IMMPlah

Documentation

Assignment 1,2,3

Nume : Emanuel Golban

Grupa: 30245

1. REQUIREMENTS

The first module of the system consists of an online platform designed to manage patients, caregivers and medication. The system can be accessed by three types of users after a login process: doctor, patient and caregiver.

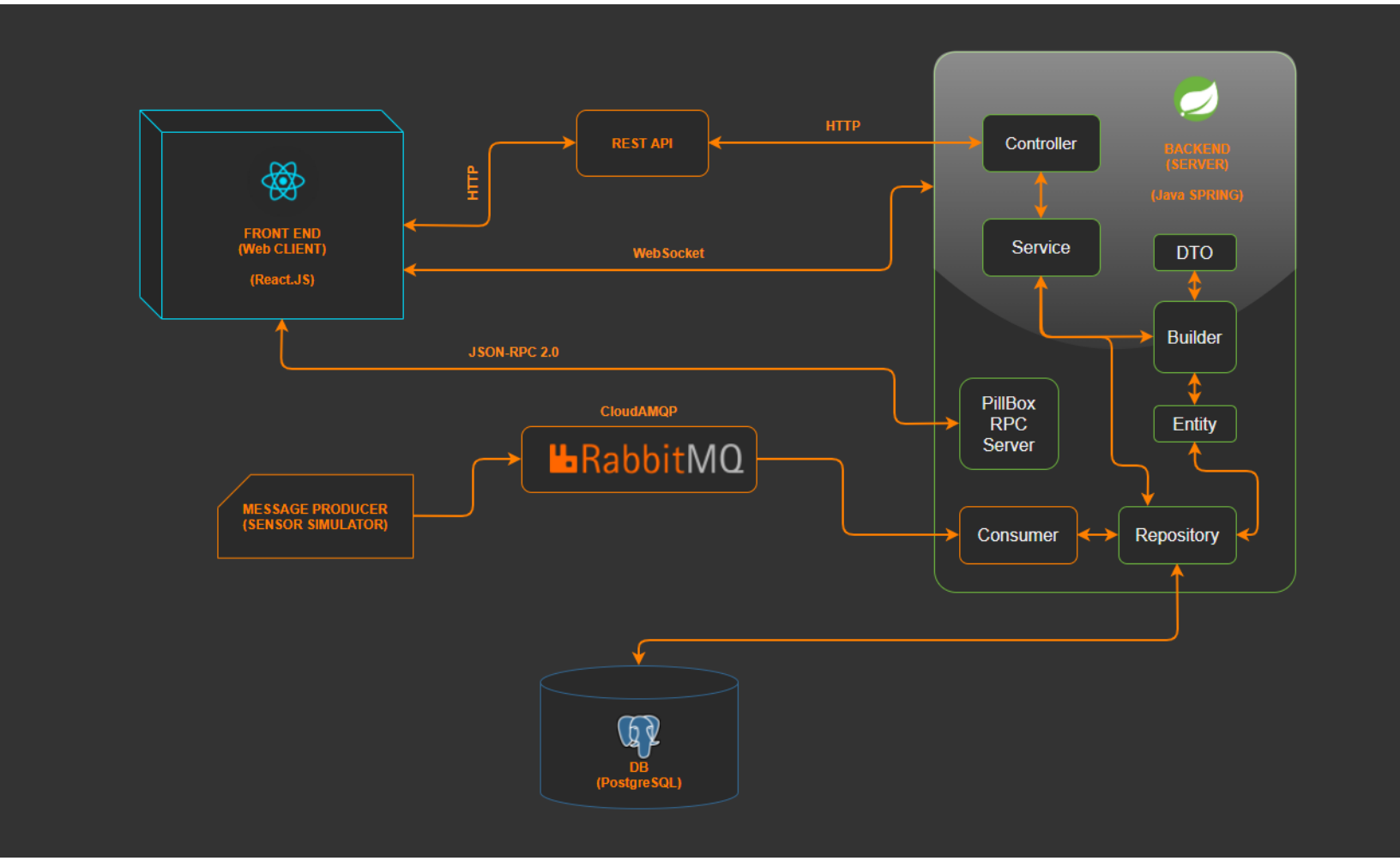
The doctor can perform CRUD operations on patient accounts (defined by ID, name, birth date, gender, address, medical record) caregiver accounts (defined by ID, name, birth date, gender, address, list of patients taken care of) and on the list of medication (defined by ID, name, list of side effects, dosage) available in the system.

The medical record of a patient must contain a description of the medical condition of the patient.

Furthermore, the doctor can create a medication plan for a patient, consisting of a list of medication and intake intervals needed to be taken daily, and the period of the treatment. The patients can view their accounts and their medication plans.

