**Bazar.com**

**Introduction**

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

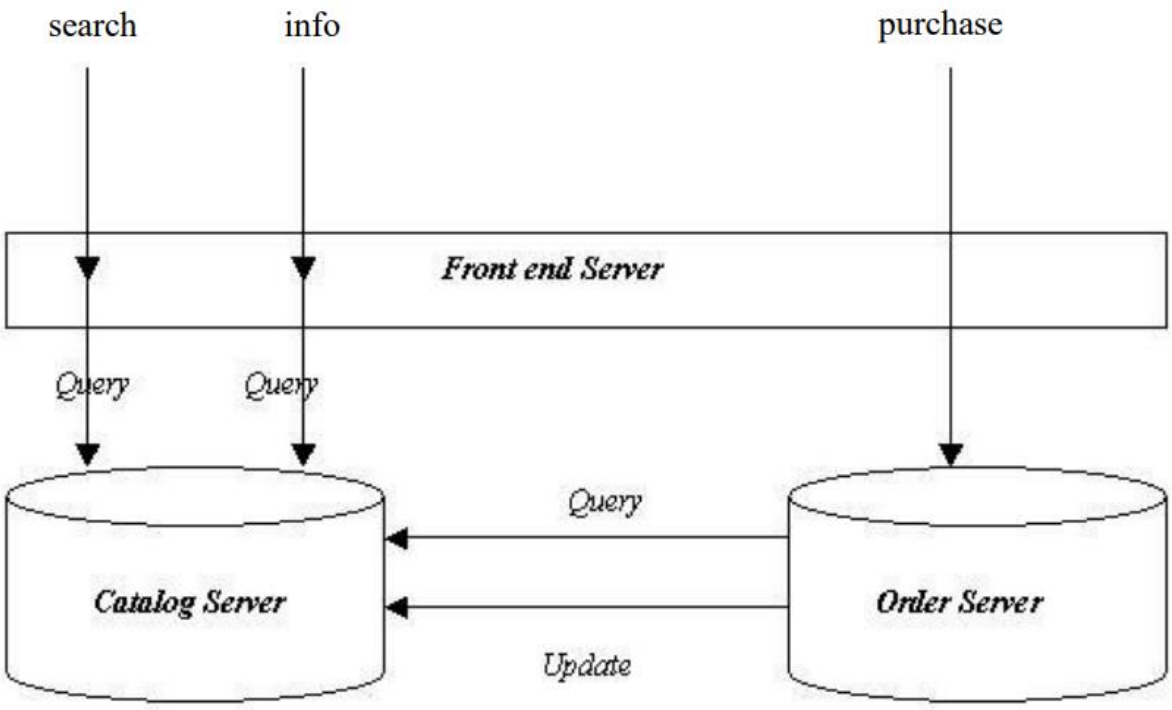
**As a front end part** , we used postman app to represent our front end side. By using postman app

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 explains 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)

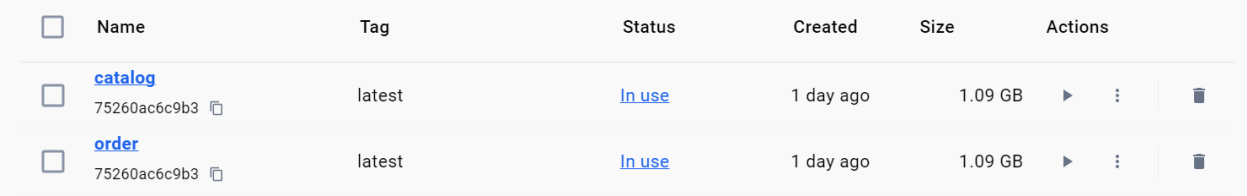
Steps:

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.

