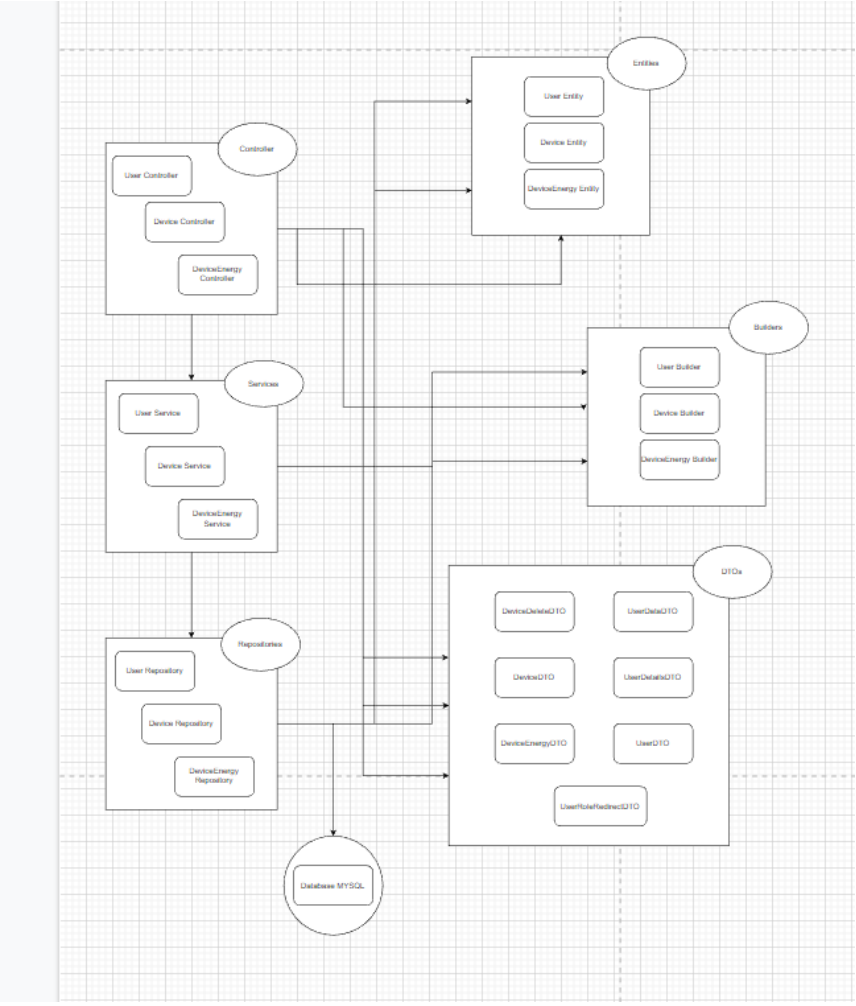
Aplicatie de monitorizare a energiei pentru device-uri

Orsan Tudor Alexandru, 30641, TI

Aceasta aplicatie a fost creata pentru a monitoriza cat de multa energie consuma device-urile clientilor ce folosesc aplicatia. Scopul aplicatiei este sa poata fi rulata de pe cloud, si sa poata monitoriza aceste date in real time.

1) Arhitectura conceptuala:

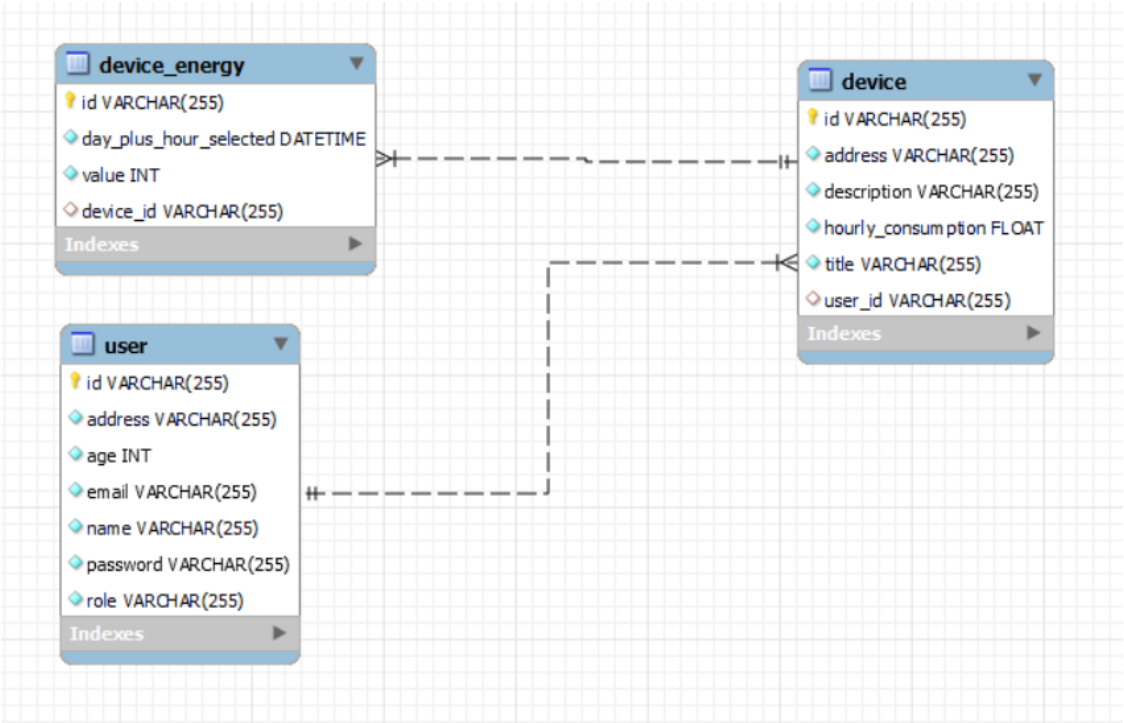


Pentru aceasta diagrama:

Aplicatia se bazeaza pe urmatoarele concepte:

- In backend, exista entitati, care sunt reprezentari exacte ale tabelelor din baza de date. In loc sa lucram cu ele, avem obiecte DTO, cu care lucram peste tot in aplicatie, si care sunt modificate astfel incat sa se potriveasca cerintelor noastre. Pentru a face conversiile intre obiecte, exista clase Builder.
- Flow-ul aplicatiei ramane acesta: Repositori-urile comunica cu baza de date pentru a face queri-uri tabelelor, dupa care service se uita dupa exceptii la date, sau foloseste functiile din repo-uri pentru a manipula datele, iar in controller se fac metodele de Get si Post, cele care ori primesc date din Frontend, ori trimit date inapoi la Frontend, depinde de situatie. Comunicarea cu frontend se face cu status-uri, iar datele sunt trimise de obicei ca DTO-uri, care se convertesc in obiecte json pentru ca React sa poata lucra mai usor cu ele.
- Tot pentru partea de comunicare, este important de retinut faptul ca aplicatiile de Frontend si Backend comunica prin request-uri si response-uri HTTP, prin care se trimit date esentiale ce pot fi manipulate in ambele parti. In functie de cum avem functionalitatile, vom face procesarile ori intr-o parte, ori in cealalta. Cand vine vorba de schimbat datele totusi, cel mai bine este sa o faci in backend.
- Cand vine vorba de Frontend, componentele sunt reprezentate astfel: API pentru a putea avea acces la controller, si pentru acces la metodele sale (Get, Post, etc...). Validatori pentru cand se face Post, sa trimitem datele in mod corect. Table pentru cand este reprezentarea unui tabel anume. Form sau un derivat al acestuia pentru cand se doresc trimiterea datelor sau primirea lor. Container pentru forma exacta a paginii (cu toate componentele de mai sus), care va fi reprezentata ca URL in app.js.
- Aceasta este structura prezentata pe scurt, cat despre functionalitati, sunt cele cerute la tema 1, legate de crearea user-ilor, device-urilor, crearea relatiei intre cele 2 entitati, si vizualizarea datelor unui user (cat si datele deviceuri-lor sale).