The caregivers can view their associated patients and the corresponding medication plans

2. SYSTEM ARCHITECTURE



A. FRONT END

Aceasta este componenta cu care interactioneaza user-ul si a fost implementata folosind Node JS si React.JS ca librarie JavaScript.

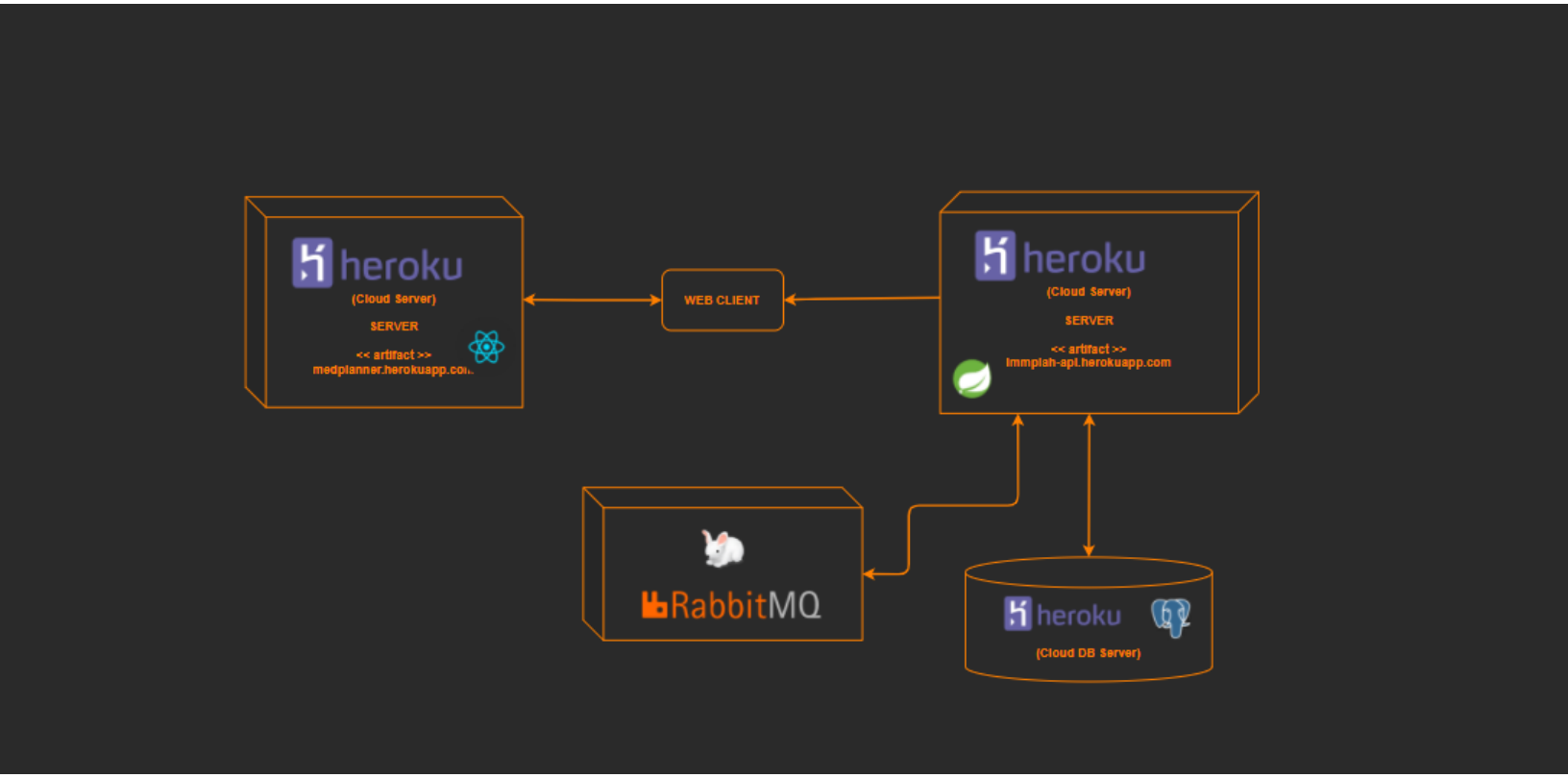
Componenta de Front end are rolul de a trimite request-uri catre server (folosind un API) pentru a trimite/primi informatii si a permite utilizatorului sa acceseze si sa modifice date de pe server.

