

SISTEME DISTRIBUITE

Tema 3 **Documentație**

Remote Procedure Call (RPC) Medication Dispenser

Cioarea - Cristescu Roxana

Grupa: 30643

Cuprins

1. Arhitectura conceptuală a sistemului distribuit.....	3
2. Diagrama UML de Deployment	4
3. Build and execution considerations	5

1. Arhitectura conceptuală a sistemului distribuit

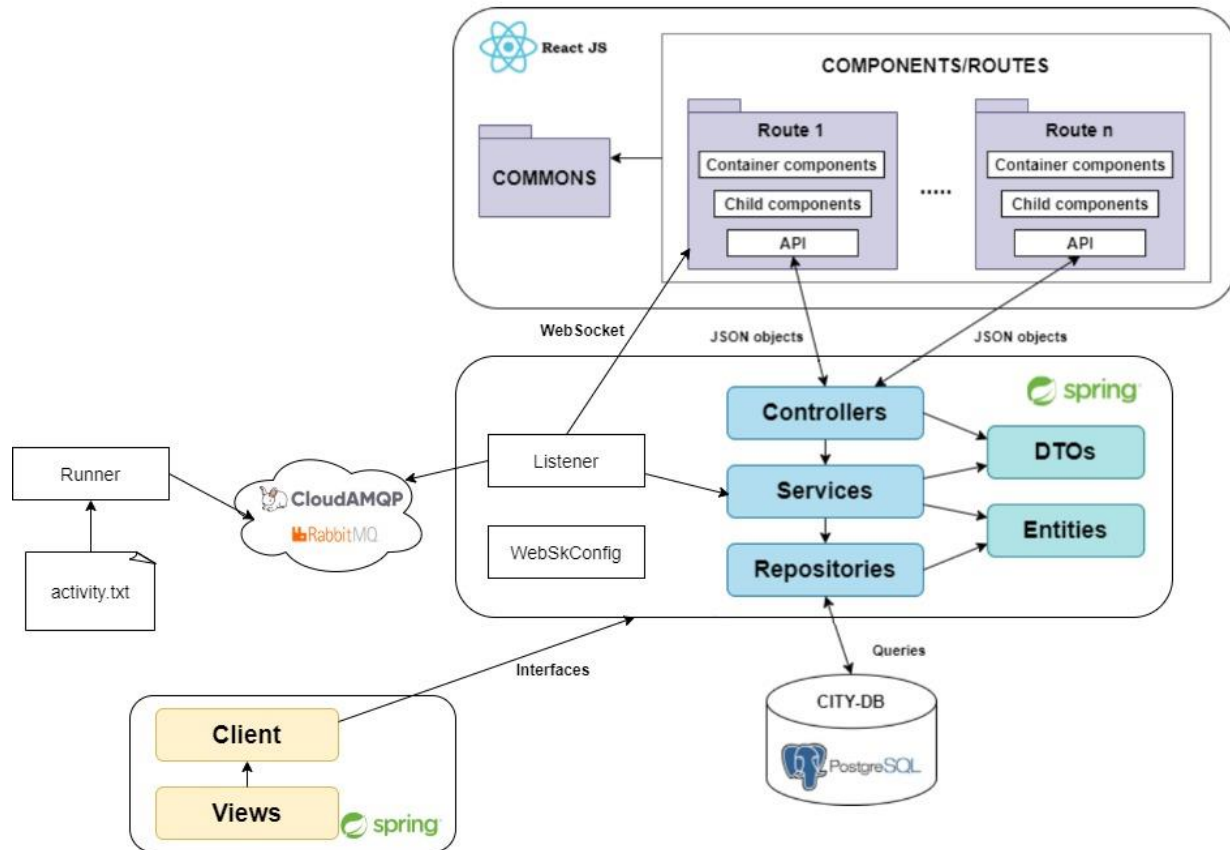


Figura 1. Arhitectura conceptuală a sistemului distribuit

Sistemul distribuit este alcătuit din trei aplicații de sine stătătoare care comunică între ele prin intermediul componentelor API (frontend) și a layer-ului Controllers (backend), cu baza de date comunică Repositories,

Producerul folosește RabbitMQ pentru a trimite date backend-ului care mai apoi folosește WebSockets pentru a trimite notificări, astfel realizându-se funcționalitățile cerute.

Din arhitectura aplicației React reies principalele componente frontend (Presentation), acesta sunt mai multe rute care conțin o componentă principală, urmată de componente secundare precum tabelurile și o clasă API. Partea de backend (Bussines) are o arhitectură bazată pe layere fiecare Entitate are cel puțin un DTO aferent care este trimis sau primit de Controller-ul specific, Controller-ul comunică cu baza de date prin intermediul claselor Repositories care sunt accesate folosind clasele Services. Modulul Producer are rolul de a simula citirea și transmiterea unor date de la niște senzori care monitorizează activitatea pacienților. Aici se citesc informațiile din fișier și se trimite câte un set de date la fiecare secundă.