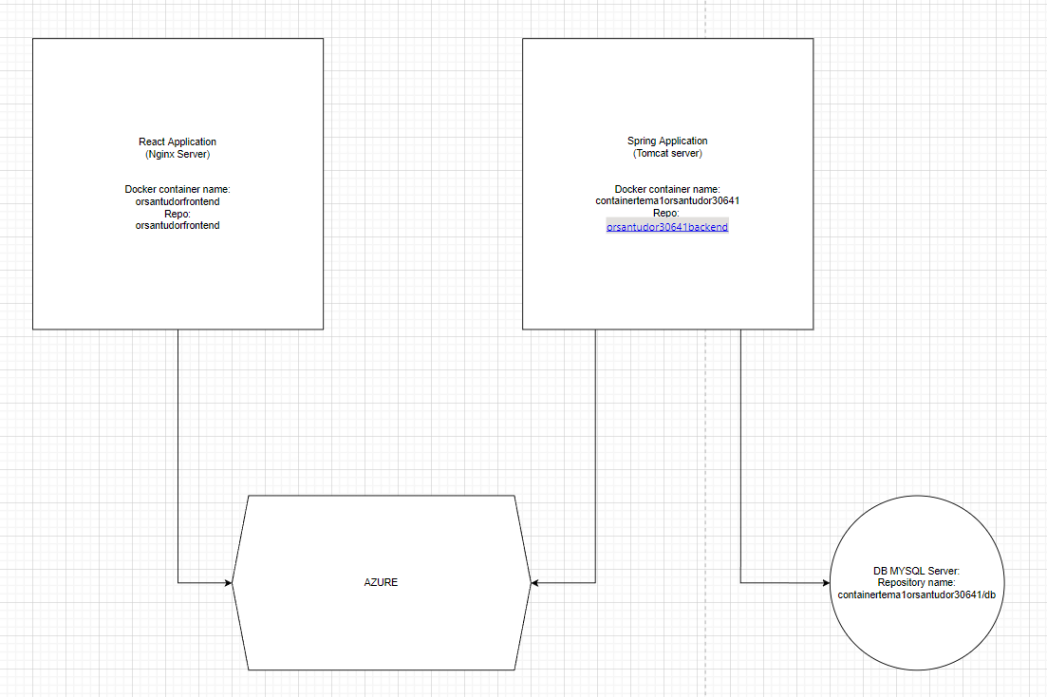
2) Diagrama bazei de date:



Pentru aceasta diagrama:

In baza de date, am creat 3 tabele. Cel de user poate contine atat clienti, cat si administratori (rolul este un atribut obligatoriu). Un User poate avea N device-uri, iar un Device poate avea N DeviceEnergy-uri. Asta inseamna ca un User poate avea oricate Device-uri, iar un Device poate avea oricate entrie-uri pentru consum, pe zile / ore diferite. Doar User si Device au operatii CRUD create in aplicatie, pentru DeviceEnergy sunt introduse datele manual, urman ca la tema 2 sa se adauge in alte moduri aceste date.

3) Diagrama de deploy:



Pentru aceasta diagrama:

Cand vine vorba de deploy, am facut deploy separat pentru partea de BD, Backend si Frontend. Exista un repository pentru fiecare, si ele comunica intre ele dupa release-urile fiecarui container individual. Este ca si cum ai rula local fiecare aplicatie, dar in loc sa fie rulate local, pe un server propriu, sunt puse pe cloud, si ruleaza cat timp containerele raman deschise. Este o metoda foarte eficienta de a tine o aplicatie deschisa, doar atata timp cat este nevoie de ea. Pentru fiecare release nou, se schimba portul pentru componente, dar ele ruleaza la fel. Baza de date trebuie, de asemenea, reinstaurata dupa oprirea-pornirea containerelor.

In final, cand toate elementele aplicatiei ruleaza in mod corect, partea de BD, Backend, Frontend, Azure, aplicatia poate fi vazuta ca un exemplu bun de justificare a cloud-ului. Toate componentele aplicatiei sunt incarcate pe cloud, si astfel se poate rula doar cat timp este necesara rularea lor, nu este nevoie de intretinerea serverelor administratorului, este o alegere buna atunci cand doresti crearea unei aplicatii noi si eficiente.