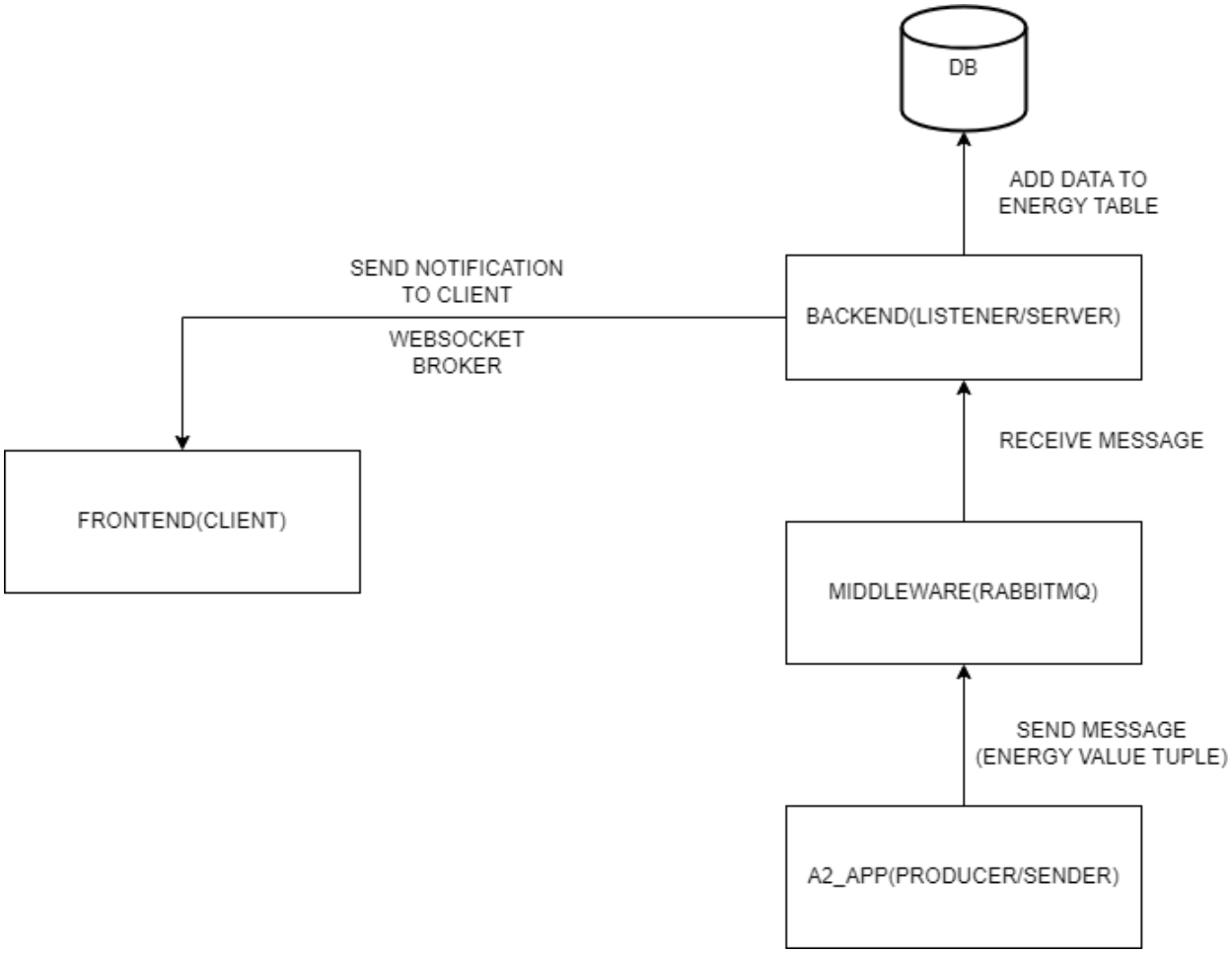
DOCUMENTAȚIE ASSIGNMENT 2 – ONLINE PLATFORM

Realizator: Pop Alexandra

Grupa: 30641, TI

1. **Arhitectura conceptuală**



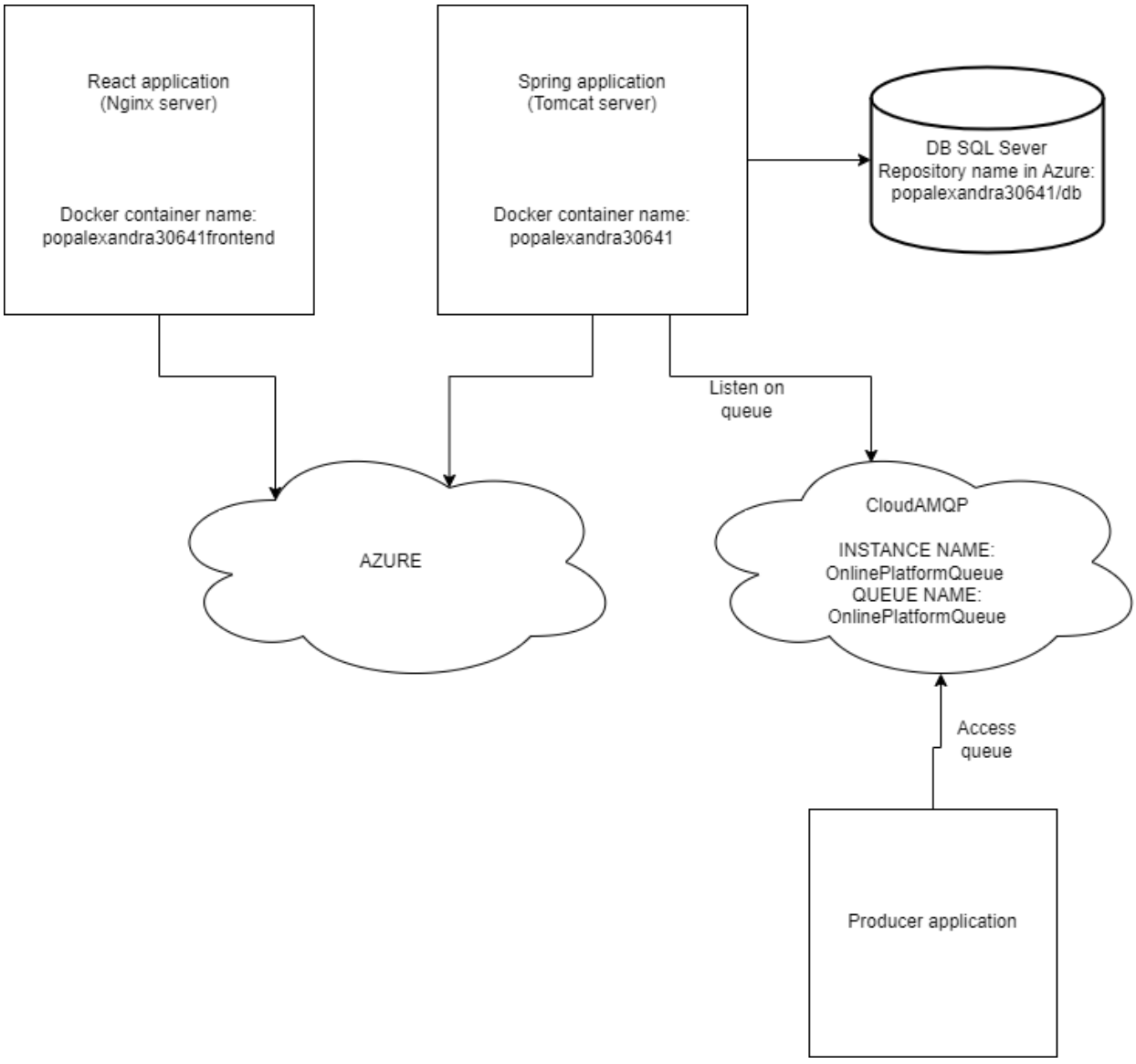
Din punct de vedere al arhitecturii sistemului distribuit realizat, pe lângă aplicațiile de frontend (react) și backend (spring boot) am adăugat o aplicație ce transmite mesaje către backend prin intermediul unei cozi, creată pe cloud (CloudAMQP), ce este administrată de un broker (middleware) RabbitMQ.

Aplicația de producer conține o clasă de configurare a cozii către care transmite mesaje (AMQPConfig.java), o clasă pentru transmiterea efectivă a mesajelor (MeasurementsSender.java), o clasă pentru citirea din fișierul ”.csv” (MeasurementsCsvReader.java) și o clasă de tip dto pentru transmiterea mesajelor sub o anumită formă (MeasurementDTO.java).

Producer-ul de mesaje transmite tuple cu structura (timestamp, device id, value), citite dintr-un fișier ”.csv” la fiecare 10 minute, simulând un device de măsurare a energiei. Pentru programarea citirii din fișier la fiecare 10 minute am folosit adnotarea @Scheduled din Spring Boot la metoda de ”send” cu atributele fixedDelay = 600000 (10 minute in ms; programează ca metoda să fie executată odată la timpul dat de acest parametru, delay-ul calculându-se între finalul unei execuții și începutul următoarei) și initialDelay = 1000 (1 secundă in ms; aplică un delay de 1 secundă până la începerea primei execuții a metodei). Transmiterea datelor în coadă se face prin conversia lor în format JSON.

