Rehabilitation information system

Technical solution description

Author: Valeriia Pegushina

Deutsche Telekom IT Solutions (T-Systems)
2021

Content

Task

First Part

It is necessary to implement an application that simulates the work of an information system for automating the workflow of a medical rehabilitation institution. The subject area and technical requirements are described in more detail below:

For doctors:

1. Adding a patient
2. Patient discharging
3. Prescribing procedures and medications
4. Editing prescriptions
5. Cancelling prescriptions

For nurses:

- View all treatment events.
- Filtering treatment events by date (for today, for the next hour) and by patient.
- Changing the status of treatment events from “in plan” to “completed” and from “in plan” to “cancelled”.

As a result, a multi-user application of the client-server type with a network connection should be developed.All data is stored on the server side. Each client can download some data, after each change operation, the data must be synchronized with the server. The application must handle hardware and software errors.

Second Part

A separate client application for the electronic scoreboard should be implemented. The app should display a list of all events scheduled for the current day. The data must be loaded at startup and stored on the client side. Data reloading is carried out in case of receiving a notification from the server about changes in the list of events (new ones are added or old ones canceled).

Used Instruments and Technologies

IDE: IntelliJ IDEA 2020 (Ultimate Edition)

Project build management tool: Apache Maven

Application Server: Wildfly 22.0.0 Final

Servlet Container: Tomcat 8.5.58

Database: PostgreSql 12

Testing instruments and libraries:

- Junit 4
- Mockito

Backend:

- Apache ActiveMQ
- Jackson
- JPA
- JSP
- JSF
- EJB
- Spring Boot
- Hibernate
- JavaMail
- REST
- Spring Security

Frontend:

- Bootstrap
- HTML/CSS
- JavaScript
- JQuery
- Primefaces

Archeticture

First application

The application architecture is based on the implementation of the MVC design pattern**:**

- The Model provides data and reacts to controller commands by changing its state.
- The View is responsible for displaying model data to the user in response to model changes.
- The Controller interprets the user's actions, notifying the model about the need for changes.

The structure of the application is shown on the picture below:

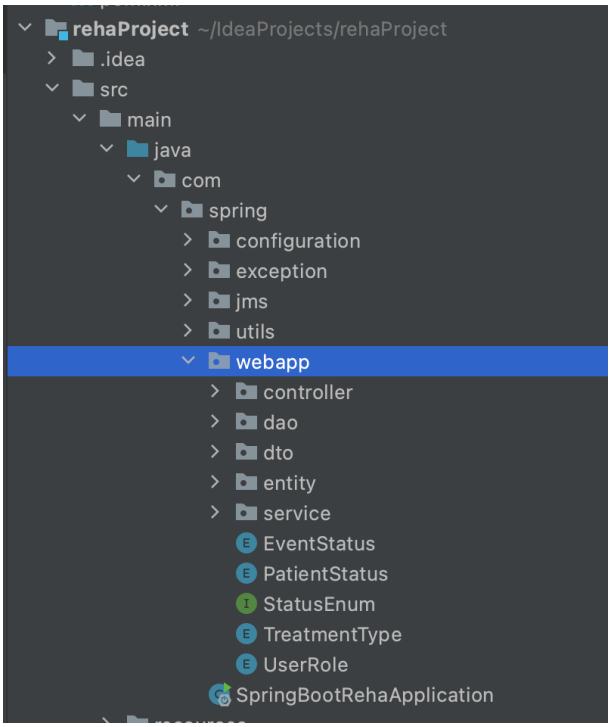

Figure 1

Controllers layer

Figure 2 shows the controllers. Access to them and to a specific endpoint depends on the user's role. This layer uses DTO to interact with the view layer. The controller passes the A Data Transfer Object to the service, where the service layer decides how to act with this object, and it also gives the DTO back to the controller.

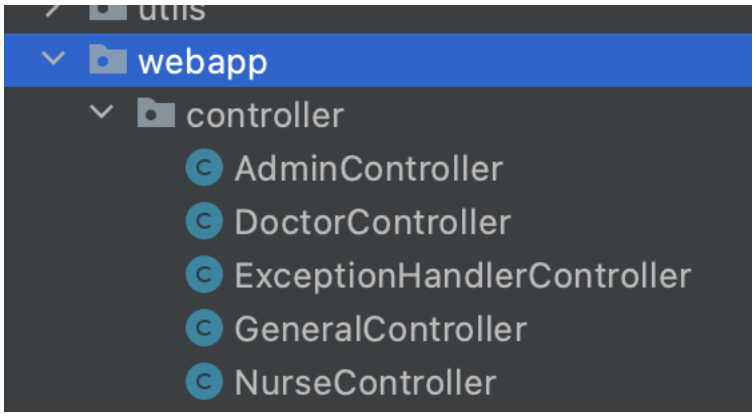

Figure 2

Model layer

Figure 3 shows the controllers. For each DTO there is a corresponding Entity(Figure 5). DAOs (Figure 4) are developed for each entity that interact with the database.

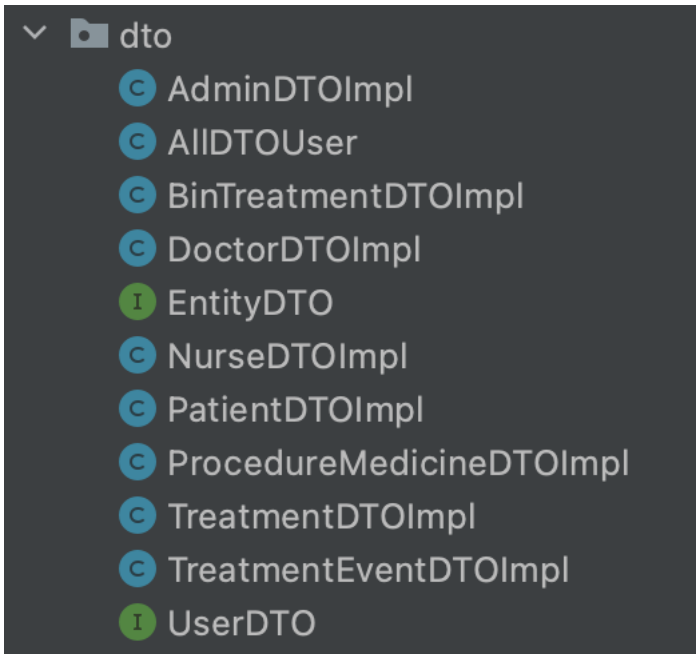

Figure 3

DAO layer

Persistence level (on the application side) is realized using Hibernate.

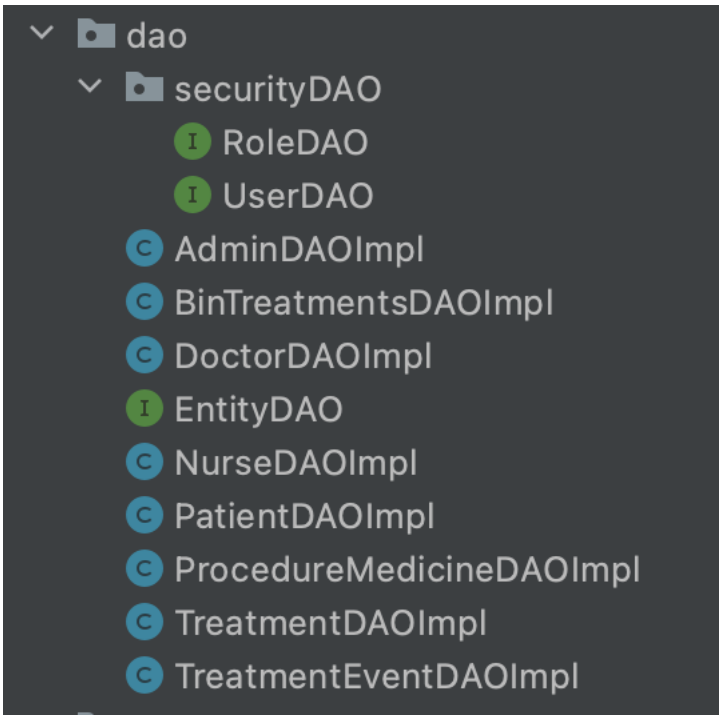

Figure 4

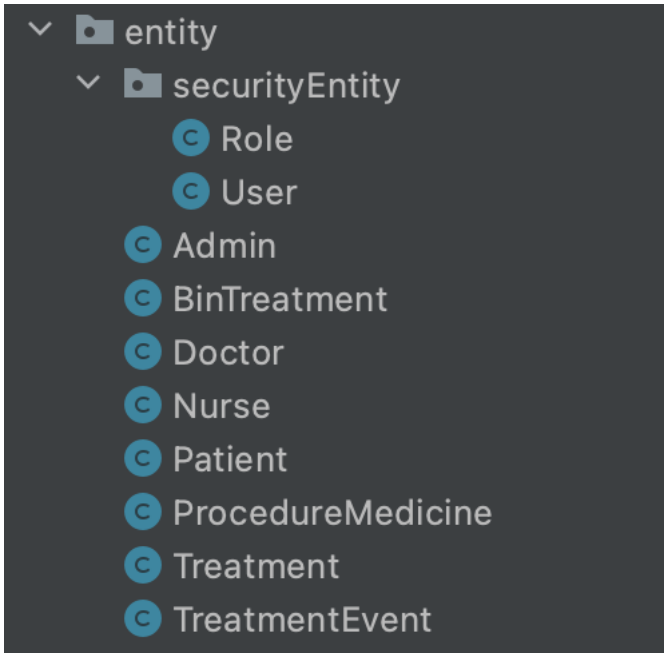

Figure 5

Service layer

Each service is responsible for the business logic of the entities. For example, TreatmentServiceImpl is responsible for interacting with the entity and DTO, and PatientServiceImpl, in addition to its entities, implements business logic using other services.

