MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY

OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE
COMPUTER SCIENCE DEPARTMENT

DISTRIBUTED SYSTEMS

Assignment 1

Request-Reply Communication Paradigm

Online Energy Utility Platform

Gafita Diana
Grupa 30243

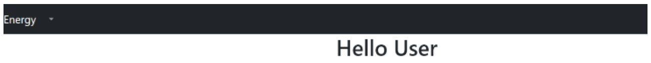
2022

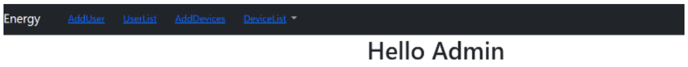
1. Requirements

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 <timestamp, energy consumption> in the database.

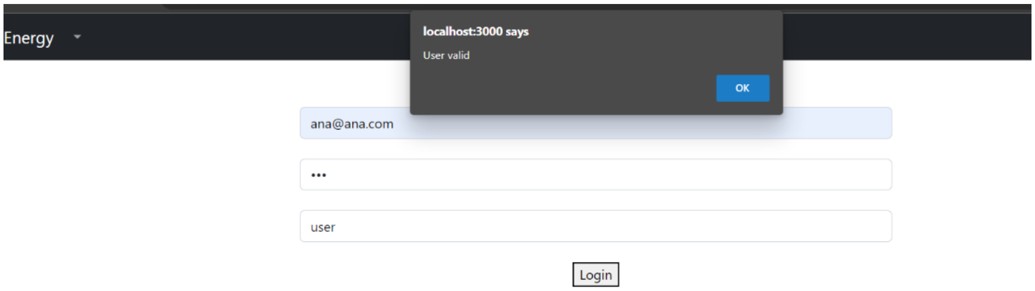
2. Tasks completed

For this project I used React for frontend and .Net Core API for the backend part. From all the requirements I managed to create the frontend for the login page with user redirect based on users role. If you are logging in as user you will be redirected to an user page and if you use an administrator account you will be redirected to an admin page which contains CRUD operations for users and devices. We have a database in MySql Workbench linked to the backend part which is the used in the frontend by fetching the api that we nead.





If the user is valid we will have a popup saying “user valid” otherwise it will said “invalid user” and you will not be re-directed to another page.



