MedHelper

Healing Hands. Caring Hearts.

Rehabilitation clinic information system

Technical solution description

Author: Julia Dalskaya

Deutsche Telekom IT Solutions (T-Systems)
2020

Content

1. Task

There are two parts in the project.

First Part

Creating an application that simulates the operation of an information system for automating document flow in a medical rehabilitation facility. The application must provide the following functionality:

For doctors:

- Adding a patient.
- Patient discharge (implies completion of all appointments from the moment of discharge).
- Prescribing procedures and medications.
- Editing prescriptions (changing period, dose, and pattern).
- Cancelling prescriptions.

For nurses:

- View all treatment events.
- Filter treatment events by date (for today, for the next hour) and by patient.
- Change the status of treatment events from “scheduled” to “completed” and from “scheduled” to “cancelled”.

The main goal is to create a multi-user client-server application. All data are stored on a server side. Each client may load some data. After each modification operation, the data must be synchronized with a server.

Second Part

Implement a separate client application for the electronic board. The app should display a list of all treatment events scheduled for the current day. Data must be loaded at startup and stored on a client side. Data is reloaded when the server notifies of changes in the list of treatment events.

2. Used Instruments and Technologies

IDE:

- IntelliJ IDEA 2020.1.2 (Ultimate Edition)

Project build management tool:

- Apache Maven 3.6.3

Application Server:

- WildFly 20.0.1 Final

Servlet Container:

- Tomcat 9.0.331

Database:

- MySQL 8.0.20

Git

Backend:

- Apache ActiveMQ Artemis (embedded in WildFly)
- Apache Commons DBCP 1.4
- iText PDF 5.5.13.1
- Jackson 2.11.0
- JavaMail 1.6.2
- JPA 2.1
- JSF 2.3
- JSP 2.3
- EJB
- Hibernate
- Lombock 1.18.12
- Logback 1.2.3
- MapStruct 1.3.1.Final
- REST
- Slf4j 1.7.3
- Spring Framework 5.2.6 RELEASE
- Spring Security 5.2.1 RELEASE

Frontend:

- Bootstrap 4.5.0
- DataTables 1.10.21
- HTML/CSS
- Font Awesome
- JavaScript
- JQuery
- Primefaces 8.0

Testing instruments and libraries:

- JUnit 5
- Mockito 3.5.7
- Selenium 4.0.0
- JaCoCo 0.8.3
- SonarQube 8.4.1

3. MedHelper App

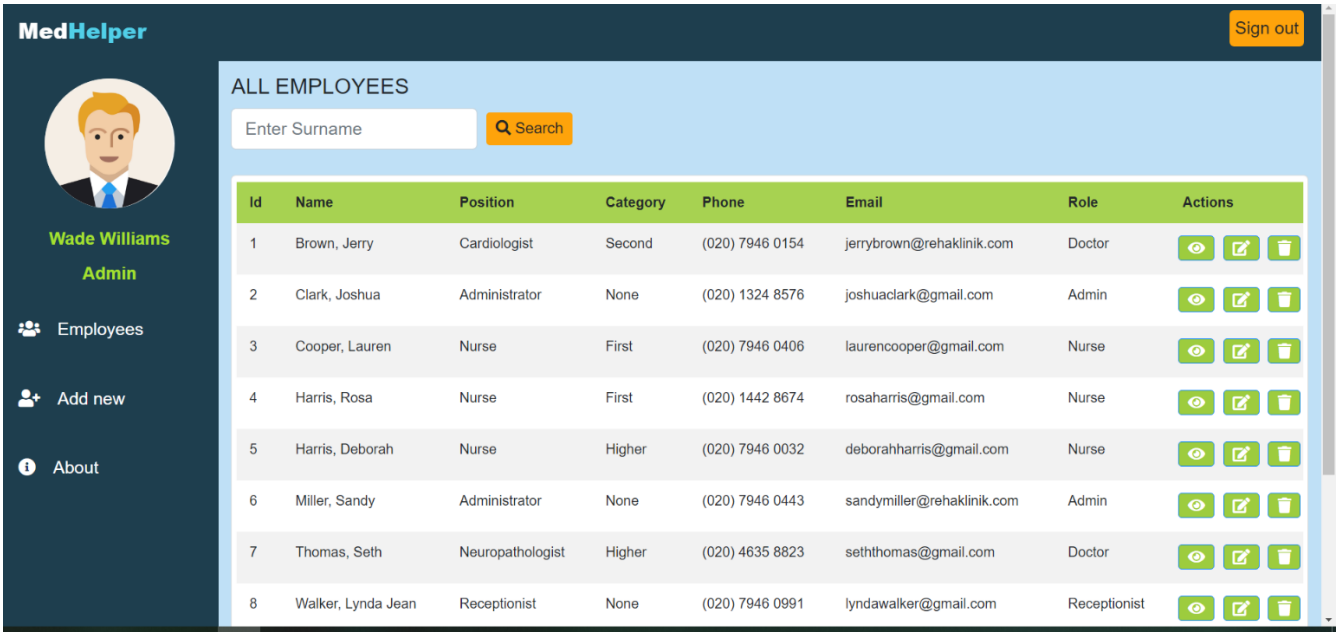
3.1. Application Description and Additional Features

According to the task, the app has three actors: a doctor, a nurse, and a patient. But in a real clinic we have more actors. For example, someone should enter and edit employee data. Also, when patients come to the clinic, they first have to deal with a registry. Therefore, two more actors were added – administrator and registry employee (**feature**).

So, there are five roles in application:
ADMIN, DOCTOR, NURSE, PATIENT, RECEPTIONIST.

1. Administrator (additional feature)

The main task of the administrator is to enter information about new employees in the database. Also he has a right to edit or to delete this information.



Pic.1. Admin's main page

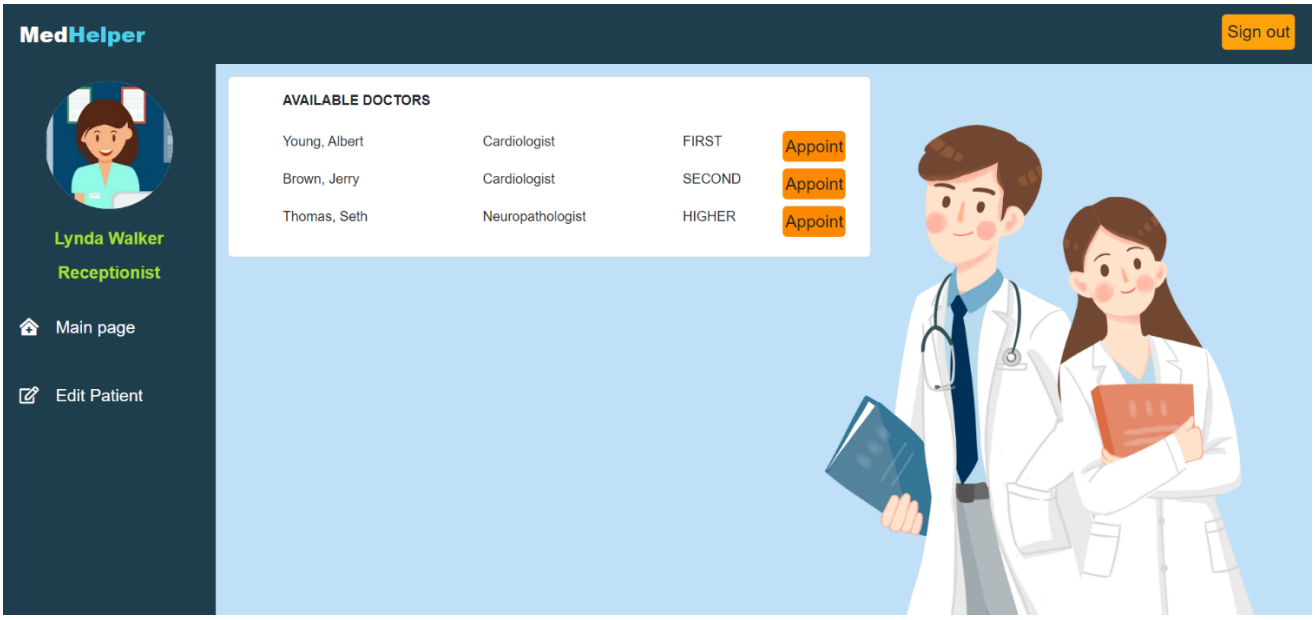
2. Receptionist (additional feature)

In original task a doctor should add a patient to database. But in real life he shouldn't do that. Therefore, this requirement has been changed and the role of a receptionist has been introduced.

Receptionist is responsible for maintaining the patient data. This employee should enter information about new patient in the database. Also he has a right to edit or to delete this information.

After adding a new patient to the database, a medical record is automatically created for him, which, in addition to information about the patient, contains information about hospitalization and diagnoses.

After that receptionist can assign a doctor to a patient.

