DISTRIBUTED SYSTEMS

Assignment 1

Request-Reply
Communication Paradigm

# Online Medication Platform

Chișluca Miruna

30243

# Project 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.

## 1. Conceptual architecture of the distributed system

The system implemented for this assignment is a client-server system, consisting of an .Net Web API (server application) and a React single-page application (client application). These two applications communicate through the request-reply paradigm, meaning that the client app sends requests to the server application which, as response, sends replies.

The API application implements the layered architecture pattern. The database layer consists of a generic repository that performs data access operations on the PostgreSQL database. The repository encapsulates the operations performed on the database (insert, update, delete, get by id). The database repositories are included in a unit of work which provides the ability to save the changes made to the database.

The business layer contains services that implement concrete methods needed for the business logic, making use of the unit of work and the generic repository methods. The presentation layer contains controllers that will receive http requests from the client and in exchange they will send a response containing the needed information.

The ORM used to create the database was Entity Framework. Each table corresponds to the entities defined in the application. Besides the entities defined, there are also some models defined in the application that include only the necessary information to be displayed to the client application. The mapping between the actual database entity and the object models sent as response is made with the use of AutoMapper.

The client application is a React single-page application. The client application has 3 main routes for the 3 roles that exist in the application (doctor, caregiver and patient), a login route, and 3 more routes for the doctor page, namely patient details, caregiver details and medication details.

The login page contains 2 fields for username and password and the login button. After successfully logging in, you are redirected to the corresponding page according to the role.

The doctor homepage contains 3 tabs: Patients, Caregivers and Medication. The Patients tab displays a table with general information about the patients associated to the doctor that is logged in. The Caregivers tab displays a table with general information about all the caregivers existent in the database. The Medication tab displays a table with information about the medication existent in the database. Clicking on a row of any of these 3 tables redirects the user to a page showing the corresponding details. For a patient, the page displays the general information and a table with the medication plans associated to the patient; for a caregiver, the page displays the general information and a table of the associated patients; for medication, the page displays the information about the medication. Each of these 3 details pages contains a button for editing and a button for deletion. In addition to the 2 buttons, the patient details page also

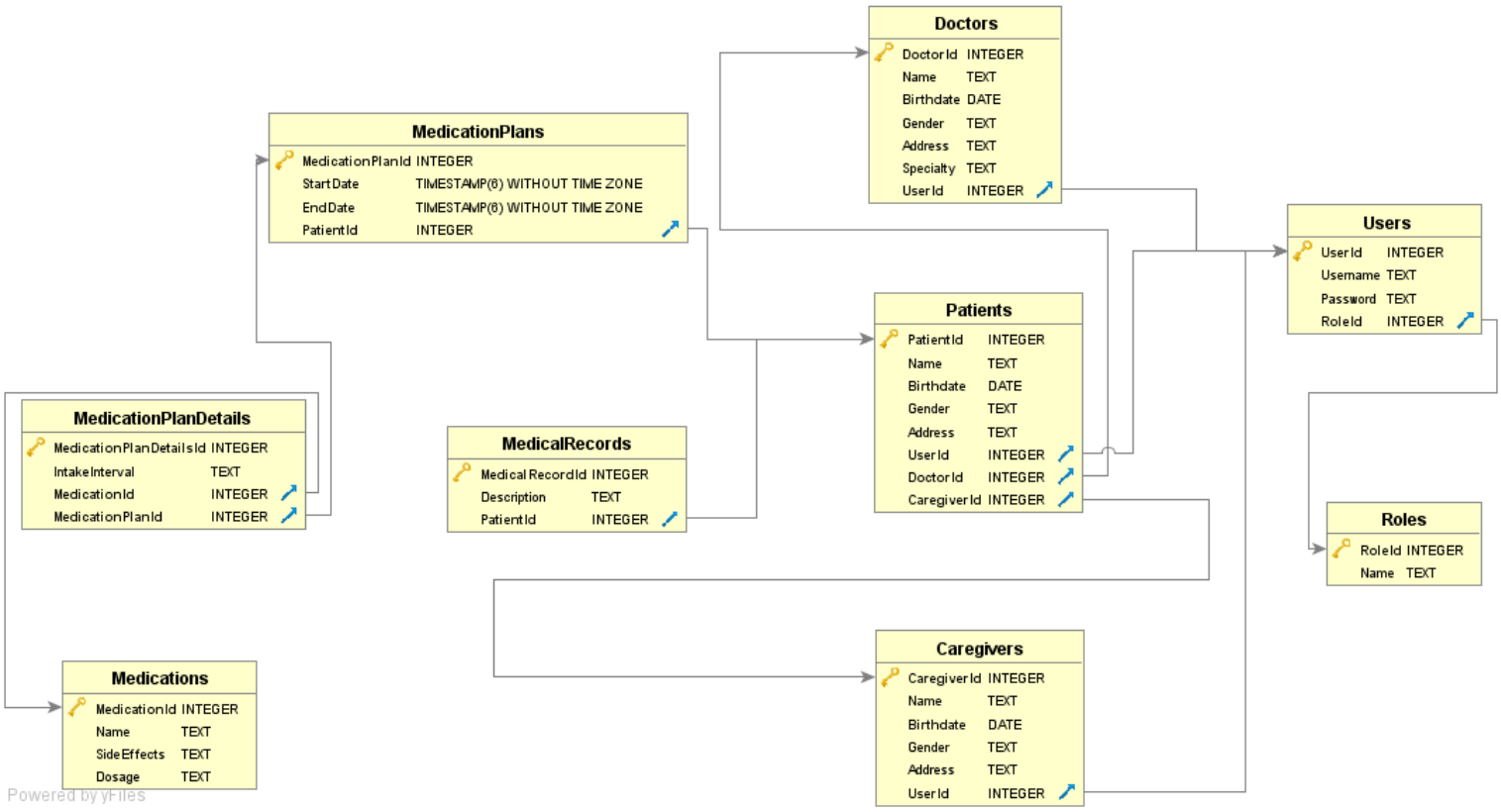
has a button for adding a new medication plan. The doctor homepage has in each of the 3 tabs a button for adding a new patient, caregiver or medication.

