**DISTRIBUTED SYSTEMS**

**Assignment 1**

**Request-Reply**

**Communication Paradigm**

**Online Energy Utility Platform**

**Cerghizan Gabriel-Ioan**

**Grupa 30241**

Contents

[**1.** **Arhitectura conceptuala** 3](#_Toc119571024)

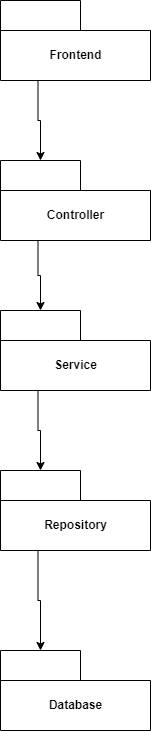
[2. Diagrama bazei de date 4](#_Toc119571025)

[3. Diagrama de deployment 5](#_Toc119571026)

[4. Fisierele Docker 6](#_Toc119571027)

[5. Pasi necesari pentru a porni aplicatia 8](#_Toc119571028)

# **Arhitectura conceptuala**



Arhitectura proiectului se bazeaza partea de backend pe un REST API avand ca Spring Boot ca si tehnologie, o baza de date in MySQL si pe serviciul de frontend scris in React.js.

Interafata grafica trimite requesturi catre api, iar acesta, dupa ce proceseaza datele trimite raspunsuri inapoi, bazandu-se pe JWT Tokens.

REST API e constituit dintr-o tehnologie bazate pe ideea de autentificare si autorizare prin intermediul tokenilor. Tokenii de tip JWT contin informatie despre utilizator, precum username, roluri si timp de validitate. Dupa ce un token expira, este necesara reautentificarea sau improspatarea tokenului de access prin intermediul unui alt token de improspatare.

Controllerele sunt primele layere care sunt accesate dupa autorizare. Acestea contin toata logica si se folosesc de servicii pentru a prelucra informatia venita din requesturi si din baza de date.

Serviciile sunt bazate pe conceptele SOLID, astfel incat fiecare metoda stie sa faca o singura sarcina si nu comunica intre ele.

Repository-urile fac legatura cu baza de date avand ca si unealta JPA Repository, un feature al Java care prelucreaza mai usor obiectele si faciliteaza comunicarea cu baza de date.

# Diagrama bazei de date

Diagram

Description automatically generated

Baza de date este alcatuita din 5 tabele: User, Device, User\_role, Role si History. Fiecare user poate sa aiba mai multe device-uri sau niciunul, fiind legat de tabela Device printr-o cheie straina. Fiecare Device poate sa aiba un istoric alcatuit din mai multe citiri ale senzorului, fiind legat de tabela History print-o cheie straina. Un utilizator poate sa aiba mai multe roluri, de aceea, ca sa nu existe legaturi de tipul many-to-many, am ales sa mai adaug o tabela intermediara intre user si role. Aceasta contine id-ul utilizatorului si id-ul rolului pe care acesta il detine.

# Diagrama de deployment

Diagram

Description automatically generated

Cele 3 servicii (frontend, backend si baza de date) sunt introduse in interiorul unor containere de tip Docker care ruleaza programul intr-o masina virtuala de tip Linux oferita de catre Docker pentru a usura utilizarea aplicatiei. Pentru a rula aplicatie este necesara o singura comanda “docker-compose up”. Aceasta construieste imaginile si porneste containerele astfel incat aplicatia o sa fie functionala fara a se porni alt IDE.

Frontendul ruleaza pe portul 80, acesta fiind mapat la portul 3000 si comunica cu aplicatia backend prin intermediul unui proxy. Backendul ruleaza pe portul 8000 si comunica cu baza de date prin intermediul unui network create in aplicatia Docker si ruleaza pe portul 3306.

# Fisierele Docker

Backendul si baza de date sunt configurate in aceleasi fiser Dockerfile in backend. Fisierul Dockerfile afferent frontendului este separat. Toate 3 sunt legate prin intermediul unui fisier docker-compose.yml aflat in radacina proiectului.

Text

Description automatically generated

Text

Description automatically generated

# Pasi necesari pentru a porni aplicatia

1. **Se descara repository-ul de pe Gitlab de la urmatorul link:** <https://github.com/gabicerghizan/DS2023_30241_Cerghizan_Gabriel_Assignment_1>
2. **Intr-un terminal aflat in radacina aplicatiei descarcate de la link-ul de mai sus, se ruleaza comanda** *“docker-compose up -d*”, **aceasta construieste imaginile si porneste containerele.**
3. **Se acceseaza intr-un browser de internet urmatorul link:** <http://localhost:80>