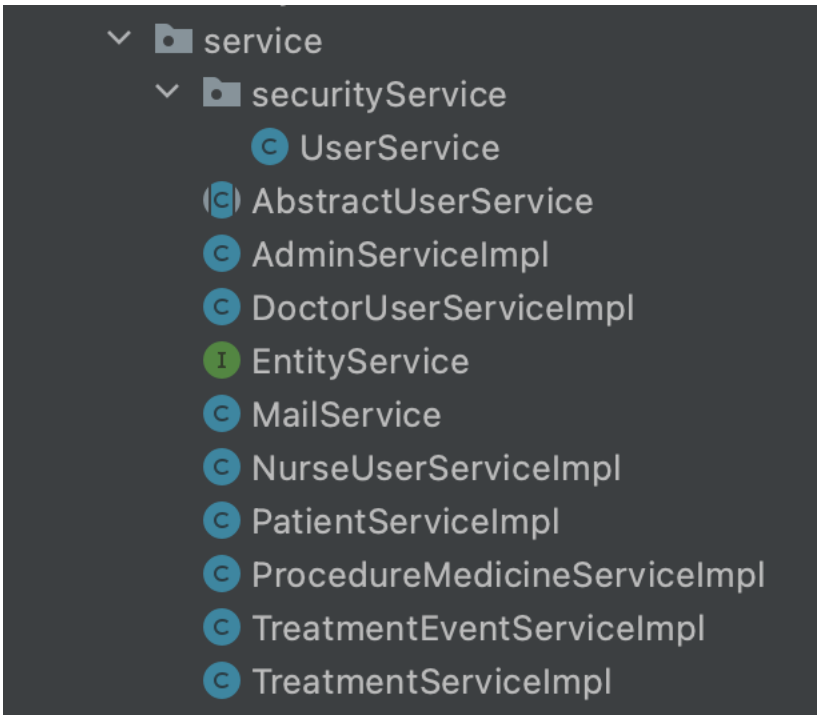

Figure 6

View layer

This layer contains jsp-files. Taglibs were also used in the work of jsp pages - JSP Standard Tag Library (JSTL) and Spring tags. Also were used Bootstrap, CSS, JQuery and JavaScript.

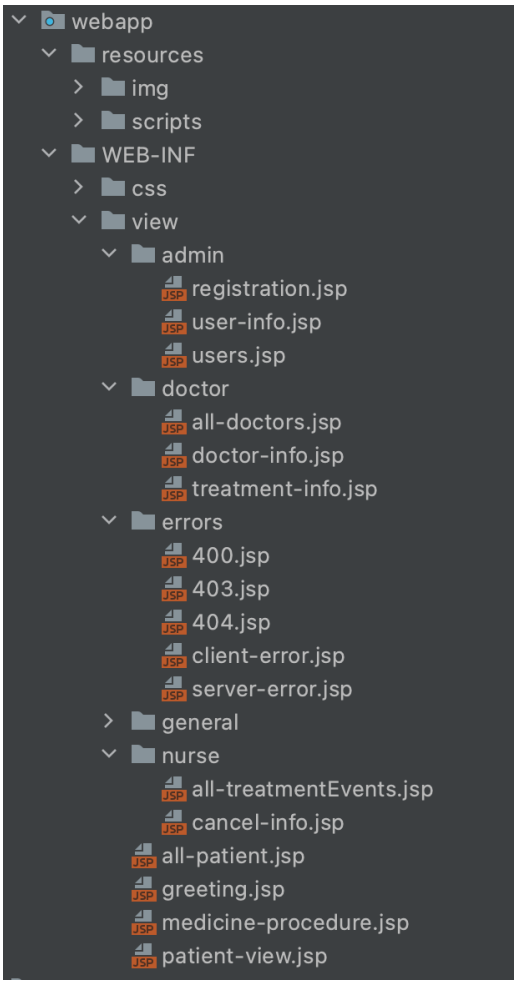

Figure 7

Second application

Second application is a one-page application based on EJB and JSF frameworks. JmsMessageTreantmentEvent is a managed bean used as model for keeping events data. JSF page is used as view for representing data. EventService helps to get json data from endpoint. JmsListener literally listen messages from ActiveMq and gives instructions to EventBean that it need to update page.

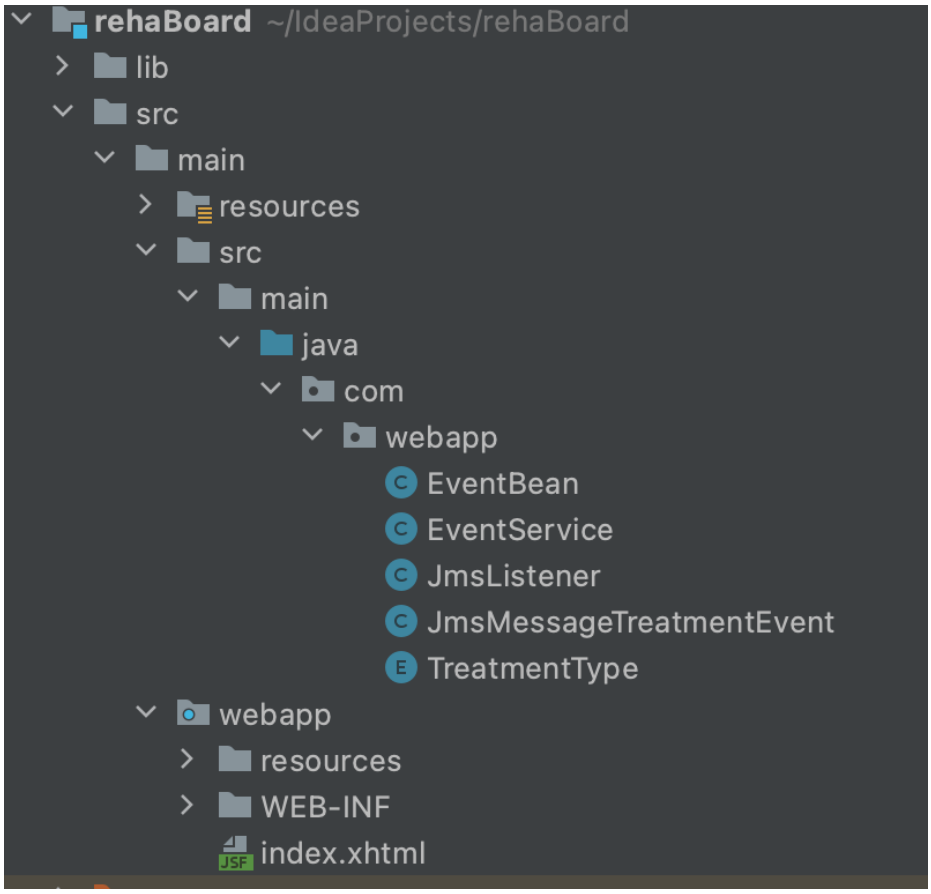

Figure 8

Database schema

Figure 9 shows database structure. Nurses, admins and doctors entities are connected with users through usernames. It considers in service's layer.

