T-Systems RUS

Java School 32

**Medical Center**

Technical Solution Description

Maria Boldareva

06/05/2020

**Content**

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Abstract………………………………………………………………………………. | | 3 |
| 2 | Technologies and Frameworks……………………………………………………...... | | 3 |
| 3 | Database Shema…………………………………………………………...…………. | | 4 |
| 4 | Implementation………………………………………………………..……………... | | 5 |
|  | 4.1 | Subject Area…………………………………………………………...………... | 5 |
|  | 4.2 | Functionality……………………………………………………..……………... | 5 |
| 5 | Architecture………………………………………………………...………………… | | 6 |
| 6 | Modules………………………………………………………..……………………... | | 7 |
| 7 | Presentation Layer………………………………………………………..…………... | | 8 |
| 8 | Business Logic Layer………………………………………………………………… | | 8 |
| 9 | Data Access Layer……………………………………………………………………. | | 9 |
| 10 | Main Pages…………………………………………………………………………… | | 10 |
| 11 | JUnit Tests……………………………………………………………………………. | | 13 |
| 12 | Building and Deployment…………………………………………………………….. | | 13 |
| 13 | Logging………………………………………………………………………………. | | 13 |
| 14 | Code Statistics………………………………………………………………………... | | 14 |

1. **Abstract**

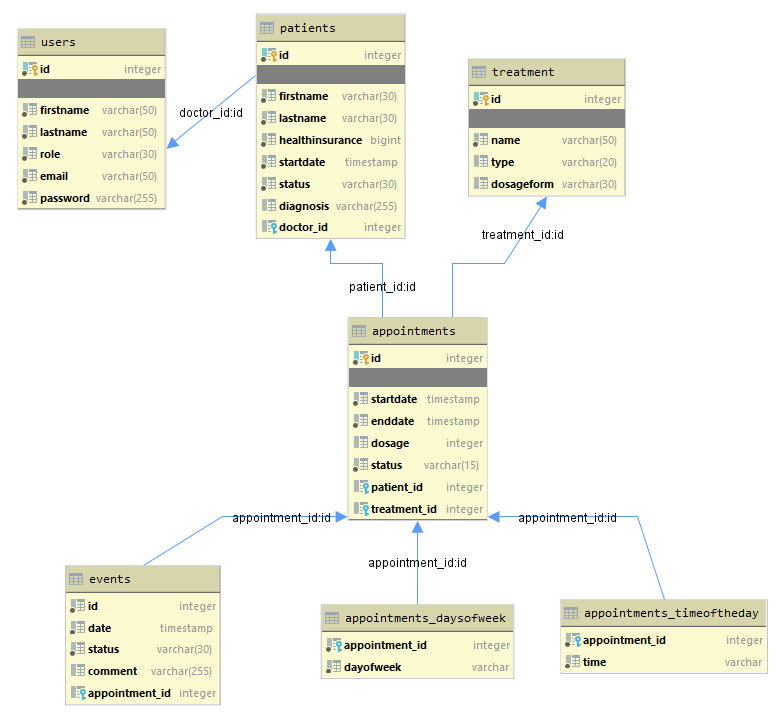
Medical clinic workflow refers to the myriad of tasks, processes, and people that work together to deliver the actual patient treatment. This precise flow of people and processes directly impacts the patient experience in some pretty obvious ways. Therefore, automation of some current processes is major objective for medical organization that allows to establish internal office communication and notation processes. Inadequate tools for working with electronic document management provide data security and updating in appropriate way.

The system represents automated medical clinic workflow. It provides correct filling of health records and interaction of medical stuff throughout the whole patient’s treatment process including patient registration, adding appointments and performing therapeutic measures.

1. **Technologies and Frameworks**

* Docker 3.5
* IntelliJ IDEA 19.3.3;
* Wildfly 14.0.0;
* PostgreSQL 9.1;
* Maven 3.5.4;
* Lombok 1.8.12;
* Log4j 1.2.17;
* Sonar 3.4.0.905
* JPA 2.2;
* Spring Framework 5.1.0;
* JUnit 4.12;
* Mockito 2.8.9;
* JSP;
* EJB 3.2
* JSF 2.2.18;
* REST Web Services;
* RichFaces 4.5.17;
* WebSockets;
* AJAX.

1. **Database Shema**



Tables and Their Relations:

* Table “Users” contains firstName, lastName, role email and password.
* Table “Patients” contains firstName, lastName, healthInsurance, startDate, status, diagnosis and has a link to the table “Users” for keeping information about doctor.
* Table “Treatment” contain name, type and dosageForm.
* Table “Appointments” contains startDate, endDate, dosage, status and has links to the table “Patients” (for keeping information about the patient who is added concrete appointment) and to the table “Treatment” (for keeping information about prescribed treatment).
* Table “Events” contains date, status, comments and has a link to the table “Appointments” for keeping information about the appointment that causes this event.
* Table “Appointments\_DayOfWeek” keeps the unidirectional relation to the table “Appointment” and information about the day pattern.
* Table “Appointments\_TimeOfTheDay” keeps the unidirectional relation to the table “Appointment” and information about the time pattern.