After logging in as a caregiver, the page displayed contains a table of all the patients associated to the caregiver. The table has the same functionality as the patients table displayed in the doctor page, meaning that it displays information about the patient and their medication plans when clicking a row.

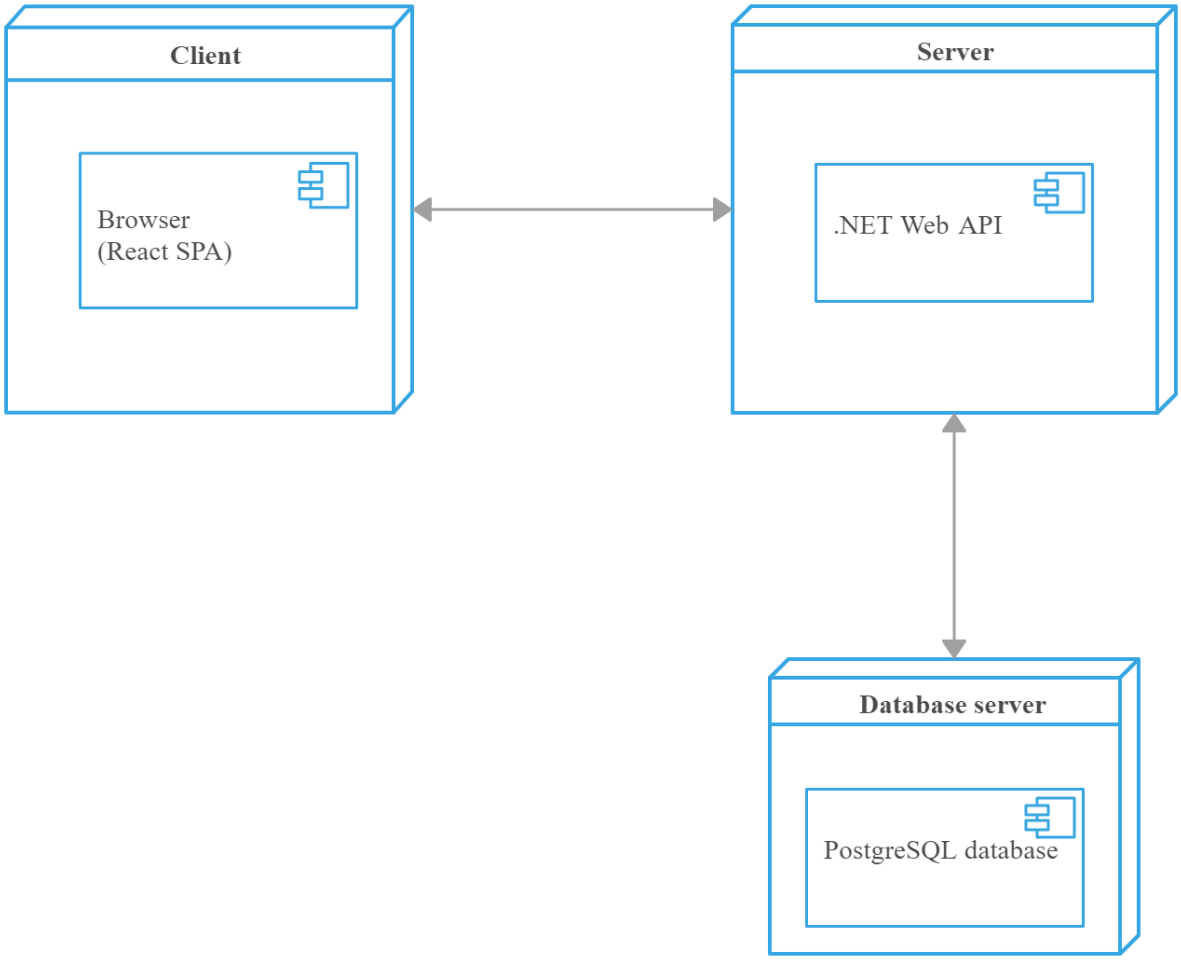
The page for the patient role contains 2 tabs. The first tab show information about the patient and the second tab contains a table with all the medication plans associated to the patient.

Regarding the security of the application, authentication is made based on tokens. When a user logs in, the request made to the server gets a response containing a JWT token, considering that the login credentials are correct. This token is stored in the local storage of the client application. The token is created in the API and contains the user id, role id and the corresponding id for doctor, caregiver or patient, depending on their role. This token is then used in the client application to redirect the user to the page corresponding to his role. Also based on this role id, the user is not allowed to navigate to other pages of the application that are not associated with their role.

## 2. Database design



## 3. UML Deployment diagram




## 4. READ ME

The application can be found here: https://medplatformapp.herokuapp.com/

Some doctor accounts are:

- Username: doctor1 / doctor2 / doctor3
- Password: abcd123

Caregiver accounts:

- Username: caregiver1 / caregiver2 / careviger3
- Password: abcd123

Patient accounts:

- Username: patient1 / patient2 / patient3
- Password: abcd123