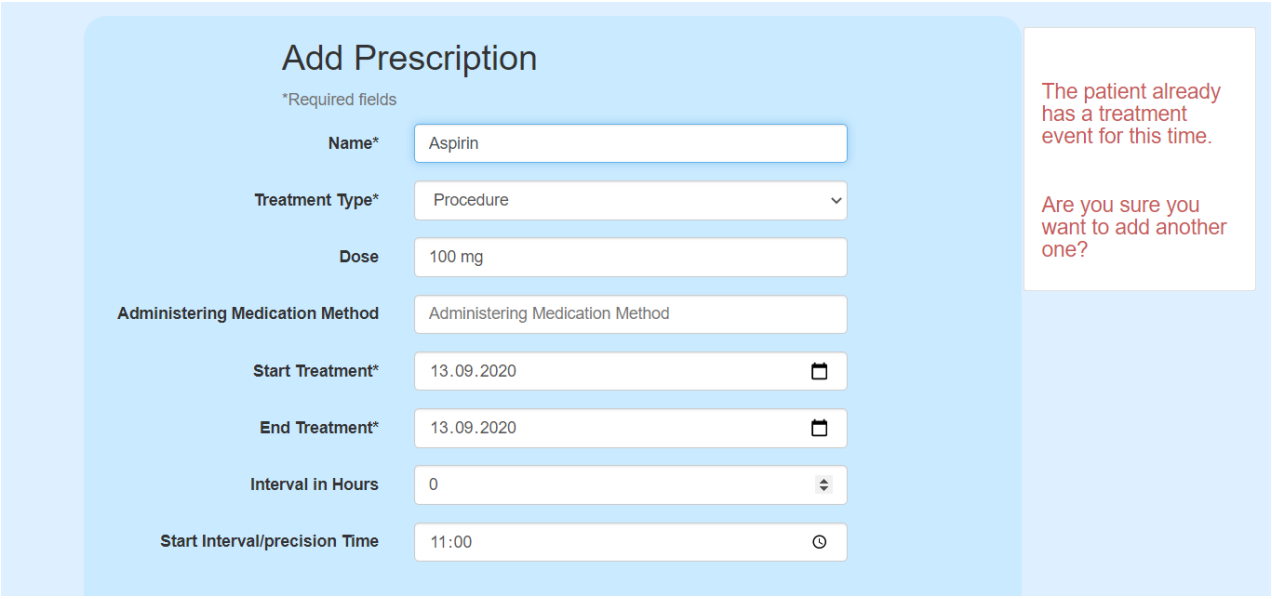


Pic.2. Appointing a doctor example

3. Doctor

A doctor has a possibility to:

- discharge a patient;
- prescribe procedures and medications;
- edit prescriptions (changing treatment period, dose and treatment time pattern);
- cancel prescriptions;
- add, edit and delete clinical diagnosis **(additional feature)**;
- work with treatment events (as well as nurse can do it) **(additional feature)**;
- filter treatment events by name and patients by surname;
- make CRUD operations with his own patients only **(additional feature)**;
- view information about all other patients **(additional feature)**;
- in case of adding duplicate prescription doctor receives a notification that such prescription has already been assigned to a patient and an offer to change the existing one or assign it to other dates **(additional feature)**;
- in case of adding any additional medicine or procedure on the same date and time that the patient already has another prescription doctor receives a warning message. And he can decide what to do further: add prescription or change time or/and date (pic.3) **(additional feature)**;



Pic.3. Warning message

- when doctor adds a new prescription to a database, all needed treatment events are generated automatically;
- when he cancels selected prescription all non-completed events are automatically canceled;
- when patient is discharged all his prescriptions are marked as finished;
- each prescription has few statuses: TBD ("To be done"), CANCELLED and DONE. When new prescription is created TBD status is automatically assigned. When all treatment events related to this prescription are completed prescription status is automatically changed to DONE **(additional feature)**.

4. Nurse

A nurse has a possibility to:

- view all treatment events;
- filter treatment events by date (for today, for the next hour) and by patient;
- change the status of events from "PLANNED" to "COMPLETED" and from "PLANNED" to "CANCELLED";
- if a nurse did not mark a treatment event in time as completed or canceled, it's automatically assigned the status "OVERDUE". Overdue treatment events are viewed on a separate page **(additional feature)**.

5. Patient (additional feature).

Additionally, the ability to use the app by the patient was added. After authorization he can see all information about his diagnosis and his treatment schedule. In addition, if desired, he can send this information to his email address - and get it as a PDF document.

3.2. Database model

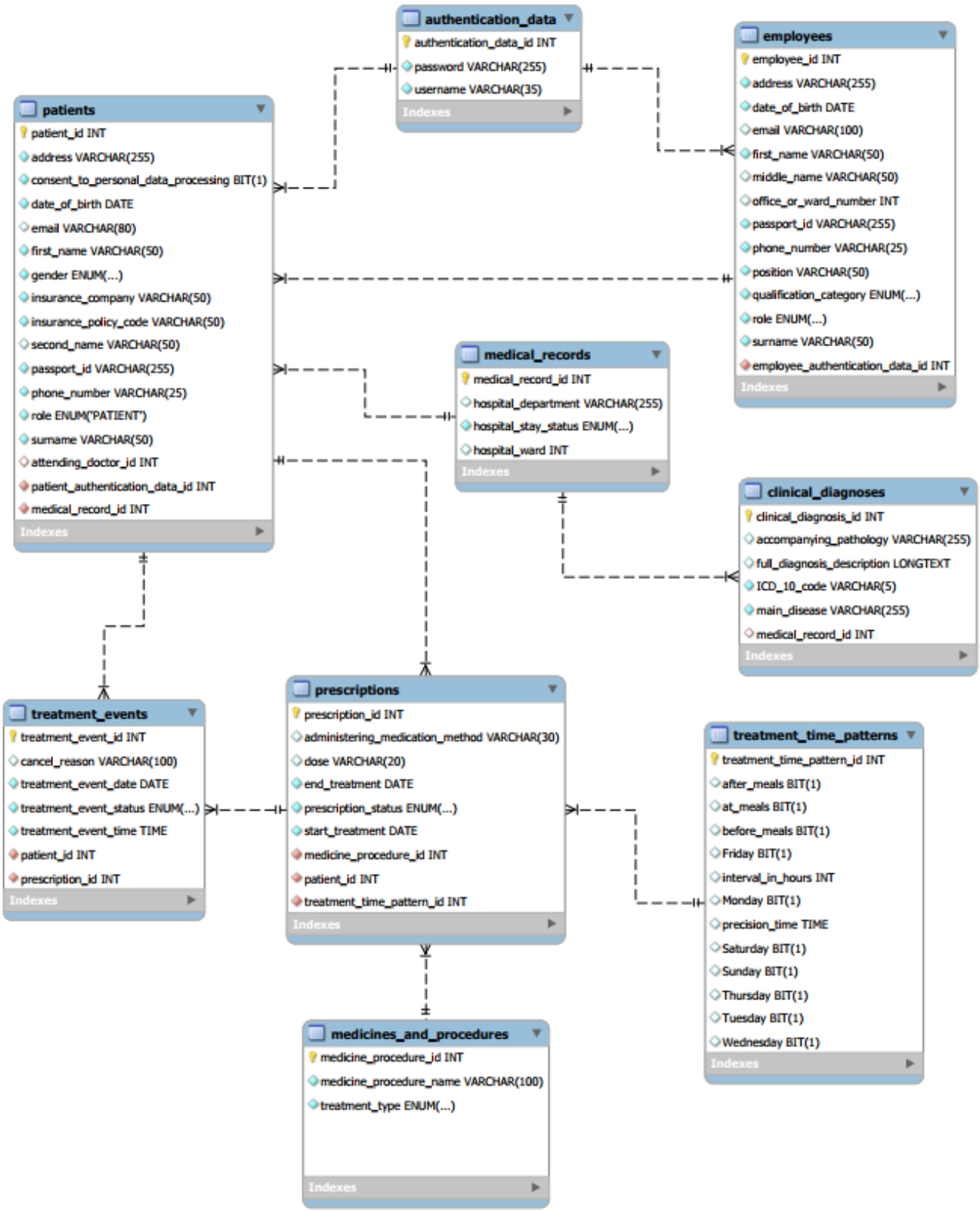


Table name	Description
employees	Stores data about employees such as administrator, doctor, nurse and receptionist. Every employee has certain role. Every doctor can treat many patients.
patients	Stores data about patients. Every patient should have one attending doctor, one medical record, many diagnosis, prescriptions and treatment events.
authentication_data	Stores users' authentication data such as username and password. Passwords are stored in an encrypted form. Usernames must be unique.
medical_records	Stores patient's medical record data such as information about patient's hospitalization status, hospital department and ward where the patient is currently being treated (useful information for a nurse).
clinical_diagnosis	Stores data about patient's clinical diagnosis – such as main disease name and ICD-10 code. The combination of columns "main_disease", "ICD_10_CODE" and "medical_record_id" must be unique.
prescriptions	Stores data about patient's prescriptions such as medicine dose and treatment period dates. Patient can have many prescriptions.
medicine_and_procedures	Stores data about medicine or a procedure – its name and type.
treatment_time_patterns	Stores data about the time pattern of taking the prescribed medication or procedure. Here is an information about the precise time and time interval of taking the medicine, specific days of the week, and additional information about taking the medicine before, during or after meals. Depending on the pattern, treatment events are generated in a certain way.
treatment_events	Stores data about the patient's treatment events (necessary information for the nurse's work). Every treatment event should