1. **Implementation**

**4.1 Subject Area**

There are 5 types of entities:

* Patient
  + First Name
  + Last Name
  + Diagnosis
  + Health Insurance
  + Start Date
  + Doctor
  + Status
* Appointment
  + Patient
  + Start Date
  + End Date
  + Day Pattern
  + Time Pattern
  + Treatment
  + Dosage
  + Status
* Event
  + Appointment
  + Date
  + Status
  + Comment
* Treatment
  + Name
  + Type
  + Dosage Form

**4.2 Functionality**

The application describes all subject areas from the task.

There are two roles: doctor and nurse. The permissions to perform certain operations are assigned to specific roles. Every user is assigned particular role, and through those role assignments acquire the permissions needed to perform particular system functions. Doctor has access only to patient list and all operations connected with it. Nurse can work with events, but has no access to patient data.

Working with the application doctor has such opportunities:

* Adding patient;
* Patient discharge (implies completion of all appointments: cancelled or done);
* Prescribing treatments: medicine procedures and medicines;
* Editing appointments (changing period, dosage, day pattern and time pattern);
* Appointment cancellation.

Creating an appointment involves generating all events that are provided by this appointment. Making changes to the appointment leads to cancellation of current events, adding to them comments “by Doctor” and creating new events. In case of cancellation of the appointment, all the events provided by it are also canceled and added comment “by Doctor”.

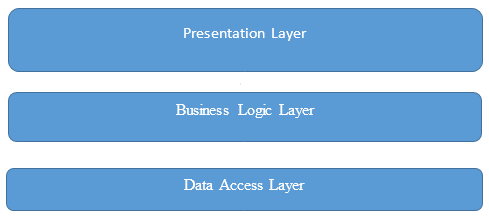
Nurse functionality includes:

* View all events;
* Filtering events by date (today, the next hour) and by patient;
* Changing event status from “planned” to “done” and from “planned” to “canceled”.

When the event status is transferred to “canceled”, the nurse must indicate the reason for the cancellation in the field “comment”.

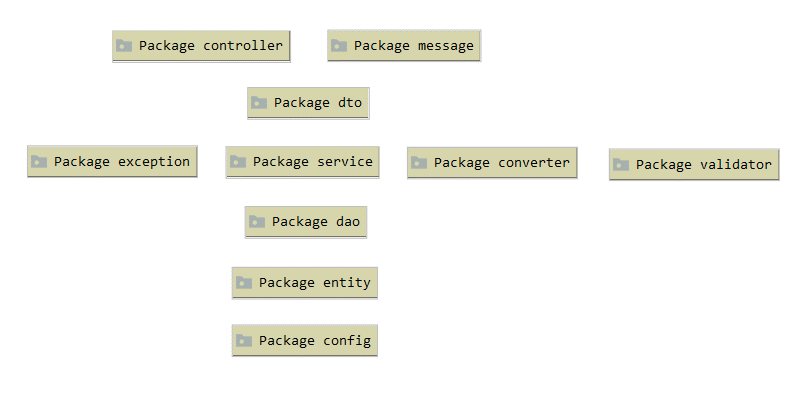
**5 Architecture**

The application has a multi-layermonolit architecture.



* Presentation layer is represented by Controllers and JSP pages with JavaScript.
* Business logic layer consists of the Services and Exceptions.
* Data Access Layer is represented by DAO and Entities modules.

**6 Modules**



• Package Controllers contains controllers for GET, POST queries from front-end. It is the transmission link between services and front-end. It also contains queries for updating, setting and removing operations using services.

• Package Messages contains classes for sending and receiving messages from other applications using JMS(WildFly).

• Package DTO contains DTO classes. The instances of these classes pass between services and controllers.

• Package Services contains classes that describe business logic and processing data from data base.

Package Converter contains classes that perform conversions between model data objects and data transfer objects.

Package Validator contains classes for checking validity of data.

• Package Exceptions contains custom exception classes for project logic.

• Package Dao contains classes that is used for communication with the database.

• Package Entities contains classes that describe project entities.

• Package Config contains configurations classes. This project use configuration classes instead of xml configurations files.

**7 Presentation Layer**

User Interface is responsible for displaying information to the user and interpreting the user’s commands. The system uses JavaServer Pages for rendering view. It enables to write dynamic, data-driven pages for web applications. Adding the JavaServer Pages Standard Tag Library provides support for common, structural tasks such as iteration and conditionals.

There is autocomplete in some forms of the application. It is implemented with the help of Ajax Autocomplete for jQuery that allows to easily create autosuggest boxes for text input fields.

In the application there are used the following libraries:

• Jquery;

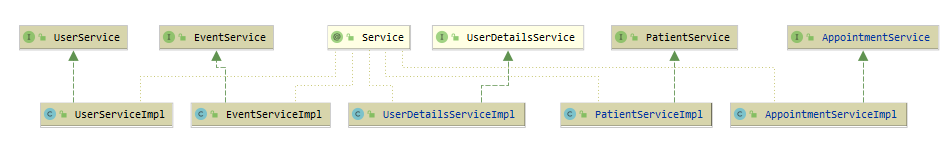
• Jquery-ui;

• Sweetalert;

• Bootstrap.

