MINISTRY OF NATIONAL EDUCATION

TECHNICAL UNIVERSITY OF CLUJ-NAPOCA ROMANIA

Distributed Systems

Assignment 1

Online Medication Platform

Student: Horatiu-Stefan Spaniol

Group: 30442

Table of contents

Requirement Analysis……………………………………………………………………………………3

Project Conceptual Architectural…..………………………………………………………………………..4

Project Deployment Diagram……………………………………………………………………………….5

Database Design…………………………………………………………………………………………….6

Build and Execution Considerations………………………………………………………………………..7

Bibliography…………………………………………………………………………………………………..7

Requirement Analysis

Assignment Specification 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.

Functional Requirements

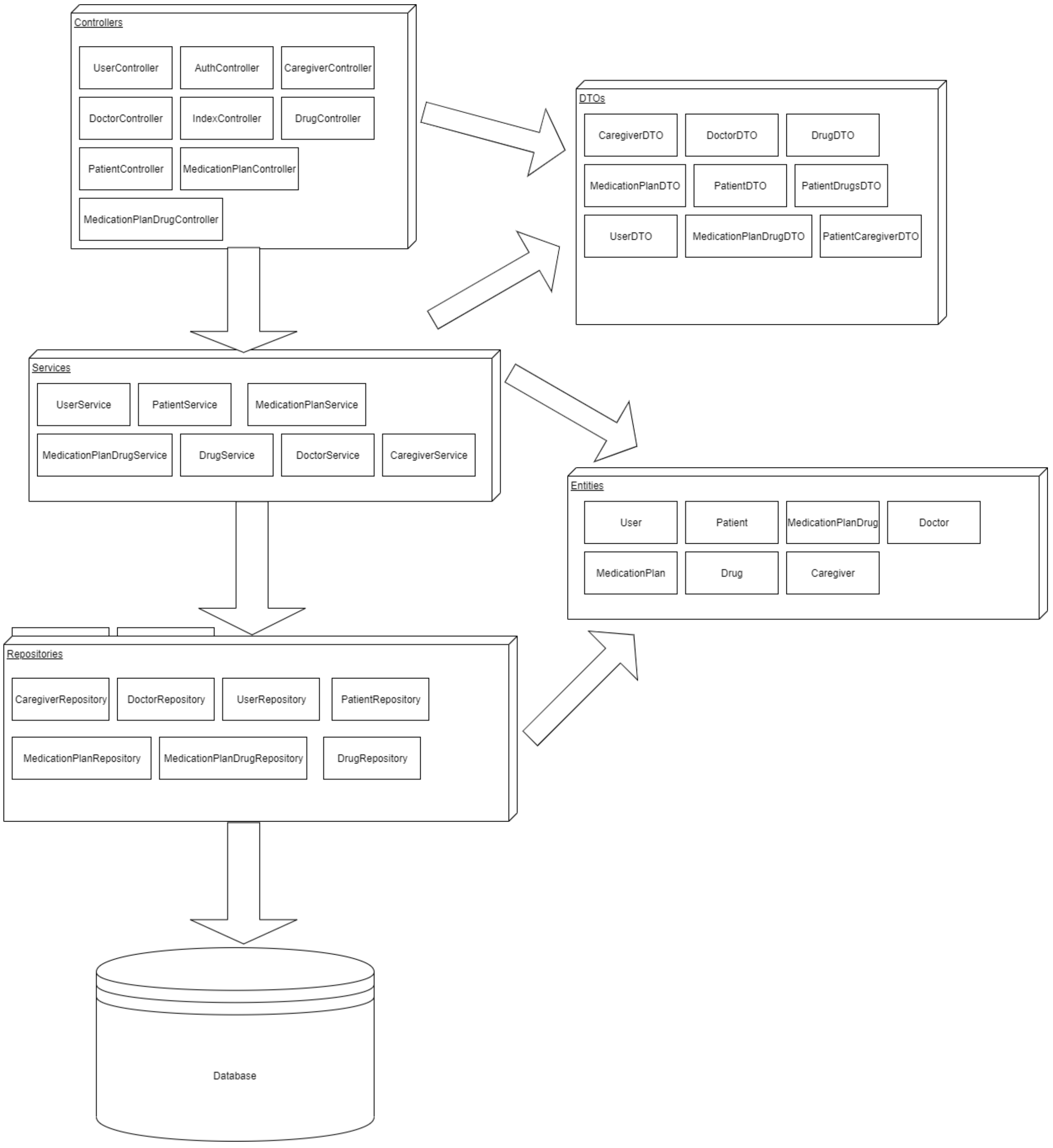
- Users log in. Users are redirected to the page corresponding to their role.
- The users corresponding to one role will not be able to enter the pages corresponding to other roles(e.g. by log-in and then copy-paste the admin URL to the browser)
- Patient Role
    - A patient can view in his/her page the accounts details and the medication plans prescribed by the doctors
- Caregiver role:
    - A caregiver can view on his/her page all the patients associated in a list or table.
- Doctor Role:
    - CRUD operations on patients
    - CRUD operations on caregivers
    - CRUD operations on medication
    - Create 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

Non-functional Requirements

- Security: use authentication in order to restrict users to access the administrator pages (cookies, session, etc.)

