A picture containing logo

Description automatically generated**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

DISTRIBUTED SYSTEMS

Assignment 2

Asynchronous Communication

Sensor Monitoring System and Real Time Notification

**Viman Andrei-Liviu**

**Grupa: 30246**

Cuprins

[Enunțul problemei 3](#_Toc118912107)

[Instrumente utilizate si justificarea limbajului ales 3](#_Toc118912108)

[Cerințele funcționale 3](#_Toc118912109)

[Modelul Use-Case 4](#_Toc118912110)

[Designul arhitectural 4](#_Toc118912111)

[Diagrama de clase 5](#_Toc118912112)

[Diagrama bazei de date 6](#_Toc118912113)

[Diagrama de deployment 6](#_Toc118912114)

# Enunțul problemei

Scopul acestui proiect este de a dezvolta o aplicație care sa gestioneze consumul dispozitivelor pe care un utilizator le are. Aplicația are doua tipuri de utilizatori: un administrator care gestionează baza de date si asociază dispozitive clienților, dar si clienții care pot sa își vizualizeze dispozitivele asociate dar si câteva statistici legate de consumul lor. In continuare a trebuit sa simulam niste senzori prin citirea unor date dintr-un fisier csv si incarcarea mesajelor intr-o coada de mesaje. La aceasta coada a trebuit sa conecta 2 clienti care sa citeasca si sa implementam un sistem de notificare de mesaje in timp real pentru clientii conentati la aplicatie.

# Instrumente utilizate si justificarea limbajului ales

Înainte sa mă apuc de implementarea acestei aplicații, a fost nevoie sa înțeleg mai bine ce am de făcut si cum trebuie sa funcționeze aceasta aplicație a trebuit sa folosesc StarUML pentru a construi diagrama cazurilor de utilizare iar mai apoi diagrama de clase care sa respecte arhitectura structurala. Numai după ce am făcut aceste lucruri m-am apucat efectiv de implementarea proiectului ales in Java.

Pentru a reuși sa implementez in Java am folosit mediul de dezvoltare IDE IntelliJ IDEEA 2021.3.2. Am ales sa folosesc acest ide deoarece este cel mai prietenos si cel mai ușor de folosit. Am ales sa folosesc limbajul Java pentru implementarea proiectului deoarece este un limbaj foarte des întâlnit intre programatori si este cel pe care îl știu cel mai bine. Pe de alta parte am ales Java deoarece are Java Spring cu care sunt destul de familiarizat, având nevoie de el pentru a implementa interfața grafica. Recunosc ca astfel am reușit sa învăț foarte multe lucruri noi in acest limbaj.

Pentru implementare a 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.

# Cerințele funcționale

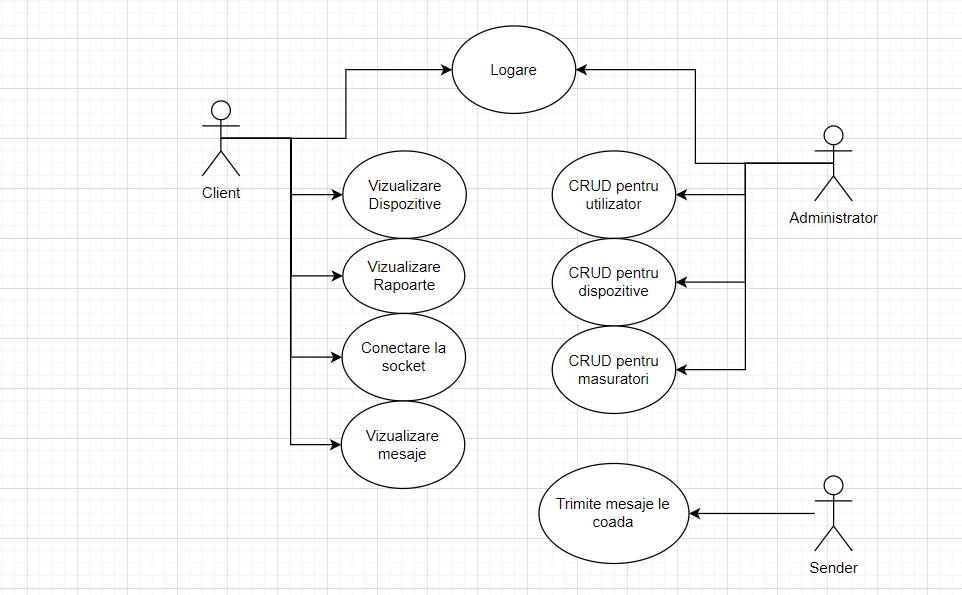
Aplicația își propune sa poate fi folosita de doua tipuri de utilizatori care pot sa aibă diverse drepturi de acces la baza de date administratorul având cele mai multe drepturi si clientul cele mai puține.