**8 Business Logic Layer**

Business requirements are handled by service layer. Service layer contains interfaces with their implementation. Service has references to DAO interfaces. The diagram below depicts the relations between all service interfaces and their implementation:



Service layer methods are annotated with @Transactional annotations. Creating and updating operations have this annotation with Propagation.REQUIRED. Other methods are read-only (@Transactional(readOnly=true)).

Business logic is implemented using the following services:

* UserService provides method for operating with users.
* EventService contains operations for getting, creating, updating events and searching by fields and "like" fields.
* UserDetailsService loads user-specific data. The interface requires only one read-only method, which simplifies support for new data-access strategies.
* PatientService is used for creating, updating and getting patients.
* AppointmentService responds for creating, updating and getting appointment. This service is closely connected with EventService since every manipulation with appointment leads to event changes.

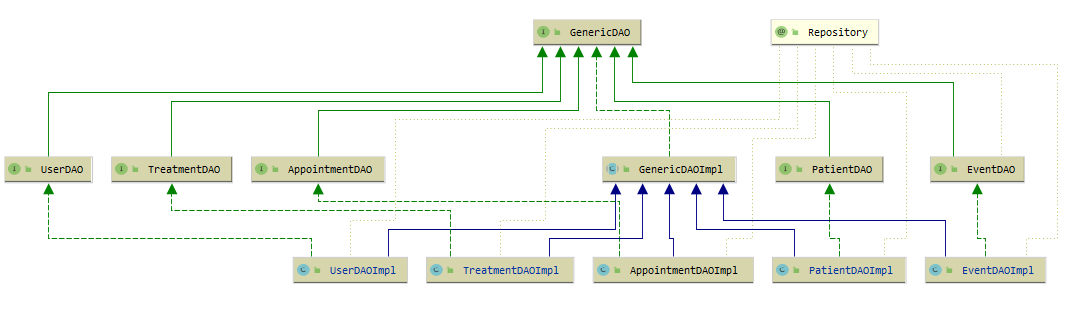
**9 Data Access Layer**

Data Access Layer is represented by DAO and Entities modules.

Entities are located in the entity package. Each entity is described by a set of attributes and may be related with other entities. There are 5 entities in the project:

* Appointment has attributes: id, startDate, endDate, days, time, dosage, status, patient, treatment and has many-to-one relationship with Patient and one-to-one relation with Treatment.
* Event has date, status, comment and has a many-to-one relationship with Appointment.
* Patient’s attributes are id, firstName, lastName, diagnosis, healthInsuranse, startDate, status and doctor. It is in a many-to-one relationship with User.
* Treatment has attributes: id, name, type, dosageForm.
* User has id, firstName, lastName, role, email and password.

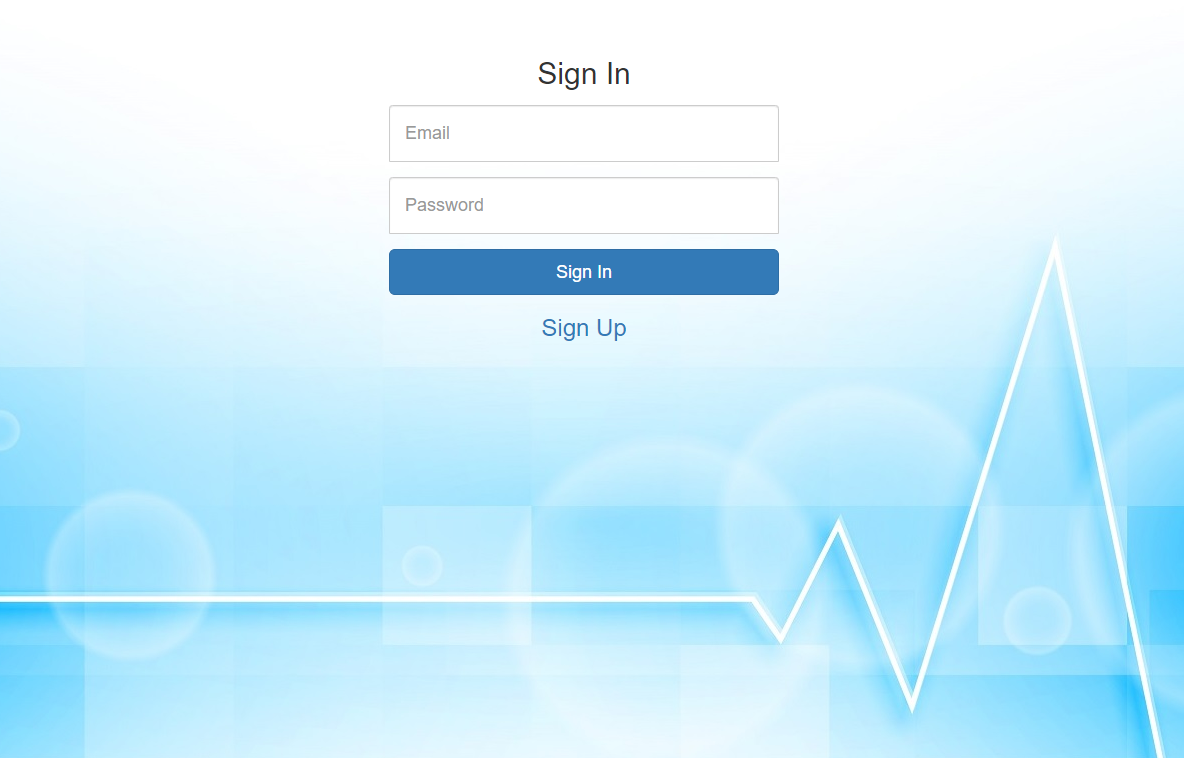
Data Access Object classes are responsible for encapsulating the details of the persistence layer and providing a CRUD interfaces for entities. JPA entity manager in DAO classes is used for performing operations in the database.



