

Stecko Daiana

Grupa: 30242

DISTRIBUTED SYSTEMS

Assignment 2

Asynchronous Communication

Sensor Monitoring System and Real-Time Notification

**Curpins**

**1.**Tema

**2.**Functionalitate

**3.**Instrumente utilizate

**4.**Design-ul Arhitectural

Diagrama de cazuri

Diagrama de deployment

Diagrama de clase

**5.**Diagrama de model

**6.**Dezvoltari ulterioare

**7.**Concluzie

**8.**Bibliografie

**1.Tema**

Cerința pentru acest proiect a fost să realizăm o aplicație pentru afișarea unui simulator al unui senzor, iar valorile pentru acest senzor se citesc dintr-un fișier csv. Clienții dețin dispozitive care pot înregistra măsura consumului de energie produs de un dispozitiv.

Acești senzori trimit date, la o anumită perioadă selectată de noi către server.

Aplicația a fost implementată prin 3 module: message producer, message broker și message consumer, iar mesaje vor fi extrase din coada.

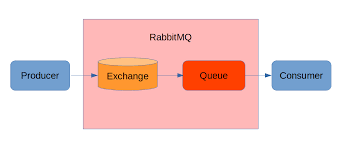
Un simulator de senzori va simula un senzor care citește date din fișiere (sensor.csv), o valoare la fiecare 10 minute. Modulul va conține un cronometru sincronizat cu ceasul local.

Simulatorul de senzori este prezentat ca o aplicație autonomă, adică o aplicație desktop, pentru a citi activitățile monitorizate de senzor din fișierul sensor.csv, configurat ca producător de mesaje și trimite datele eșantionului monitorizat la coada definită.

**2.Functionalitate**

Aceasta aplicație presupune adăugarea unor noi utilizatorii. Acești utilizatori vor avea asignate device-uri, iar pentru fiecare device avem măsurători. Pentru fiecare entitate, am realizat deasemenea, pe lângă posibilitatea de inserare, și editare și ștergere.

Pentru realizarea simulării din aceasta temă, am adăugat pentru un device din baza mea date, care este asignat unui anumite utilizator, măsurători. Măsurătorile sunt adăugate la un anumit timp, dat de mine, în baza de date, și se face o medie în fiecare ora pentru a se observa daca consumul într-o ora nu depășește consumul maxim admis al unui dispozitiv într-o ora.



**3.Instrumente utilizate**

La realizarea backend-ul m-am folosit de Spring Boot-Intellij IDEA. Am mai lucrat in trecut cu acest program, nu a fost ceva foarte diferit. Pentru frontend am utilizat VisualStudio Code cu Reactjs. Nu am mai utilizat foarte mult in tecut, a fost ceva mai nou, din acest motiv m-am confruntat cu diferite probleme, neestiind initial cum functioneaza. Insă in final am reusit să mă familearizez si lucrurile au decurs bine.

Am folosit ca baza de date postgres.

Pentru implementare am folosit o coada care este situata deja pe cloud de la rabbitMQ si am integrat conexiunea in tema trecuta pentru a putea avea acces la baza de date si la toate funcționalitățile de data trecuta si mai exact de aplicatia de front si de back.

In proiectul de Sender simulator are loc citirea măsurătorilor din fișierul sensor.csv si punerea lor în coadă. În proiectul de backend folosit la assignmentul anterior, am adăugat clasă Receiver care preia datele din coada și le afișează în consolă. Datele sunt transmise și preluate la un interval de 10 minute.

