

**An-Najah National University**

**Faculty of Engineering and IT**

**Computer Engineering department**

**Distributed OS**

**Dr. Samer Arandi**

**Second semester 2023-2024**

**Course Project (Part 1)**

**Micro Webservices + REST**

**شهاب الدين خراز - 12027747**

**Introduction:**

As we got more familiar with REST from the first homework, this time in this homework, we will go to another level and try to be more familiar with **services**, specifically **micro webservices**, we will practice them by building the smallest bookshop, Bazar.com.

**System Design:**

The system is -mainly- implemented as a two-tier web design, Front-end, and Back-end, where **microservices** are used at each. The Back-end tier consists of two microservices: Catalog and Order. The Front-end tier will run like a web Proxy, it will take the client requests and forward them to the catalog or order based on the desired operation.

We have a set of operations that can be done, let us take a quick look at them:

1. Search(topic)
2. Search(itemName)
3. Info(itemNumber)
4. Purchase(itemNumber)
5. Update(cost)
6. Update/stock(increase)
7. Update/stock(decrease)

All these operations calls are exposed as **REST** calls on top of HTTP, also there is an endpoint for each of them.

To gain more knowledge, and put more **Distribution**, we will use **Docker** containers, where each microservice will be encapsulated in a Docker container. Then, we will run the three containers and send requests from the host device.

**How it works:**

Let us take a deeper look at how the system works, in other words, what happens behind the scenes when the client makes a request.

To understand this, let us suppose the client makes the following request:  
 GET https://serverIP/search/ distributed%20systems

Here is what happens:

1. The request will be sent to the front-end service.
2. The front-end service will forward the request to the catalog or order service using an HTTP REST call, based on the desired operation. (catalog in the case of this request)
3. The catalog service will do the actual functionality (get all books under the distributed systems topic)
4. The books will be returned as JSON objects to the front-end service, then the front-end service will return the response to the client.

**Design tradeoffs:**

One tradeoff is about search, we can search by topic or by item, to do this, we can make a hierarchy in the URL or simply make a general endpoint and specify the rest in the request body. To do the second approach, we need to handle more functionality on the server side, so I make the first approach, this is an example of a tradeoff the ease over flexibility.

**Possible improvements:**

Split the catalog service into two services (Vertical distribution) or duplicate the whole service on two servers (Horizontal distribution) and introduce a load balancer. This will improve the overall performance and supports more scalability.

**Note:**

Although we cannot be 100% sure without introducing Tests, there are no cases where the program is known not to work correctly (at least for all possible requests mentioned previously and in the project description).

**How to run the program:**