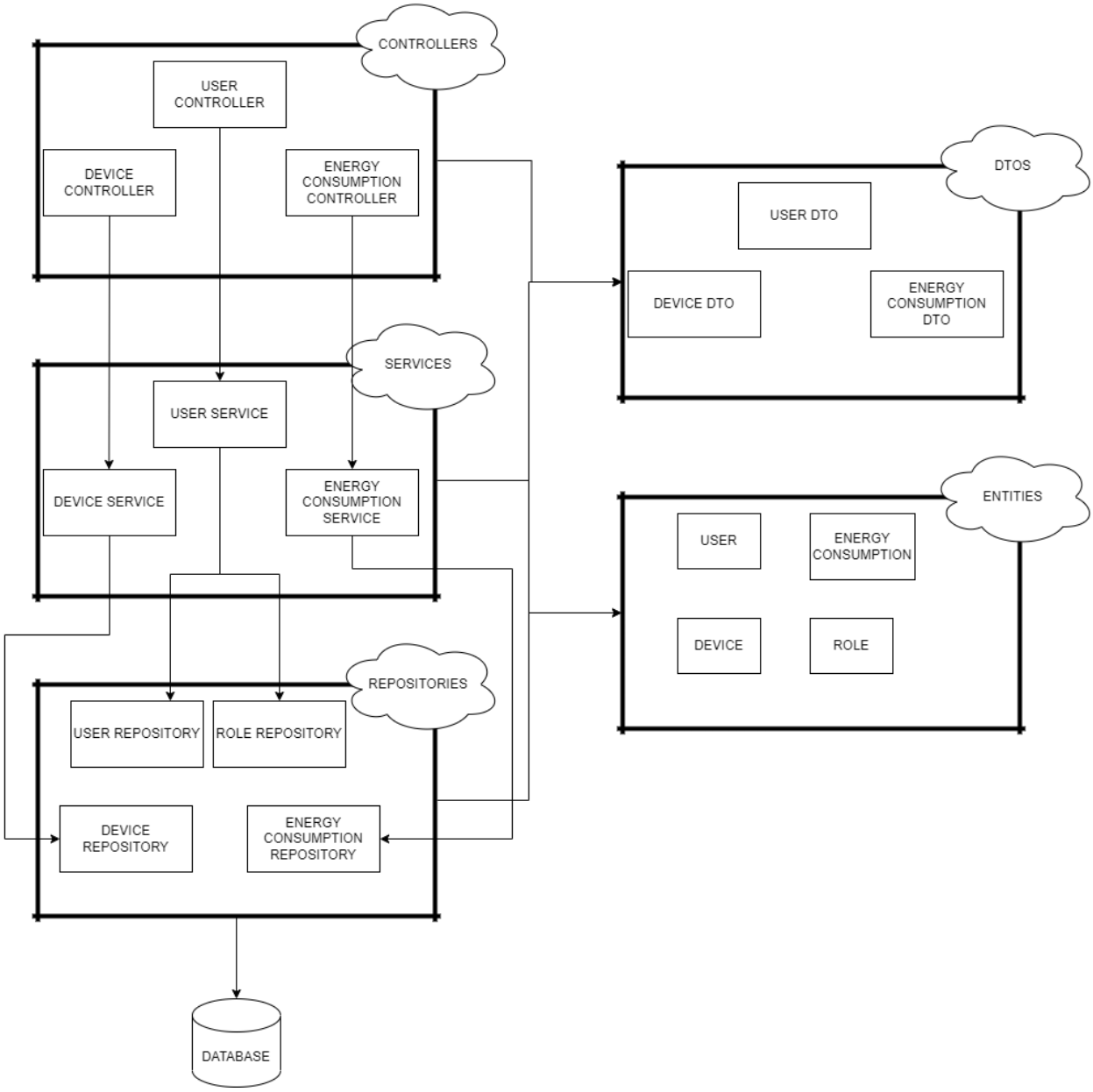
DOCUMENTATIE ASSIGNMENT 1 – ONLINE PLATFORM

Realizator: Pop Alexandra

Grupa: 30641, TI

1. Arhitectura conceptuală



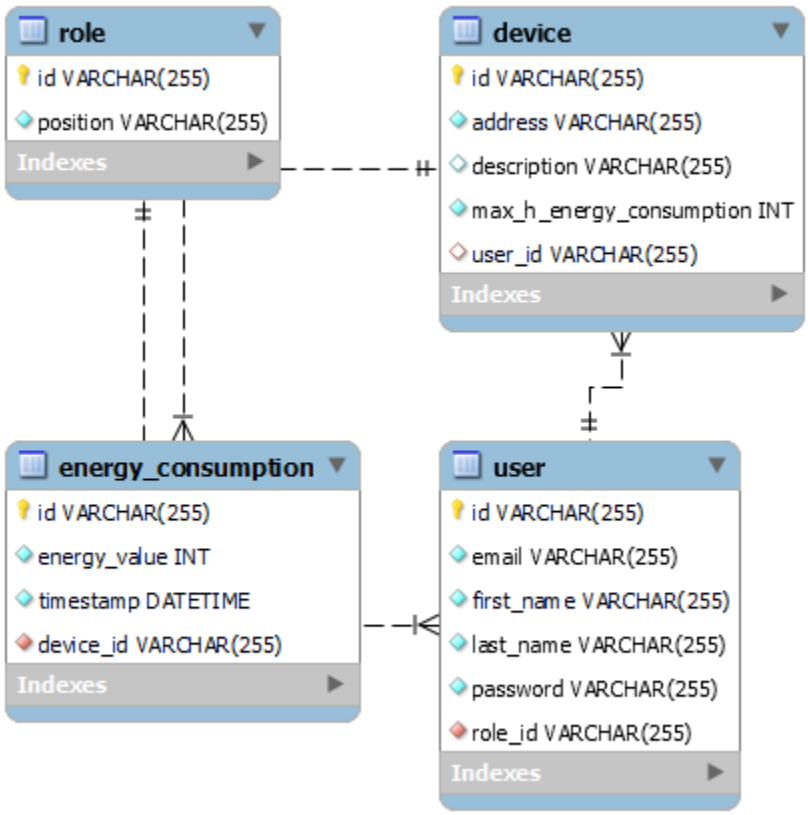
Principalele controllere ale aplicației sunt cele de UserController (unde avem request-urile pentru vizualizarea tuturor userilor, vizualizarea unui user după id, adăugarea unei noi instanțe de user, editarea unui user după număr variabil de parametri și ștergerea unui user după id), DeviceController (unde avem aceleași tipuri de request-uri ca la user) și EnergyConsumptionController (conține doar request-uri de vizualizare tuple pentru un anumit device după id-ul acestuia și de adăugare a unei instanțe noi în db). În plus, mai avem un controller pentru entitatea ”Role” (RoleController), unde avem request de adăugare rol, folositor doar pentru adăgarea de instanțe noi fără a folosi DBMS-ul bazei de date.

Există clase de service pentru fiecare entitate a aplicației, acestea realizând o legătură între Controllere și Repositories și conversia de la DTO la entități și invers.

De asemenea, există repository pentru fiecare entitate, și doar cel de EnergyConsumptionRepository are un query manual, ce face extragerea tuplelor din db după id-ul device-ului de care aparțin prin foreign key.

În aplicație au fost create diverse DTO-uri pentru vizualizare, salvare, editare entități, dar în diagramă sunt evidențiate doar niște DTO-uri dummy, pentru a forma o viziune asupra conținutului aplicației.

