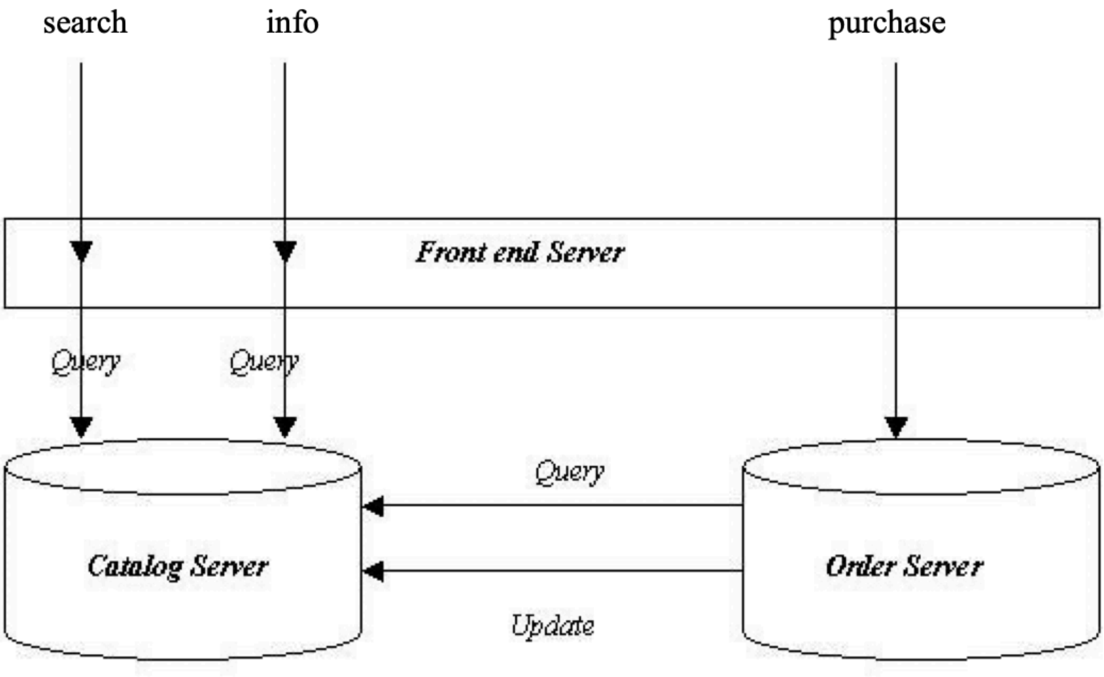
DOS Lab 1

Bazar.com

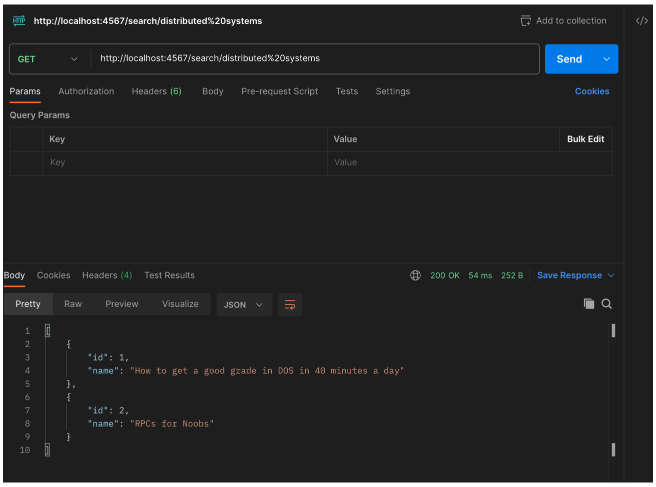
Mohammad Aker and Yousef Abd-Alsalam
Dr. Samer Arandi

In this part of the project, bazar.com was implemented as the picture shows below:



The project is divided into 3 modules:

1. Catalog server: responsible for communicating with the books file, it houses 2 queries, "search" that the topic as a parameter and returns the books that are categorized into that topic, and "info" that takes a book ID as a parameter to return its name, stock, and price. In addition to these 2 queries, an endpoint was created to meet the needs of the order server (updateStock) that takes the book ID as a parameter to decrement the stock by 1. Shown below are "search" and "info" queries:

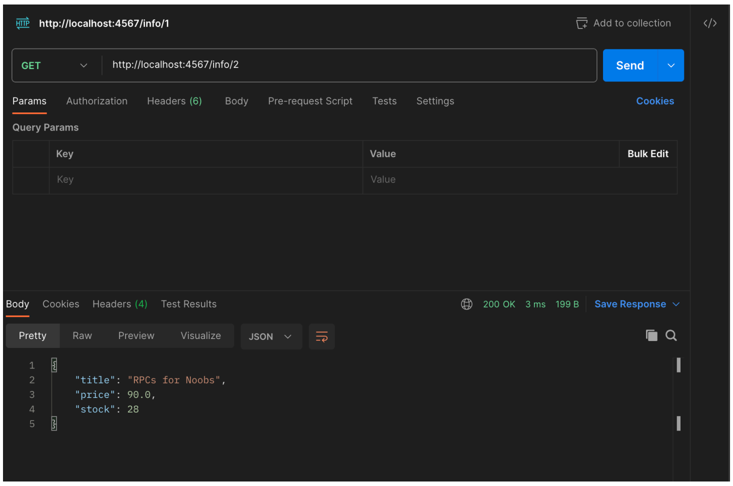


http://localhost:4567/search/distributed%20systems

GET http://localhost:4567/search/distributed%20systems

200 OK 54 ms 252 B

```
[
    {
        "id": 1,
        "name": "How to get a good grade in DOS in 40 minutes a day"
    },
    {
        "id": 2,
        "name": "RPCs for Noobs"
    }
]
```

