LAB 2 – PART 1

Ra'ed Khwayreh + Yaccob Assi

1-Cache

In frontend server we create array

```
static Book cache[]=new Book[5];
```

To store requests

For example when the request come to the front and it's for info, frontend server check if this request stored in the cache

```
get("/info/:id", (req, res) -> {


      res.type("application/json");

      Book bookCheck = getBookByID(req.params(":id"));
      if(bookCheck!=null) {
          System.out.println("From Cache");
          if(bookCheck.quantity.equals("0")&&bookCheck.title.equals("")) {
              return  new Gson().toJson("Book not exist");
              }else  return  new Gson().toJson(bookCheck);

      }
```

```
public static Book getBookByID(String id) {

    for(int i =0 ; i<cache.length ; i++) {
        if(cache[i].id.equals(id)) {
            return new Book(cache[i].title,cache[i].quantity,cache[i].price);
        }
        }
    return null;
}
```

If not exist in the cache it will forward the request for categories server.

```
else {

        String port="";
        String ip="";
        if(to_cat) {port="5555";
        ip="http://192.168.1.21";
        }
        else {port="6666";
        ip="http://192.168.1.11";
        }
        to_cat=!to_cat;

System.out.println("From Catalouge");

URL url = new URL(ip+":"+port+"/info/"+req.params(":id"));

System.out.println(url);

HttpURLConnection con = (HttpURLConnection) url.openConnection();
con.setRequestMethod("GET");


BufferedReader in = new BufferedReader(new InputStreamReader(con.getInputStream()));
        String inputLine;
        String content ="";
        while ((inputLine = in.readLine()) != null) {
            content+=inputLine;
        }
        in.close();

        System.out.println(content);
            String s[]=content.split(",");
            if(!content.equals("Book not exist")) {
```

```
                Book book_cash = new Book(s[0],s[1],s[2],s[3],s[4]);
                Book book_show = new Book(s[1],s[2],s[3]);

                 putBook(book_cash);
                 return  new Gson().toJson(book_show);
             }
             else {
                Book book = new Book(req.params(":id"),"","0","","");
                putBook(book);
                return  new Gson().toJson("Book not exist");


             }

 }});
```

And here we go to the second task which is replication.

2- Replication

Frontend server not always send to the same category server it switch between server after each request.

And after the response came to the frontend it will store it in the cache.

It same for search requests.

3- Consistency

In order requests, frontend do the same between replication order servers

```
post("purchase/:id", (request, response) -> {
    String port;
    String ip;
    if(to_order) {port="9999";ip="http://192.168.1.21:";}
    else {
        port="8888";
        ip="http://192.168.1.11:";
    }
    to_order=!to_order;

    response.redirect(ip+port+"/purchase/"+request.params(":id"));
    return null;
});
```

Frontend sent the request for one of order servers.

After that the order server forward the request for one of categories server, after that order server which received the request and do it, it send.

Send request to the other categories server to update the values

Each category server when receive update request it will implement this function

```
public static Object Update(Request request, String jdURL) throws SQLException {


    try {
        Connection connection = DriverManager.getConnection(jdURL);
        Statement statement = connection.createStatement();
        ResultSet resultSet = statement.executeQuery(
                "Select * from categories where id = '" + request.params(":id") + "' and quantity >= 1");
        int i = 0;
        while (resultSet.next()) {
            i++;
        }
        if (i >= 1) {
            statement.execute(
                    "UPDATE `categories` SET `quantity`=`quantity`-1 WHERE id = " + request.params(":id") + "")



            connection.close();

        }} catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    return null;



}
```

And then the frontend server apply this update on cache.