DISTRIBUTED SYSTEMS – UTCN

ASSIGNMENT 1 – DOCUMENTATION

Campan Tudor-Cosmin

30433

**Conceptual Architecture of the Online Platform**

My requirement was to create an online platform to manage users, their associated smart energy metering devices, and the monitored data from each device.

There were 2 types of users:

->ADMIN: he can:

-CRUD on normal users;

-CRUD on devices;

-LINK user to device;

->USER: he can:

-VIEW his/hers devices;

-SEE graphics with consumption in time(not yet implemented by me);

For this assignment I used a synchronous layered architecture. Basically, the backend code which receives request, communicates DB (and possibly other services) and sends response back to client.

As technologies, I used Java for back-end, with the Spring Framework, and VueJS for front-end.

As for DB, I implemented the schema in MySQL. I used JPA for repositories.

As I only had 2 big entities : User and Device, I used one package for each. Each entity has a model, repository, dto, mapper,service and controller.

I did the mapping on user-device by introducing an userId in the device table.

I managed the requests through REST API, introducing the correct dependencies.

To be mentioned, I used **gradle**.

My package organisation looks like this:

Graphical user interface, text

Description automatically generated

I used auxiliary libraries to make my work easier, such as **Lombok** or Mapstruct.

For Security I used **Spring Security,** generating a token for every user, having roles for each user etc. I documented my spring security part from the web.

I used docker to deploy the app, having 3 images:

-Fron-end image

-Database image

-Backend image

Then, I used docker-compose.yml to compose them.

Graphical user interface, text, application

Description automatically generated

**DB Design**

I had 4 tables on my database:

-User

-Role

-Device

-User\_role

The user\_role is the table which links users to roles as many users can have many roles.

@JoinTable(name = "user\_roles",  
 joinColumns = @JoinColumn(name = "user\_id"),  
 inverseJoinColumns = @JoinColumn(name = "role\_id"))

There is also a link between users and devices.

The table diagram looks like this:

A screenshot of a computer

Description automatically generated with medium confidence

**Deployment Diagram**

Graphical user interface

Description automatically generated with medium confidence

**Build&Execution Consideration**

DB: - configure mySql and the credentials&port for mySql found in the application.

Back-end: -open repo as IntellijIDEA Project, it should work as is.

Back-end : run yarn-install in terminal to configure, then yarn serve.