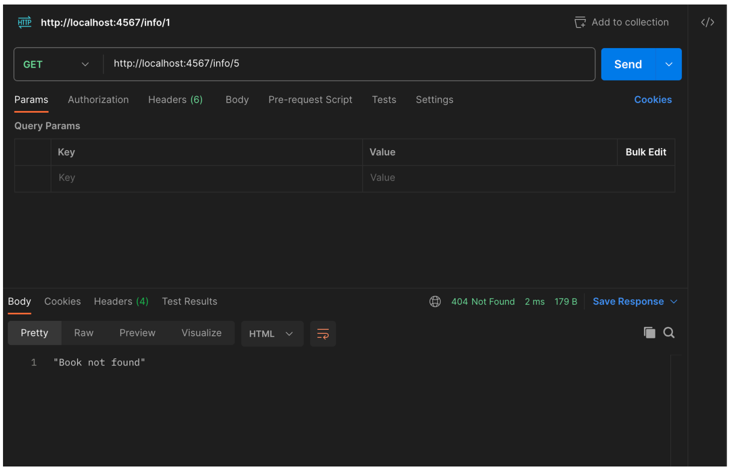


http://localhost:4567/info/1

GET http://localhost:4567/info/2

200 OK 3 ms 199 B

```
{
    "title": "RPCs for Noobs",
    "price": 90.0,
    "stock": 28
}
```

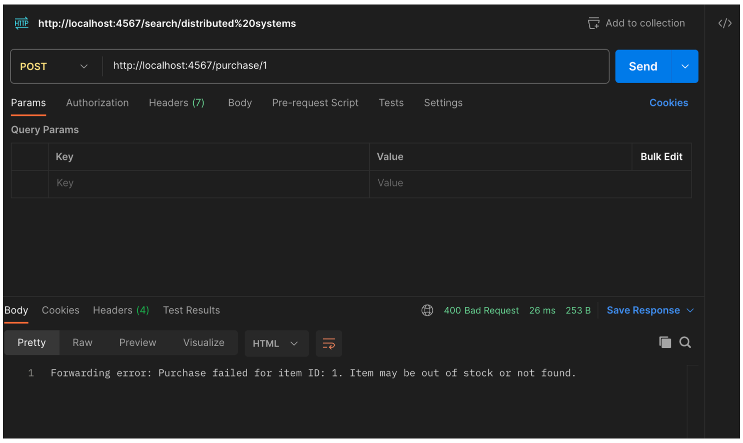


GET http://localhost:4567/info/5 Send

404 Not Found 2 ms 179 B

```
"Book not found"
```

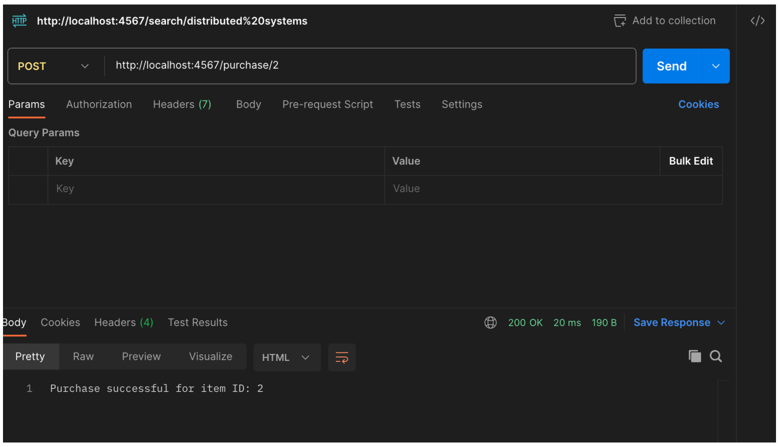
2. Order server: this server has only 1 request (purchase) that takes the book ID as a parameter and sends an "info" query to the Catalog server, if the stock is not 0, then it sends a "updateStock" request that fulfills the purchasing process. Shown below are examples of when a purchase fails and when a purchase is a success:



POST http://localhost:4567/purchase/1 Send

400 Bad Request 26 ms 253 B

```
Forwarding error: Purchase failed for item ID: 1. Item may be out of stock or not found.
```



POST http://localhost:4567/purchase/2 Send

200 OK 20 ms 190 B

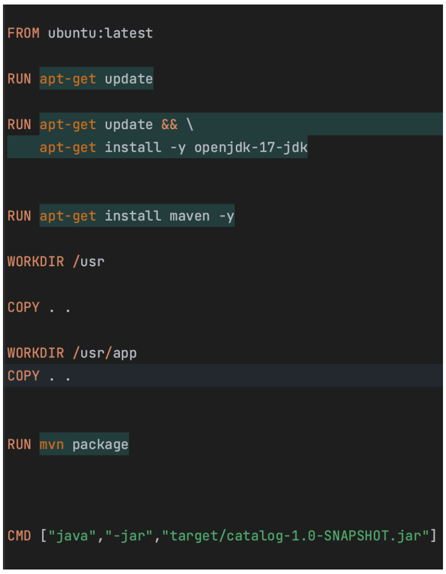
```
Purchase successful for item ID: 2
```

3. Gateway server: this server is only responsible for forwarding the requests to the Catalog and Order servers and forwarding the responses to the client. All the above requests were on the Gateway server (notice the same port).

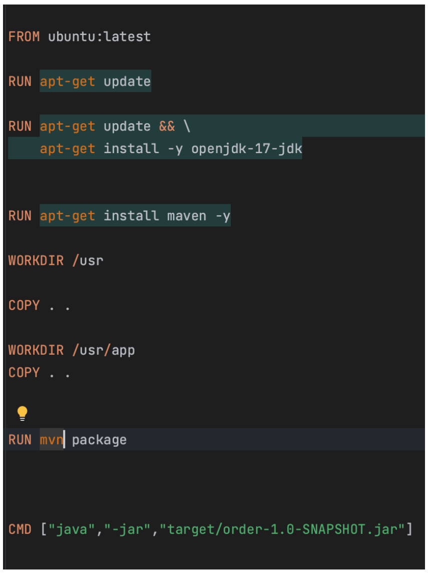
Dockerization:

3 docker containers (1 for each server) were required. To further emphasize the importance of Docker, we added a docker-compose file that when running it, runs all the server at once with the dependencies clarified in it. We used the same docker file for each server, excluding the JAR file name.

