase cost of a given book
localhost:3007/books/decCost/80fc	Decrease cost of a given book

2) Catalog server: we build catalog server to handle these received http requests:

Search by id request (from front end – postman)

Search by topic request (from front end – postman)

Search by title request (from front end – postman)

Update cost request (from front end – postman)

Update count request (from front end – postman)

Purchase request (from order server)

3) Order server: we build this server to handle purchase requests

Purchase request (from front end – postman)

Steps:

- 1 - We used Node.js as backend tool to build catalog server and order server.
- 2 – We running both of server at the beginning locally to test all operation.
- 3 – Then we docarize catalog server and order server to make both of server are executable does not matter what is the hosting operating system so we can easily distribute components of our project.

From dashboard of docker we run catalog and order container , then we run catalog server and order server .

<input type="checkbox"/>	Name	Tag	Status	Created	Size	Actions
<input type="checkbox"/>	catalog 75260ac6c9b3	latest	In use	1 day ago	1.09 GB	
<input type="checkbox"/>	order 75260ac6c9b3	latest	In use	1 day ago	1.09 GB	

Now both of servers are running. We will send now http requests from postman:

We will send these requests within ibraheem prohect collection

▼	Ibraheem Project
	GET Search By Id
	GET Search by topic
	GET Search By title
	PUT Purchase a book
	PUT Increase count of given book
	PUT Decrease count of given book
	PUT Increase cost of a given book
	PUT Decrease cost of a given book

Search by Id:

GET localhost:3007/books?id=d725 **Send**

Params Auth Headers (7) Body Pre-req. Tests Settings **Cookies**

raw JSON **Beautify**

1

Body 200 OK 4 ms 256 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": "d725",
4     "title": "RPCs for Noobs",
5     "cost": "5$",
6     "topic": "distributed systems",
7     "count": 11
8   }
9 ]
```

Search by topic:

GET localhost:3007/books?topic=distributed systems **Send**

Params Auth Headers (7) Body Pre-req. Tests Settings **Cookies**

Body 200 OK 6 ms 383 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": "a95b",
4     "title": "How to get a good grade in DOS in 40 minutes a day",
5     "cost": "5$",
6     "topic": "distributed systems",
7     "count": 3
8   },
9   {
10    "id": "d725",
11    "title": "RPCs for Noobs",
12    "cost": "5$",
13    "topic": "distributed systems",
14    "count": 11
15  }
16 ]
```

Search by title:

GET

localhost:3007/books?title=Xen and the Art of Surviving Undergraduate School

Send

Params

Auth

Headers (6)

Body

Pre-req

Tests

Settings

Cookies

Query Params

<input checked="" type="checkbox"/>	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	title	Xen and the Art of Surviving Und...			
	Key	Value	Description		

Body

200 OK5 ms292 BSave as example

Pretty

Raw

Preview

Visualize

JSON

```
1 [
2   {
3     "id": "80fe",
4     "title": "Xen and the Art of Surviving Undergraduate School",
5     "cost": "5$",
6     "topic": "undergraduate school",
7     "count": 15
8   }
9 ]
```

Purchase a book:

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

- URL: `localhost:3008/books/purchase/80fe`
- Method: `PUT`
- Body: `Purchase is done`
- Status: `200 OK`, `53 ms`, `179 B`