localhost:3000 says

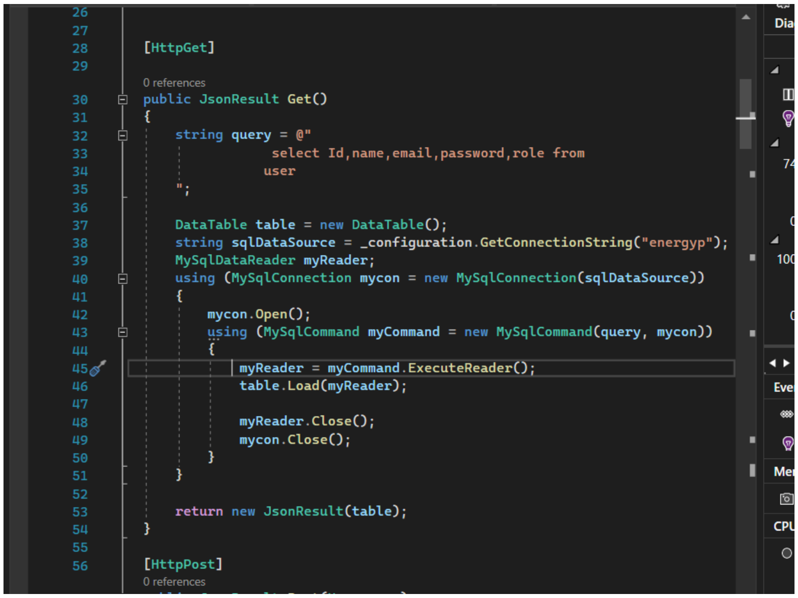
User valid

ana@ana.com

•••

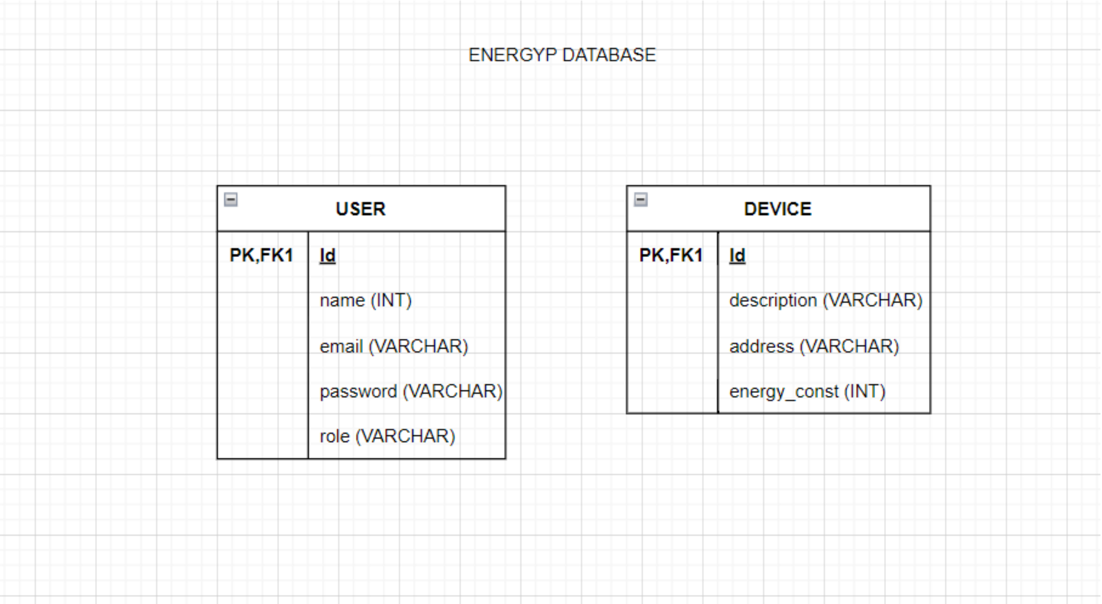
user

In the backend part we have the Models folder where there are 2 models for the user and for the device, I used them in the Controller files for the API part: get, post, put, delete.



```
[HttpGet]

0 references
public JsonResult Get()
{
    string query = @"
                select Id,name,email,password,role from
               user
    ";

    DataTable table = new DataTable();
    string sqlDataSource = _configuration.GetConnectionString("energyp");
    MySqlDataReader myReader;
    using (MySqlConnection mycon = new MySqlConnection(sqlDataSource))
    {
        mycon.Open();
        using (MySqlCommand myCommand = new MySqlCommand(query, mycon))
        {
            myReader = myCommand.ExecuteReader();
            table.Load(myReader);

            myReader.Close();
            mycon.Close();
        }
    }

    return new JsonResult(table);
}

[HttpPost]
0 references
```

3. DB Design

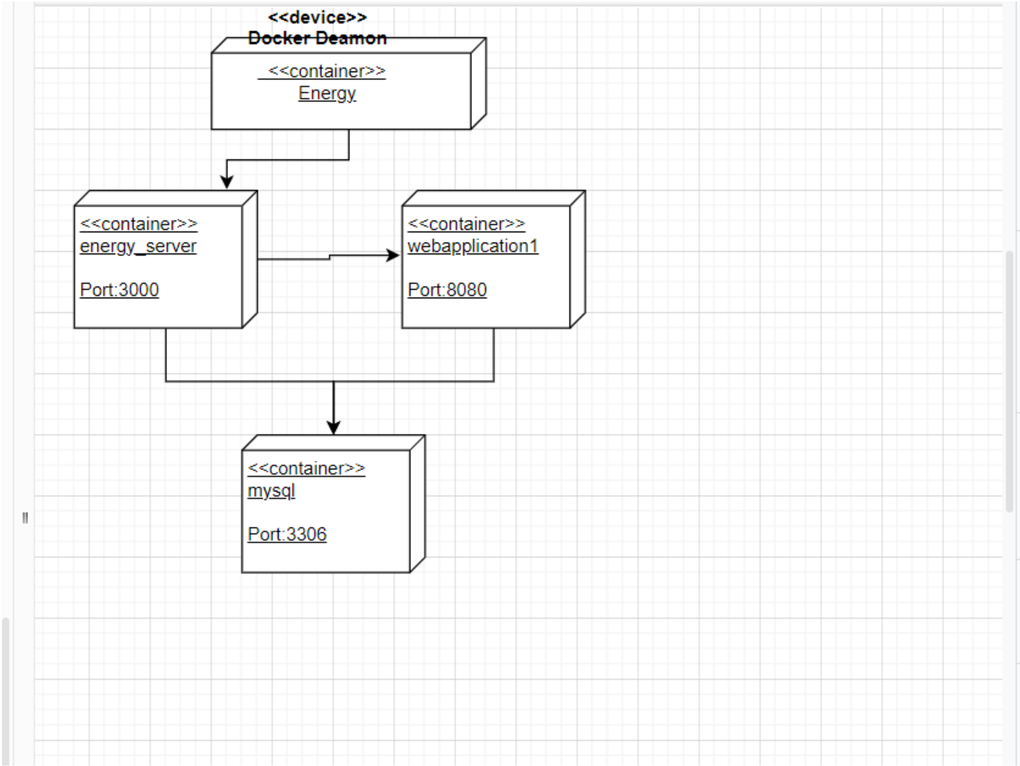


ENERGYP DATABASE

USER	
PK,FK1	Id
	name (INT)
	email (VARCHAR)
	password (VARCHAR)
	role (VARCHAR)

DEVICE	
PK,FK1	Id
	description (VARCHAR)
	address (VARCHAR)
	energy_const (INT)

4. UML Deployment Diagram


<<device>>
Docker Deamon
<<container>>
Energy
<<container>>
energy_server
Port:3000
<<container>>
webapplication1
Port:8080
<<container>>
mysql
Port:3306