DISTRIBUTED SYSTEMS

Assignment 1

**Request-Reply Communication Paradigm**

Online Energy Utility Platform

Student: Ștefan-Andrei CHICHIȘAN

Group: 30441

Course teacher: Cristina POP

Laboratory teacher: Oana MARIN

Table of contents:

1. Proposed task
2. Conceptual architecture of the online platform
3. DB design
4. UML deployment diagram
5. Build and execution considerations
6. **Proposed task**

The first assignment of the distributed systems course is to design an online platform should be designed and implemented to manage users, their associated smart energy metering devices, and the monitored data from each device. The system can be accessed by two types of users after a login process: administrator (manager), and clients. The administrator can perform CRUD (Create-Read-Update-Delete) operations on user accounts (defined by ID, name, role: admin/client), registered smart energy metering devices (defined by ID, description, address, maximum hourly energy consumption), and on the mapping of users to devices (each user can own one or more smart devices in different locations). After the mapping is done, for each device the energy consumption is stored on hourly basis as tuples of the form <timestamp, energy consumption> in the database.

1. **Conceptual architecture of the online platform**

Backend

The backend is written in Java programming language using Spring Boot framework. The architecture used in the development is a 3-tier REST service (Controller – Service -Repository). The architecture is well suited for web applications in general having the components well spread and organized.

All the packages implement entities, DTOs, mappers, services, controllers and repositories for only one main component. For example, the package *device* has a repository interface that implements the repository for users, a controller class that has all the request methods, a service class that connects the requests to the repository. It also has DTOs and mapper (transforms the DTO to model/entity and vice-versa). There is a thumb rule that models/entities shouldn’t be sent or received through the controller, that’s what DTOs (Data Transfer Object) are for, to send the data between the frontend and backend using JSON type information.

Diagram

Description automatically generated

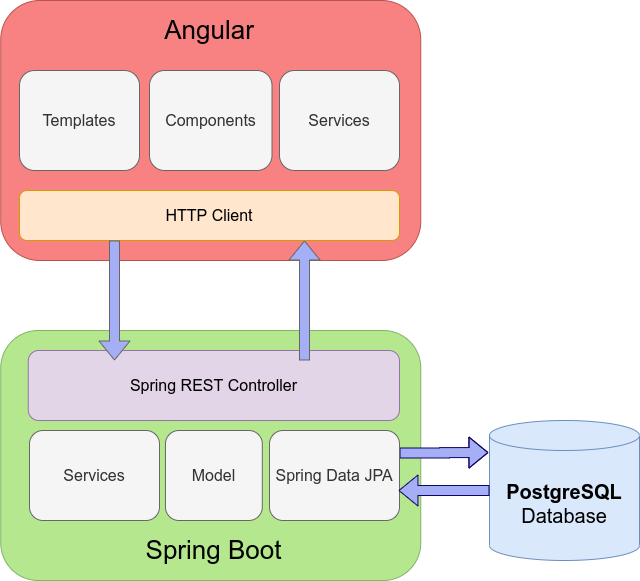
Frontend

The front part of the application was designed and implemented using Angular framework, TypeScript, HTML and CSS (SCSS). The application can be only accessed by the administrator, the clients being redirect to a page error.

There are a few main pages and dialog boxes displayed. For the administrator privileges: users page, devices page and measurement page and for each of them dialog boxes for creating/editing/deleting. For the client page, as I said above, he will be redirected to an error page.

Diagram

Description automatically generated



*Application architecture schema*

1. **DB design**

The used database for this project is PostgreSQL are there are 5 tables in total create and used by the application.

Tables and their contents:

* *users* : holds information about user entities
* *role* : a table that contains all the roles the application can have
* *device* : holds information about the device
* *measurement* : holds information about the measurement values of devices
* *device\_measurement* : a joined table that is created by Hibernate as a relationship between the device and measurement.

A piece of paper with writing on it

Description automatically generated with low confidence

1. **UML deployment diagram**

Diagram

Description automatically generated

1. **Build and execution considerations**

Localhost:

* Create database in PostgreSQL using Hibernate
* Set database credentials in the *application.properties*
* Create the host variable in Angular project with the API of the backend
* Run the backend on the port configured for the apache-tomcat server
* Start frontend by using *ng serve* which uses webpack dev server
* Check the endpoints with the Postman API
  + Here is my own collection : <https://www.getpostman.com/collections/a14a5fcf4238c4fccd57>