Project Conceptual Architectural

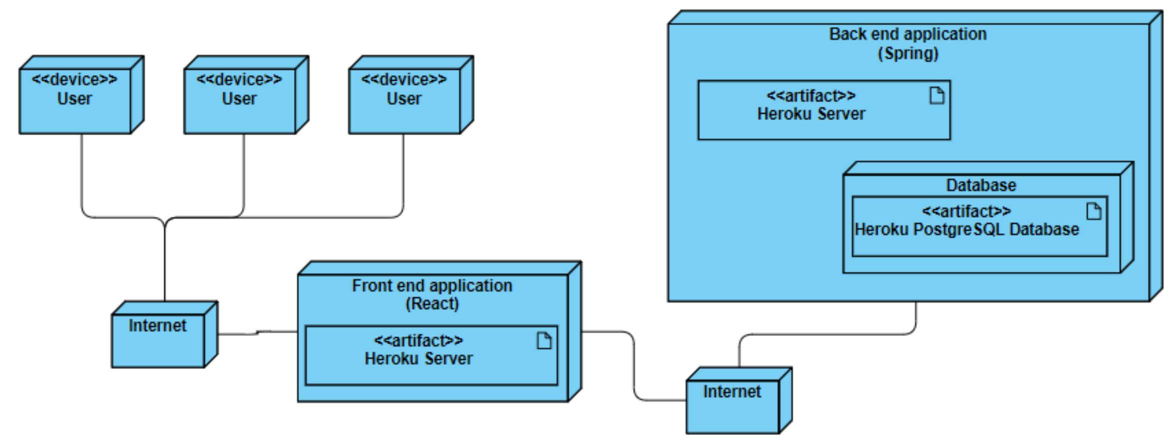
The project conceptual architecture followed was the one provided in the requirements.



As we can see in the figure above, the architecture is very straight-forward.

The controllers receive information as requests or DTOs; if needed, they transform these requests into DTOs to be sent to the services, which in turn transform the DTOs into entities to be sent to the repositories that access the database. Services also sometimes receive information directly from the controllers as requests, without transforming these requests into DTOs beforehand. Repositories return entities to the layer above, Services, which transform the entities into DTOs or into some other custom response to be sent back to the Controllers.