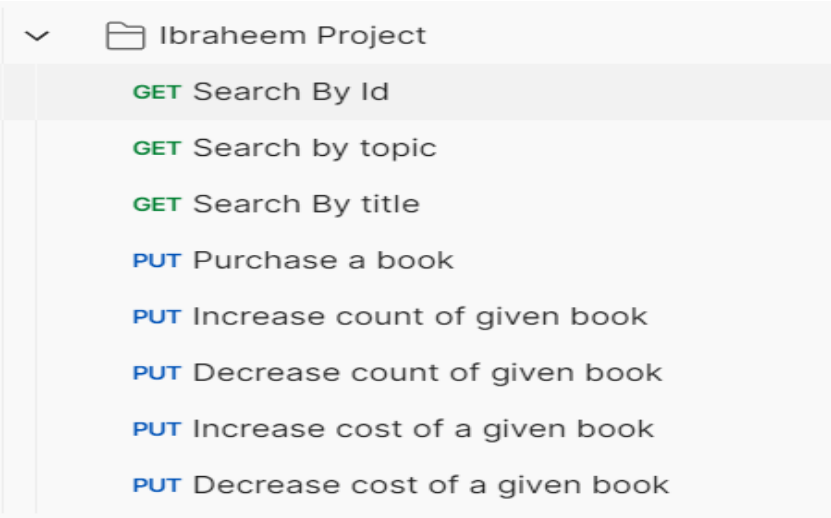
From dashboard of docker we run catalog and order container , then we run catalog server and order server .



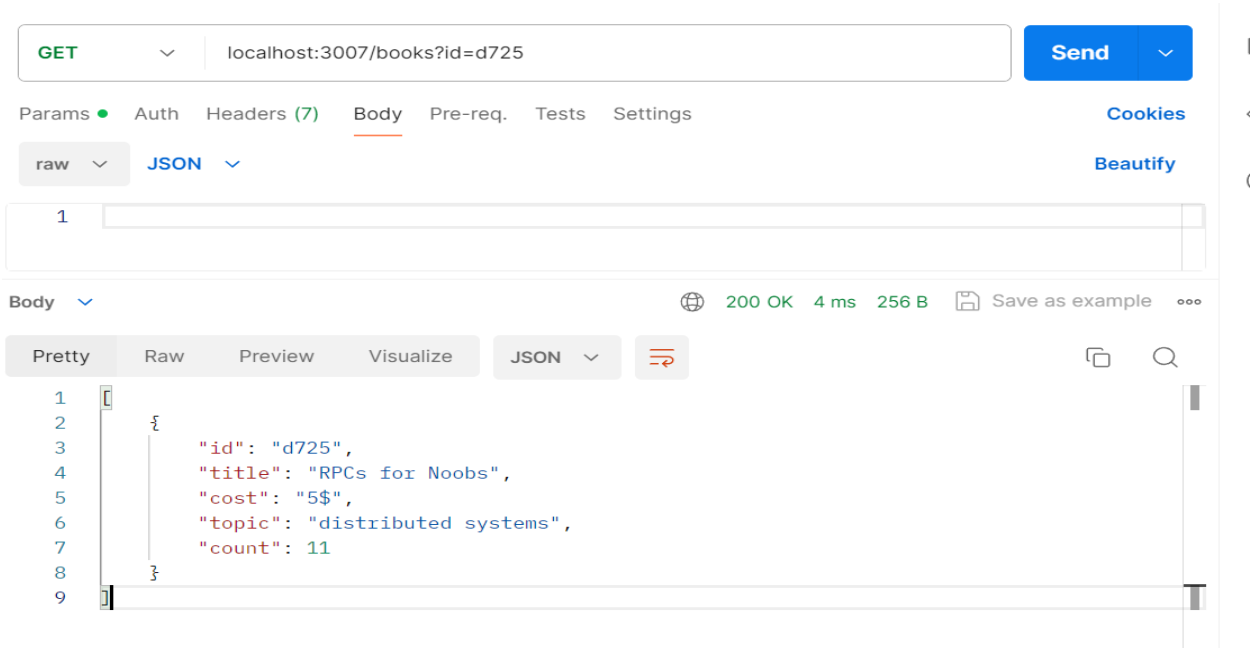
| Name | Tag | Status | Created | Size | Actions |
|---|---|---|---|---|---|
| catalog 75260ac6c9b3 | latest | In use | 1 day ago | 1.09 GB | |
| order 75260ac6c9b3 | latest | In use | 1 day ago | 1.09 GB | |