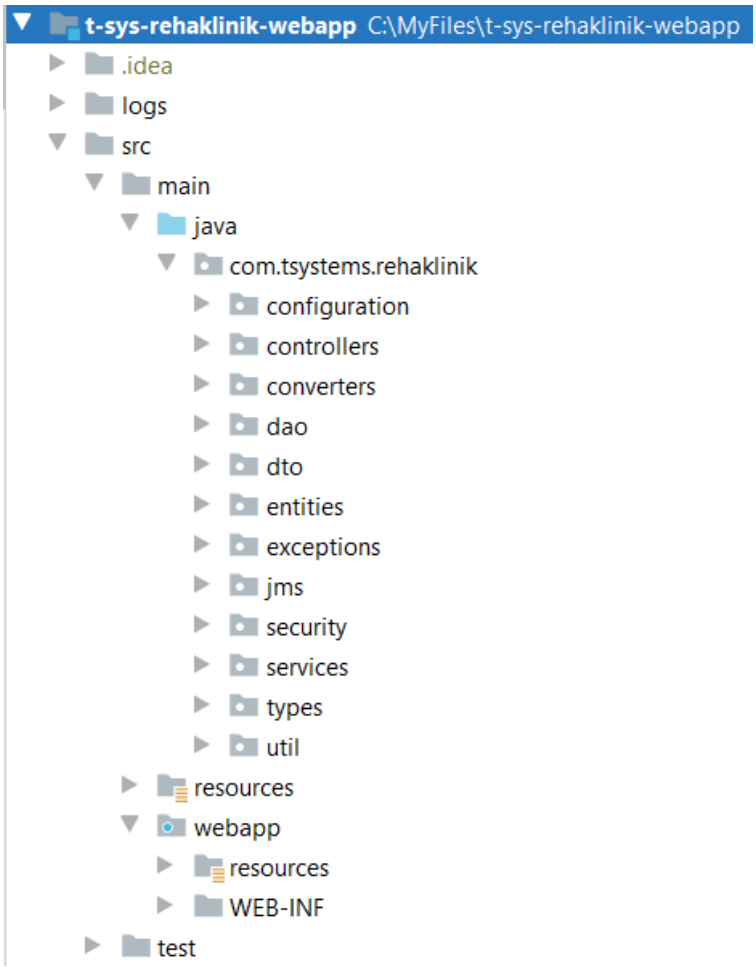
	have precision date and time, and also patient's id to whom this event is applied.

3.3. Architecture

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

- The Model contains only the pure application data, it contains no logic describing how to present the data to a user.
- The View presents the model's data to the user.
- The Controller exists between the view and the model. It listens to events triggered by the view and executes the appropriate reaction to these events.

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

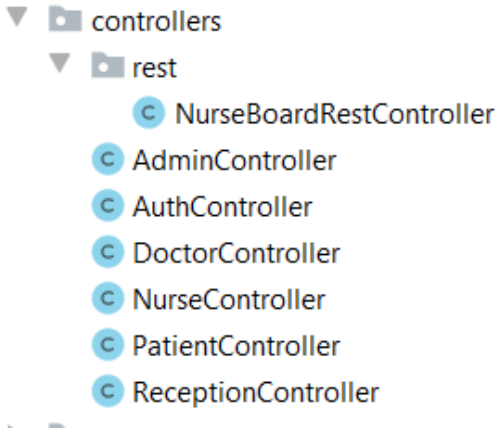


Pic.4. The structure of the application

Controllers layer

From the view, the user's request goes to a controller. The specific choice of controller depends on the user's role. For example, all requests from doctor's pages are directed to DoctorController, from nurse's pages they are directed to NurseController and so on. Such solution also makes it easier to provide access to app functionality depending on the user's role.

And there is one rest controller for communication between two separate parts of application.



Pic.5. Controllers

In this layer we deal only with DTO objects that forms on the view layer. A Data Transfer Object is an object that is used to encapsulate data and send it from one subsystem of an application to another. In this case – from controllers to services and in the opposite direction.

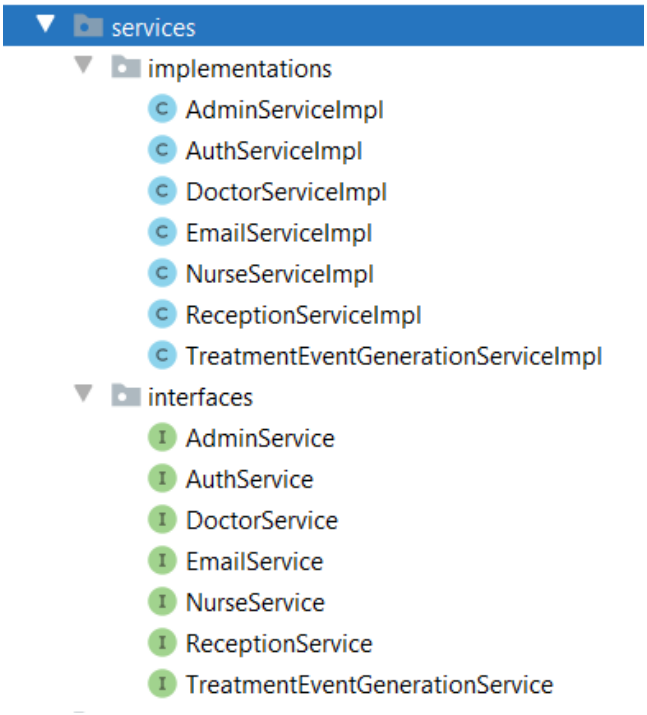
Pic.5 DTO'

There isn't any business logic in controllers. However, at this level, the basic validation of values that come from the view layer takes place.

There is module with converters to convert DTO objects to domain objects, and vice versa. DTO objects are converted both using the MapStruct code generator and “manually”.

Service layer

To provide business logic, following services exist:



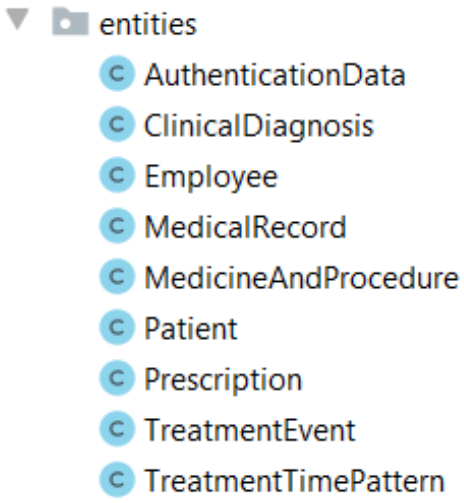
Pic.6. Services

- **AdminService** encapsulates business logic for working with employees.
- **AuthService** encapsulates business logic for authorization.
- **DoctorService** contains business logic of the doctor's work with patient's medical records, clinical diagnoses, prescriptions and treatment events:
    - create, edit, delete prescription;
    - cancel prescription;
    - assign a status to a prescription such as “Done” or “To be done”;
    - check duplicates of prescriptions;
    - check existing of prescriptions on the same date and time;
    - create, edit and delete clinical diagnosis;
    - set and edit hospitalization information;
    - manage treatment events (set to complete, cancel or delete);
    - get all needed information;

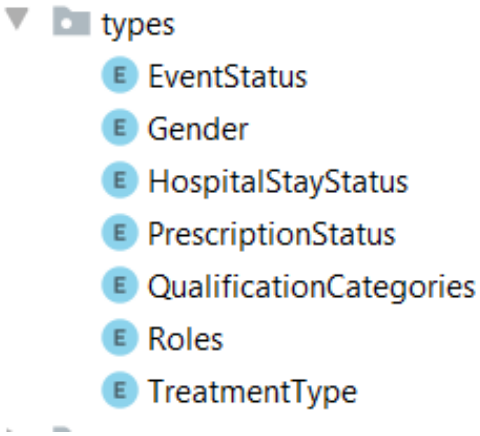
- **EmailService** contains business logic of email sending.
- **NurseService** contains business logic of the nurse's work:
    - manage treatment events (set to complete or cancel);
    - view information about treatment events;
    - search treatment events by patient’s surname.
- **ReceptionService** encapsulates business logic for working with patients.
- **TreatmentEventGenerationService** is responsible for generating of treatment events when adding or editing a prescriptions.