B. BACK END

Componenta de back-end include urmatoarele:

- Entitati
  - Acestea formeaza model-ul aplicatiei si sunt folosite pentru a mapa relational obiectele cu tabelele din baza de date folosind Hibernate.
- DTO-uri
  - Obiectele folosite pentru transferul de date
- Buildere
  - Acestea fac conversia din entitati in DTO-uri si vice-versa.
- Repository-uri
  - Acestea sunt folosite pentru a accesa efectiv datele din baza de date.
- Controllere
  - Acestea gestioneaza request-urile primite de catre server si apeleaza controllerele aferente fiecarui tip de REQUEST.
- Servicii
  - Acestea realizeaza operatiunile de CRUD si sunt accesate de catre controllere.

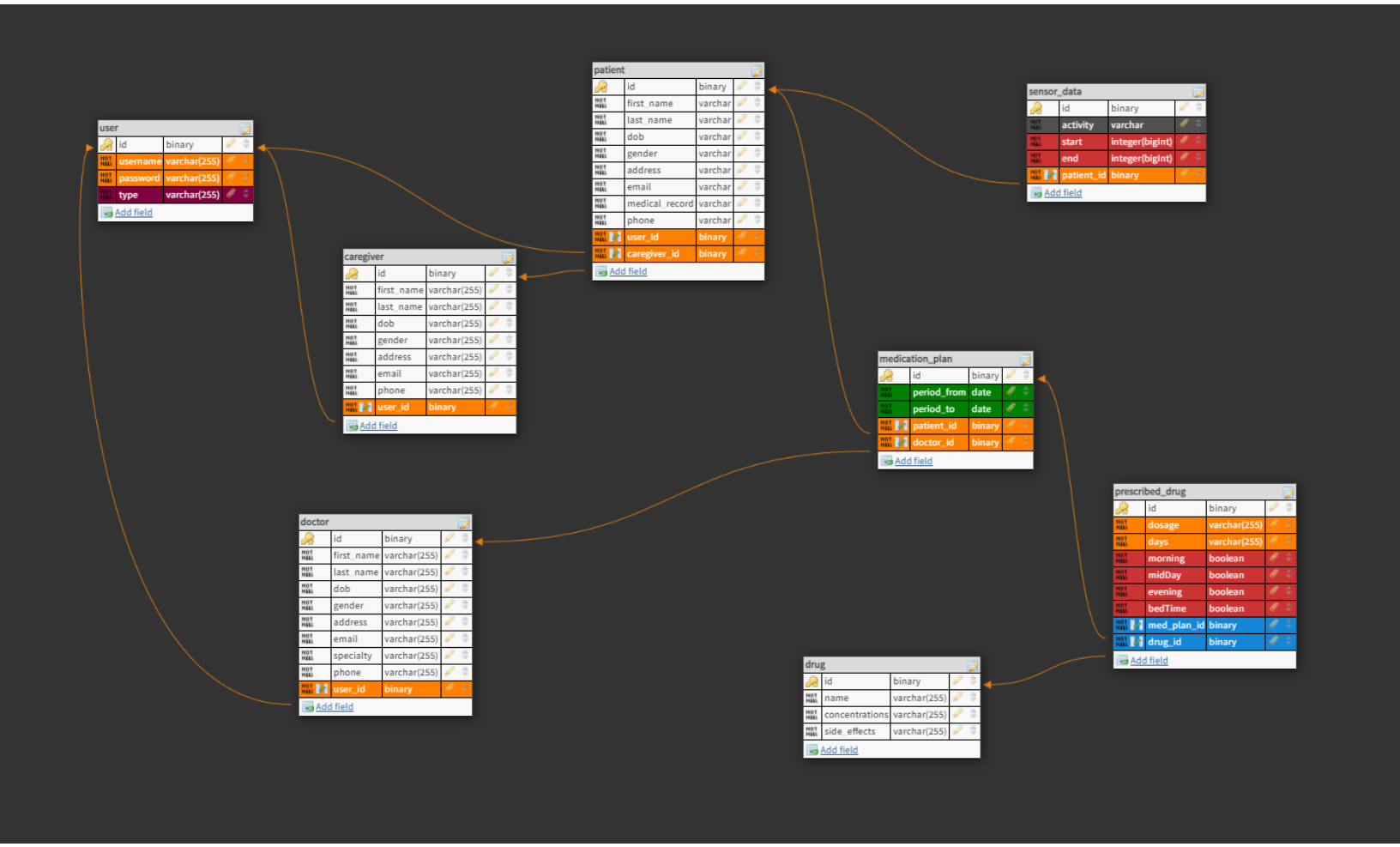
3. DEPLOYMENT



Deployment-ul s-a realizat folosind Heroku Cloud, atat pentru partea de back-end cat si pentru front-end.

