

Lab 1: Bazar.com: A Multi-tier Online Book Store

Name	ID
Ra'ed Khwayreh	11822076
Yacoob Assi	11718673

In this lab we used SPARK – JAVA



First thing we created database for our books using **sqlite** and we stored books.

```
sqlite> select * from categories;
1|How to get a good grade in DOS in 40 minutes a day|25|45|distributed systems
2|RPCs for Noobs|31|25|distributed systems
3|Xen and the Art of Surviving Undergraduate School|13|30|undergraduate school
4|Cooking for the Impatient Undergrad|39|50|undergraduate school
```

Then we built category server, this server receives most requests:

- 1- Search from front.
- 2- Info from front.
- 3- Search from order server.

Then we built frontend server and order server.

How our project work?!

When the client sends the request for frontend server, frontend server check the request to forward it for appropriate server. Like when it checked it and see /info it'll forward it for category server.

Category server received the request then checked it if info or search and if POST/GET.

If the request GET then it will check the URL to take the arguments and if it's POST, it'll check the body of the request.

The category server connected with database to get data from it. When it received data, it converts each element to Object of book then convert it to JSON to return it for client.

And in this picture explain it for info after check request type:

```
public static String infoUrlArgument(Request request, String jdURL) {
    Book book;
    try {
        Connection connection = DriverManager.getConnection(jdURL);
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement
            .executeQuery("Select * from categories where id = " + request.params(":id") + "");
        book = new Book(resultSet.getString("title"), resultSet.getString("quantity"),
            resultSet.getString("price"));
        return resultSet.wasNull() ? new Gson().toJson(new OrderStatus("Failed", "Book not exist")) : new Gson().toJson(new Status(new OrderStatus("Success", "Book exist"), book));
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return "Problem";
}
```

connect to database and get the result then convert the result to object of Book;

Return the result for client as json

This was for category server.

When the frontend received request for order it'll forward the request for order server. When the order server received the request it'll send request for category server to check if the book exist, and if it's existed it will update book's quantity (-1).

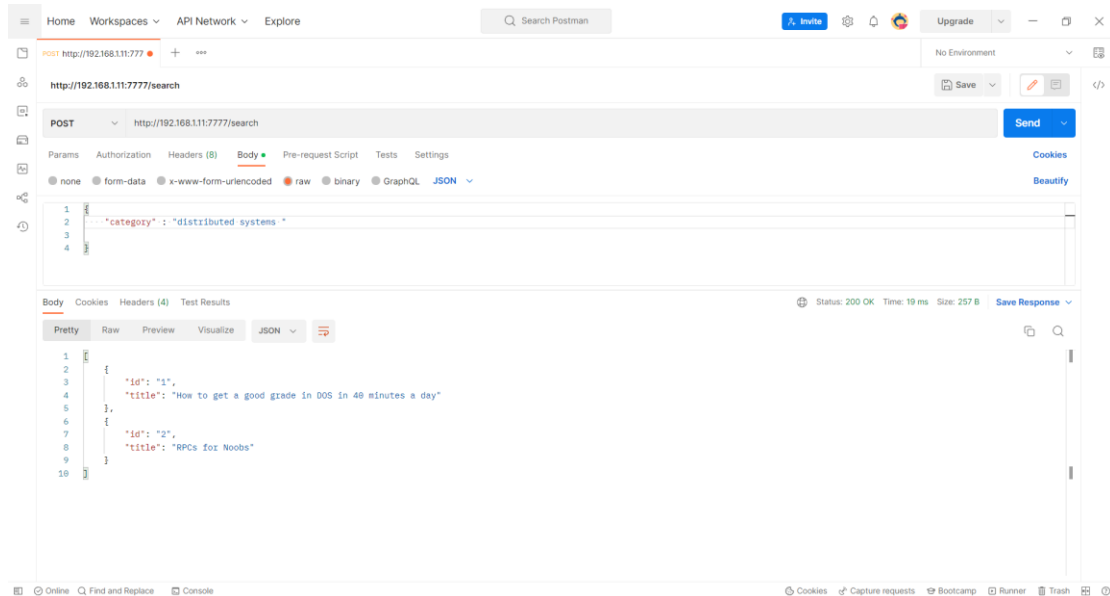
Some results:

1- Get info for book of id = 3, passing arguments in URL:

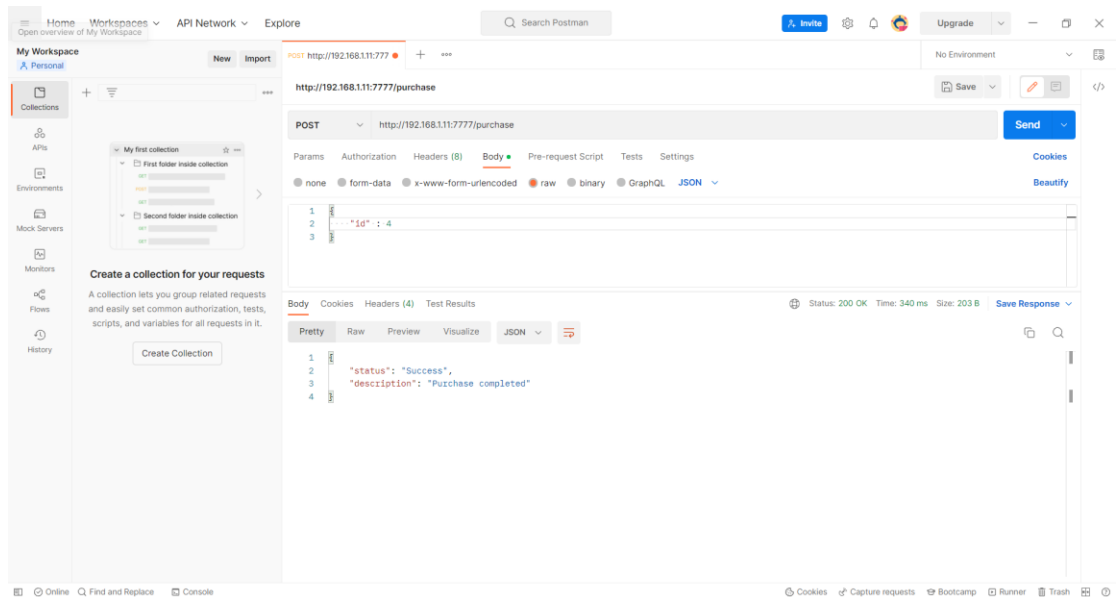
The screenshot shows a REST client interface with a GET request to `http://192.168.1.11:7777/info/3`. The response is a JSON object with the following structure:

```
{
  "orderStatus": {
    "status": "Success",
    "description": "Book exist"
  },
  "book": {
    "title": "Xen and the Art of Surviving Undergraduate School",
    "quantity": "13",
    "price": "30"
  }
}
```

2-Search for distributed systems category passing arguments in the body:



3-Purchase book passing arguments in the body:



4-Purchase not existing book passing arguments in the body:

