ONLINE ENERGY UTILITY PLATFORM

Nadu Laura Andreea

Group 30442

Table of Contents

Assignment Objective

Main Objective

Create an online energy platform to keep track of the energy consumption of the customers' devices.

Sub-Objectives

O1. Analyse the problem and identify the requirements

O2. Design the online platform

O3. Implement the backend process

O4. Implement the User Interface

O5. Test the platform

Problem analysis, modelling, scenarios, use cases

Analysing the problem

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 **customers**. 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 in the database.

Functional requirements:

- Users log in. Users are redirected to the page corresponding to their role.
- Administrator/Manager Role:
  - CRUD operations on users and devices.
  - Create user-device mappings.
- User/Client Role
  - Can view on his/her page all the associated devices.
  - Can view the daily energy consumption for each of his/her associated devices as line charts or bar charts per day (OX- hours; OY- energy value [kWh] for that hour). The day should be selected from a calendar.
- The users corresponding to one role will not be able to enter the pages corresponding to the other role (e.g., by log-in and then copy-paste the admin URL to the browser).

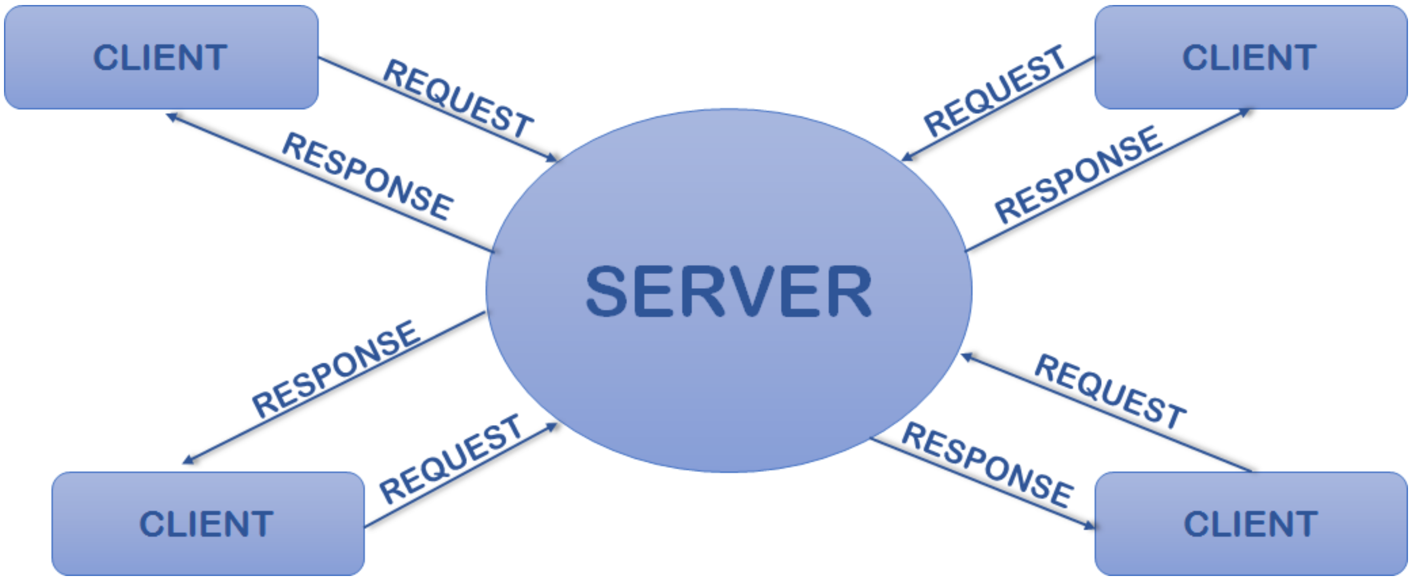
Non-functional requirements:

- Security: use authentication to restrict users to access the administrator pages (cookies, session, etc.)

Conceptual Architecture

The platform will use the Client – Server architecture, which refers to a system(**server**) that hosts, delivers and manages most of the resources and services which the **client** requests. This architecture supposes that all requests are delivered over a network and it is also referred to as the networking computing model or client network.

In our case, the requests will be made from the frontend of the Angular Client application which will then be interpreted by the Spring Boot backend application which will then return a response to the client.



The Client has the following folder structure:

- components – contains the components used for rendering the UI (html, scss and ts)
- constants – contains files with useful constants throughout the application, as well as definitions for the backend endpoint calls
- guards – contains guard services which are used to restrict the user from accessing forbidden pages
- models – contains entity/model definitions of the useful objects mapped from the backend
- services – contains services related to each model. They are injectable in each component that might want to operate on a said model, or make an API call regarding it.