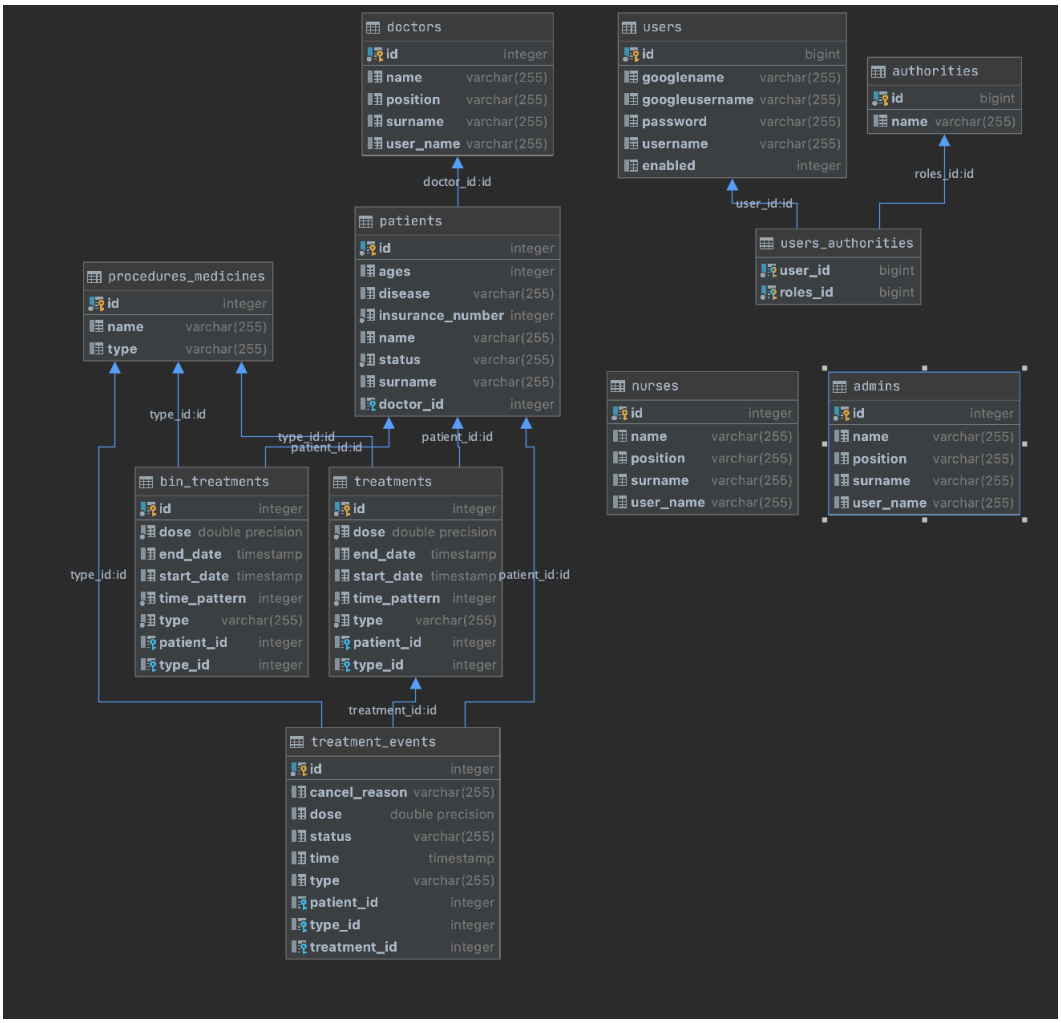

Figure 9

Table name	Description
doctors	Stores data about doctor(about personal information, username and patients)
nurses	Stores data about nurses.

admins	Stores data about admins.
patients	Stores data about patient and treatments.
treatments	Stores data about patient's prescription.
treatment_events	Stores patient's treatment event(event time, dose, name of medicine and procedure and so on).
procedure_medicine	Stores information about medicine and procedures.
bin_treatments	Stores data about previous patient's prescription.

User cases

Administrator side

This application gives the administrator the ability to perform the following actions:

- add, delete and edit users
- add, remove and edit patients
- add, remove and edit treatments
- to assign new doctors to patients

Figure 10 shows admin's main page.

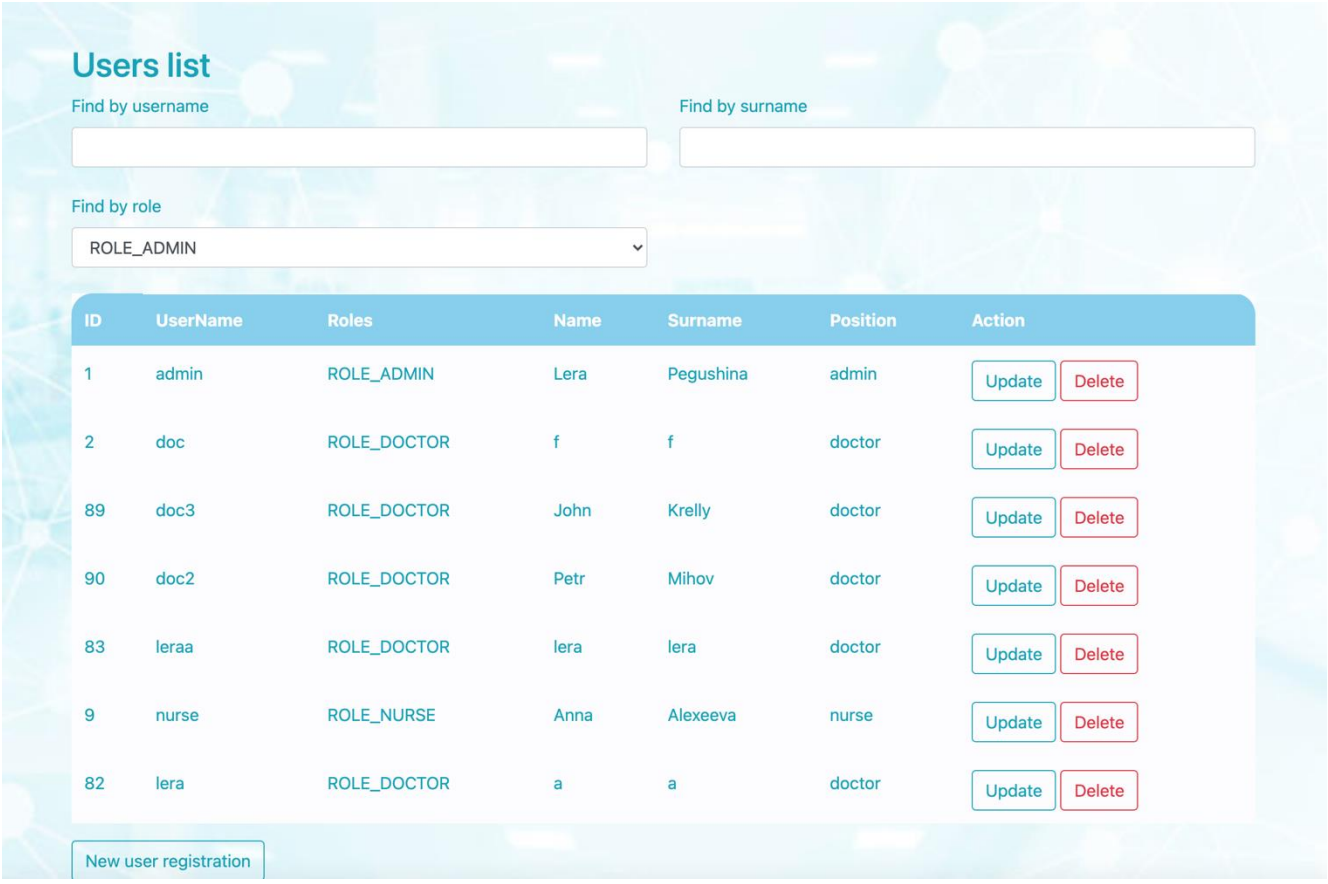

Figure 10

If the administrator deletes a doctor, the patient remains without a doctor and the administrator has the opportunity to reassign a doctor to this patient. Also, the administrator has access to all functions and pages that are only for nurses or doctors.

When administrator adds new user with email, for this user sends email with password and username.

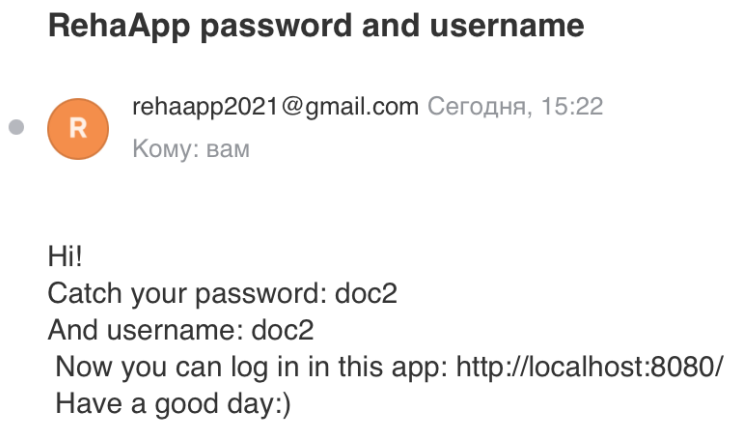

Figure 11

Doctor side

A doctor has a possibility to:

- add and edit patient and patient info
- add, edit and remove treatment

Figure 12 shows doctor's main page.

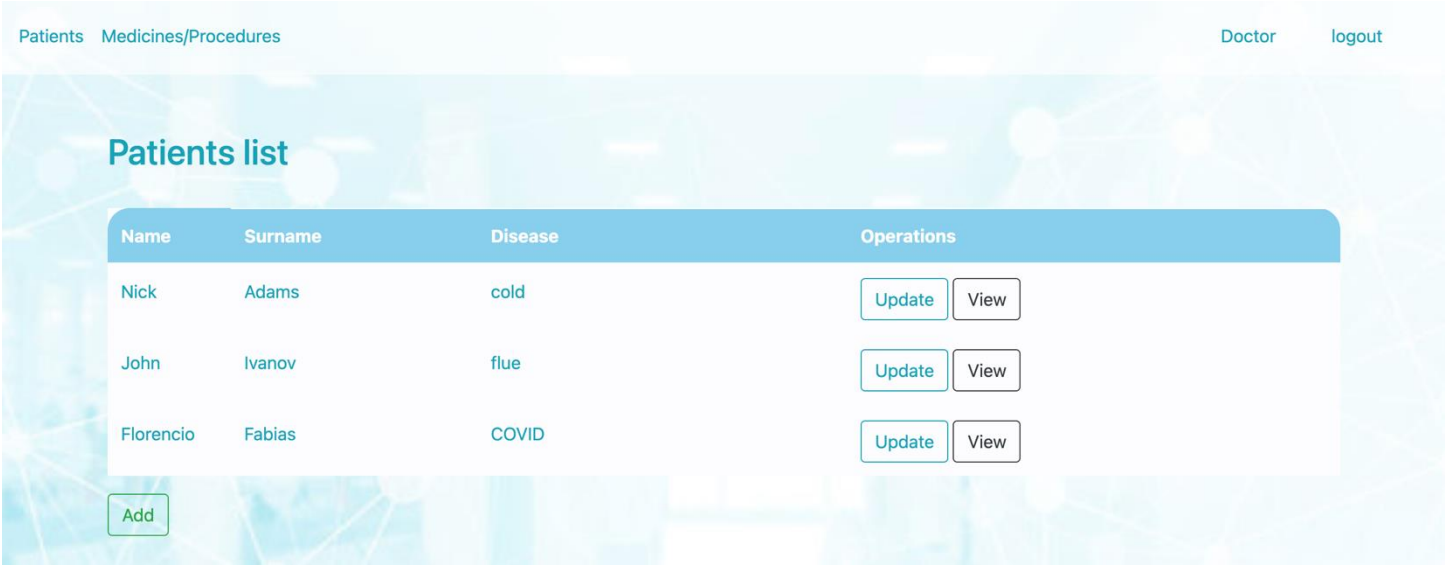

Figure 12

When doctor wants to add or edit information about patient or treatment, he gets page (figure 13,14), where he can whatever it needed.