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

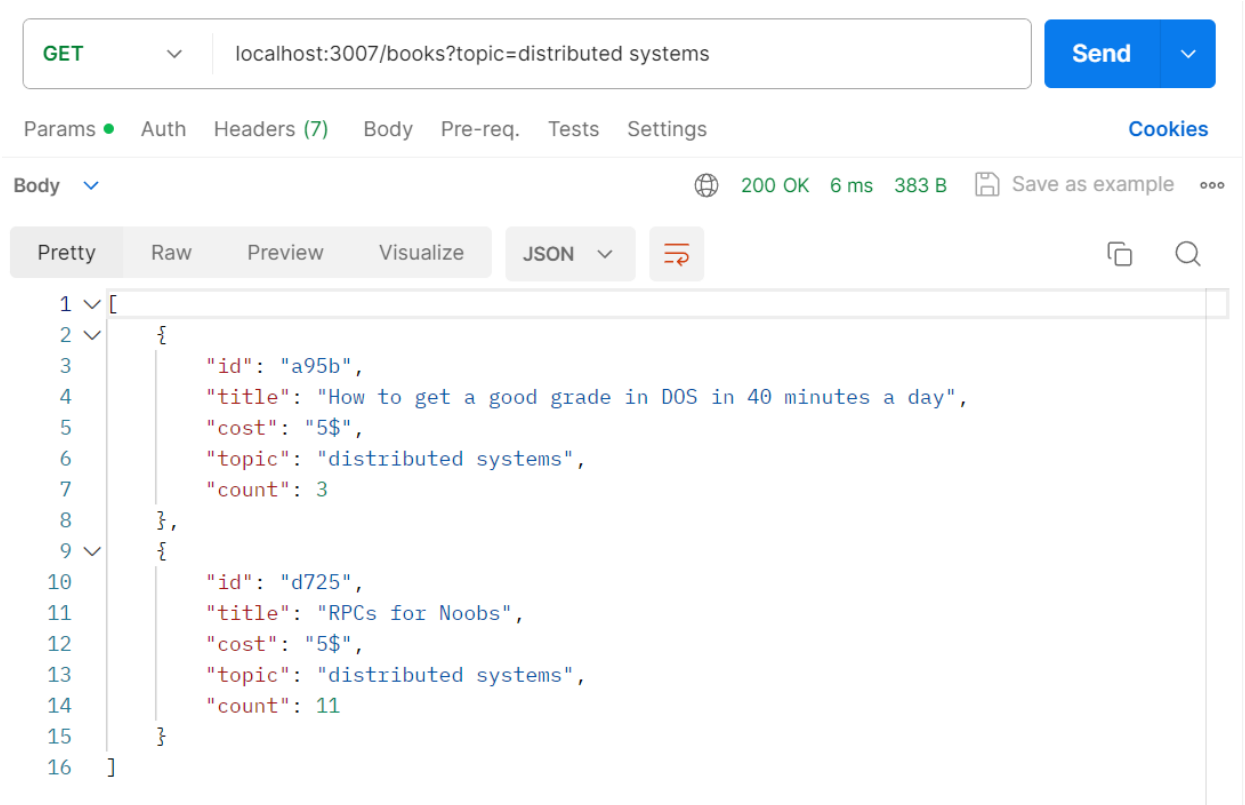
We will send these requests within ibraheem  prohect collection