După conectare administratorul poate insera utilizatori noi, poate adaugă dispozitive utilizatorilor adăugați iar pentru fiecare dispozitiv poate sa adauge măsurători. Pentru toate aceste trei entități acesta poate sa facă toate operațiile unei baze de date, adică inserare, actualizare dar si ștergere.

Clientul nu poate sa își adauge singur dispozitive doar sa își vizualizeze dispozitivele asociate de administrator. Din parte de front am reușit sa fac ca acesta sa nu poată sa acceseze pagina administratorului. El poate sa vadă istoricul consumului lui de când are dispozitivul sau dispozitivele.

Pentru a simula senzorii am ales 2 dispozitive din baza de date care sun assignate unor utilizatori. Masuratorile sunt inserate odata al 10 minute in baza de date si se face o medie in fiecare ora pentru a se observa daca consumul intr-o ora nu depaseste consumul maxim admis al unui dispozitiv intr-o ora. Daca aceasta valoare este depasita, atunci se trimite o notificatre in timp real la utilizator daca este conectat si o sa ii apara un alert cu un mesaj specific de eroare.

# Modelul Use-Case



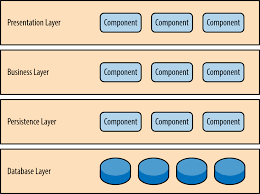
# Designul arhitectural

Un design arhitectural este punctul de plecare al unu proiect si descrie modul in care vor trebui scrise clasele. El trebuie respectat de la început pana la sfârșitul proiectului cu strictețe. Construind acest design se stabilesc structura componentelor proiectului din care fac parte: baza de date, beckendul si frontendul. Tot in aceasta etapa se stabilesc elementele functionale si se stabilește si designul interfeței utilizator.

Baza de date folosita la acest proiect este o baza de date relaționala creata in Postgres SQL, care conițe trei tabele unul pentru utilizatori, unul pentru dispozitive si unul pentru masuratori. Pentru partea de backend am ales limbajul Java împreună cu frameworkul spring care face ca implementarea backendului sa fie mult mai ușoară. Pentru frontend am ales sa folosesc React JS deoarece am mai lucrat cu Java-Script.

Arhitectura folosita este una de tip client server unde clientul trimite mai multe requesturi spre server, clientul așteptând la rândul lui răspunsuri de la server. Aplicația este construita folosind un design de tipul Layer Arhitecture in care se separa implementarea in mai multe straturi, stratul pentru baza de date, stratul pentru business, stratul pentru persistenta si stratul pentru prezentare așa cum se poate observa si in imaginea de mai jos.

Pentru coada am implementat aplicatia cu trei module un message producer care trimite mesaje pe coada, un broker care este chiar coada si doi message consumeri care consuma mesajele din coada si le insereaza in baza de date.

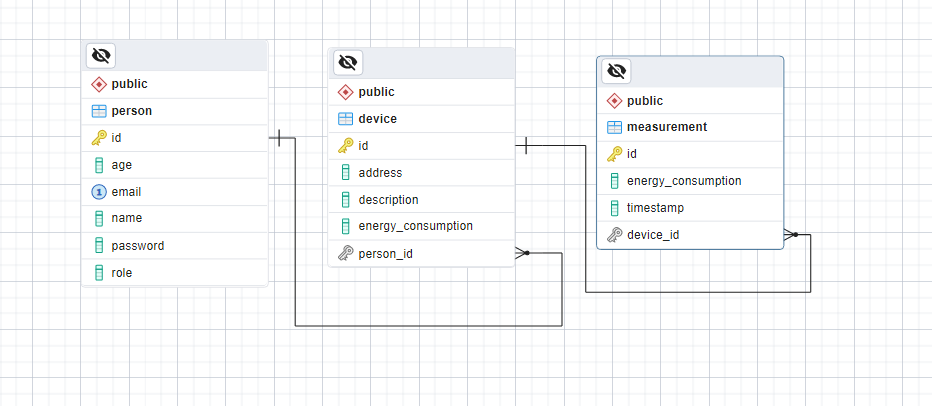


# Diagrama de clase

O imagine care conține text, negru, monitor, ecran

Descriere generată automat

# Diagrama bazei de date



# Diagrama de deployment