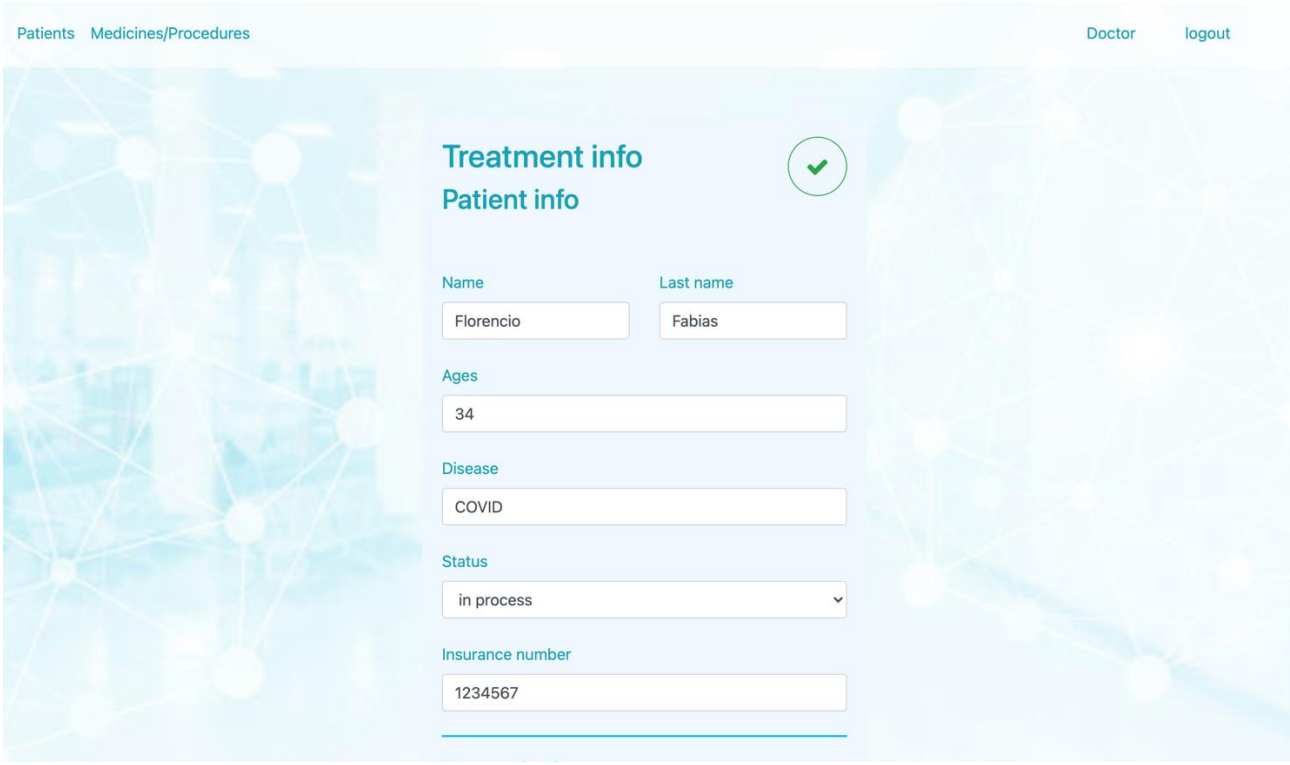

Figure 13

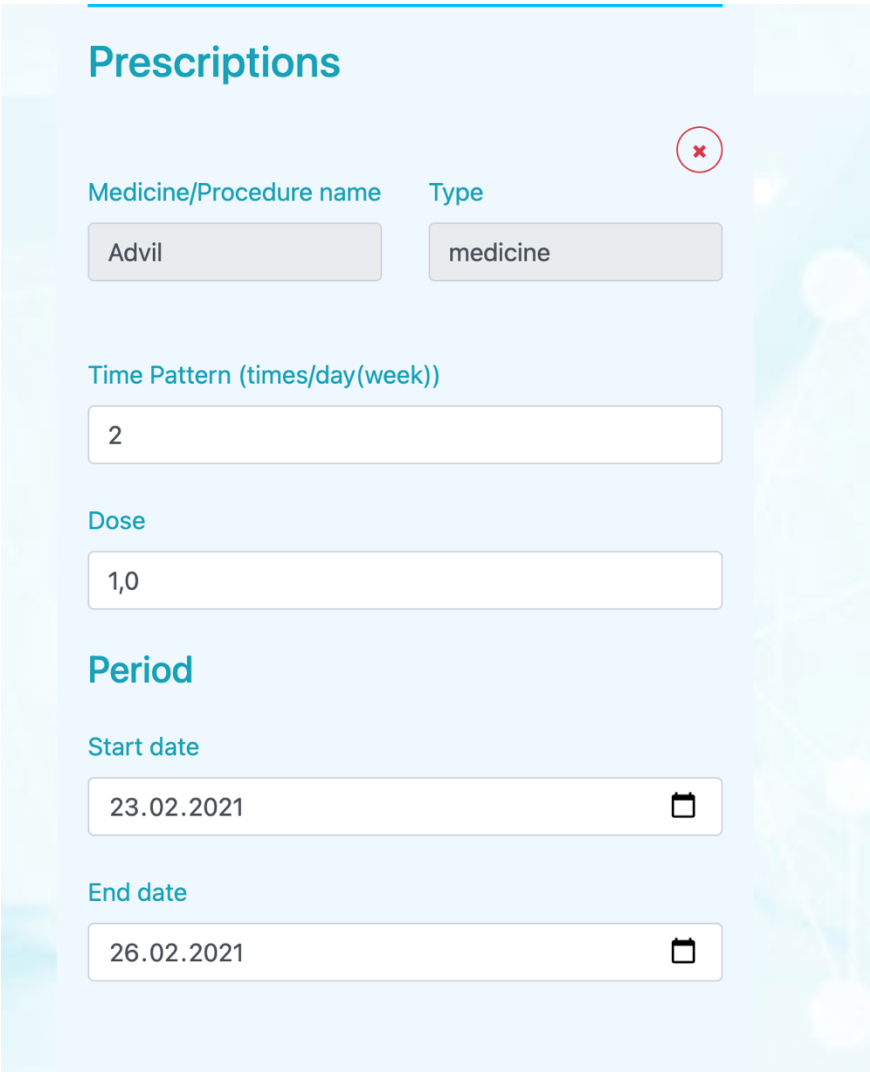

Figure 14

Also, after treatment removing(canceling), we still observe this treatment, but in bin section (Figure 15).

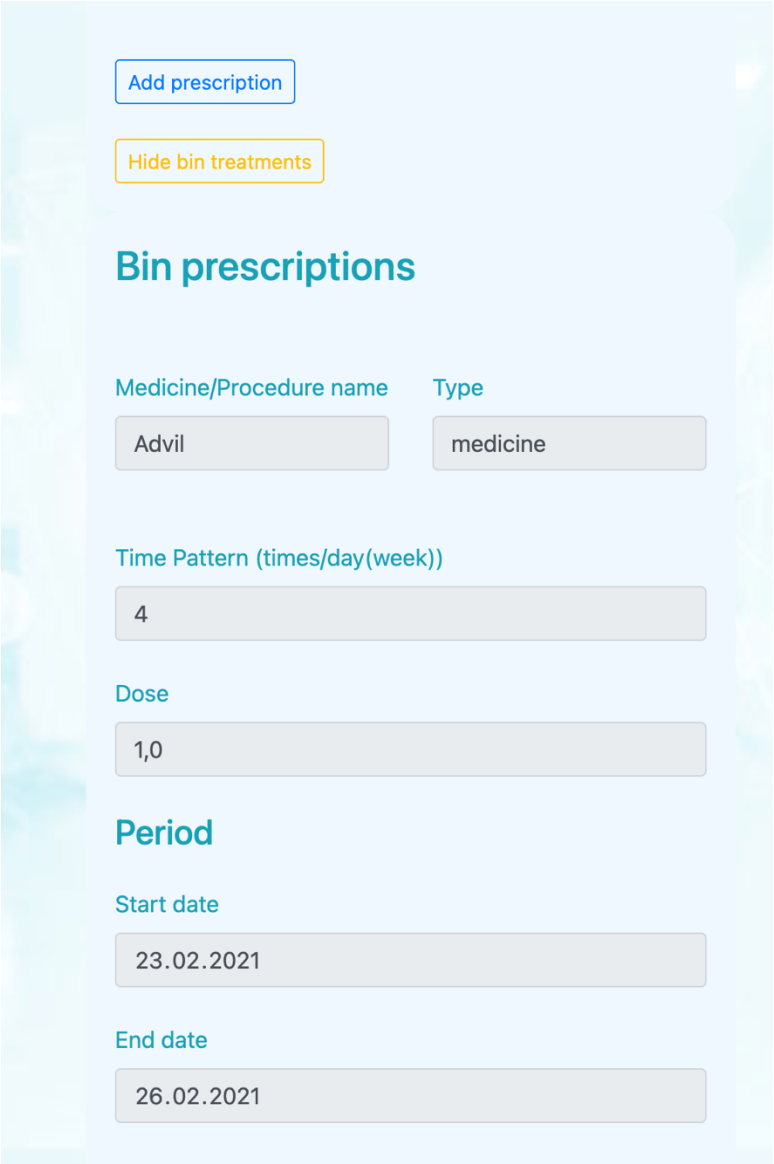

Figure 15

Nurse side

A nurse has a possibility to:

- view treatment's events;
- filter treatment events by any date, for today, for nearest hour, by treatment type and by patient's surname;
- change the status of events from "in plan" to "completed" and from"in plan" to "canceled", when she cancels event, it necessary to input canceling reason;

If the nurse did not notice the event in time, the row with this event in the table will be highlighted in gray.