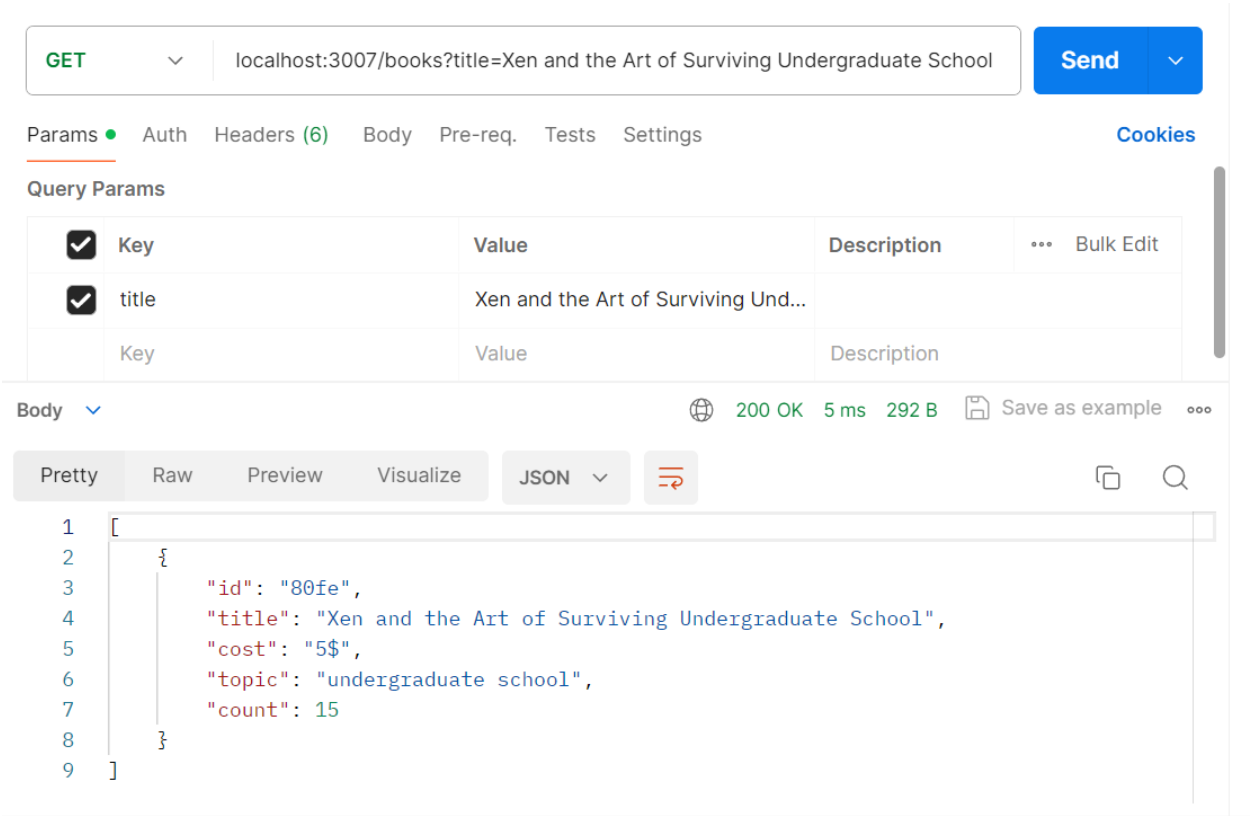
Search by Id:



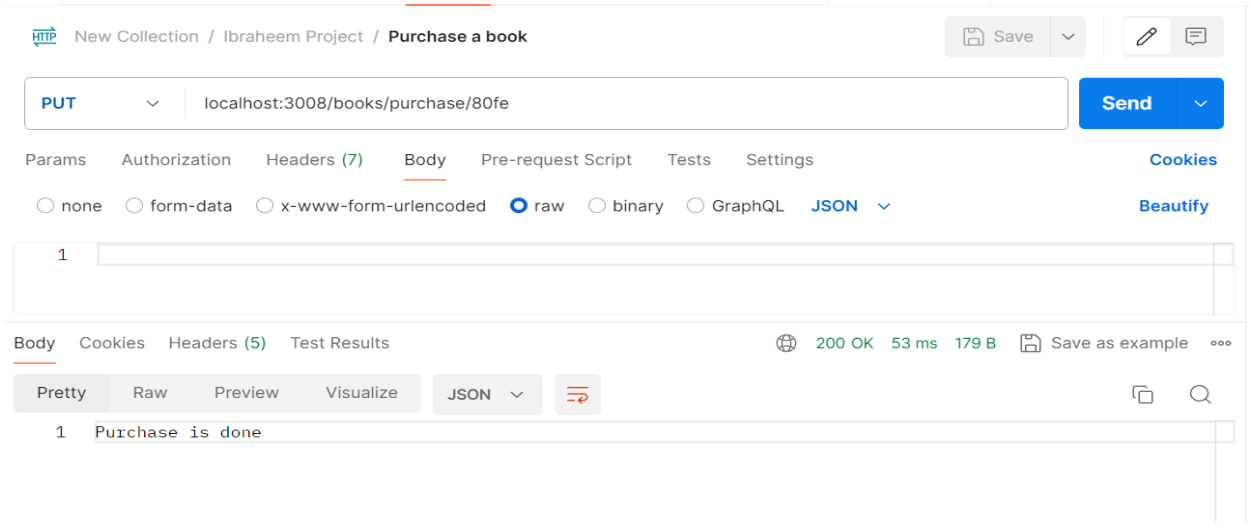

Search by topic:



Search by title:



Purchase a book:



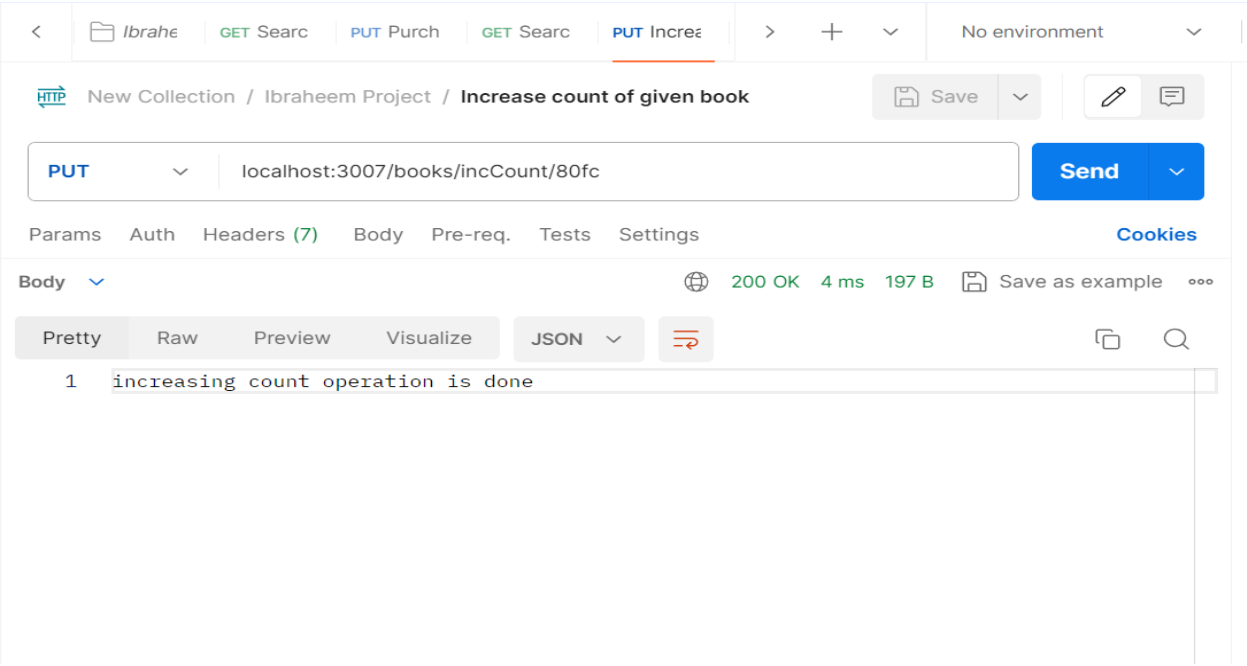

Before purchase the count of book is:


```
{
  "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:



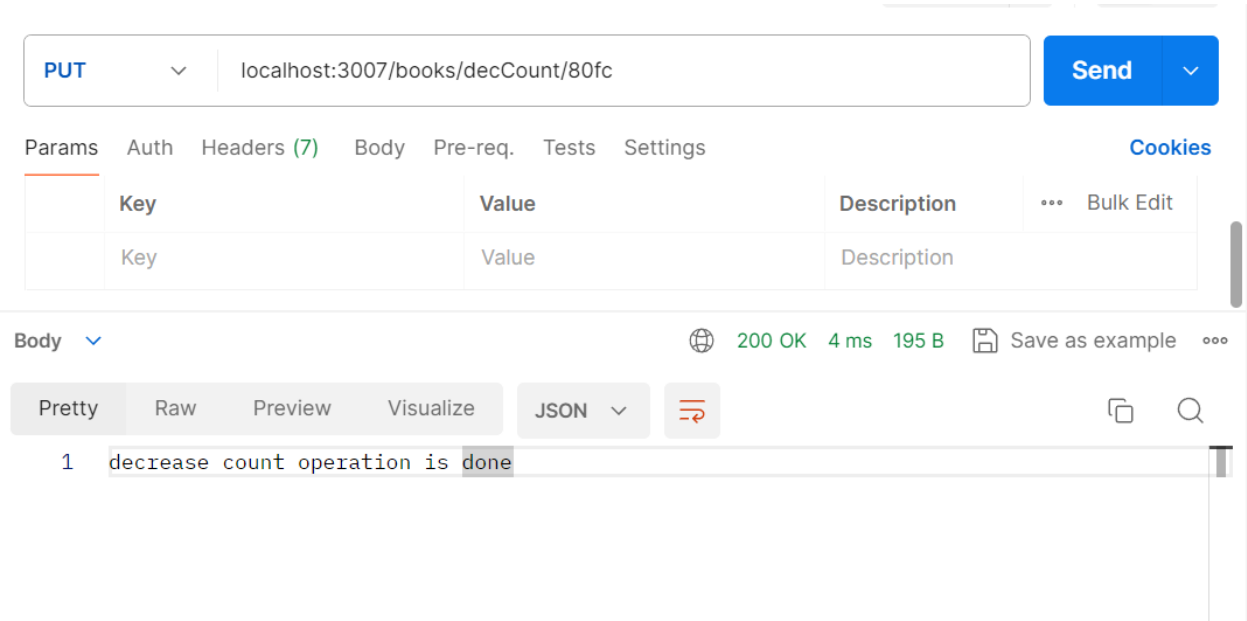

Before increase count of given book:


```
{
  "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:



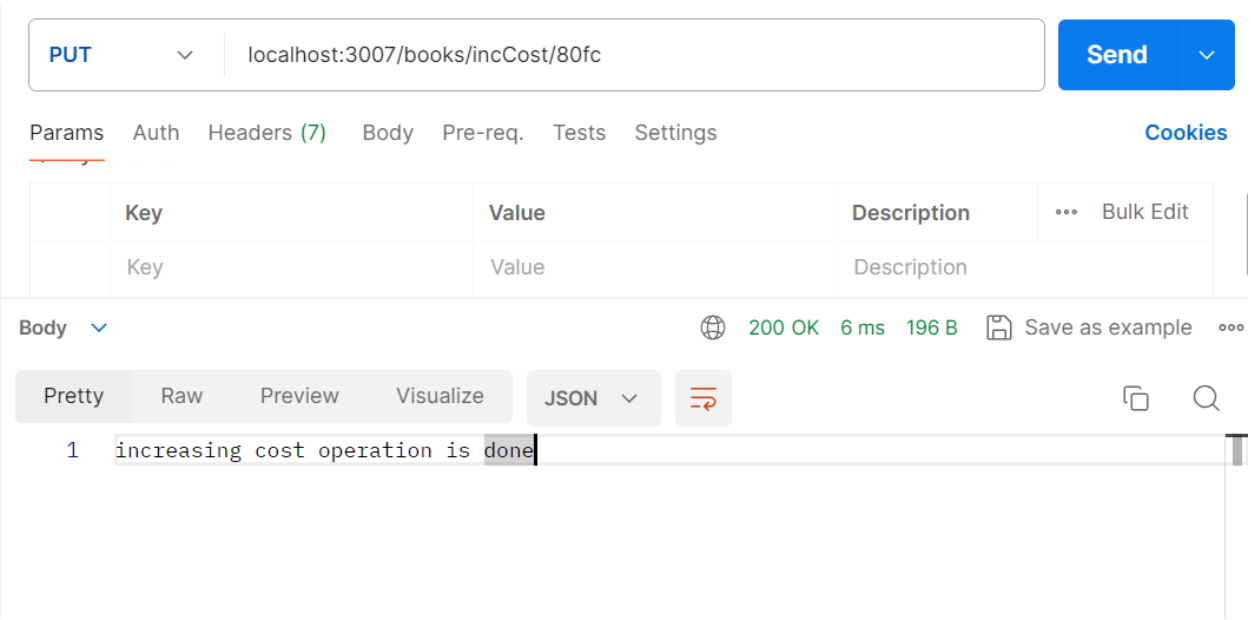

Before decrease count of given book:


```
{
  "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:



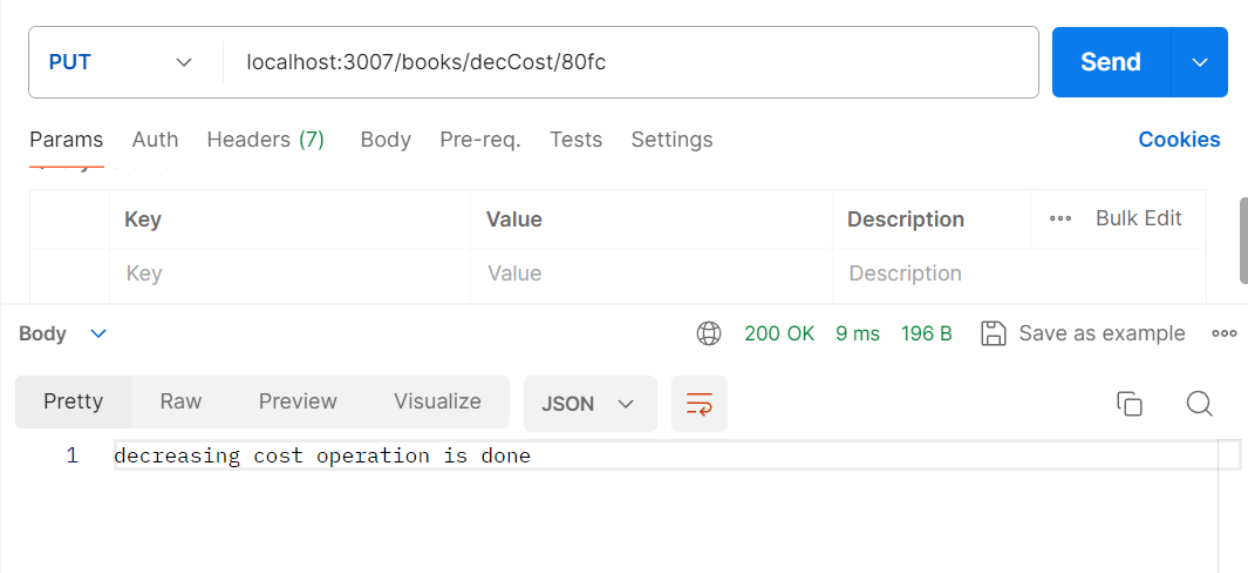

Before increase cost of a given book:


```
{
  "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:




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: