Bazar.com: A Multi-tier Online Book Store

Areen Ateeq (11923874)

Frontend Server:

It performs all website operations (info, search, and purchase), the first two operations trigger queries on the catalog server, the purchase operations triggers a request to the order server.

search:

```
2 usages
class SearchController(ViewSet):
    def retrieve(self, request, pk):
        books = CatalogService.search_books(pk)
        if books is not None:
            return JsonResponse(data={'data': books['data']}, status=200)
        else:
            return JsonResponse(data={'status': 'no books with this name'}, status=200)
```

• info:

```
class InfoController(ViewSet):
    def list(self, request):
        books = CatalogService.get_books()
        if books is not None:
            return JsonResponse(data={'data': books['data']}, status=200)
        else:
            return JsonResponse(data={'status': 'no books with this name'}, status=200)

def retrieve(self, request, pk):
        book = CatalogService.get_book_by_id(pk)
        if book is not None:
            return JsonResponse(data={'data': book['data']}, status=200)
        else:
            return JsonResponse(data={'status': 'no books with this name'}, status=200)
```

•purchase:

```
class PurchaseController(ViewSet):
    def update(self, request, pk):
        status = CatalogService.purchase(pk, request.data.get('item_number'))
        if status is not None:
            return JsonResponse(data={'data': 'Purchase done'}, status=200)
        else:
            return JsonResponse(data={'data': 'Not allowed to purchase'}, status=200)
```

Catalog Server:

Catalog server: this server has two types of operation. Update and query on information and search.

I build this model (ORM):

```
class Catalog(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200)

7 usages
class Book(models.Model):
    id = models.AutoField(primary_key=True)
    name = models.CharField(max_length=200)
    catalog = models.ForeignKey(Catalog, on_delete=models.CASCADE)
    count = models.IntegerField(default=0, null=False)
    cost = models.PositiveIntegerField(default=0, null=False)
```

Id of book → AutoField which means id will auto increment

Name → name of book with maximum size

Count → how much books I have (must be not null)

Cost → Cost of book

implementation GET method: returns all matching entries

```
def list(self, request):
   books = Book.objects.all()
   return JsonResponse({'data': BookSerializer(books, many=True).data})
```

Get method -> book ID: an item is specified and all relevant details are returned, Filter depend on id. (book id [in class] = pk[id in URL])

```
def retrieve(self, request, pk):
   book = Book.objects.filter(id=pk).first()
   return JsonResponse({'data': BookSerializer(book, many=False).data})
```

Update function

```
def update(self, request, pk):
    book = Book.objects.filter(id=pk).first()
    if not book:
        return JsonResponse({'data': 'Resource Not Found'}, status=422)
        book.count = int(request.data.get('item_number'))
        book.cost = int(request.data.get('item_cost'))
        book.save()
    return JsonResponse({'data': 'Purchase Done'})
```

Check the book will be update exist in system, if not return JsonResponse to indicate resource not found. Then get count and cost if it exist.

Search function

The pk represent here the topic of the book(catalog name)

```
2 usages
class SearchController(ViewSet):
    def retrieve(self, request, pk):
        item_name = pk
        books = Book.objects.filter(catalog__name__contains=item_name).all()
        return JsonResponse({'data': BookSerializer(books, many=True).data})
```

Order Server:

Purchase operation delivered to Order Server. Order Server send query to Catalog Server, check if book exist then make update to Catalog Server to decrease quantity by item number.

```
class PurchaseController(ViewSet):

def update(self, request, pk):
    book = CatalogServer.get_book_data(pk)
    if book['data']['count'] >= int(request.data.get('item_number')) and book['data']['count']
    CatalogServer.purchase_book(
        int(pk),
        int(book['data']['count']) - int(request.data.get('item_number')),
        int(book['data']['cost'])
    )
    return JsonResponse(data={'data': 'successful'}, status=200)
    else:
        return JsonResponse({'data': 'Not Allowed to purchase more than existing'}, status=403]
```

To take data from Catalog I code a catalog_service.py which have many function will be send request to Catalog server.

Finally:

I have three services, uploaded them in three containers, each service has docker file

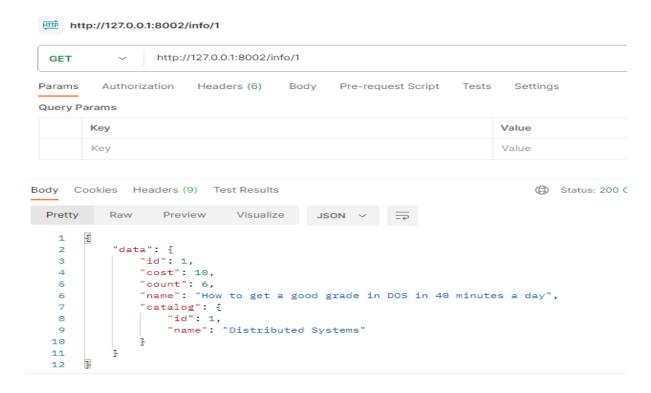
To make communication between them I build composer file to connect all of them on same network which called "my network".

Then I run all services by these command on cmd:

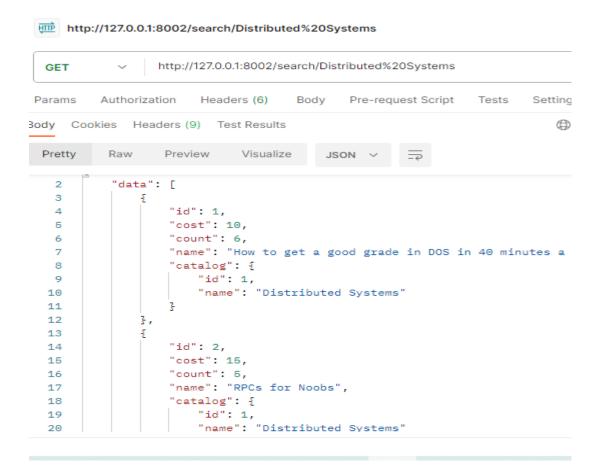
```
:\Users\ALFALAK\cd Downloads
:\Users\ALFALAK\Downloads>cd AREEN
:\Users\ALFALAK\Downloads\AREEN>cd Areen DOS
:\Users\ALFALAK\Downloads\AREEN\Areen DOS>docker-compose up -d
[+] Building 0.0s (0/0)
[+] Running 3/0

[ Container areendos-catalogweb-1 Running
[ Container areendos-orderweb-1 Running
[ Container areendos-feweb-1 Running
] Container areendos-feweb-1 Running
[ Container areendos-feweb-1 Running
] Container areendos-feweb-1 Running
```

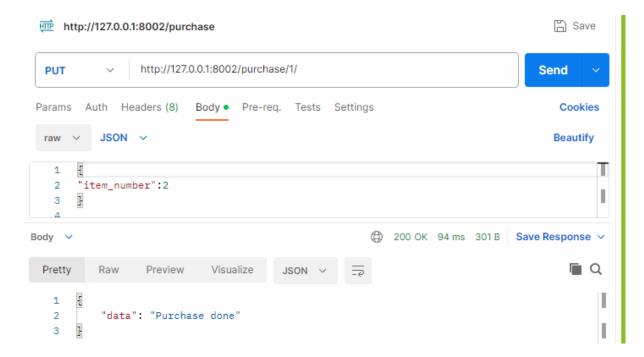
/info/id → Frontend (info)



Search depend on topic - Frontend



Purchase by id - Frontend

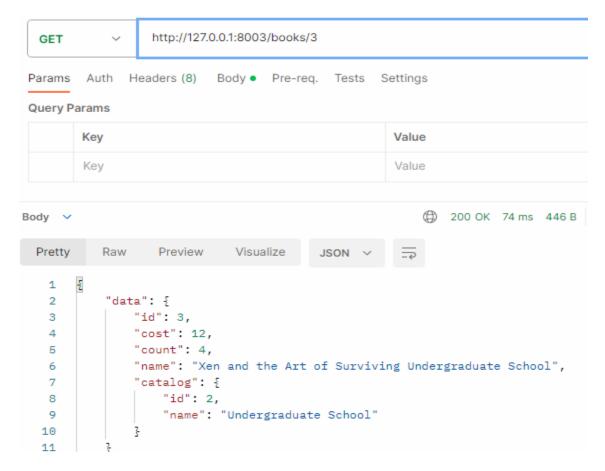


Use Catalog Server:

Search on topic and then return titles of books.

```
GET
                   http://127.0.0.1:8003/search/Distributed%20Systems
arams
                        Headers (6)
                                          Body
                                                   Pre-request Script
                                                                         Tests
                                                                                  Settings
Query Params
                                                                               Value
        Kev
ody Cookies Headers (9) Test Results
                                                                                       Status: 200 OK Time: 6
                     Preview
           Raw
                                 Visualize
Pretty
                                              JSON ~
                     "name": "How to get a good grade in DOS in 40 minutes a day",
  8
                      "catalog": {
                          "id": 1,
"name": "Distributed Systems"
  9
 10
 11
 12
 13
                     "id": 2,
 14
                     "cost": 15,
"count": 5,
"name": "RPCs for Noobs",
 15
 16
 17
                      'catalog": {
    "id": 1,
    "name": "Distributed Systems"
 18
 19
 20
```

Info/id



Update cost and quantity