Figure 16 shows nurse's main page.

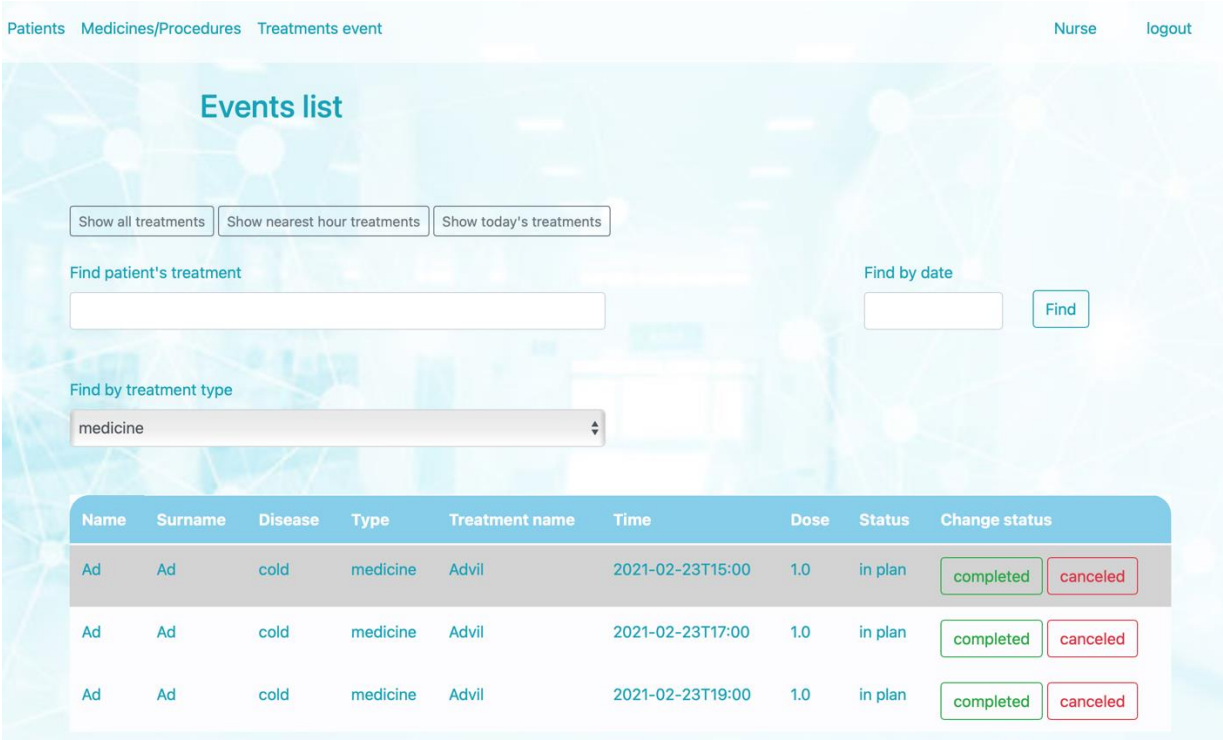

Figure 16

Nurse board (second application)

Nurse's board is a one-page application which deploys on WildFly application server. When something changes for today's events, first application sends message via ActiveMq. Second application listens same broker and when message gets there, application receives it immediately. Then application send REST request to take a today's events, specific message is pushed to the JSF through websocket and it lets the table know to update. For this purpose AJAX was used.

Events

Patient name	Patient surname	Name	Dose	Type	Status	Cancel reason	Treatment time
Ad	Ad	Advil	1.0	medicine	in plan		2021-02-23 15:00:00
Ad	Ad	Advil	1.0	medicine	in plan		2021-02-23 17:00:00
Ad	Ad	Advil	1.0	medicine	in plan		2021-02-23 19:00:00

Figure 17

Unit testing

Unit-tests caver more than half service layer logic. It was tested only necessary methods with specific logic. There are written using Junit4 and Mockito.

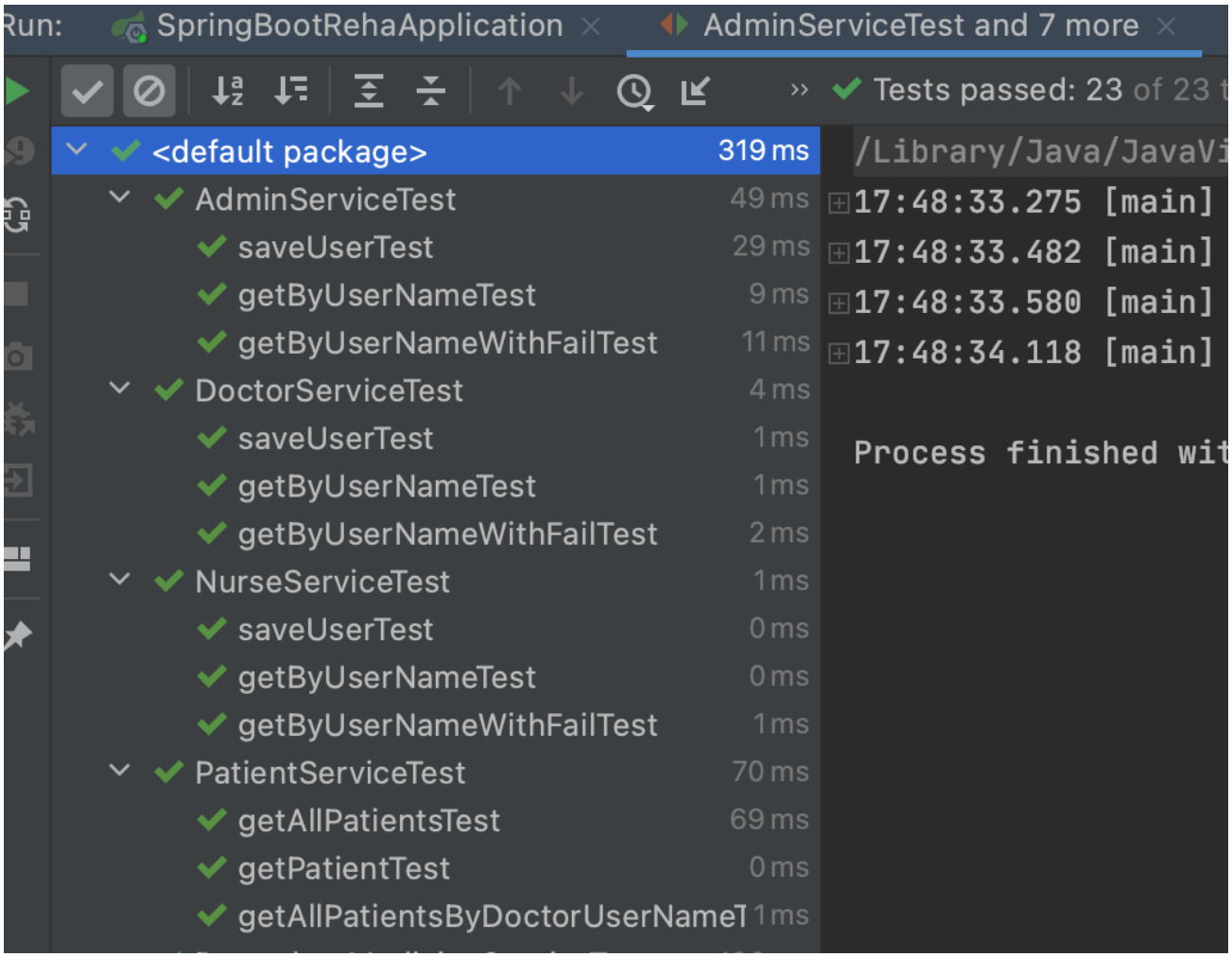

Figure 18

Logging

Logback is used for logging. All logs are saved in a file and necessary logs input into console:

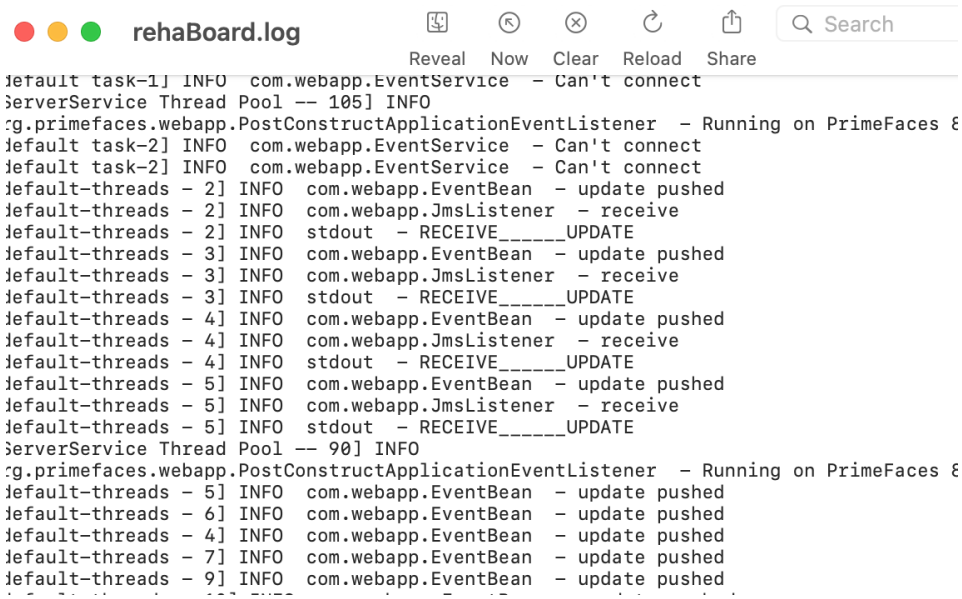

Figure 19

UI

Figure 21 -login page.

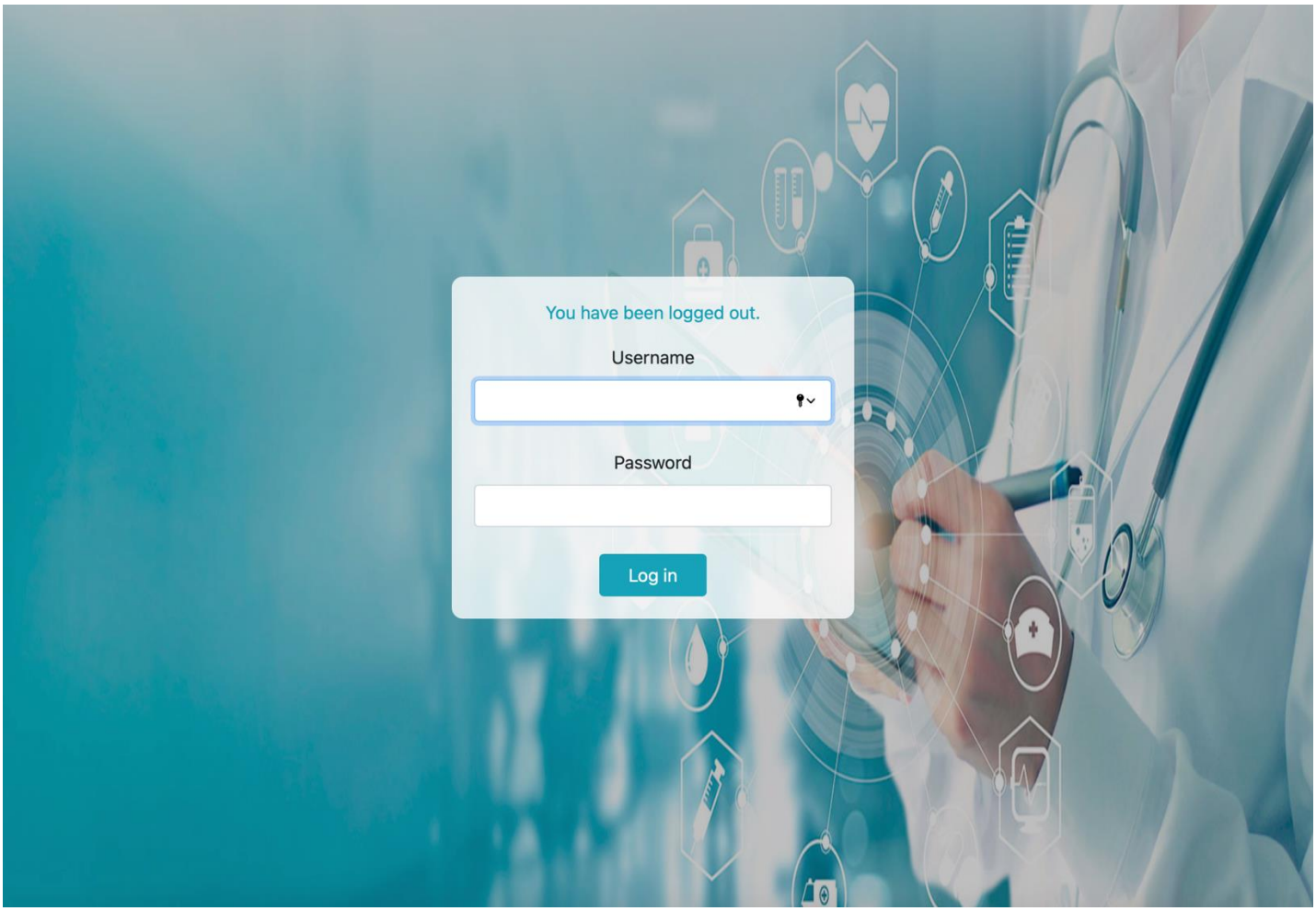

Figure 20

Figure 22 – treatment's events page.

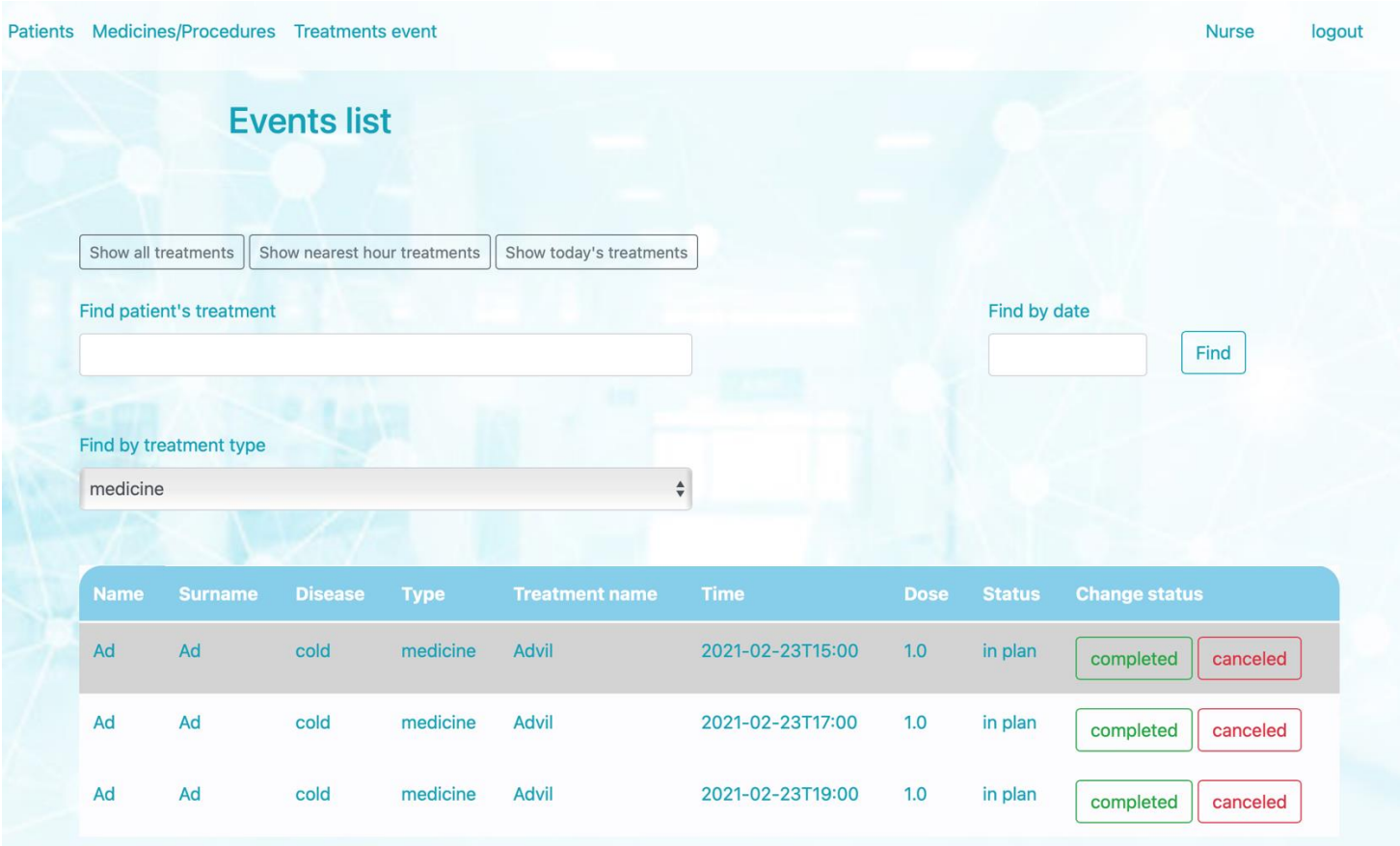

Figure 21

Figure 23 – edit form for patient and treatment.