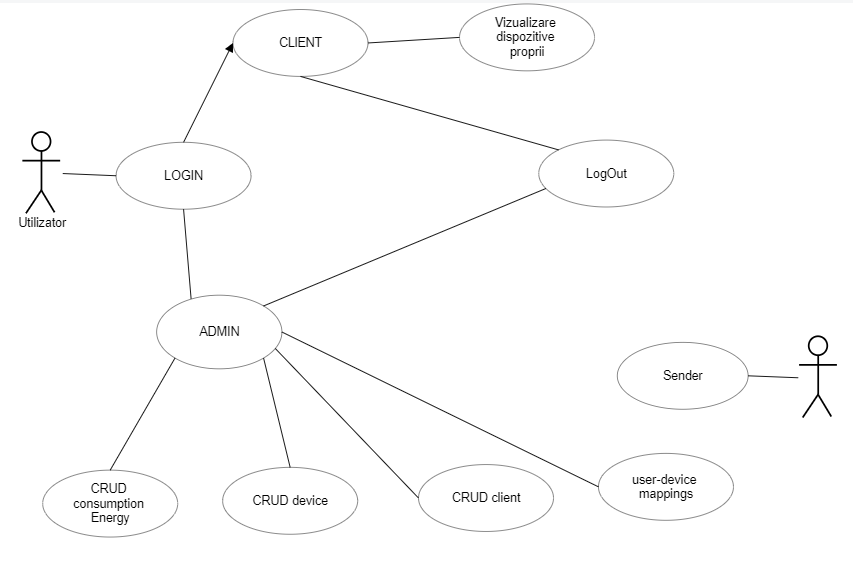
**4.Design-ul Arhitectural**

O arhitectură conceptuală descrie un stil arhitectural stratificat. Fiecare strat într-un stil arhitectural stratificat este un pachet de software care are o interfață bine definită și câteva dependențe bine-cunoscute cu alte straturi. Fiecare strat implementează o funcție tehnică în cadrul serviciului. De exemplu, un strat de acces la date este responsabil pentru încapsulare mijloacele tehnice pentru accesarea unei baze de date. Toate cererile de acces la date către o bază de date trec prin stratul de acces la date pentru acea bază de date.

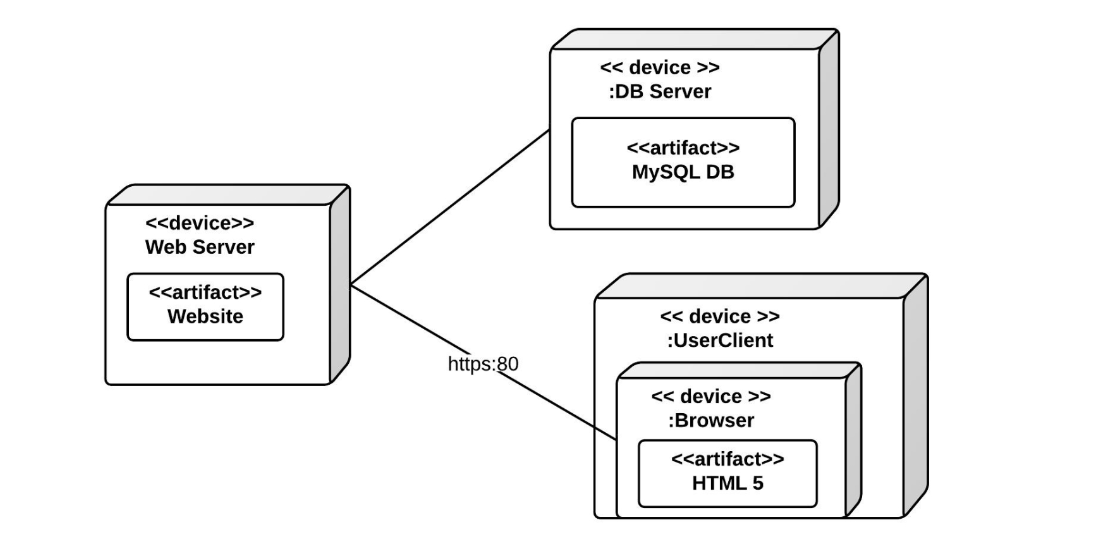
Sunt descrise ideile designului la nivel general, fiecare strat îndeplinind un anumit rol.

La nivel de coadă am implementat aplicația după trei module un message producer, acesta trimite mesaje pe coada, un message broker care este însuși coada și doi message consumeri care consumă mesajele din coadă și le inserează în baza de date.

