Design Documentation for Lab 2

Architectural Overview

To give an overview of the architecture we had to emulate a bookstore running on a single server with a single buyer. The bookstore components (front-end, catalogue and order) are deployed as three processes and a fourth process representing the client performing interface calls to the front-end process.

The clients will be interacting with the APIs implemented by the front-end microservice. The API endpoints available are:

GET $FRONT_END_PUBLIC_IPv4_DNS:5004/search/<topic_name>
GET $FRONT_END_PUBLIC_IPv4_DNS:5004/lookup/<book_id>
POST $FRONT_END_PUBLIC_IPv4_DNS:5004/buy/<book_id>

Also, the update API implemented by the catalog service will be available for the clients. The endpoint is as follows:

curl --header "Content-Type: application/json" --request PUT --data '{"id": 1, "stock":2000, "cost":2000}' http://$CATALOG_PUBLIC_IPv4_DNS:5002/catalog/update

Technical Overview

The following is a brief description of each microservice implemented.

1. **Front End Server:** The users will interact with this server. It is responsible for the processing and abstracting back-end servers from the users. The implementation of this server is under the folder `./FrontendService`. It consists of only one microservice that has the functionality of searching for a book by topic, looking up for a book by it's id and buying the book by it's id.

2. **Catalog Server:** This is one of the microservices that forms the back-end server. It is responsible for maintaining the catalog of the books availables and the cost and quantity of the books. Also, it implements the functionality of querying the catalog by `topic or id` and also updating the cost and quantity of books available. The implementation of this microservice is under the folder `./CatalogService`.

3. **Order Server:** This is the second microservice in the back-end server. It is responsible for updating the quantity of the books available everytime it receives a request if the book is available in the catalog. It also maintains a log of all the purchase orders it receives. The implementation of this microservice is under the folder `./OrderService`.

Logging on Catalog and Order Server

1. Catalog Service

Logging happens in `**logfile.json**` inside the Docker container catalog-service. A buy request will be logged in the "buy" key of the json object and a query request will be logged in the "query" key of the json object. To check the logs run the following command while the container is running

```
$ docker exec -it catalog-service bash
```

Inside the container, check the content of logfile.json:

```
$ cat logfile.json
```

2. Order Service

To view the logs for the order-service call the following API endpoint while the container is running

```
GET $ORDER_PUBLIC_IPv4_DNS:5007/log
```

Evaluation and Measurement

1. Time to complete 1000 sequential requests by one user

INFO:root:Total time: 90191.7736530304 ms
INFO:root:Average time: 90.1917736530304 ms

2. Time to complete 1000 sequential requests by 5 user at a time

INFO:root:Total time: 136533.04195404053 ms
INFO:root:Average time: 136.53304195404053 ms