Project Deployment Diagram

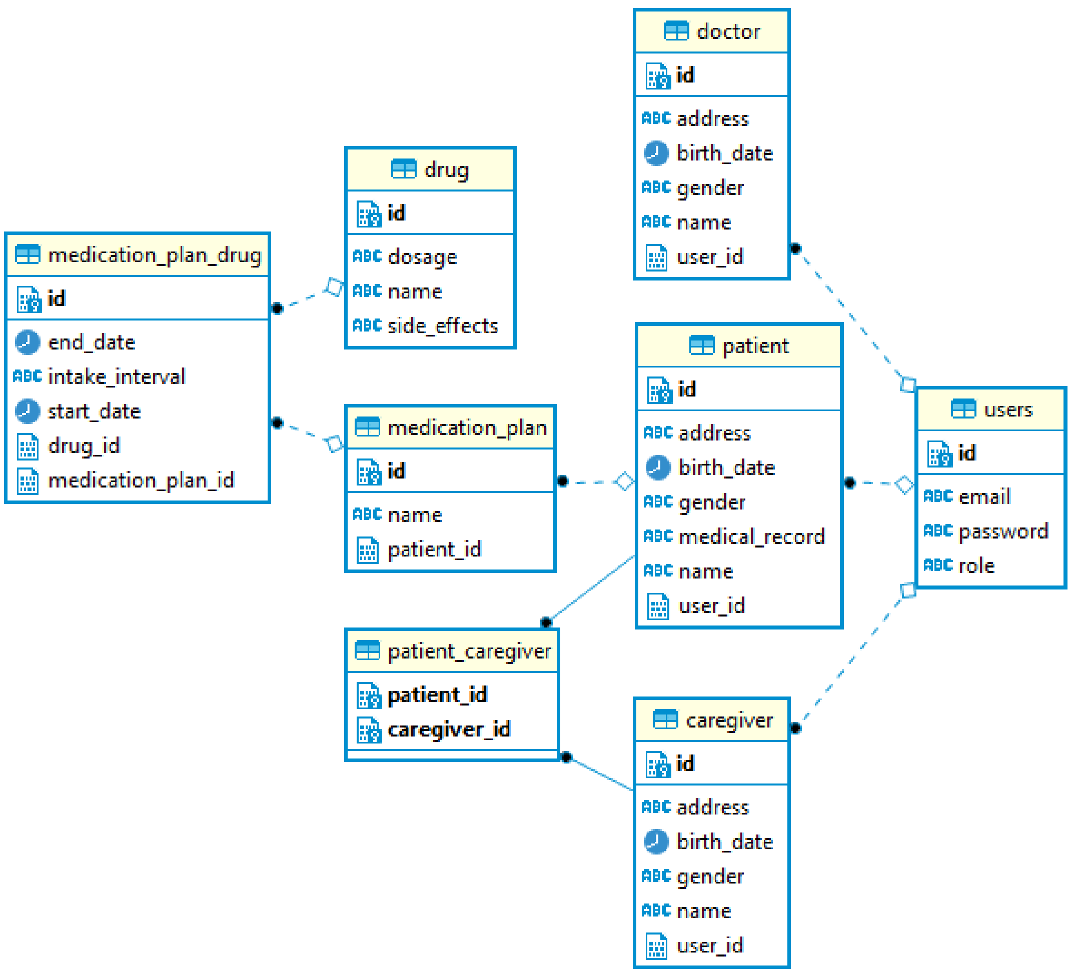


The deployment diagram shown above presents the physical relation between the applications that are deployed on nodes.

The user node symbolizes the fact that the user can use any device to access the application(i.e. computer, mobile phone, tablet); however, this device it can be seen, this device would have to communicate through the Internet node with the Front end application, which is deployed on a Heroku server. This application in turn, communicates through the Internet with the Back end application that runs on another Heroku server. The back end application also has a Heroku PostgreSQL Database, which was added prior to deploying the Backend application on that respective Heroku server.

Database Design

The database design process went through 5 versions, and the one below is the final version.



The “users” table contains 3 string fields that hold an email, password and the role of the user. This role is either a patient, caregiver or doctor.

The “doctor” table has account information of the doctor, and a one-to-one relation to the “users” table, indicating that a doctor has a user account associated to it and that user account belongs to just that one doctor.

The same principle applies to the “patient” and “caregiver” tables, which contain personal account information alongside the foreign key to the “users” table. However, the “patient” table and “caregiver” table are also in a many-to-many relationship, which can be seen illustrated by the “patient_caregiver” table. This depicts the fact that a patient can have many caregivers associated to them, and a caregiver can also have many patients associated to them.(i.e. Patient P1 has associated Caregivers C1, C2 and Patient P2 has associated Caregivers C2, C3, which leads that Caregiver C2 has associated Patients P1 and P2).

The “patient” table also has a one-to-many relationship with the “medication_plan” table, as one patient can have many medication plans, but a medication plan can only belong to one patient. This medication plan contains the name of the specified medication plan, and a list of drugs that need to be taken(and more information about the way this intake should be done) by the patient associated with the medication plan.

The drugs table is a standalone table that contains drugs, with relevant information such as dosage and side effects.

The table “medicationplan_drug” is another many-to-many relationship between the “medicationplan” table and the “drugs” table, as it contains a foreign key towards the mentioned drug and a foreign key to the mentioned medication plan. It also contains extra information about intake intervals and the period of the intake. This table was added because intake intervals is not the same for different people even though the drug is the same.

Build and Execution Considerations

In order to build and execute this project locally, there are some prerequisites. The first one of these, is have a Java JDK installed. The project was built with Java 11, so at least 11 is necessary to correctly build and execute the project. The second prerequisite would be the pgAdmin application for creating and connecting to a PostgreSQL database locally and to also host this database server. The third prerequisite would be a web browser, be it Microsoft Edge, Google Chrome, Mozilla Firefox or any other browser. The fourth and final prerequisite would be the Intellij IDEA application, which has the Java JDK mentioned beforehand associated with it.
In order to deploy the application, GitLab’s Continuous Integration/Continuous Development pipelines and the Heroku cloud were used, and the application can be found on the following links:

Backend: https://ds2020-spaniolhoratiu-1.herokuapp.com

Frontend: https://ds2020-spaniolhoratiu-1-front.herokuapp.com

Bibliography

https://gitlab.com/ds_20201/react-demo

https://gitlab.com/ds_20201/spring-demo

https://bezkoder.com/spring-boot-react-jwt-auth

https://en.wikipedia.org/wiki/Deployment_diagram