Assignment 3 – REST and HATEOAS

Important points to include in the planning are therefore:

- *How should the server(s) be able to scale if you encounter challenges with availability at high load?*
- *If you store the state of the client on the server side, how does it affect scalability?*
- *How can REST/HATEOAS help? Why?*
- *How can caching/proxy help. Is it relevant here?*
- *How easy is it to add more servers to handle higher loads over time?*

Documentation requirements:

- *Deployment diagram with examples of different types of requests. Use the UML deployment diagram if you don't have a particularly well-founded alternative.*
- *Comments on deployment diagram/architecture selection (maximum one page)..*
- *Comments in code.*
- *Self-documenting structure of web service.*

Planning points

1. The server should be scalable due to the scalability and resource sharing features of REST.
2. Benefits of HATEOAS include that the server does not need to know anything regarding state of clients and can respond to requests in client applications and resources. Therefore, high scalability.
3. Caching/proxy can reduce the scalability since it will require more space and data to store, especially for big data.
4. Adding more servers can be challenging depending on how the data is stored and affects the availability aspect of databases.

Deployment Diagram

5. The selected architecture is based on the following from assignment 1:
    a. Patient Look Up/Register
        i. Unique Identifier
        ii. Name
        iii. Symptoms
        iv. Identifiable by department, section and post
    b. Authorised Users
        i. Authorised Users have to identify themselves for security.

    For (5a) the patient or patients is to be registered with the provided information. The registered patients have to be able to searchable by id. The server has to be able to provide a list of all patients currently in the system. Lastly, the option to delete patients by id from the system is there when the patient leaves the hospital. The structure of Patient can be seen as:

```
{
        “id”:
        “name”:
        “symptom”:
        “department”:
        “section”:
        “post”:
}
```

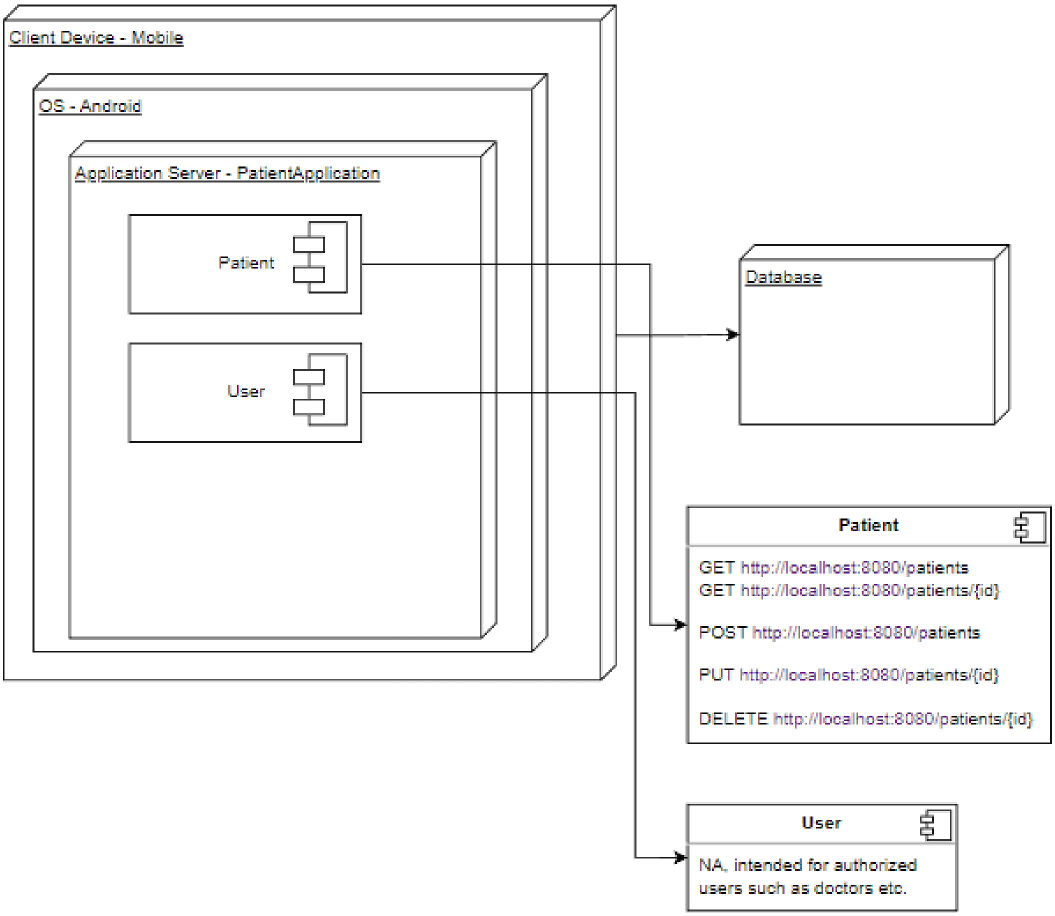
Client Device - Mobile
OS - Android
Application Server - PatientApplication
Patient
User
Database
Patient
GET http://localhost:8080/patients
GET http://localhost:8080/patients/{id}
POST http://localhost:8080/patients
PUT http://localhost:8080/patients/{id}
DELETE http://localhost:8080/patients/{id}
User
NA, intended for authorized users such as doctors etc.

Figure 1 Deployment Diagram