Pe partea de backend am adăugat partea de listener pentru mesajele transmise de producer către coadă, aceasta reprezentând consumer-ul mesajelor. Clasa ”EnergyDataReceiver.java” implementează listener-ul cu ajutorul adnotării @RabbitListener pe coada în care transmite producer-ul, preia tuplele, verifică datele din interior, face o sumă a valorilor pe oră și adaugă suma finală în baza de date, iar dacă aceasta depășește valoarea maximă a device-ului pentru care sunt transmise tuplele (valoare preluată din baza de date) va transmite o notificare către partea de frontend (client-ul căruia îi este assignat device-ul respectiv). Această notificare este transmisă prin websockets, se deschide un canal între backend și frontend cu ajutorul unui broker, iar backend-ul poate transmite mesaje către frontend fără ca cel din urmă să facă request-uri continue, la anumite momente de timp. Configurarea acestui mijloc de comunicare se face în backend în clasa ”WebSocketConfiguration.java”, iar în frontend în fișierul ”client-container.js”.

2. UML Deployment diagram



Aplicațiile de frontend, backend și baza de date sunt puse pe cloud-ul de Azure, în aceleași containere de la assignment-ul 1, însă am făcut un nou release cu datele nou adăugate.

Aplicația de sender este rulată pe mașina locală și transmite datele într-o coadă care este stocată în CloudAMQP.

Backend-ul și sender-ul pot comunica foarte ușor deoarece coada este într-un mediu cloud, nu local.

3. 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.

Simulatorul de device-uri se poate clona pe masina locală cu comanda ”git clone https://github.com/DS202230641PopAlexandra/DS2022_30641_Pop_Alexandra_2.git” în fișierul în care se dorește stocarea proiectului. Prin pornirea acestuia, se va începe transmiterea de date, ele putând fi văzute în consola proiectului sau în consola backend-ului.