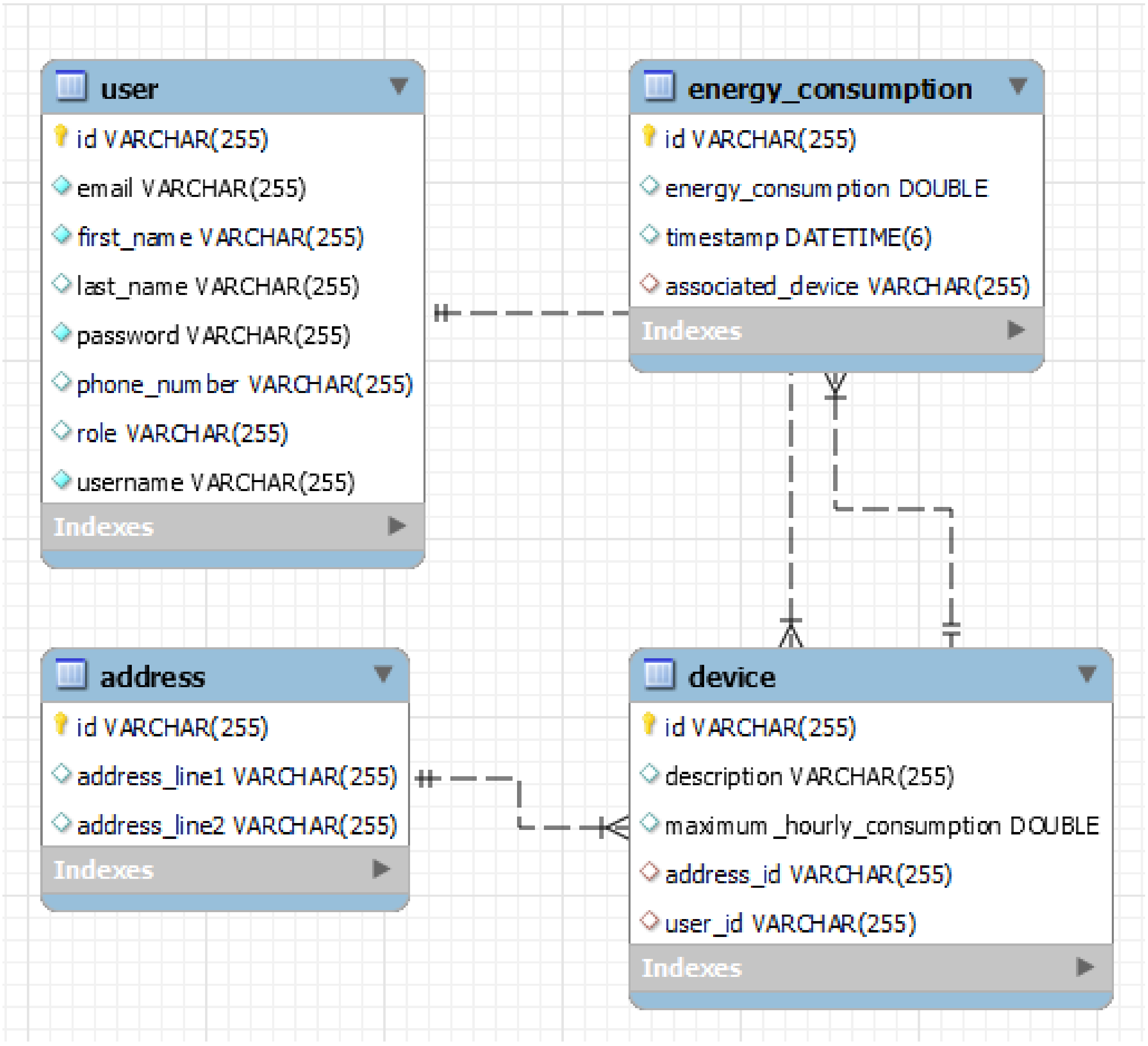
The Server has the following folder structure:

- security – contains server's configurations related to Security, CORS, JWT and others
- controllers – defines and exposes the APIs to the client
- dtos – contains mappers, validators and classes which will be mapped from Database entities to be secure objects which will be exposed to the Client
- entites – contains the database entity structure
- repositories – contains the entry points of the database entities
- services – contains all the logic which is used by the exposed APIs

