DS Assignment 1

Online Energy Utility Platform

ANAMARIA RAITA

Group 30442

# Problem specification

An online platform should be designed and implemented to manage users, their associated smart energy metering devices, and the monitored data from each device.

# Requirements

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 in the database.

## Functional requirements

➢ Users log in. Users are redirected to the page corresponding to their role.

➢ Administrator/Manager Role:

o CRUD operations on users and devices.

o Create user-device mappings.

➢ User/Client Role o Can view on his/her page all the associated devices.

o 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

## Non-functional requirements

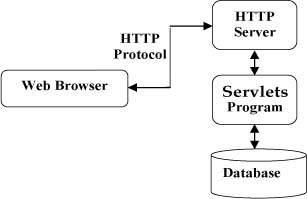
* Security: use authentication in order to restrict users to access the administrator pages (cookies, session, etc.)

# Implementation technologies

For implementing this project, I used the following technologies: REST services for backend application, in this case Java Spring REST and ReactJS which is a javascript-based technology for the frontend. The database was implemented in PostgreSQL.

The communication between the client and the server is done using Java Web Servlets. Java Servlets are programs that run on a Web or Application server and act as a middle layer between a request coming from a Web browser or other HTTP client and databases or applications on the HTTP server.

Using Servlets, you can collect input from users through web page forms, present records from a database or another source, and create web pages dynamically.



Hibernate ORM was used in order to handle the operations on the Postgres Database. ORM stands for Object-Relational Mapping (ORM) is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc. Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieve the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the work in persisting those objects based on the appropriate O/R mechanisms and patterns.

## System Architectural Design

### Backend Architectural pattern description

The Layered Architecture tries to divide the code and implementation into different layers, where each layer will have a fixed responsibility. The diagram of a common layered is sh architecture is shown below:

Diagram

Description automatically generated

Layered Architecture

**Presentation Layer:**

 The presentation layer handles the HTTP requests, translates the JSON parameter to object, and authenticates the request and transfer it to the business layer. In short, it consists of views i.e., frontend part.

**Business Layer:**

 The business layer handles all the business logic. It consists of service classes and uses services provided by data access layers. It also performs authorization and validation.

**Persistence Layer:**

The persistence layer contains all the storage logic and translates business objects from and to database rows.

**Database Layer:**

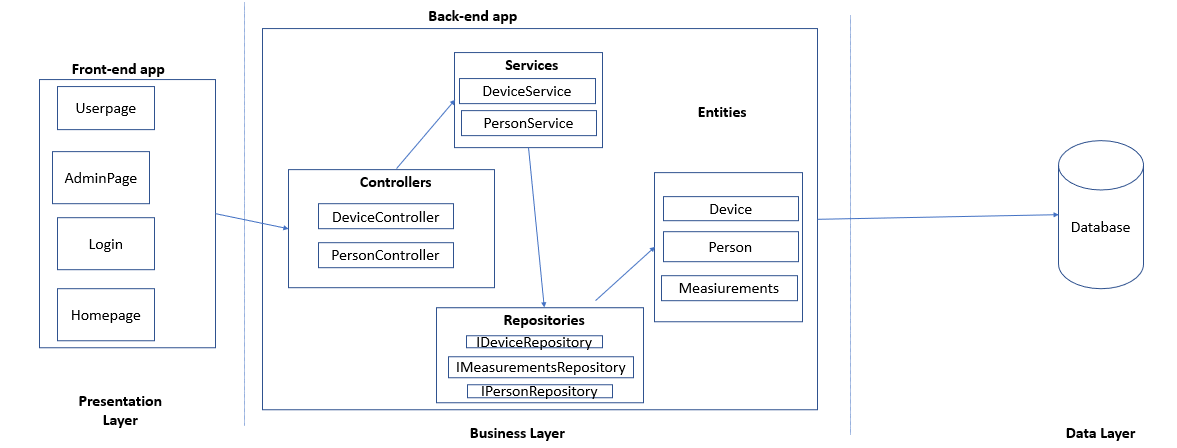
In the database layer, CRUD (create, retrieve, update, delete) operations are performed.

### Diagrams

Diagram

Description automatically generated

Deployment Diagram



Conceptual Architecture

# Data model

All the data from the application is kept in a PostgresSQL Database.

Person and Devices are in a 1:M relationship, as a person can have multiple devices, but a device belongs to only one person.

Devices and Measurements are also 1:M relationship, as a device has many measurements but a measurement corresponds only to a certain device.

Diagram

Description automatically generated