2. Diagrama UML de Deployment

Modulele aplicației finale vor fi stocate în trei servere principale, unul pentru baza de date, unul pentru aplicația de frontend și unul pentru aplicația de backend acestea comunicând între ele. Partea de backend va comunica în mod direct cu baza de date, trimițând date pentru a fi stocate sau extrăgând date prin intermediul interogărilor. Browserul web este cel care facilitează comunicarea dintre frontend și backend. Comunicarea se face prin HTTP requests, iar datele care circulă sunt obiecte de tip JSON, astfel arhitectura aplicației este una de tipul client-server. Separat, pe un alt cloud se află serverul RabbitMQ care comunica cu Backend-ul

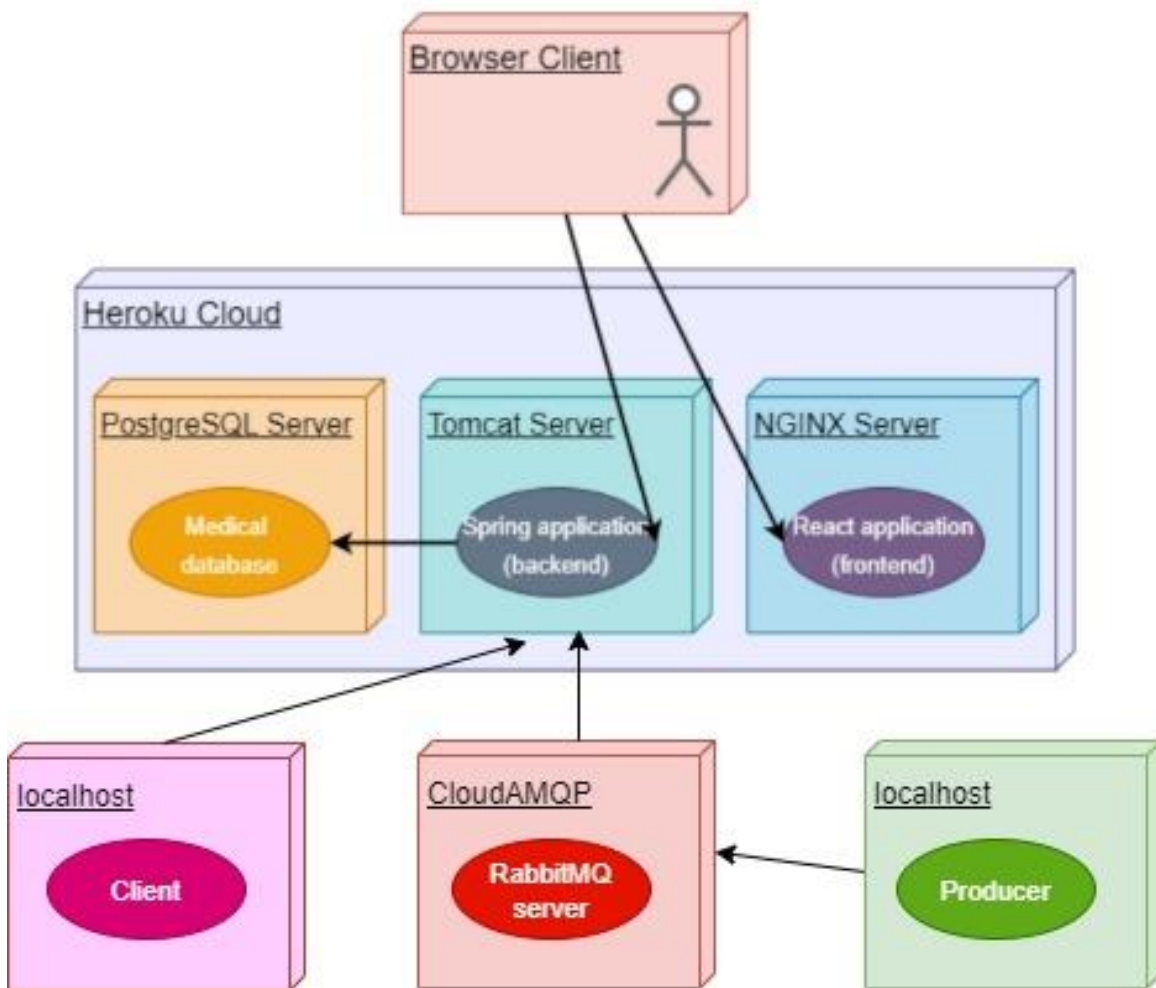


Figura 3. Diagrama UML de Deployment

3. Build and execution considerations

- **Backend:**

- Software requirements:
 - Java 11;
 - Server PostgreSQL pentru baza de date (v13.0.1) + pgAdmin,
 - Mediu de dezvoltare: IntelliJ
- Building requirements:
 - Crearea unei baze de date noua city-db;
 - Scrierea credentialelor PostgreSQL în application.properties
 - Scrierea credetialelor CloudAMQP în application.properties
- Execuție:
 - Se rulează fișierul Ds2020Application

- **Frontend:**

- Software requirements:
 - Instalare pachet npm
 - Mediu de dezvoltare: IntelliJ sau WebStorm
- Building requirements:
 - Scrierea adesei aplicatiei de backend în host.js
- Execuție
 - în terminal: npm install
 - în terminal: npm start

- **Producer:**

- Software requirements:
 - Java 11;
 - Server RabbitMQ
 - Cont CloudAMQP
 - Mediu de dezvoltare: IntelliJ
- Building requirements:
 - Scrierea credetialelor CloudAMQP în application.properties
- Execuție:
 - Se rulează fișierul Ds2020ProducerApplication

- **Client RPC:**

- Software requirements:
 - Java 11;
 - Mediu de dezvoltare: IntelliJ
- Building requirements
 - Depedentele si conexiunea Hessian
- Execuție:
 - Se rulează fișierul Ds2020Aplication