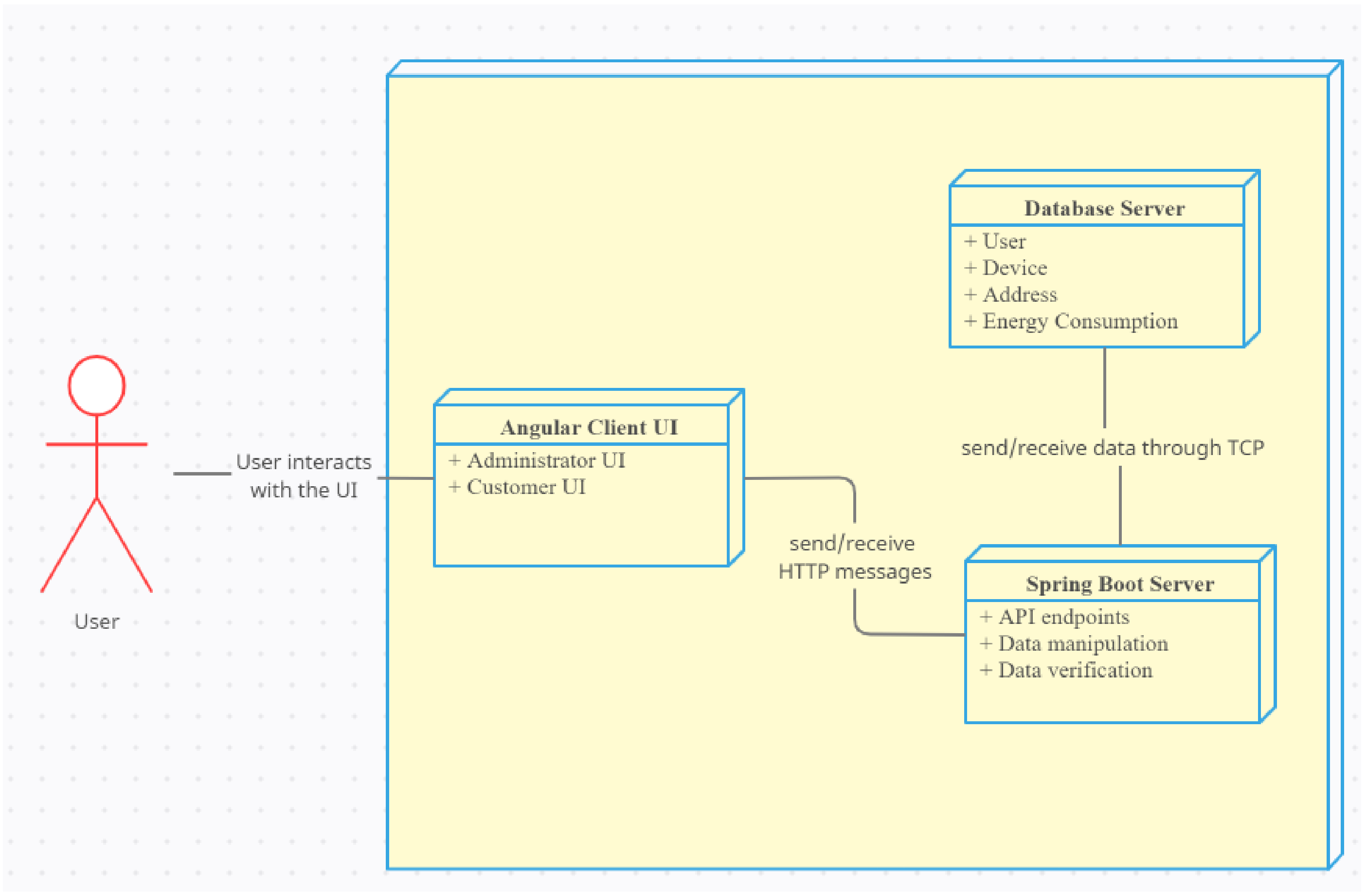
Database Design Architecture

The platform holds all the needed information in a MySQL Database, which has the following 4 tables:

- user – holds the information related to user credentials
- device – holds the information related to device technical specifications, a relation to the owner(user), a relation to the address associated to the device and a relation to the consumptions of the device
- address – holds the information related to the address associated to a device
- energy_consumption – holds the information related to hourly consumptions of the device



Deployment Diagram using Docker


User
User interacts with the UI
Angular Client UI
+ Administrator UI
+ Customer UI
send/receive HTTP messages
Database Server
+ User
+ Device
+ Address
+ Energy Consumption
send/receive data through TCP
Spring Boot Server
+ API endpoints
+ Data manipulation
+ Data verification