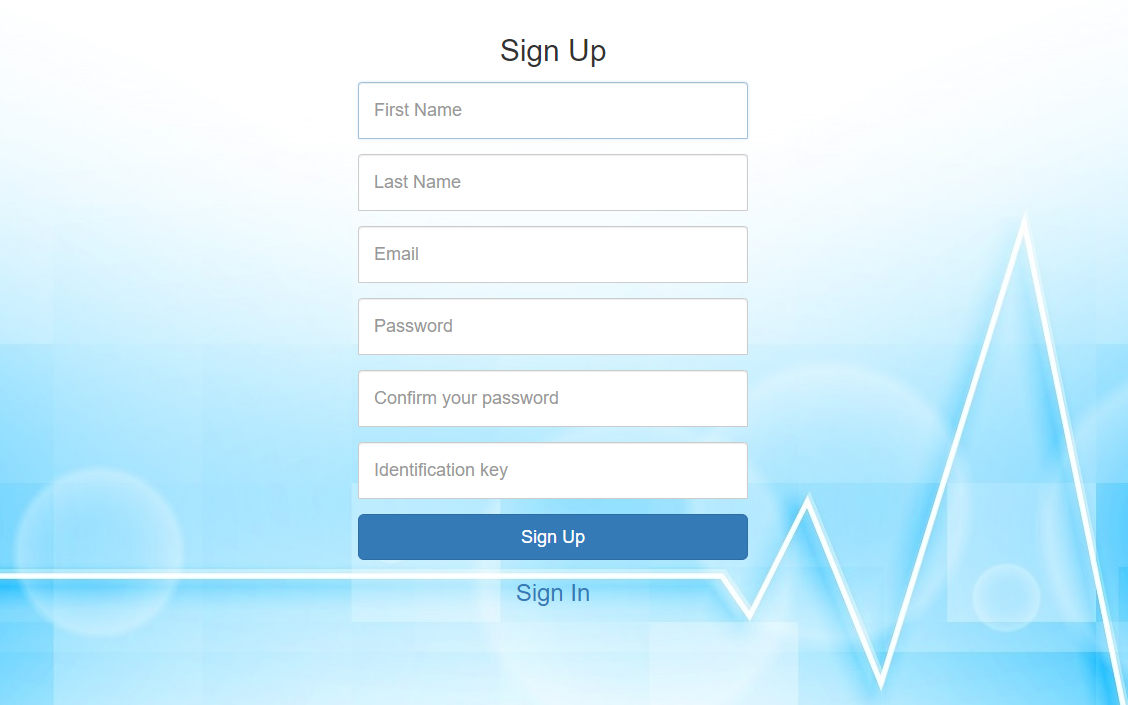
GenericDAO contains parametrized common methods for all DAO classes. Other classes extend GenericDAO with own unique methods.

**10 Main Pages**

**10.1 Authorization**

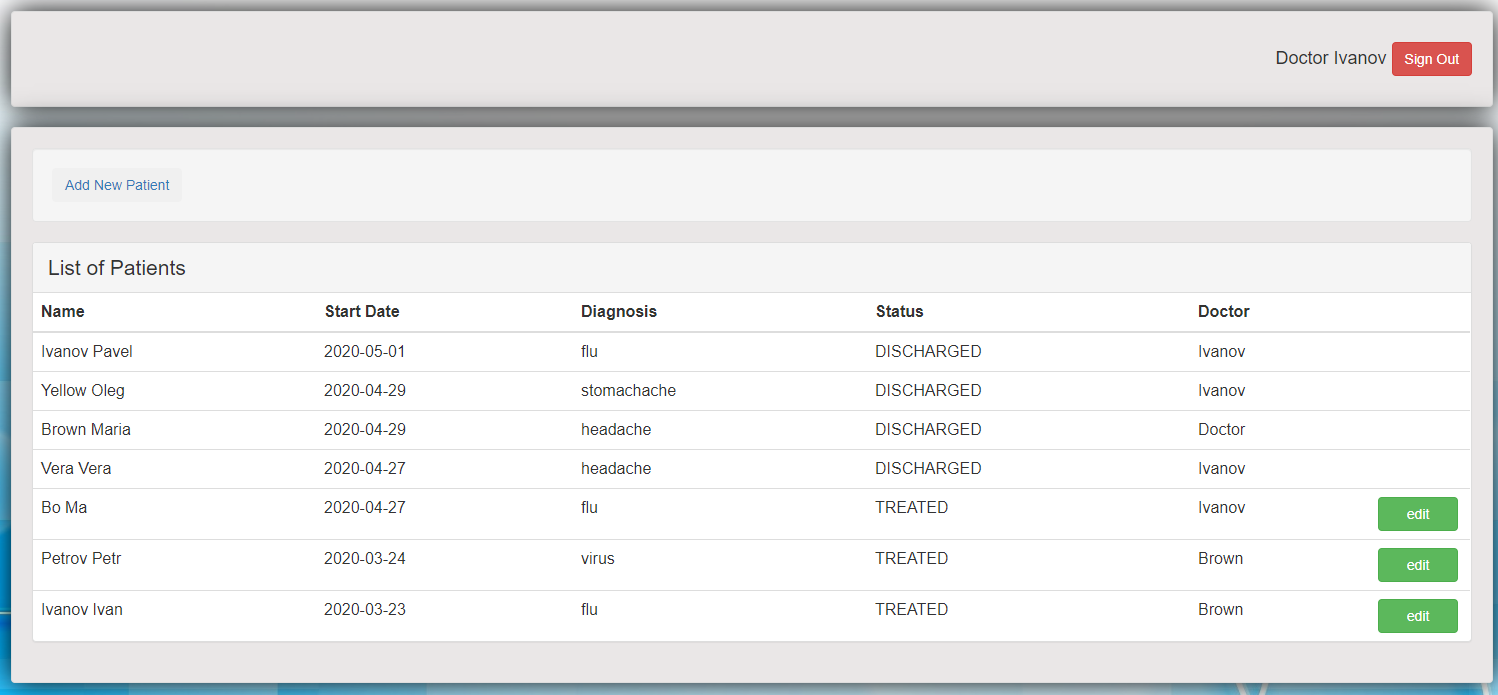


**10.2 Registration**

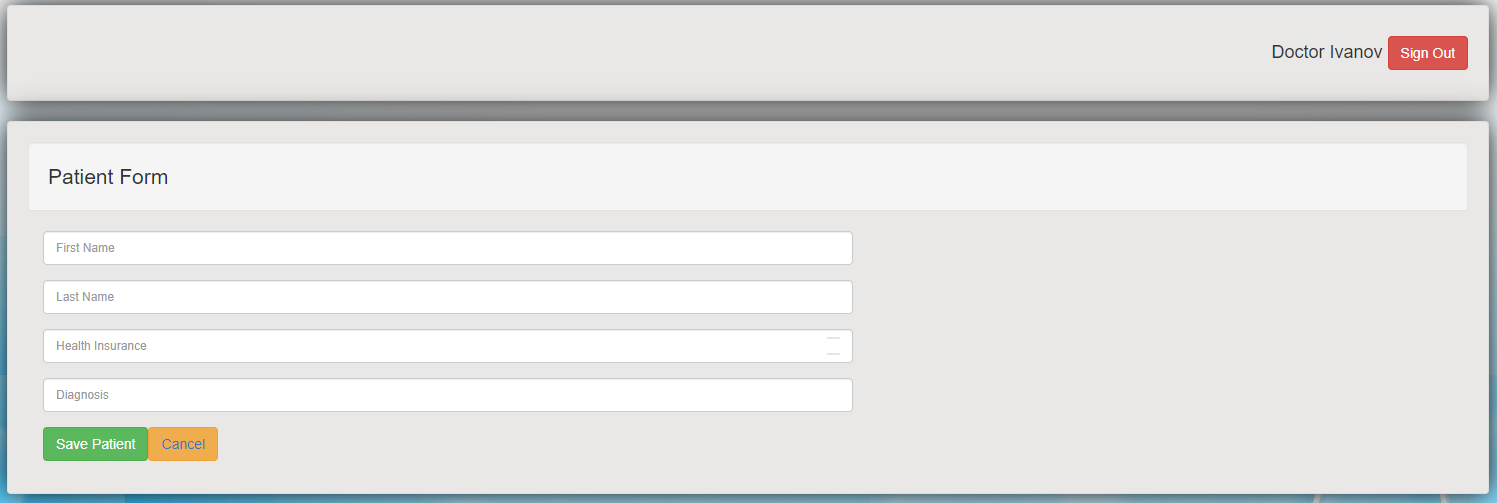


**10.3 Doctor GUI**

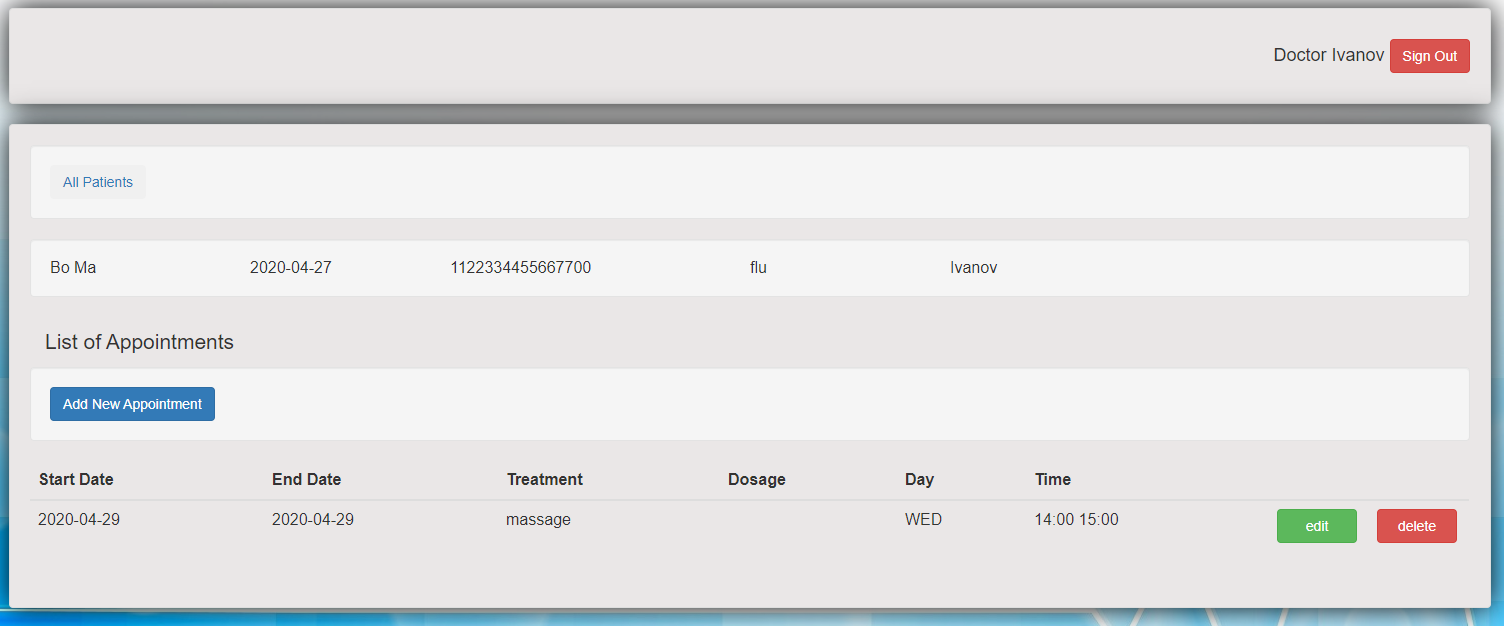
**10.3.1 Patient List**



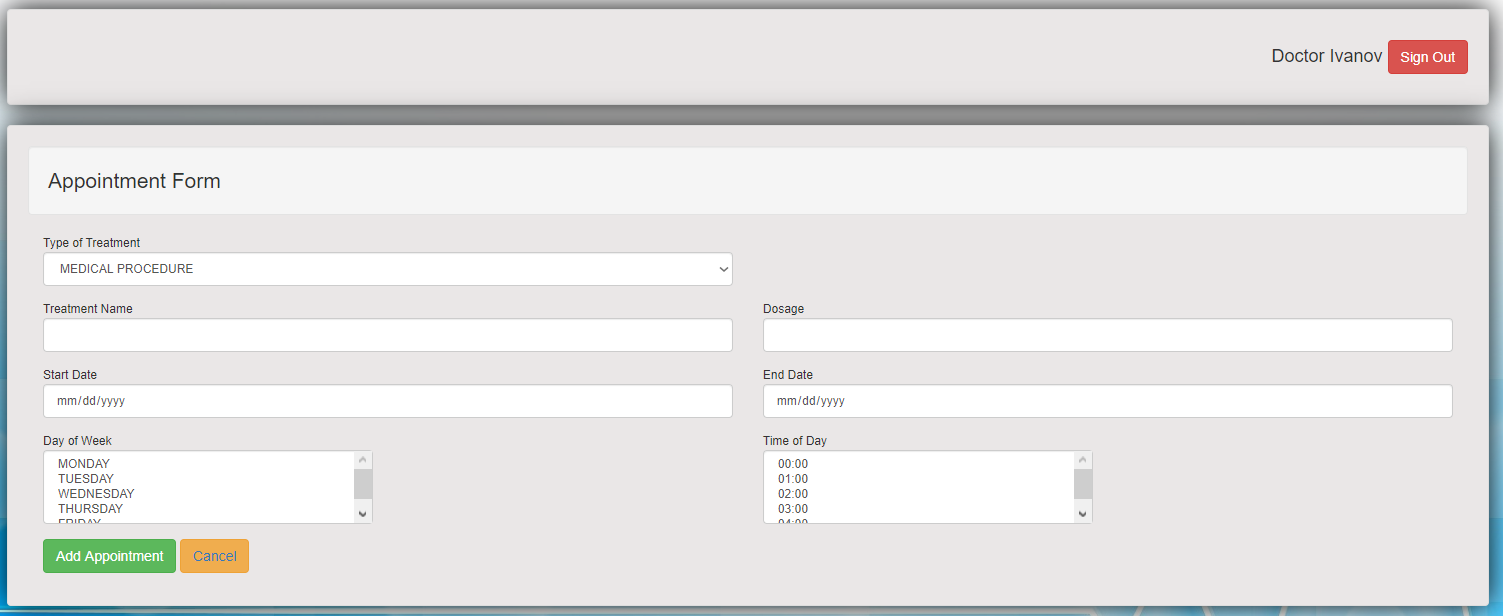
**10.3.2 Adding Patient**



**10.3.3 Patient Page**

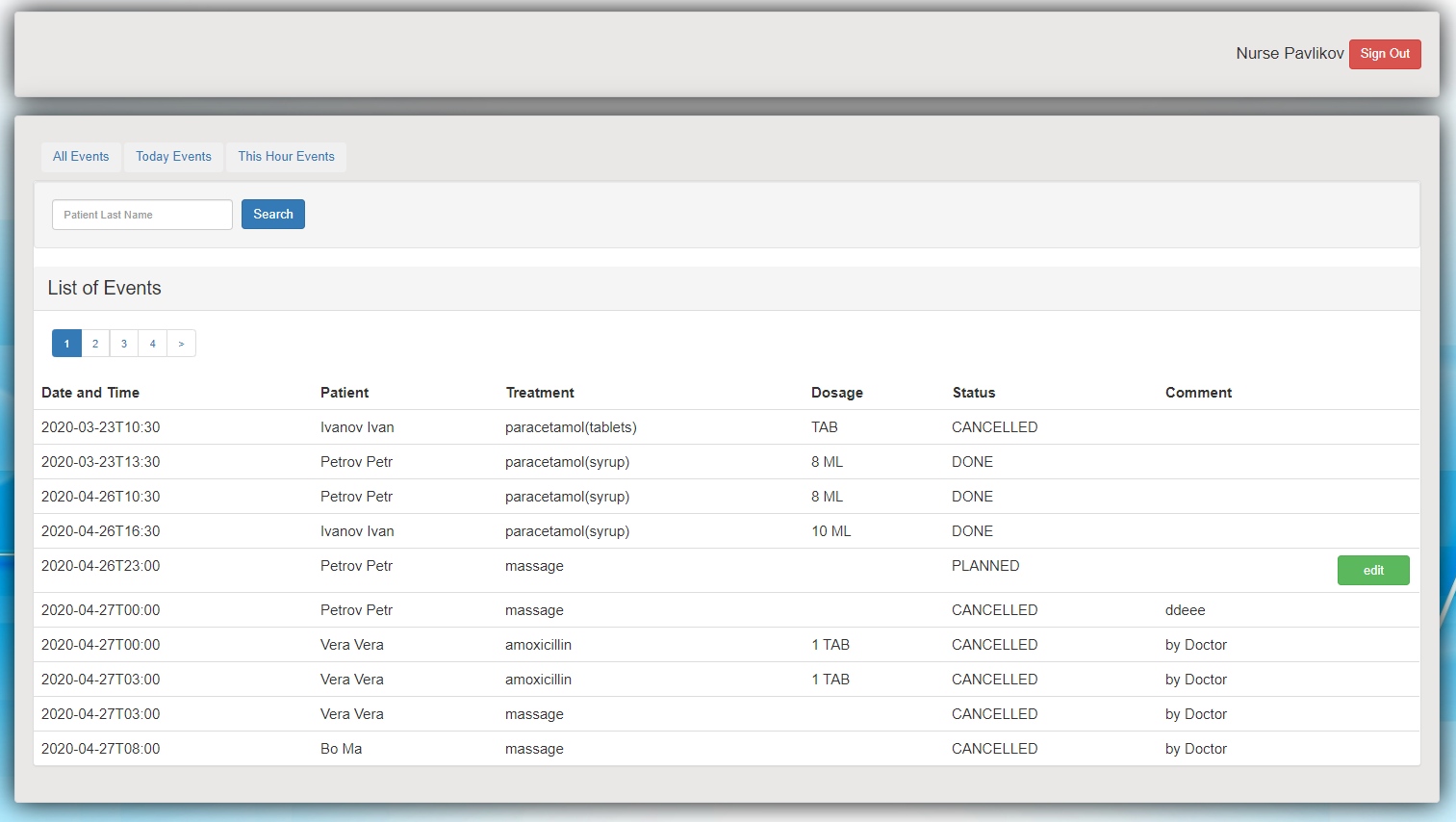


**10.3.4 Adding and Updating Appointment**

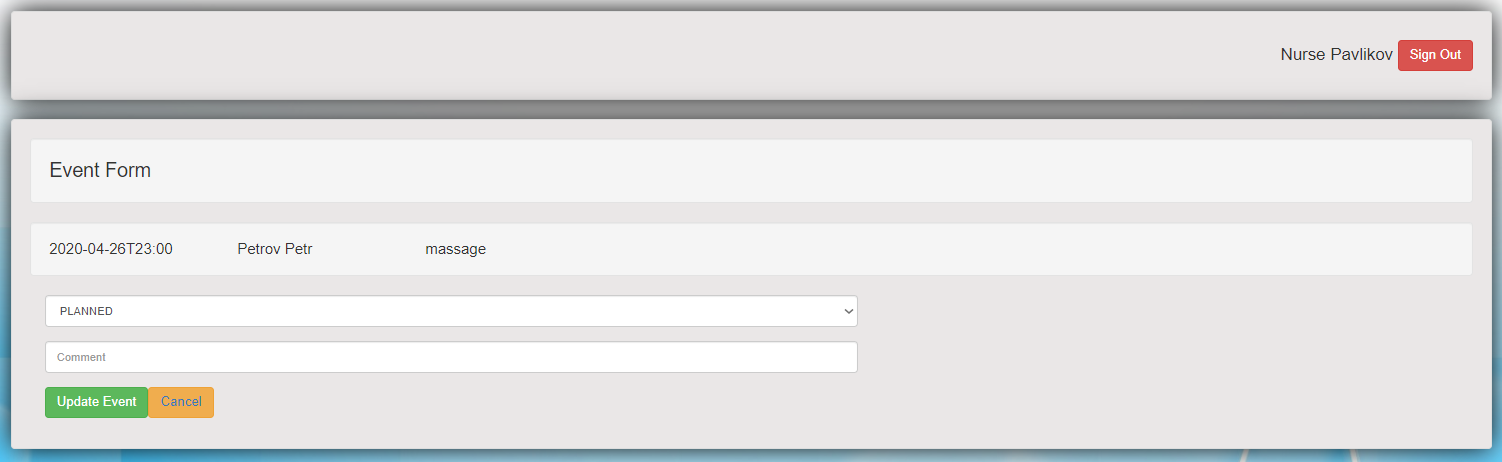


**10.4 Nurse GUI**

**10.4.1 Event List**



**10.4.2 Updating Event**



**11 JUnit Tests**

Service testing is organized with the help of MockitoJUnitRunner.class. There are 4 test classes with 24 tests:

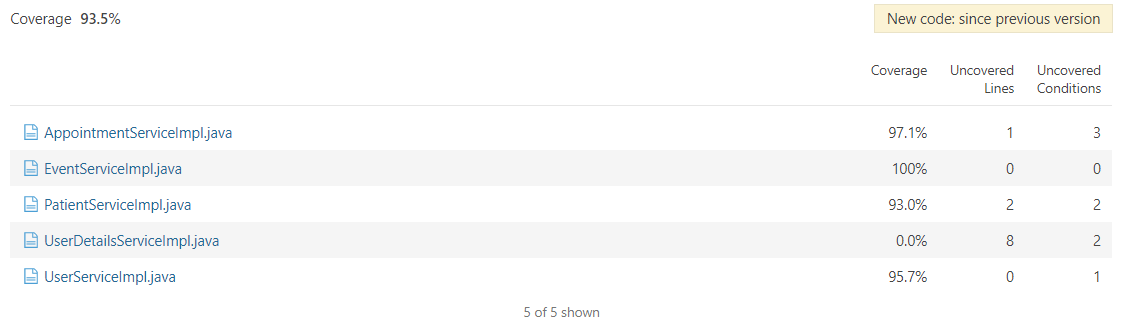
• UserServiceTest tests method from UserService: 4 tests;

• EventServiceTest provides tests for EventService methods: 9 tests;

• PatientServiceTest tests PatientService: 5 tests;

• AppointmentServcieTest tests AppointmentService methods: 6 tests.