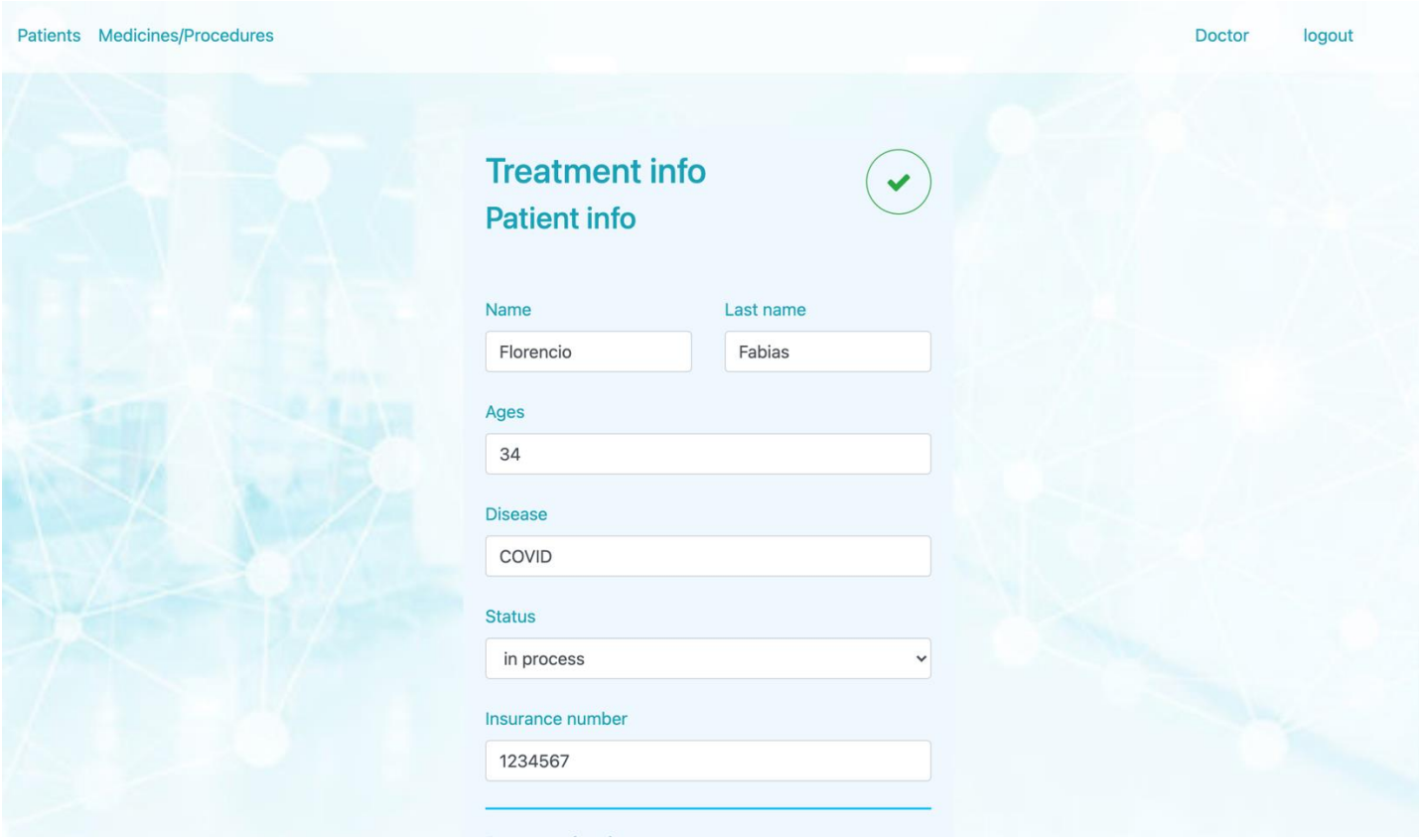

Figure 22

Figure 24 -main page for admin with users.

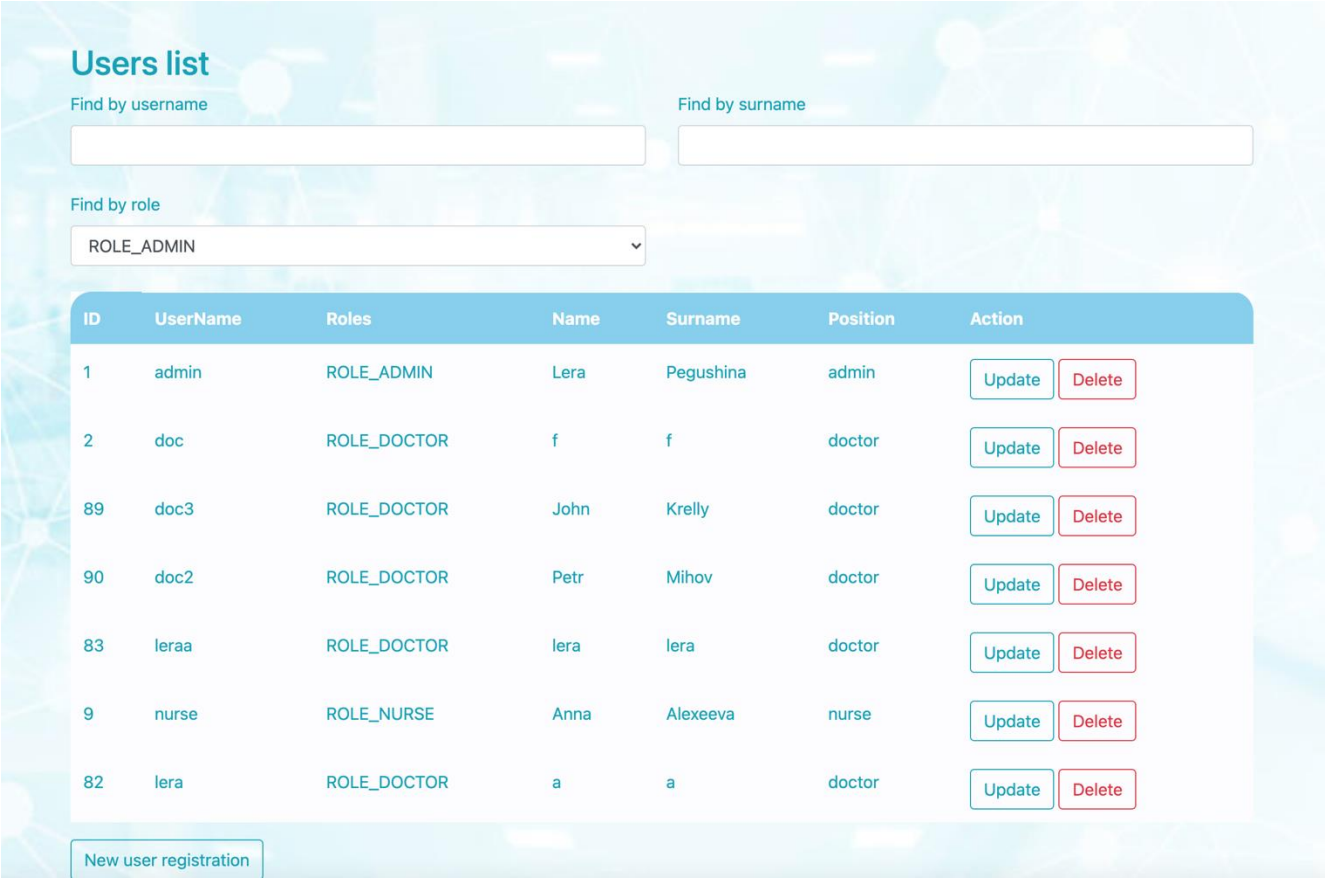

Figure 23

Figure 25 -page with all procedures and medicines.

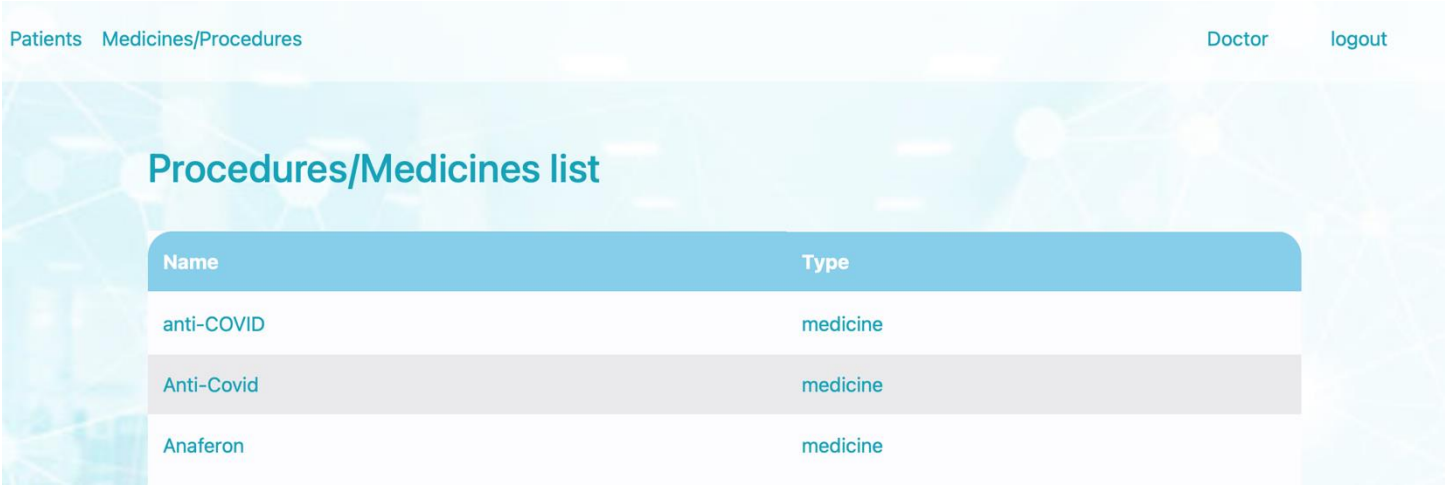

Figure 24

Figure 26 -page with patients.

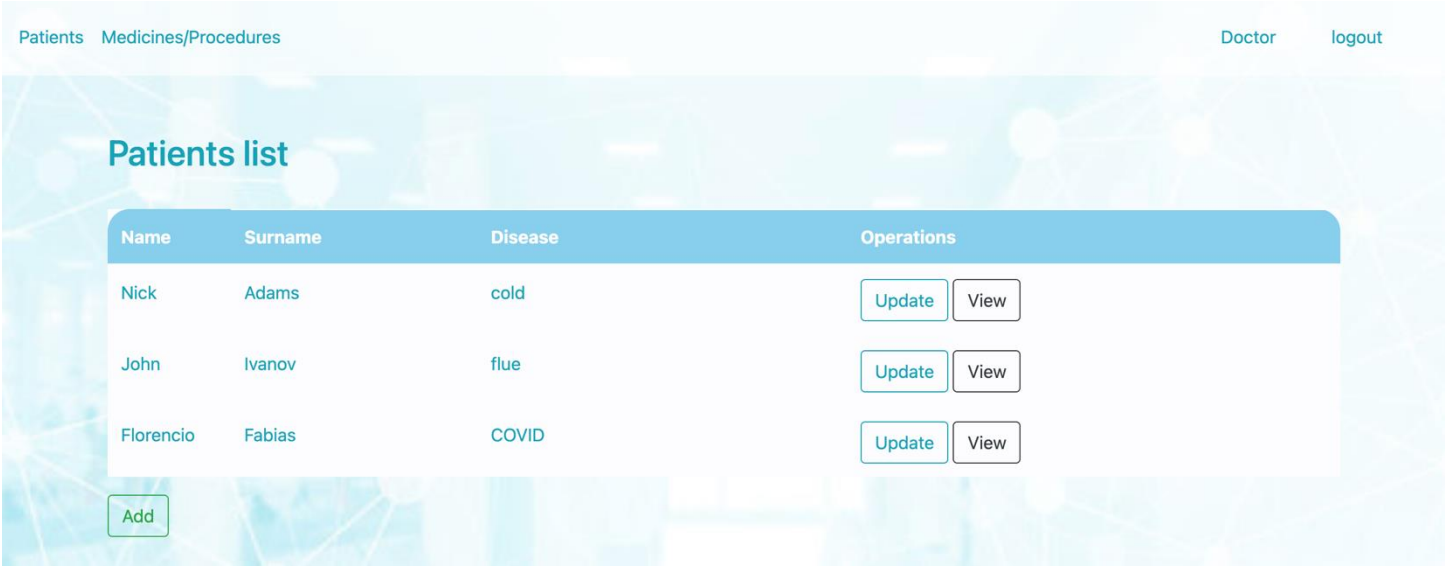

Figure 25

Figure 27 - page to input cancel reason.

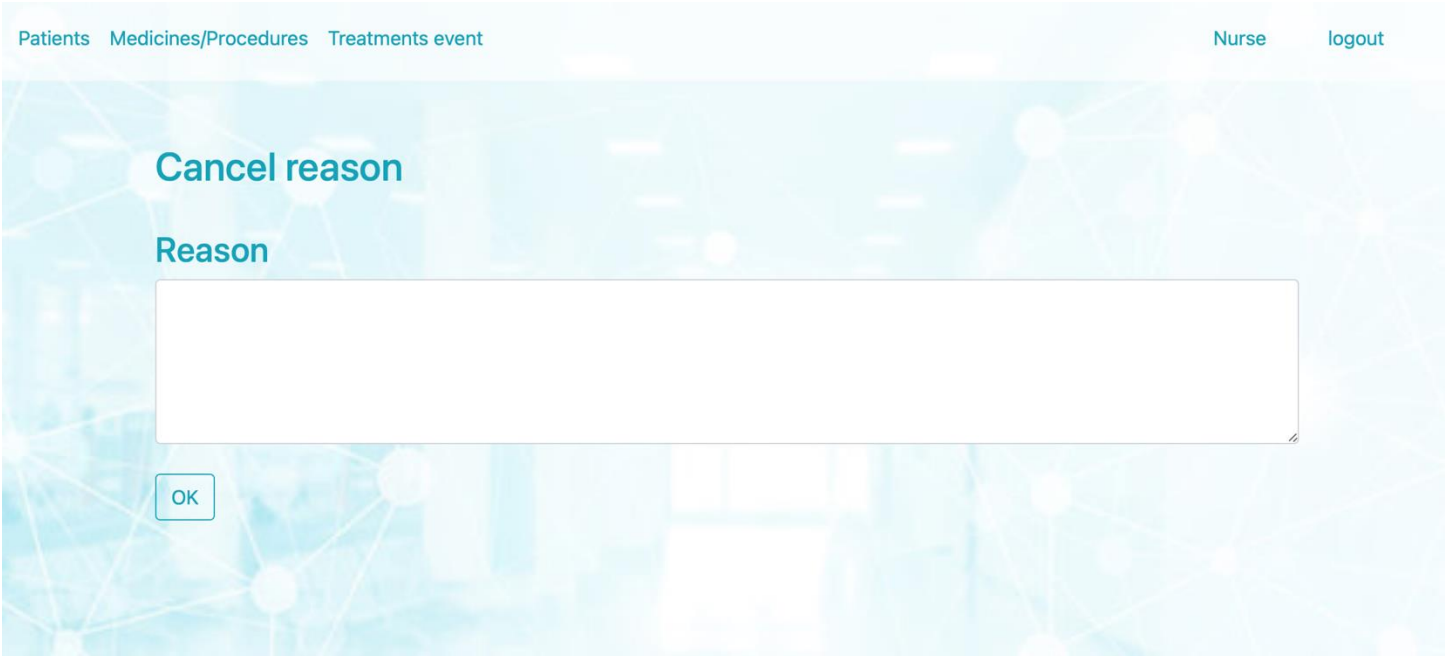

Figure 26

Figure 28 – edit user page.

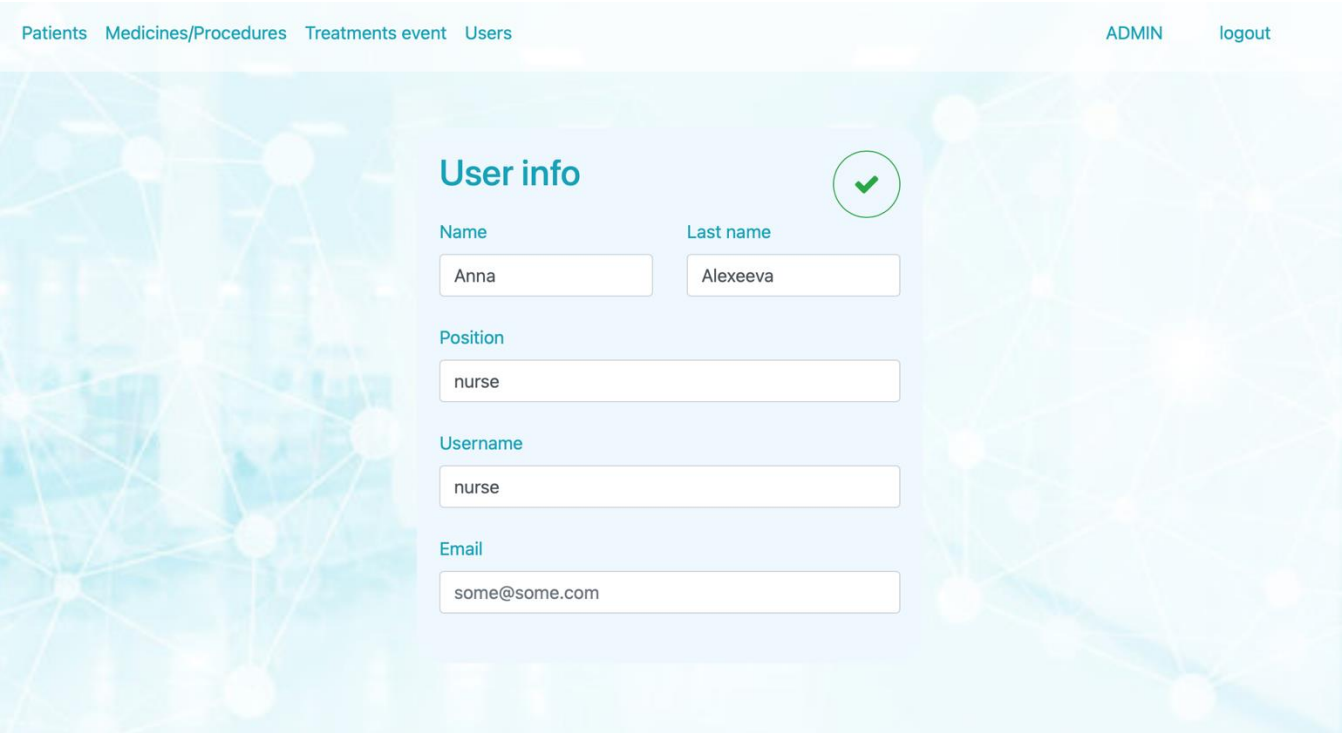

Figure 27

Known Bugs

1. Any procedure event generated at 2 pm of the current day, regardless of the time when the corresponding prescription was made.
2. Some validation not optimally implemented.
3. It is not checked whether the patient has really changed the doctor (it can be nurse, or admin and app will give an error)

Further improvements

1. Add oauth2 authorization.
2. Perform load testing to find out weak nodes.
3. Implement UI testing and Integration testing.
4. Fix bugs.
5. Code refactoring.