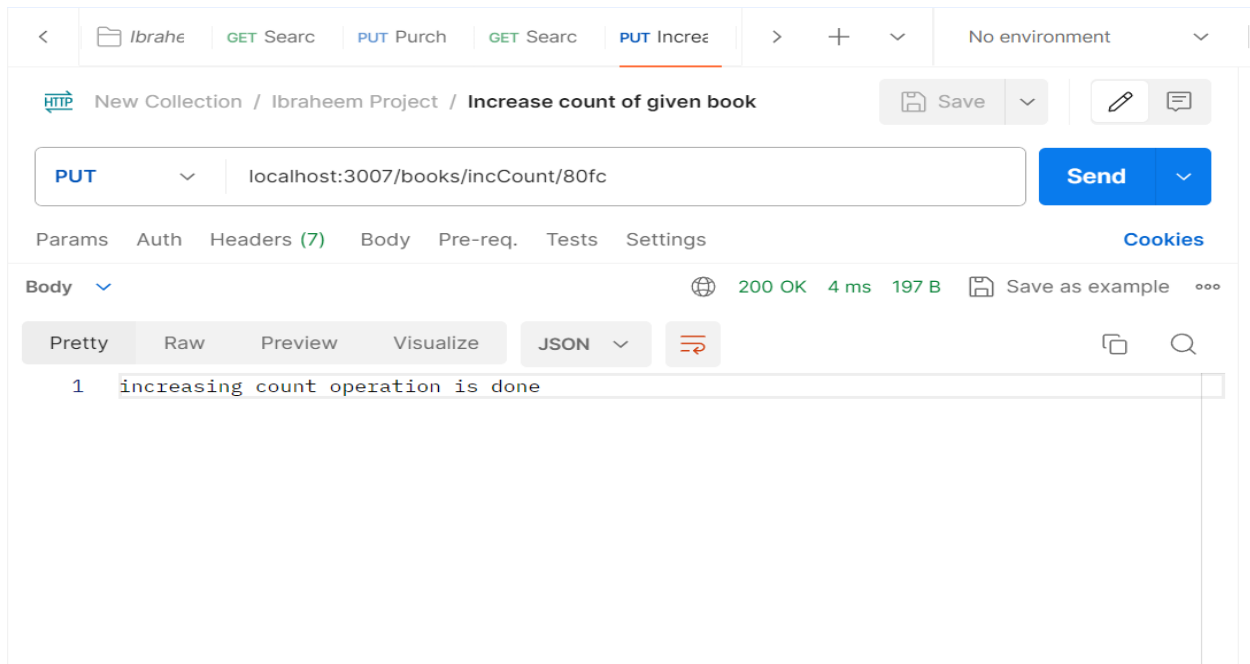
Before purchase the count of book is:

```
{
  "id": "80fe",
  "title": "Xen and the Art of Surviving Undergraduate School",
  "cost": "5$",
  "topic": "undergraduate school",
  "count": 15
},
```

And after that is:

```
{
  "id": "80fe",
  "title": "Xen and the Art of Surviving Undergraduate School",
  "cost": "5$",
  "topic": "undergraduate school",
  "count": 14
},
```

Increase count of a given book:



Before increase count of given book:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 6,
  "topic": "undergraduate school",
  "count": 30
}
```

After:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 6,
  "topic": "undergraduate school",
  "count": 31
}
```

Decrease count of a given book:

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

- Method:** PUT
- URL:** localhost:3007/books/decCount/80fc
- Buttons:** Send, Cookies
- Tabs:** Params, Auth, Headers (7), Body, Pre-req., Tests, Settings
- Table:**

Key	Value	Description
Key	Value	Description
- Body:** decrease count operation is done
- Status:** 200 OK, 4 ms, 195 B
- Actions:** Save as example, Copy, Search

Before decrease count of given book:

```
{  
  "id": "80fc",  
  "title": "Cooking for the Impatient Undergrad",  
  "cost": 6,  
  "topic": "undergraduate school",  
  "count": 31  
}
```

After:

```
{  
  "id": "80fc",  
  "title": "Cooking for the Impatient Undergrad",  
  "cost": 6,  
  "topic": "undergraduate school",  
  "count": 30  
}
```


Increase cost of a given book:

PUT localhost:3007/books/incCost/80fc Send

Params Auth Headers (7) Body Pre-req. Tests Settings Cookies

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body 200 OK 6 ms 196 B Save as example

Pretty Raw Preview Visualize JSON

```
1 increasing cost operation is done
```

Before increase cost of a given book:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 6,
  "topic": "undergraduate school",
  "count": 30
}
```

After:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 7,
  "topic": "undergraduate school",
  "count": 30
}
```

Decrease cost of a given book:

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

- Method:** PUT
- URL:** localhost:3007/books/decCost/80fc
- Buttons:** Send, Cookies
- Tabs:** Params, Auth, Headers (7), Body, Pre-req., Tests, Settings
- Table:**

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		
- Body:** 200 OK, 9 ms, 196 B. Save as example
- Response Format:** Pretty, Raw, Preview, Visualize, JSON
- Response Body:**

```
1 decreasing cost operation is done
```

Before decreasing cost of a given book:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 5,
  "topic": "undergraduate school",
  "count": 30
}
```

After:

```
{
  "id": "80fc",
  "title": "Cooking for the Impatient Undergrad",
  "cost": 4,
  "topic": "undergraduate school",
  "count": 30
}
```

Notes:

1 – Just purchase operation handled by order server , and other requests are handled by catalog server .

2 – We check possibility of success purchase operation on order server not in catalog server, and this reduce the traffic of requests between catalog and order server.

3- There is a copy of database in catalog exists in order server

here is some output from command line:

