

**Facultatea de Automatica si Calculatoare**

**Sectia Calculatoare**

**Online Energy Utility Platform**

**Documentation**

Zilai Denis

Grupa 30244; Anul 4, sem 1

Distributed Systems

Contents

[1. Project objective 3](#_Toc118756099)

[2. The analysis of the problem, modeling, scenario, use cases 3](#_Toc118756100)

[3. Design (design decisions, UML diagrams, class design, packages, User Interface) 3](#_Toc118756101)

# Project objective

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 in the database.

# The analysis of the problem, modeling, scenario, use cases

This project was modelled rather in a simplistic way, using Java as backend and Angular as frontend. After the application is started locally on localhost, on the main and single page of the application, the only user represents the admin, who is able to perform CRUD (create, read, update, delete) operations on the table that represents the users, and the table that represents the devices. The administrator can view the tables, as well as deleting entire rows, those being devices or clients; modifying the characteristics of either client or device is also available for the administrator. Those actions can be performed by using the buttons with clear instructions from the User Interface.

The DataBase used for this project was SQLite, which helped a lot with the implementation, as it is embedded within the IntelliJ Ultimate IDE. There were used three tables:

* the Client table, which stores the ids, the usernames and the passwords for every user
* the Device table, which stores the ids, the descriptions, the addresses and the maximum hour consumptions for every device
* the ClientDevice table, which “creates” the link between users and devices, the relationships being “many-to-many” (a device can be owned by multiple users, and a user can own multiple devices), that stores the ids of the clients and the ids of the devices

This project also uses the Gradle, as it is fairly easy to understand it and work with it.

One important file within the project is represented by the Entities.hbm.xml, as with the help of this file, the mapping between the model classes and the DataBase tables can be done with ease.

# Design (design decisions, UML diagrams, class design, packages, User Interface)

The project is organized in a clean manner, the classes belonging to specific packages. The packages used were:

* Domain: this package contains the model classes: Admin, Client, Device, Entity and ClientDevice
* Persistence: this package contains the Dialect needed for the SQLite, the Generic interfaces needed to implement the HBM (hibernate mapping) classes, needed for CRUD purposes
* Services: this package contains the Controller class, used for CRUD purposes, with the help of the HBM classes

DataBase diagram:

Graphical user interface, text, application, chat or text message

Description automatically generated

UML Deployment Diagram:

Diagram

Description automatically generated

# README

To run this application, it is needed to start both projects (backend and frontend). Backend application is written in Java, while the fronted application is written in Angular. After the applications have been started, a browser can be opened to start using the functionalities of the project. It has to be opened via localhost, by default the selected port has been chosen 4200.