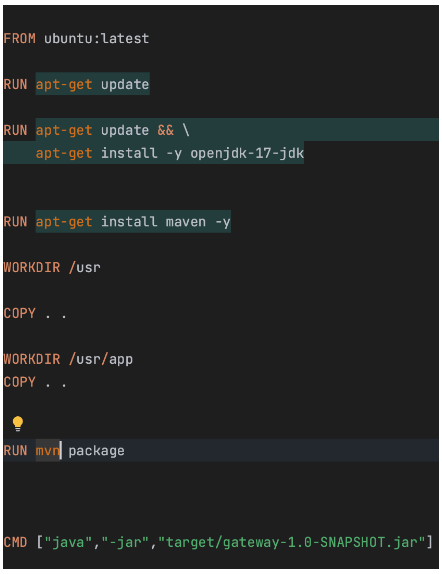
Shown below are the docker files for the 3 servers and the docker-compose file:



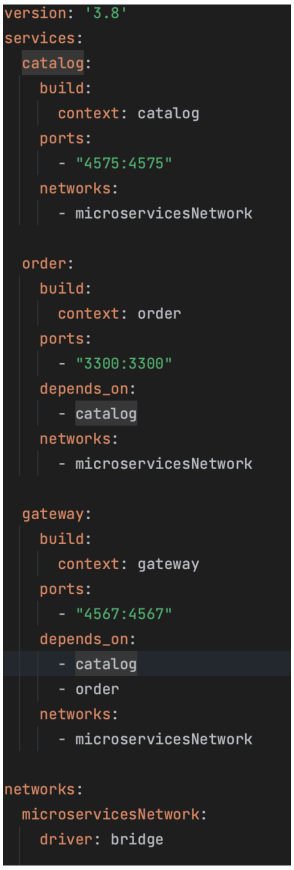
```
FROM ubuntu:latest

RUN apt-get update

RUN apt-get update && \
    apt-get install -y openjdk-17-jdk


RUN apt-get install maven -y

WORKDIR /usr

COPY . .

WORKDIR /usr/app
COPY . .


RUN mvn package



CMD ["java","-jar","target/catalog-1.0-SNAPSHOT.jar"]
```



```
FROM ubuntu:latest

RUN apt-get update

RUN apt-get update && \
    apt-get install -y openjdk-17-jdk


RUN apt-get install maven -y

WORKDIR /usr

COPY . .

WORKDIR /usr/app
COPY . .


RUN mvn package



CMD ["java","-jar","target/order-1.0-SNAPSHOT.jar"]
```



```
FROM ubuntu:latest

RUN apt-get update

RUN apt-get update && \
    apt-get install -y openjdk-17-jdk


RUN apt-get install maven -y

WORKDIR /usr

COPY . .

WORKDIR /usr/app
COPY . .


RUN mvn package



CMD ["java","-jar","target/gateway-1.0-SNAPSHOT.jar"]
```



```
version: '3.8'
services:
  catalog:
    build:
      context: catalog
    ports:
      - "4575:4575"
    networks:
      - microservicesNetwork

  order:
    build:
      context: order
    ports:
      - "3300:3300"
    depends_on:
      - catalog
    networks:
      - microservicesNetwork

  gateway:
    build:
      context: gateway
    ports:
      - "4567:4567"
    depends_on:
      - catalog
      - order
    networks:
      - microservicesNetwork

networks:
  microservicesNetwork:
    driver: bridge
```