**FACULTY OF COMPUTER AUTOMATION AND COMPUTER SCIENCE**

**DISTRIBUTED SYSTEMS**

**Assignment 1**

Online Energy Utility Platform

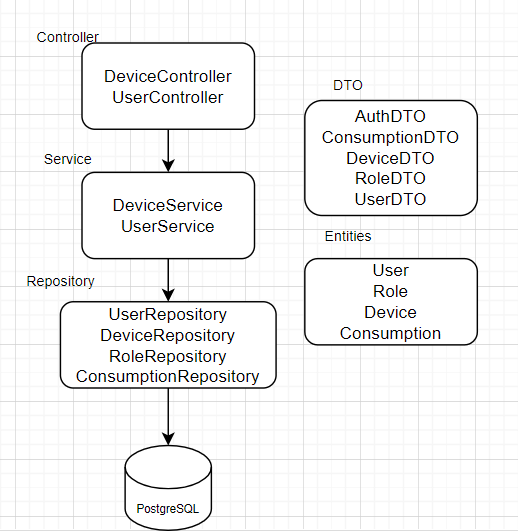
Dunca Denisa-Mihaela, gr: 30643

**Conceptual architecture of the online platform**

The project “Online Energy Platform” is a full stack application that represents a useful way for users to access information about their devices and about the energy that every device consumes by accessing the energy consumption charts and it provides an easy way of adding new users and devices for the administrators. The project is made in principle using Java and SpringBoot on backend, React Js on frontend and PostreSQL for the data base. The project is at the state of local deployment by using Docker.

The application has a login page that provides access to users and administrator by inserting the credentials, it also provides a registration for the users that don’t have yet an account. This application has two kinds of users, the clients that can easily see their devices and their energy consumptions by device, on a specific chosen date, and the administrators which can see, modify, edit, delete, and add users and devices and they can also map devices to users.

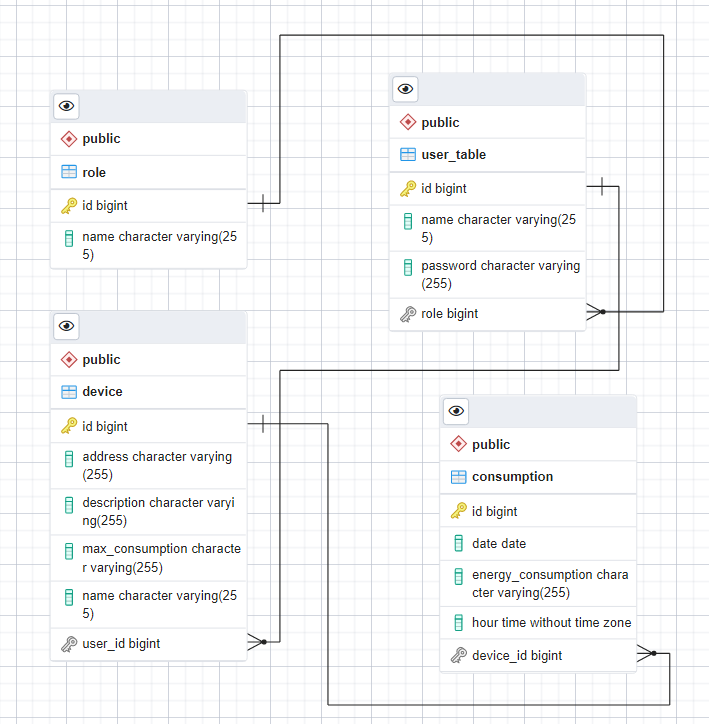
In order to create the project, I used the Layered Architecture which is a way the project is organized and is made out of four big categories of layers: presentation layer, application layer, domain layer and infrastructure layer. The presentation layer contains all the classes responsible for presenting the user interface and sends the response back to the client (Controller layer). The application layer contains all the logic in the application, all the functionalities (Service layer). The domain layer represents the entities and the business rules (Repository layer), and the infrastructure layer contains the classes responsible for the technical stuff, in this project we have DAO classes, and entities form Model.



**Data Base Design**

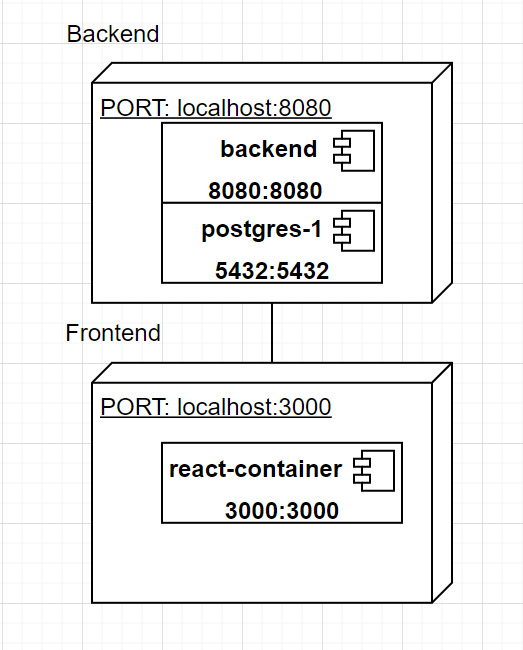
The database contains four tables, two of them are related to the users and the other two are related to the devices. The Role table contains the name of the role (user, admin) and the id for each role so that it can be mapped to the users. The User table contains the minimum information about the users (name and password) and the id of the role that each user has. Between Role table and User table there is a “one to many” relationship.

The Devices table contains all the information related to the devices such as the name, description, address, the maximum energy consumption and the most important the user ID, which is by default null so that the admin can map the device to the user. And the last table, the Consumption table contains all the energy consumption mapped by a timestamp (date and hour) and the device ID. Between Devices table and Consumption table there is a “one to many” relationship and also between User and Device table.

****

**UML Deployment diagram**

For the deploy I used Docker, and I created two containers, one for the backend and data base and one for the frontend. The backend was implemented using Java and Spring Boot and the database used was PostgreSQL and for the frontend I used React JS.

****

**Build and execution considerations**

I used firstly the Intellij IDE to implement the code on the backend and PGAdmin4 to create the data base, I also used Postmen to test the API calls and for the frontend I used WebStorm. You can start the application by running these two projects and by creating a specific database. After creating all the functionalities needed to this project, I did the deployment locally by using Docker desktop.

For both, frontend and backend I created a Dockerfile and a docker-compose.yaml. Firstly, I built the docker image by running docker-compose build command, and finally I run the docker-compose up command to create the container and run the application. After that you can also run the application from Docker Desktop.