ENERGY UTILITY PLATFORM

**& SMART HOME APPLIANCE**

Analysis and Design Document

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

Suppose that the clients have intelligent home appliances that can be controlled remotely using remote procedure call (RPC). Each such device can communicate with the server that will compute the time when the device will be started for an optimal energy consumption.

Develop a client-side application (either a desktop application or a web application based on a JavaScript framework running from the browser) for the smart appliance associated to a client that:

1. i) gets the client hourly historical energy consumption over d days in the past (𝐸𝑐𝑙𝑖𝑒𝑛𝑡𝑑(ℎ));
2. ii) gets the averaged energy consumption for the client over the past week (e.g. client baseline);

𝐵𝑎𝑠𝑒𝑙𝑖𝑛𝑒(ℎ)=17Σ𝐸𝑐𝑙𝑖𝑒𝑛𝑡𝑑(ℎ)7𝑑=1,∀ℎ∈{1..24},

𝑤ℎ𝑒𝑟𝑒 𝐸𝑐𝑙𝑖𝑒𝑛𝑡𝑑(ℎ) 𝑖𝑠 𝑡ℎ𝑒 𝑐𝑙𝑖𝑒𝑛𝑡 𝑒𝑛𝑒𝑟𝑔𝑦 𝑐𝑜𝑛𝑠𝑢𝑚𝑝𝑡𝑖𝑜𝑛 𝑓𝑜𝑟 𝑑𝑎𝑦 𝑑 𝑖𝑛 𝑡ℎ𝑒 𝑝𝑎𝑠𝑡 𝑎𝑛𝑑 ℎ𝑜𝑢𝑟 ℎ 𝑓𝑟𝑜𝑚 𝑑𝑎𝑦 𝑑

1. iii) allows the selection of a program with a duration in hours (select a duration D of a program);
2. iv) gets the best time to be started considering the baseline and the program duration to avoid energy peaks from the client (e.g. to minimize the maximum energy consumption for every hour of the day)

𝐶𝑜𝑚𝑝𝑢𝑡𝑒 𝑡𝑠,𝑡𝑒 𝑠𝑢𝑐ℎ 𝑡ℎ𝑎𝑡 𝑀𝑖𝑛(𝑀𝑎𝑥(𝐵𝑎𝑠𝑒𝑙𝑖𝑛𝑒(ℎ)+𝐸𝐷𝑒𝑣𝑖𝑐𝑒)),∀ℎ∈[𝑡𝑠,𝑡𝑒],𝑡𝑒=𝑡𝑠+𝐷

# Functional Requirements

The client application displays a chart with the client historical energy consumption over d days in the past (default d = 7).

The client application displays the client baseline as a reference consumption for the next day

The client application allows the selection of a program with a duration (either from a list of programs or by entering directly the duration)

The client application asks the server for the best start time in the next day to minimize the peaks of energy consumption. It displays the new chart of estimated consumption as the baseline summed with the device max consumption.

# Non-functional Requirements

The Smart Home Appliance application runs on java swing and is using jfreechart for showing charts.

The Remote Procedure Call used is Hessian which will make the client application call methods from the server.

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.

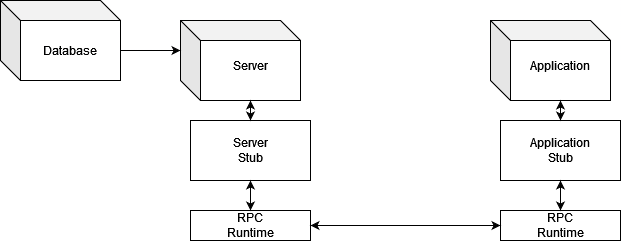
The other layers represent the backend part of the application which is also deployed and developed with Spring Boot. The database layer resides in Heroku.

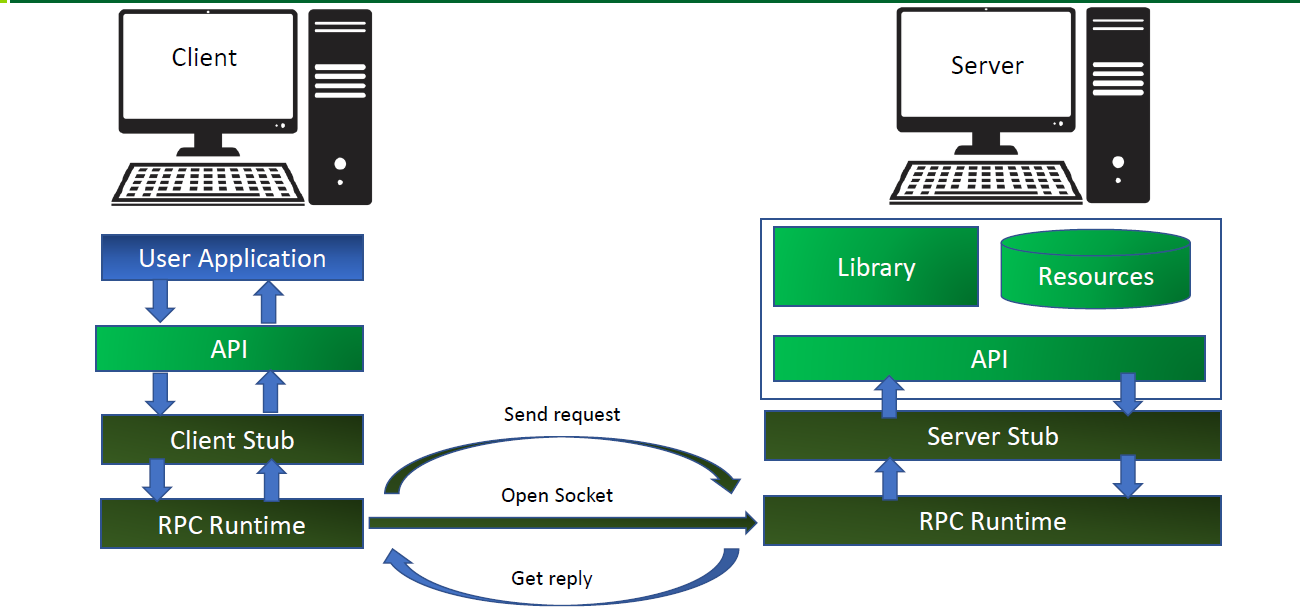
Both parts of the application are deployed using Heroku Servers as docker containers.

The Remote Procedure Call for the server is implemented using Hessian. The approach is similar to a REST API because we set an endpoint and there we have an interface with the methods that we can call on the client.  
 The Smart Home Appliance application is also using the same interface and model in order to using those methods. The architecture is NetBeans generated and is a simple GUI java swing application with login and user interface.

**2.2 Diagrams**

**Deployment Diagram**

****

**

3. Class Design

**For Backend**

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
* Double maxValue
* Device device
* List<Record> records

Record

* Long id
* Long timestamp
* Double energyConsumption

**For Smart Home Appliance application**

We use the same model as the backend for the classes that we need in our application.

AppUserRmi

- username

- firstName

- lastName

- address

- dateOfBirth

-role

- devices

DeviceRmi

- id

- description

- address

- maxEnergyConsumption

- avgEnergyConsumption

We have AppUserRmi and DeviceRmi classes which are used for creating the login option of the smart home appliance application and are sending information about the user and their devices. The login is done through RPC made available by Hessian.

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