DAO layer

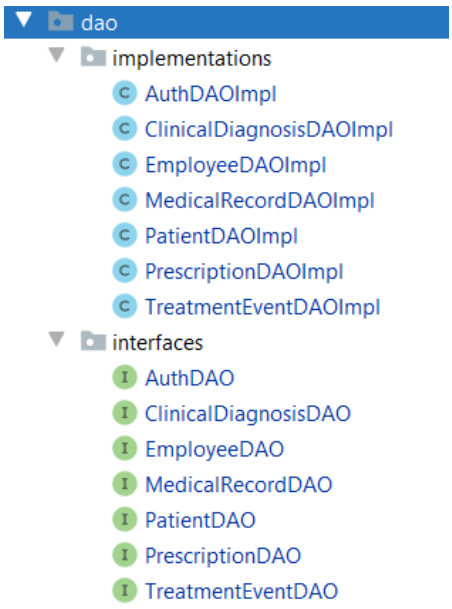
Persistence level (on the application side) is realized using Spring-ORM framework and Hibernate as JPA provider. EntityManager is used to manage persistent entities.



Pic.7. Entities



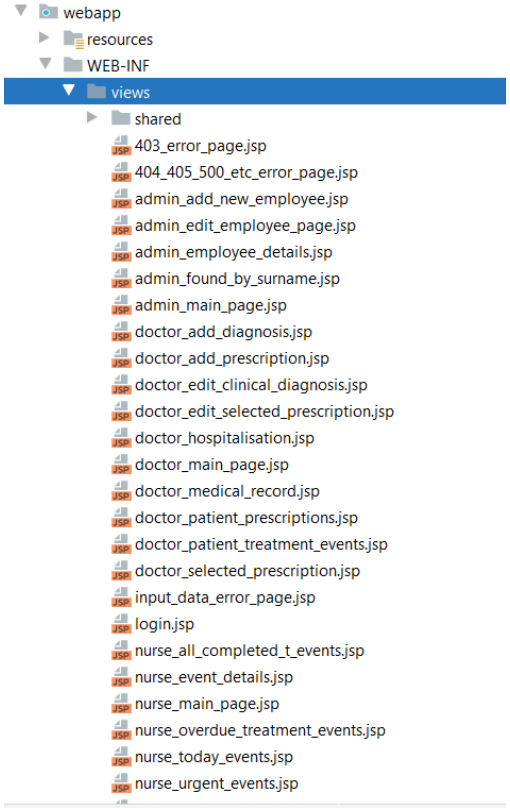
Pic.8. Enums



Pic.9. DAO layer

View layer

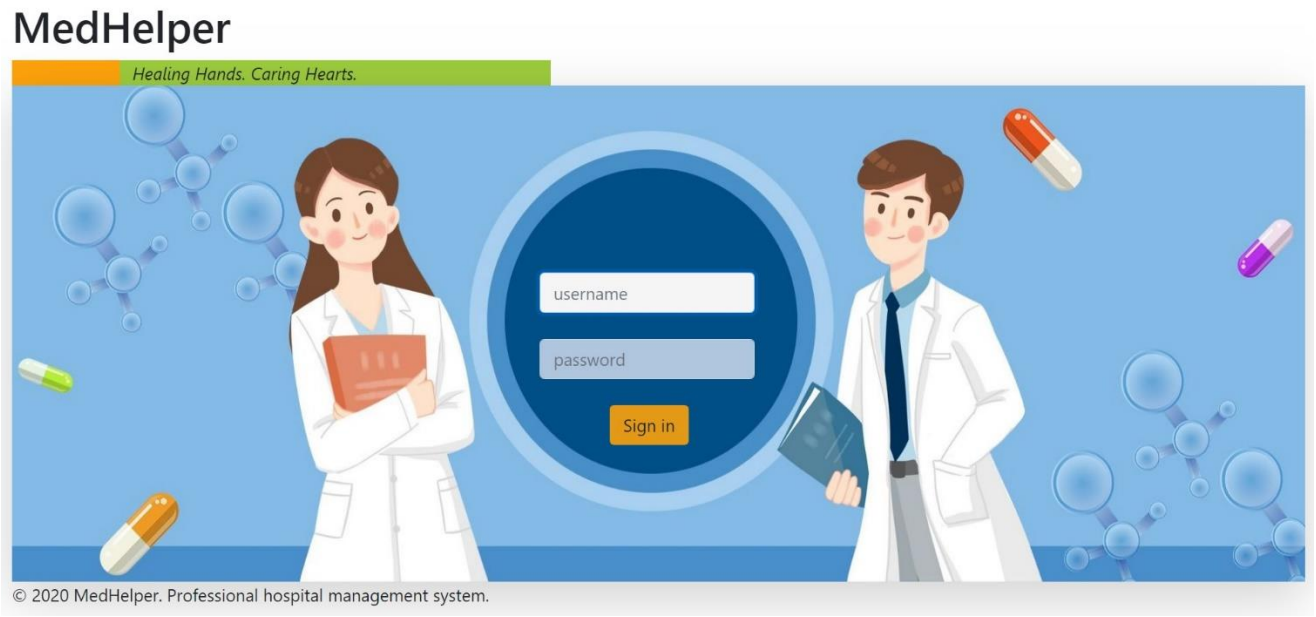
This layer contains jsp-files. Taglibs were also used in the work of jsp pages - JSP Standard Tag Library (JSTL) and Spring tags. Also were used Bootstrap, JQuery and Fontawesome library.



