ENERGY UTILITY PLATFORM

Student: Florin Petrean

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 3

3. System Architectural Design 3

4. UML Sequence Diagrams 3

5. Class Design 3

6. Data Model 3

7. System Testing 3

8. Bibliography 3

1. Requirements Analysis

# Assignment Specification

# The first module of the system consists of an online platform designed to manage clients, smart devices equipped with energy consumption sensors and monitored data from sensors. The system can be accessed by two types of users after a login process: administrator, and clients. The administrator can perform CRUD operations on client accounts (defined by ID, name, birth date, address) registered smart devices that have sensors attach to monitor energy consumption (defined by ID, description, address (location), maximum energy consumption, average/baseline energy consumption), smart sensors attached to devices (ID, sensor description, maximum value) and on the mapping of clients to devices (each client can own one or more devices that are monitored) and sensors to devices (each device has one sensor monitoring its energy consumption). Usually, the device is either a house equipped with a smart power meter or part of a house, case in which a client has several devices. Each device has one sensor monitoring its energy consumption. The sensor records periodically, at fixed timestamps, tuples of the form <timestamp, energy consumption>, where energy consumption is a counter measuring in kWh the total energy consumed by the device since it was started. Each client can view its devices, and their current and historical energy consumption, as well as the total energy consumption of their devices.

# Functional Requirements

Users log in.

Users are redirected to the page corresponding to their role.

Administrator/Manager Role:

* + CRUD operations on clients
  + CRUD operations on devices
  + CRUD operations on sensors
  + Create mapping client-device and associate sensors to devices.

Client Role

* + A client can view on his/her page all the devices and sensors.
  + A client can view their monitored energy consumption (current data and historical data)
  + View historical energy consumption for each client as line charts or bar charts for one day (OX- hours; OY- energy value [kWh] for that hour). Select day from a calendar.
  + The users corresponding to one role will not be able to enter the pages corresponding to other roles (by log-in and then copy-paste the admin URL to the browser)

# Non-functional Requirements

Security

I have implemented the login functionality of the application with JWT Token and SessionStorage.

I have implemented private admin routes and user routes in order for the user to only go at pages where they are supposed to.

I have implemented security filters on the backend part of the application so that unprivileged users cannot do requests they are not supposed to.

I have used DTOs for transferring data between backend and frontend using a custom DTO mapper.

I have also implemented generic Controller, Service and Repository in order to minimize code duplication.

2. System Architectural Design

**2.1 Architectural Pattern Description**

The application is using a layered architecture on the server and each layer of the layered architecture pattern has a specific role and responsibility within the application as suggested by the name of the layer.

Between the presentation and business layer we have controllers which puts the needed data from business layer onto a specific port in order for the presentation layer to access it.

Diagram

Description automatically generated with medium confidence

The presentation layer is separated “physically” by the other layers and deployed as a separate application in our case. The frontend part of the application was developed in React.

The other layers represent the backend part of the application which is also deployed and developed with Spring Boot.

Both parts of the application are deployed using Heroku Servers.

**2.2 Diagrams**

**Package diagram**

Diagram

Description automatically generated

**Deployment Diagram**

Diagram

Description automatically generated

3. Class Design

User

* Long id
* String username
* String firstName
* String lastName
* String address
* LocalDate dateOfBirth
* List<Device> devices

Device

* Long id
* String description
* String address
* Long maxEnergyConsumption
* Float avgEnergyConsumption
* Sensor sensor

Sensor

* Long id
* String description
* Long maxValue
* Device device
* List<Record> records

Record

* Long id
* LocalDateTime timestamp
* Long energyConsumption

4. Data Model

The data model of this application contains 4 tables: user, device, sensor, record.

*Diagram

Description automatically generated*

8. Bibliography

<https://www.springboottutorial.com/spring-boot-react-full-stack-crud-maven-application?fbclid=IwAR1_EEVsMkGDDM1qEVtTXEC2puVLmiWAatgQzOl1kDD6OiZeIUfLvvaydus>

<https://hellokoding.com/deleting-data-with-jpa-hibernate/>

A lot of stackoverflow

Course materials