2. DB design



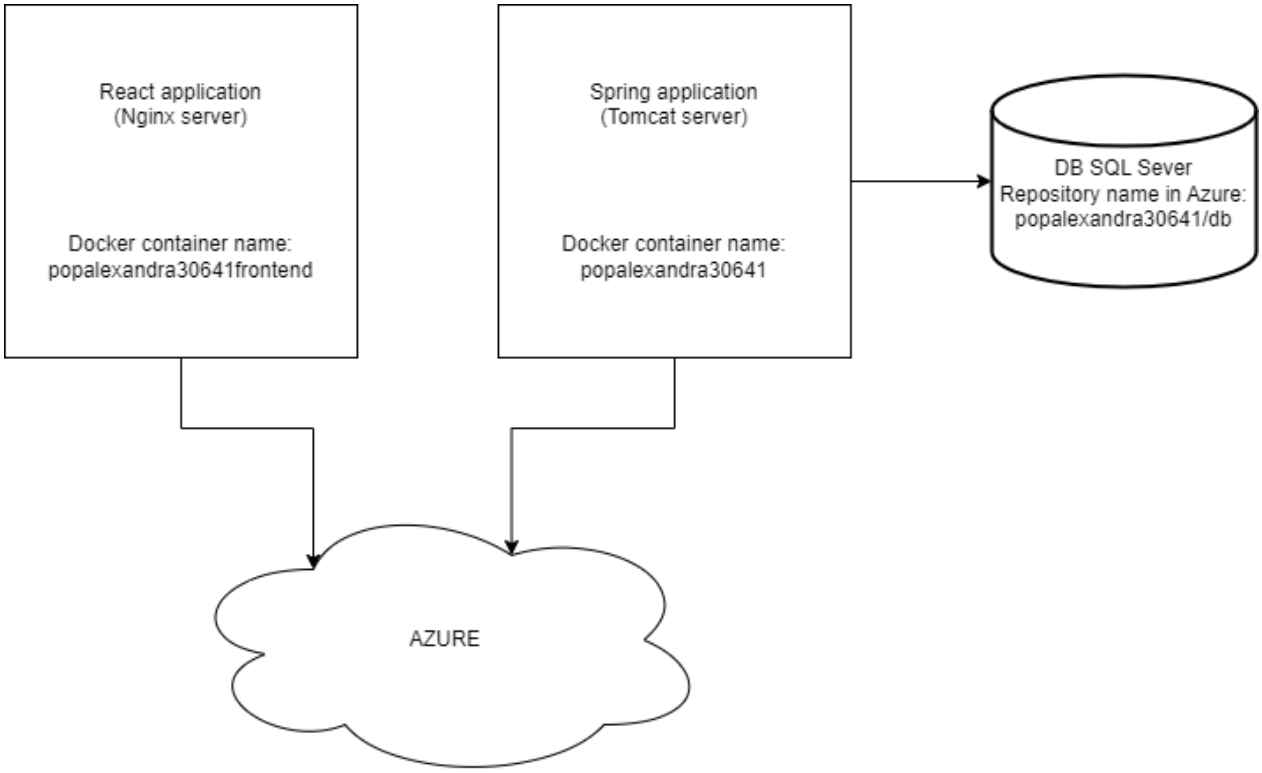
Baza de date „online-platform” a aplicației conține 4 tabele.

Tabela de „user” păstrează datele fiecărui utilizator al aplicației, și are o relație de „many-to-one” cu tabela „role” care stochează cele 2 roluri ale aplicației („ADMIN” și „CLIENT”), fiecare utilizator având un rol asociat.

Tabela „device” are o relație „many-to-one” cu tabela „user”, deci fiecare user va putea avea multiple instanțe de „device”-uri.

Ultima tabelă, „energy_consumption”, păstrează o instanță de consum pentru „device”-uri, cu data la care au fost măsurate și valoarea măsurată, pentru ca user-ul să își poată ține evidența consumurilor pe ore și pe zile.

3. UML Deployment diagram



Pentru fiecare din aplicații (frontend și backend) s-a făcut un deployment pe Azure Cloud.

Aplicația de backend este conținută în container-ul denumit ”popalexandra30641”, din resource group-ul ”AlexandraPop30641”, ce conține repository-ul atât pentru backend, cât și pentru baza de date. La fiecare repornire a container-ului creat, se reinstanțiază baza de date (deci datele anterioare nu mai sunt persistate), și adresa de IP pe care rulează aplicația.

Aplicația de frontend este conținută în container-ul denumit ”popalexandra30641frontend”, din același resource group ca aplicația de backend, și este legată la aplicația de backend care se află pe portul 8080.

La fel ca la rularea pe mașina locală a aplicațiilor, ambele containere (frontend și backend) trebuie să fie pornite pentru ca aplicațiile să fie funcționale (implicit trebuie pornită și vm ce conține agent-ul).

4. Readme file containing build and execution considerations

Pentru rularea aplicației de backend pe mașina locală, trebuie mai întâi să se introducă comanda ”git clone https://github.com/DS202230641PopAlexandra/DS2022_30641_Pop_Alexandra_1_Backend.git” în fișierul în care se dorește stocarea proiectului. Apoi se deschide proiectul în IntelliJ și se rulează fișierul ”DS2020Application.java”, iar aplicația va fi funcțională.

Pentru rularea aplicației de frontend, trebuie utilizată comanda de ”git clone https://github.com/DS202230641PopAlexandra/DS2022_30641_Pop_Alexandra_1_Frontend.git” în folder-ul dorit, apoi după deschiderea proiectului în ide-ul IntelliJ, trebuie rulate în terminalul ide-ului comenzile ”npm install” pentru ca toate librăriile specificate în ”package.json” și ”package-lock.json” să fie instalate (acestă comandă trebuie rulată la fiecare reclonare a proiectului), și ”npm run start” pentru a porni aplicația (se va deschide automat pagina html principală în browser-ul vostru default).

Pentru ca aplicația de frontend (react app) să aibă acces la datele din baza de date și pentru a putea trimite request-uri, trebuie ca aplicația de backend să fie funcțională.