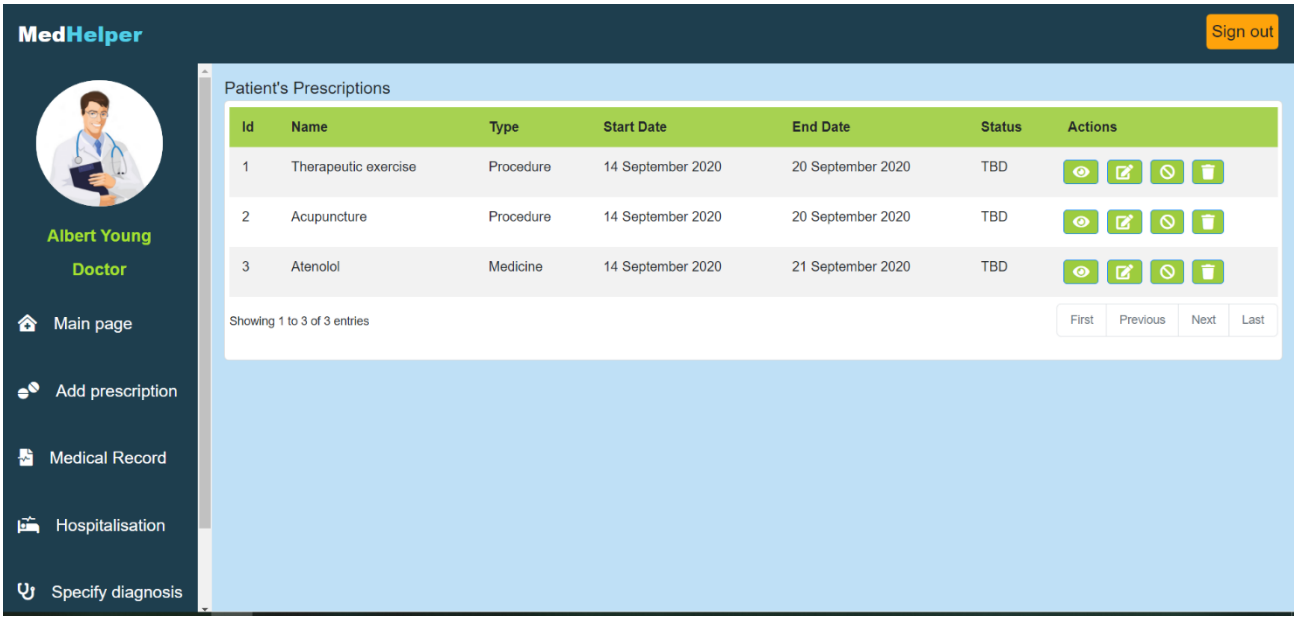
Pic.10. View layer

3.4. UI

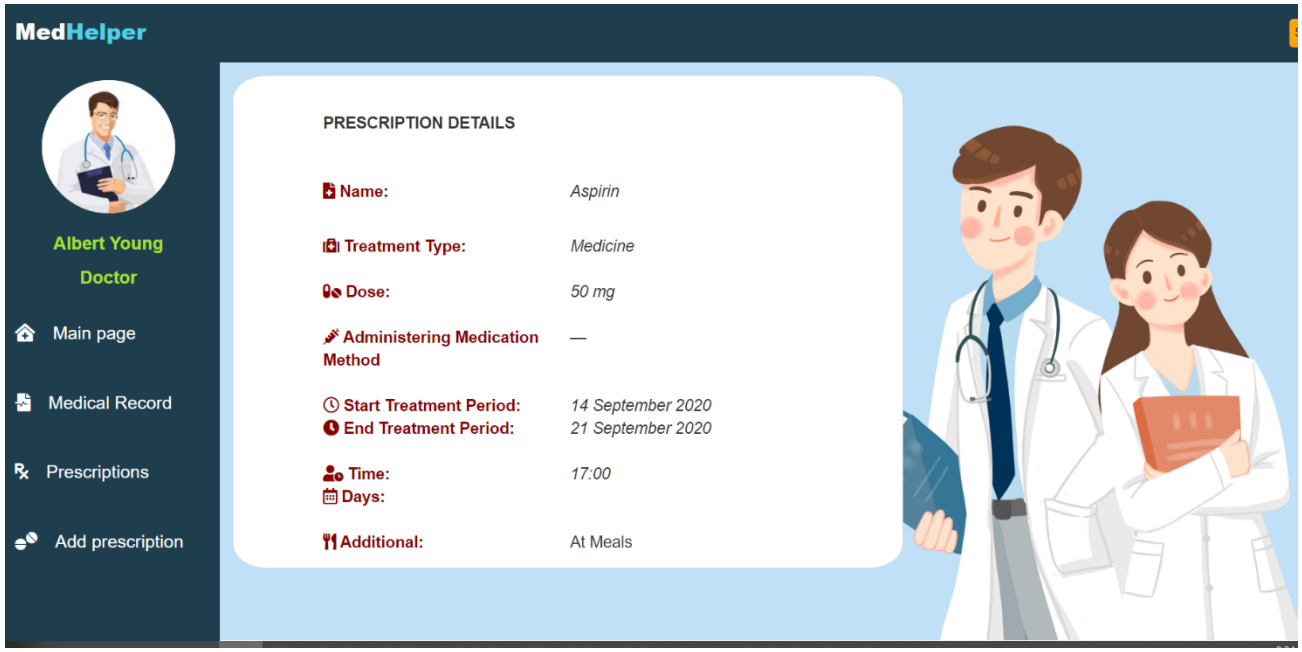
There are some examples of application's UI:



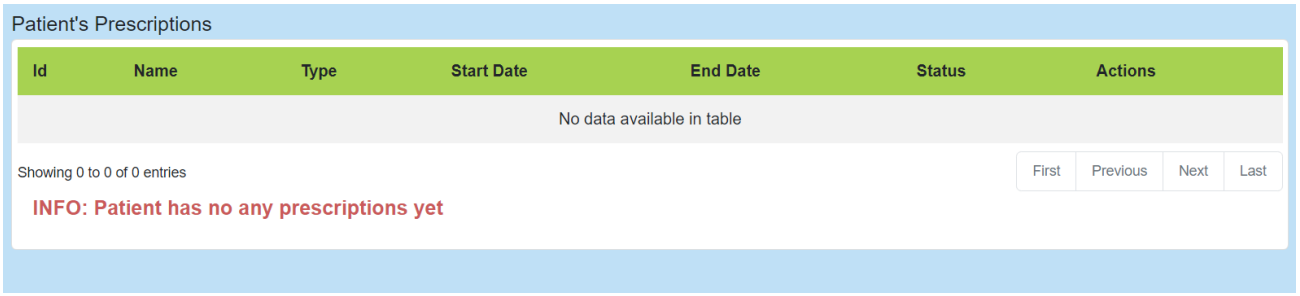
Pic.11. Login page



Pic.12. Page with patient's prescriptions

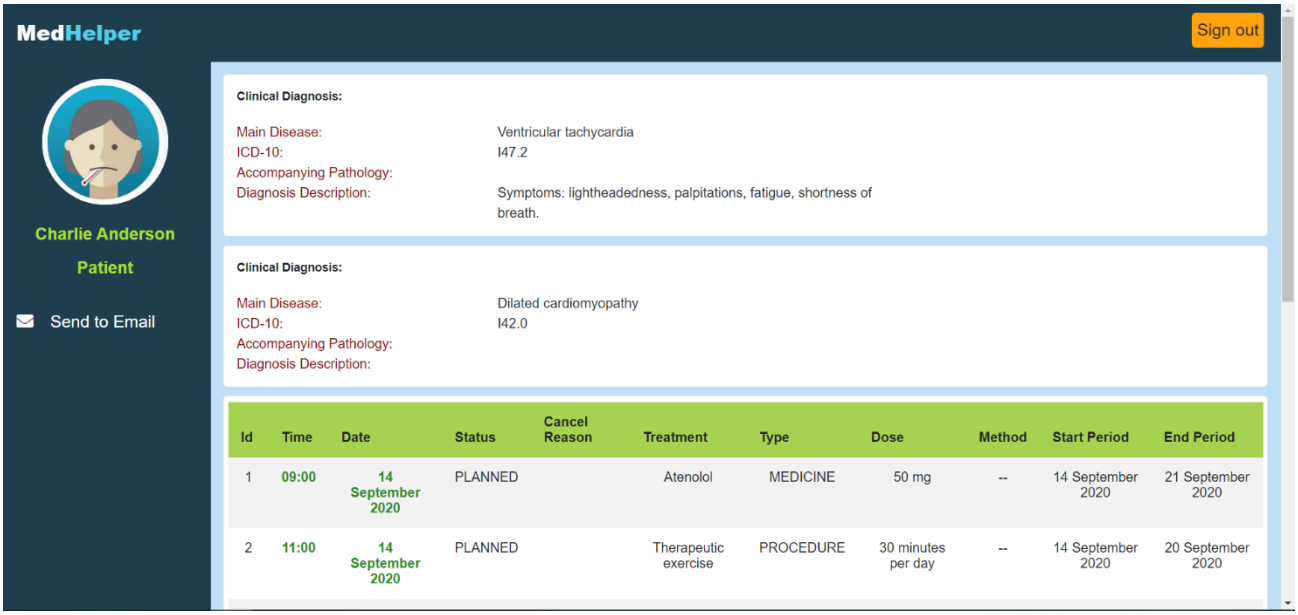


Pic.13. Prescription details page

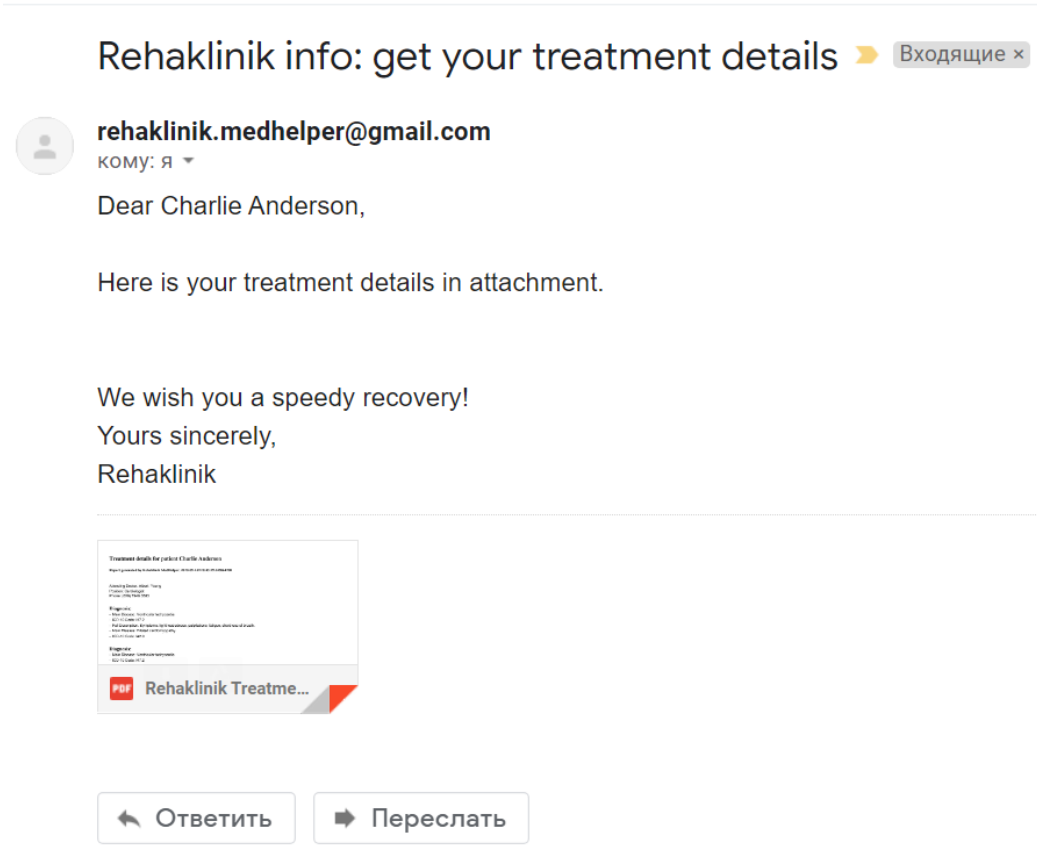


Pic. 14. Information message on a doctor page

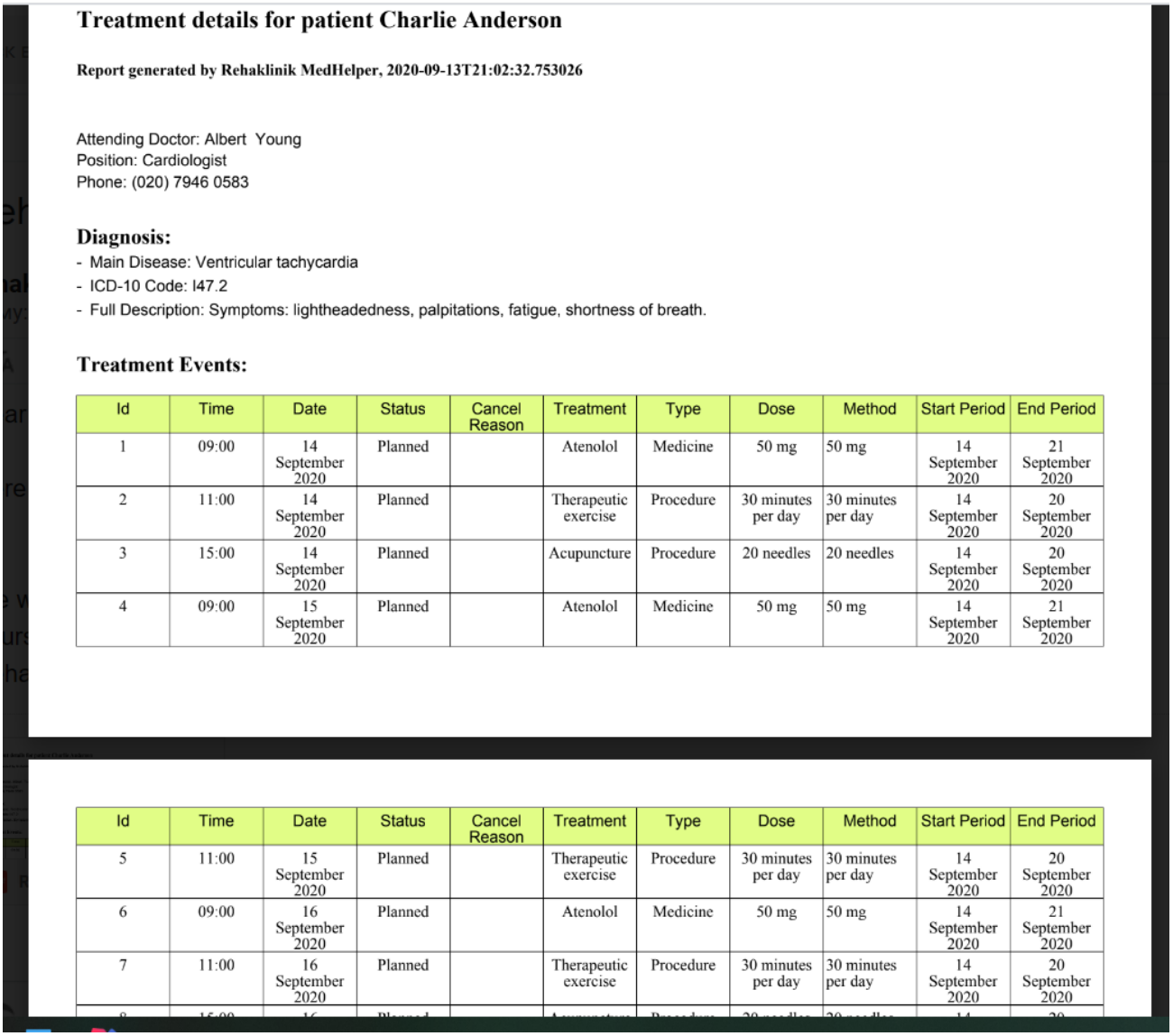
Pic.15. Main nurse’s page



Pic.16. Patient's main page



Pic.17. Email example



Treatment details for patient Charlie Anderson

Report generated by Rehaklinik MedHelper, 2020-09-13T21:02:32.753026

Attending Doctor: Albert Young
Position: Cardiologist
Phone: (020) 7946 0583

Diagnosis:

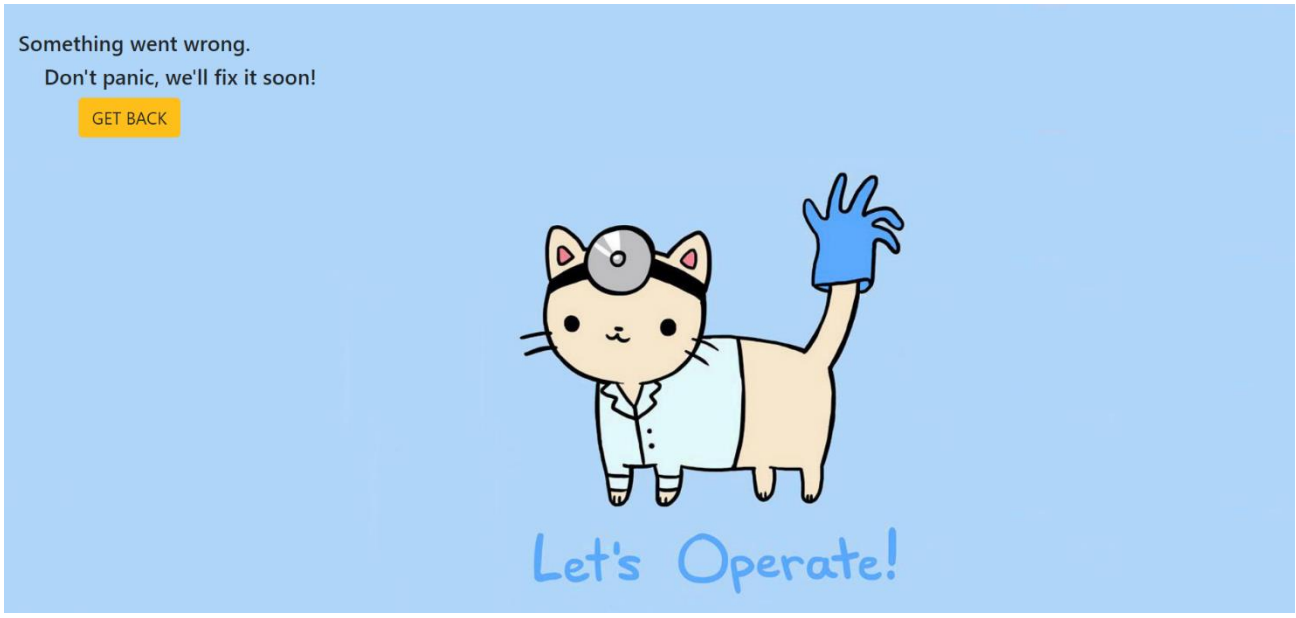
- Main Disease: Ventricular tachycardia
- ICD-10 Code: I47.2
- Full Description: Symptoms: lightheadedness, palpitations, fatigue, shortness of breath.

Treatment Events:

Id	Time	Date	Status	Cancel Reason	Treatment	Type	Dose	Method	Start Period	End Period
1	09:00	14 September 2020	Planned		Atenolol	Medicine	50 mg	50 mg	14 September 2020	21 September 2020
2	11:00	14 September 2020	Planned		Therapeutic exercise	Procedure	30 minutes per day	30 minutes per day	14 September 2020	20 September 2020
3	15:00	14 September 2020	Planned		Acupuncture	Procedure	20 needles	20 needles	14 September 2020	20 September 2020
4	09:00	15 September 2020	Planned		Atenolol	Medicine	50 mg	50 mg	14 September 2020	21 September 2020

Id	Time	Date	Status	Cancel Reason	Treatment	Type	Dose	Method	Start Period	End Period
5	11:00	15 September 2020	Planned		Therapeutic exercise	Procedure	30 minutes per day	30 minutes per day	14 September 2020	20 September 2020
6	09:00	16 September 2020	Planned		Atenolol	Medicine	50 mg	50 mg	14 September 2020	21 September 2020
7	11:00	16 September 2020	Planned		Therapeutic exercise	Procedure	30 minutes per day	30 minutes per day	14 September 2020	20 September 2020

Pic.18. Generated PDF example



Pic.19. Common error page

Pic.20. 403-error page

3.5. Security

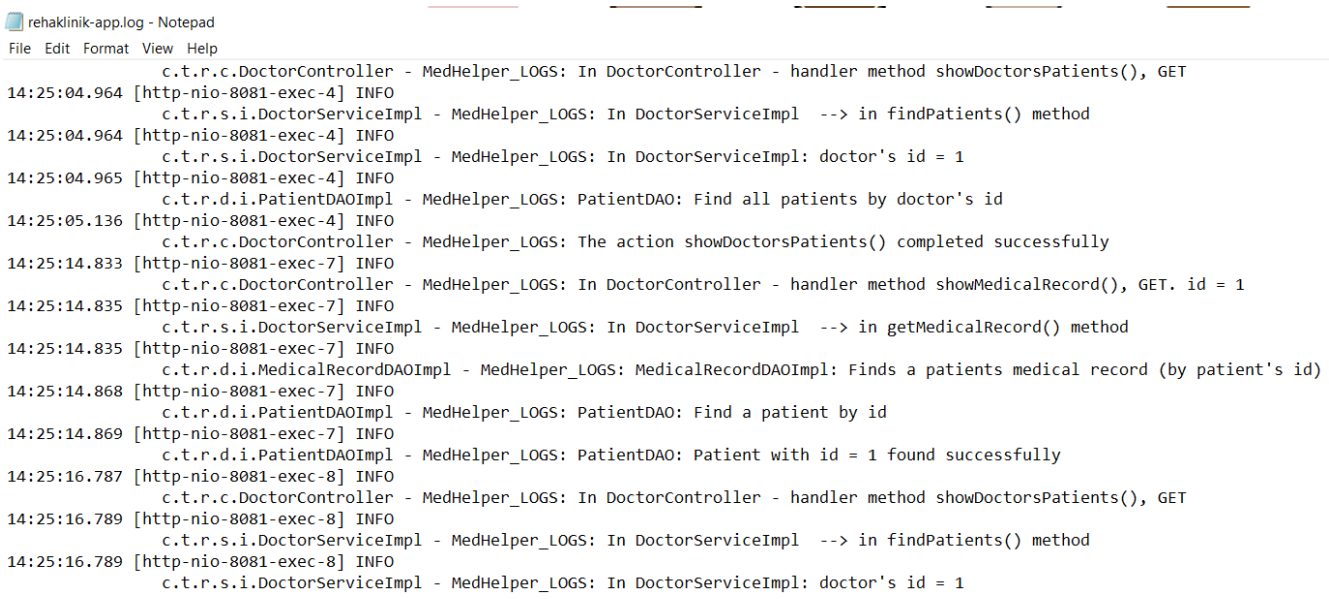
Security support is implemented using the Spring Security project.

Authorization and authentication are performed using CustomAuthProviderImpl class, that implements the AuthenticationProvider interface from Spring Security. All passwords are encrypted using the PasswordEncoder interface and the BCrypt function.

There are five roles in application, and after successful authorization users are redirected to the main page of their accounts.

3.6. Logging

Logback is used for logging. All logs are saved in a file:

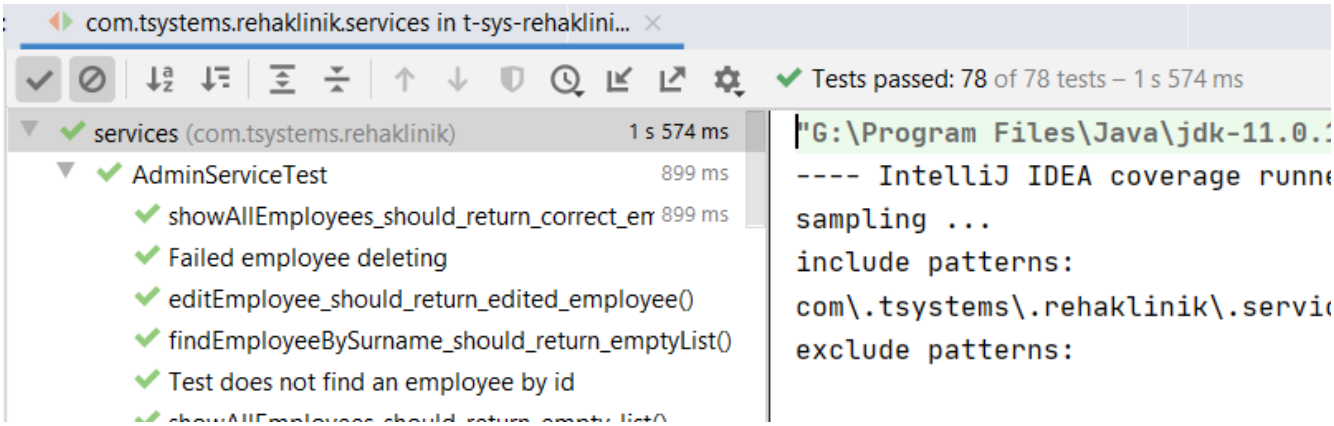


Pic.21. Logging example

3.7. Code Quality

Unit testing

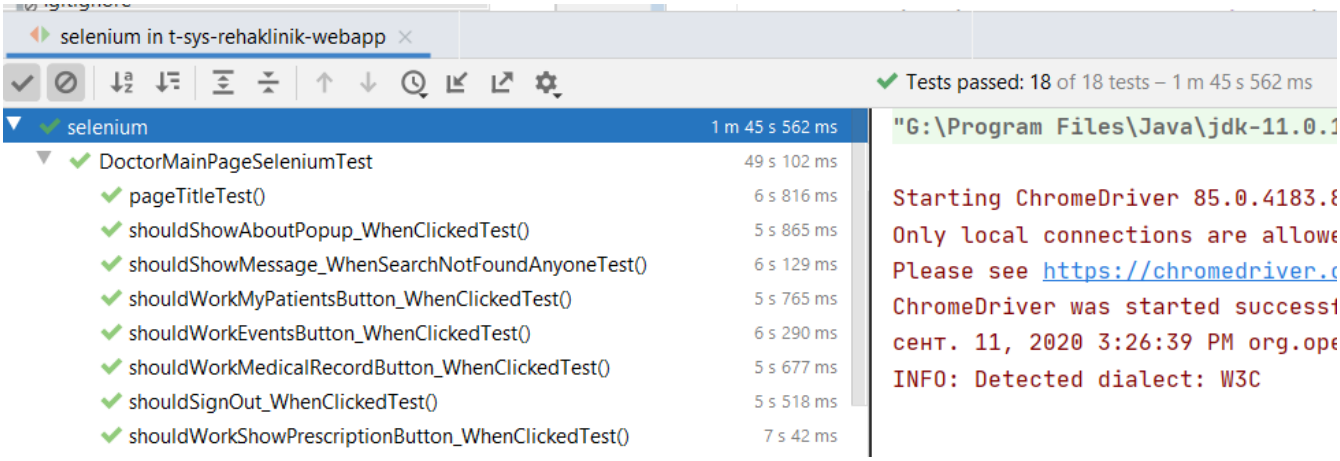
Unit-tests are written for business logic using Junit5 and Mockito. The coverage is 88.7% for the service layer.