Deployment-ul server-ului de RabbitMQ s-a realizat folosind CloudAMQP.

4.DATABASE


user
id binary
username varchar(255)
password varchar(255)
type varchar(255)
Add field
patient
id binary
first_name varchar
last_name varchar
dob varchar
gender varchar
address varchar
email varchar
medical_record varchar
phone varchar
user_id binary
caregiver_id binary
Add field
sensor_data
id binary
activity varchar
start integer(bigInt)
end integer(bigInt)
patient_id binary
Add field
caregiver
id binary
first_name varchar(255)
last_name varchar(255)
dob varchar(255)
gender varchar(255)
address varchar(255)
email varchar(255)
phone varchar(255)
user_id binary
Add field
medication_plan
id binary
period_from date
period_to date
patient_id binary
doctor_id binary
Add field
doctor
id binary
first_name varchar(255)
last_name varchar(255)
dob varchar(255)
gender varchar(255)
address varchar(255)
email varchar(255)
specialty varchar(255)
phone varchar(255)
user_id binary
Add field
prescribed_drug
id binary
dosage varchar(255)
days varchar(255)
morning boolean
midDay boolean
evening boolean
bedTime boolean
med_plan_id binary
drug_id binary
Add field
drug
id binary
name varchar(255)
concentrations varchar(255)
side_effects varchar(255)
Add field

5. RPC

Pe partea de server, am implementat serviciul de RPC folosind JSON-RPC 2.0 (jsonrpc4j http://solarcitynotes.blogspot.com/p/json-rpc.html) iar client-ul pentru modulul de pillbox a fost integrat in client-ul web. Apelul metodelor puse la dispozitie de service-ul RPC din backend sunt apelate folosind react-jsonrpc-client (https://www.npmjs.com/package/react-jsonrpc-client)

1. Interfata serviciului

- Am declarat in interfata metodele de care avem nevoie pentru a descarca planul pentru o anumita zi si pentru a anunta server-ul ca un anumit pacient a luat sau nu, un anumit medicament intr-un anumit interval

```
@JsonRpcService("/pillbox")
public interface MedPlanService {

    DayPlanDTO getTodaysPlan(@JsonRpcParam(value = "date") String todaysDate,
                             @JsonRpcParam(value = "patientId") UUID patientId);

    String markDrugAsTaken(@JsonRpcParam(value = "drug") String drug,
                           @JsonRpcParam(value = "patientId") UUID patientId,
                           @JsonRpcParam(value = "takenOn") String date,
                           @JsonRpcParam(value = "interval") String interval);

    String markDrugAsNotTaken(@JsonRpcParam(value = "drug") String drug,
                           @JsonRpcParam(value = "patientId") UUID patientId,
                           @JsonRpcParam(value = "takenOn") String date,
                           @JsonRpcParam(value = "interval") String interval);
}
```

2. Implementarea interfetei

- In clasa pentru service am implementat metodele descrise in interfata mentionata anterior.

```java
@Service
@AutoJsonRpcServiceImpl
public class MedPlanServiceImpl implements MedPlanService {

    @Autowired
    private SimpMessagingTemplate webSocket;

    private final MedicationPlanService medicationPlanService;

    @Autowired
    public MedPlanServiceImpl(MedicationPlanService medicationPlanService) {
        this.medicationPlanService = medicationPlanService;
    }

    @Override
    public DayPlanDTO getTodaysPlan(String todaysDate, UUID patientId) {
        return medicationPlanService.getDayPlan(todaysDate, patientId);
    }

    @Override
    public String markDrugAsTaken(String drug, UUID patientId, String date, String interval) {
        System.out.println(String.format("[%s] Patient with ID:[%s] has taken [%s] at [%s]", date
        return "Drug was successfully marked as TAKEN!";
    }

    @Override
    public String markDrugAsNotTaken(String drug, UUID patientId, String date, String interval) {
        System.out.printf("[%s] Patient with ID:[%s] DID NOT TAKE [%s] at [%s]%n", date, patientI
        webSocket.convertAndSend( destination: "/topic/notifications", String.format("[%s] Patient [
        return String.format("Caregiver of PATIENT with ID:[%s] has been alerted!", patientId);
    }
```

3. Service exporter

- In aceasta clasa, instantiem exporter-ul care va expune serviciul de RPC pe endpoint-ul „/pillbox”

```
@Configuration
public class PillboxServer {

    @Bean(name = "/pillbox")
    public static AutoJsonRpcServiceImplExporter autoJsonRpcServiceImplExporter() {
        AutoJsonRpcServiceImplExporter exporter = new AutoJsonRpcServiceImplExporter();
        return exporter;
    }

}
```