

**FACULTY OF AUTOMATION AND COMPUTER SCIENCE**

**COMPUTER SCIENCE DEPARTMENT**

Request-Reply Communication Paradigm

**\**

Online Energy Utility Platform

Profesor indrumator: Student: Dragusanu Octavian

Grupa:

Cuprins

[1. Cerintele aplicatiei 3](#_Toc118723997)

[2. Arhitectura sistemului 3](#_Toc118723998)

[3. Design-ul bazei de date 4](#_Toc118723999)

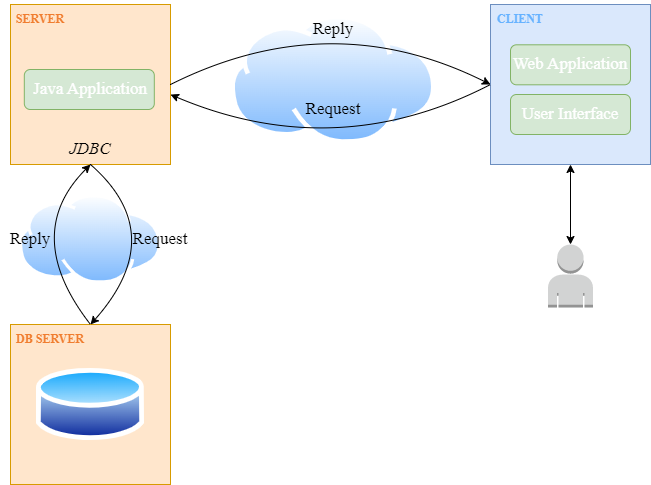
[4. Diagrama UML pentru deployment 5](#_Toc118724000)

[5. Readme 6](#_Toc118724001)

# Cerintele aplicatiei

Trebuie proiectata si implementata o aplicatie web cu React pe partea de frontend si Springboot pentru backend, care sa poata gestiona utilizatorii, dispozitivele inteligente de masurare a energiei associate fiecarui utilizator si sa se poata monitoriza consumul al fiecarui device. Sistemul poate fi accesat de catre doua tipuri de utilizatori, in urma procesului de logare: administratorul si clientii. Administratorul poate efectua operatiuni CRUD(Create-Read-Update-Delete) pe conturile clientilor, pe device-urile smart de masurare a energiei, de asemenea, acesta poate mapa device-uri utilizatorilor prin interfata web. Pentru fiecare device smart, consumul de energie este salvat in baza de date pentru fiecare ora, ca si tuple de forma <timestamp, energy consumption>.

# Arhitectura sistemului



Pentru crearea acestei aplicatii am ales sa folosesc o arhitectura de tipul client-server. Pentru partea de client, adica frontend, am folosit Javascript impreuna cu o biblioteca open-source pentru construirea interfetelor de utilizator, React. Pentru partea de backend am folosit limbajul Java impreuna cu framework-ul Springboot, iar pentru stocarea datelor(utilizatorii, device-urile, consumul device-urilor) am utilizat o baza de date relationala Postgres.

Aplicatia functioneaza in felul urmator: Un utilizator foloseste User Interface(aplicatia web) pentru a face diferite request-uri la server, server-ul ii intoarce mereu un ranspund la request-ul facut.

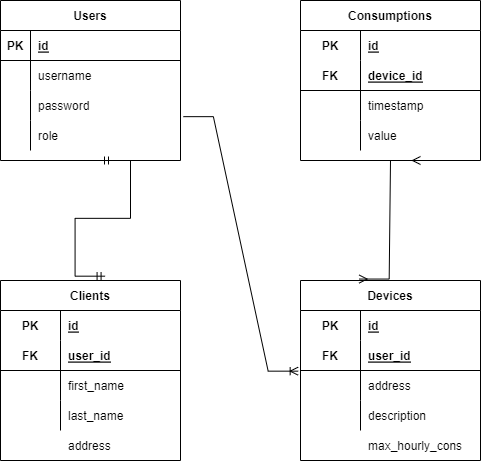
Cele 3 mari layere in care e impartita aplicatia:

CLIENT – aplicatia web la care are acces utilizatorul, prin ea se pot indeplinii toate cerintele specificate.

SERVER – aici se afla aplicatia Java, se fac mapari, se declara entitatile de care avem nevoie, se fac validari.

DB SERVER – aici se afla stocata toata informatia de care avem nevoie sa o trimitem mai departe pe frontend.

# Design-ul bazei de date



Fiecare User are asociat un Client, de aici relatia *One to One*.

Un Device trebuie sa apartia unui User si un User poate sa aiba mai multe Device-uri la locatii diferite, de aici relatia One to Many.

Un Consumption trebuie poate apartine unui singur Device, iar un Device poate avea mai multe Consumptions, relatia One to Many

# Diagrama UML pentru deployment

Diagram

Description automatically generated

In diagrama ne este prezentat modul de comunicare dintre partea de Client si cea de Server.

# Readme

Pentru a executa aplicatia pe computer-ul tau citeste README.md care se afla in cele doua repositories.

FRONTEND – [https://github.com/octaviandrg/DS202230643\_Dragusanu\_Octavian\_1\_Frontend](https://github.com/octaviandrg/DS202230643_Dragusanu_Octavian_1_Backend)

BACKEND - <https://github.com/octaviandrg/DS202230643_Dragusanu_Octavian_1_Backend>