Pic.22. Running unit tests example

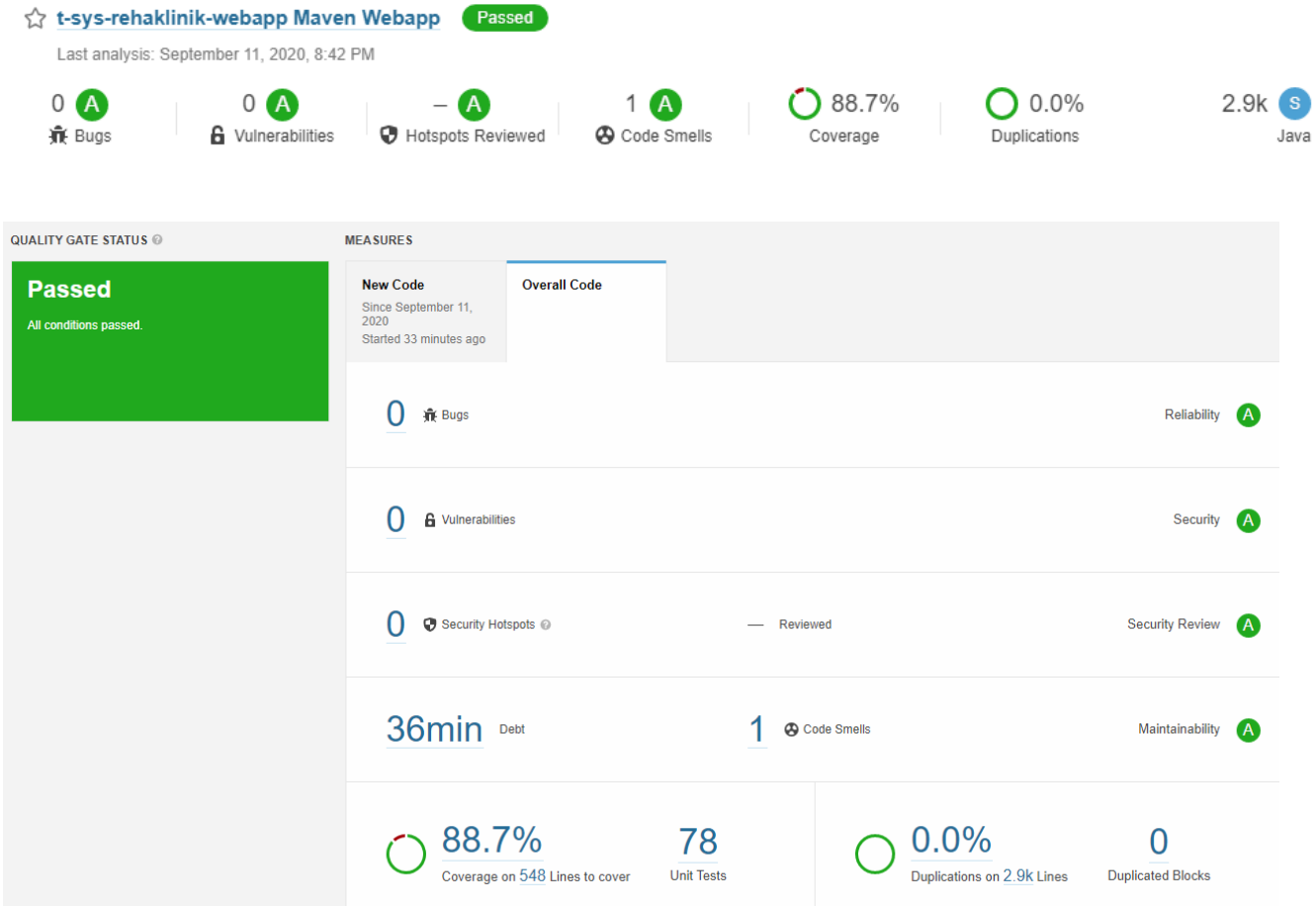
UI testing

Selenium WebDriver was used to test the application interface. Tests check the functionality of the authorization page, the doctor's main page, the patient's medical record page, the error page, check the sign out operation and some operations for adding and editing records.



Pic.23. Running UI tests example

SonarQube report



Pic.24. SonarQube report

3.8. Known Bugs

1. Multiple checkboxes are set to “checked” while adding new prescription after getting a warning message about trying to add new prescription on the same time that is already taken.

Steps to reproduce the bug:

1.1. Open page “Add new prescription”.

1.2. Add “Aspirin” on 11:00.

1.3. Set start treatment period date and end treatment period day.

1.4. Press button “Add prescription”.

1.5. Open page “Add new prescription”.

1.6. Add “Massage” on 11:00.

1.7. Set the same start treatment period date as in step 1.3. and end treatment period day.

1.8. Set checkbox “Thursday” to checked.

1.9. Press button “Add prescription”.

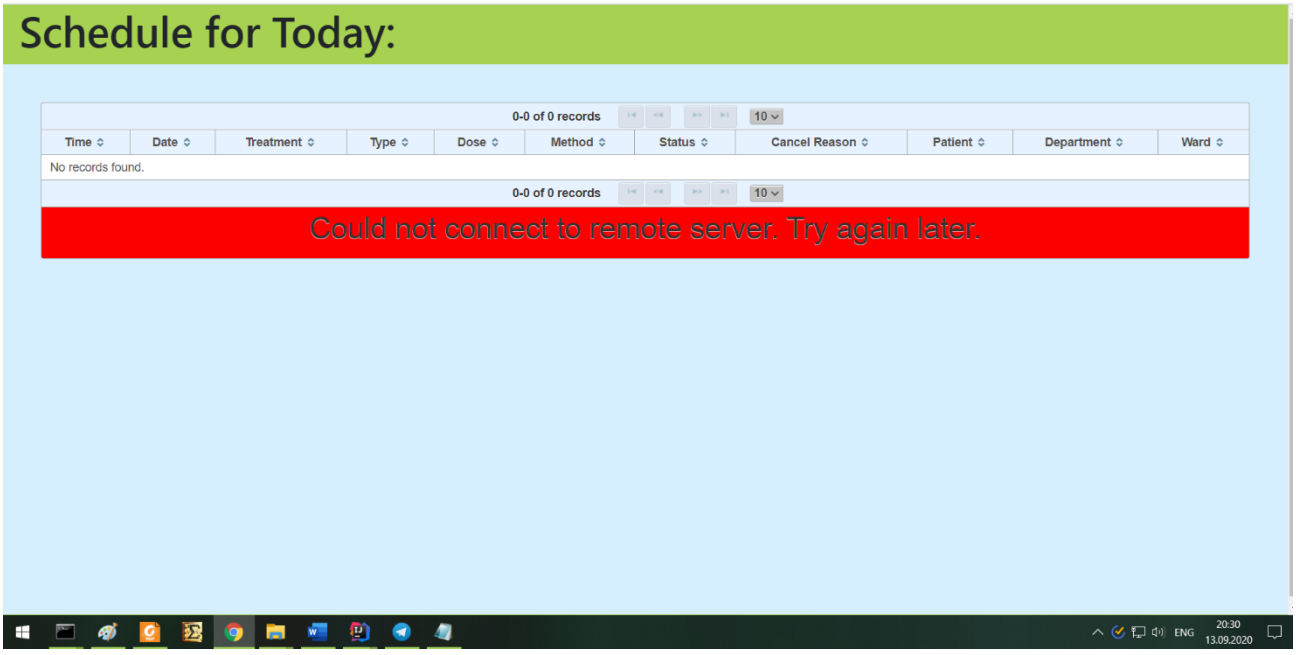
2.0. Opens page with warning message, and all checkboxes are checked instead of one.

Temporary solution:

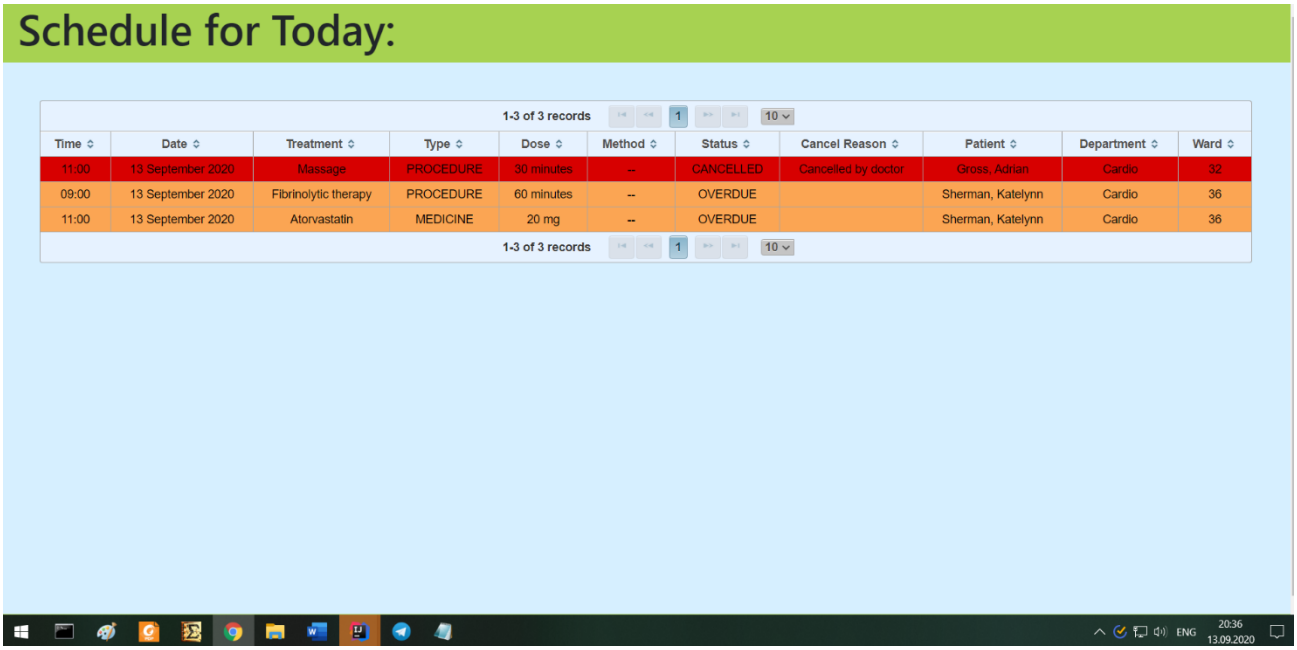
Do not set saved values into checkboxes after the page is reloaded and a warning message appears.

4. Nurseboard

Nurseboard is a simple one-page application deployed on WildFly application server. It communicates with another application via rest requests. When the main application is uploaded, it sends to this application a message about that in queue on the server (using JMS). After that second app send REST request to take a data – all today treatment events. When data receives, a message is pushed to the JSF side via websocket, after that data on this page is updating with AJAX. When second app could not connect to remote source, warning message appears.



Pic.25. Nurseboard warning message



Pic.26. Nurseboard

5. Further improvements

1. Add the ability for users to change their passwords and email.

2. Add server-side pagination to speed up the data retrieval process and save the system from having to load data that the user may not use.

3. Apply soft-deleting pattern when deleting data.

4. Add to a doctor and to a nurse an ability to receive notifications about overdue treatment events.

5. Add the ability to sort data by the specified criteria.