**Diagrama de cazuri**



**Diagrama de deployment**

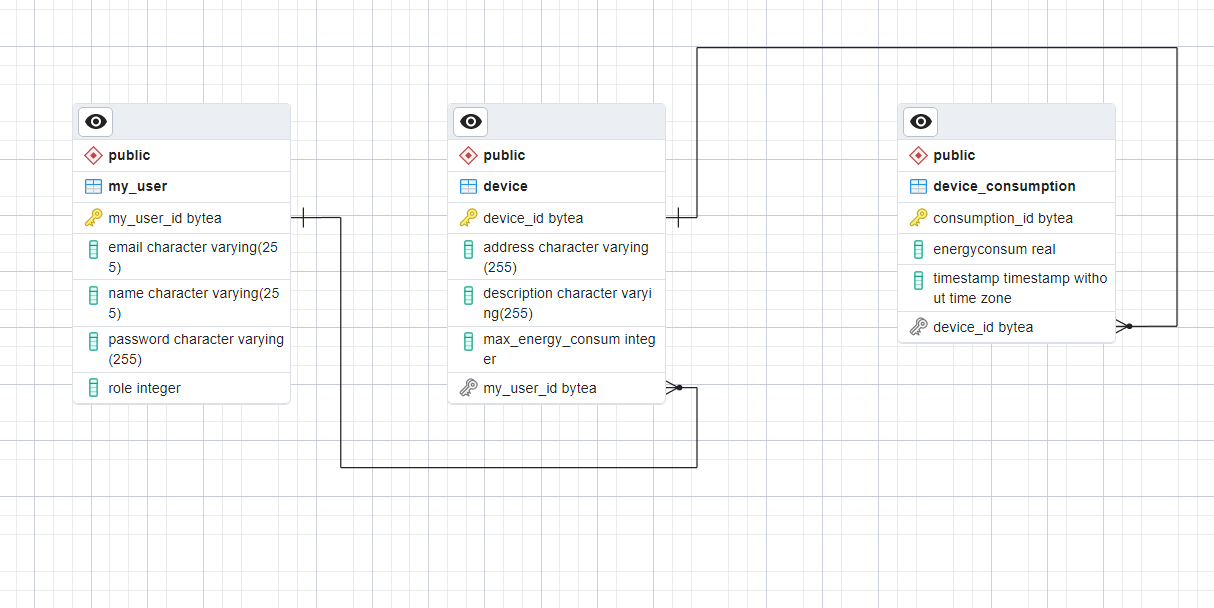
****

**Diagrama de clase**

**O imagine care conține text, monitor, captură de ecran, negru

Descriere generată automat**

**5.Diagrama de model**

****

**6.Dezvoltari ulterioare**

Cum nu am reușit să implementez toate cerințele, consider că pe viitor aș putea să mă ocup și de partea de deploy pe Azure, eu de data aceasta am reușit să fac deploy doar local. De asemenea, cum am reușit să adaug măsurători doar pentru un device, aș putea să implementez o metodă care să adauge pentru mai multe device-uri.

**7.Concluzie**

Acest proiect m-a ajutat să învăț foarte multe lucruri legate de transmiterea datelor mai ușor, până acum nu am pus foarte mult accent pe această parte. Însă odată cu proiectul acesta pot spune că am reușit să dezvolt această parte. Deci pot spune că am descoperit și odată și învățat foarte multe lucruri noi care mă vor ajuta cu siguranța pe viitor, consider că această materie ne ajută să facem mai multe lucruri deodată, ceea ce este foarte bine și în acest mod putem vedea ceea ce ne place cu adevărat.

**8.Bibliografie**

[**https://www.cloudamqp.com/**](https://www.cloudamqp.com/)

[**https://umes4ever.medium.com/real-time-application-using-websocket-spring-boot-java-react-js-flutter-eb87fe95f94f**](https://umes4ever.medium.com/real-time-application-using-websocket-spring-boot-java-react-js-flutter-eb87fe95f94f)

[**https://www.cloudamqp.com/docs/java.html**](https://www.cloudamqp.com/docs/java.html)

[**https://www.rabbitmq.com/tutorials/tutorial-one-java.html**](https://www.rabbitmq.com/tutorials/tutorial-one-java.html)

**W3schools**

**StackOverFlow**