Java Test Coverage for services is 93,5%.

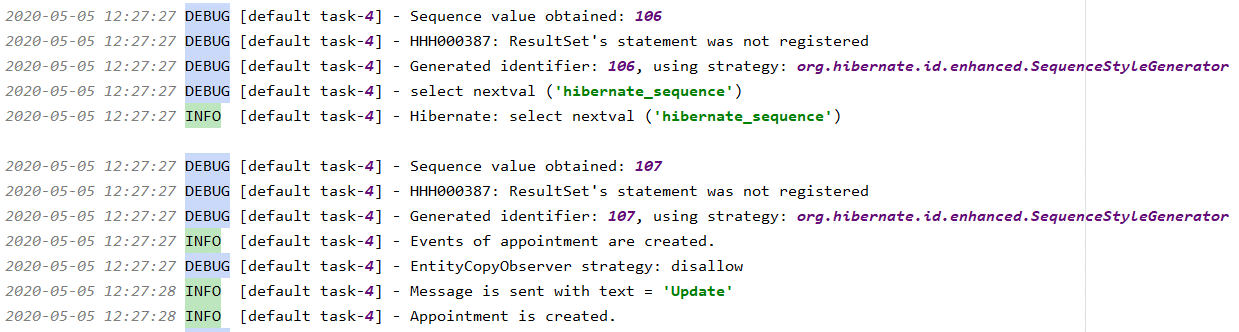


**12 Building and deployment**

The build tool of the application is the Apache Maven build system. Docker is used to containerize the applications for automated deployment. There is Docker Compose that allows to define all containers and their dependencies in a Compose file. Therefore, the Compose file provides starting all the containers at once. Moreover, starting them can be order dependent. For example, the database should be running before the application.

**13 Logging**

Log4j is used as a logging utility. It helps to do any debugging or see what’s going on inside the system. Logging information is stored in multiple files. There are 3 log4j files for publishing the log messages. They are filled one after another upon reaching the maximum size. Their maximum size is 2 MB.



**14 Code Statistics**

Statistics from Sonar is:

