Rehab web application

Technical solution description

Content

- Overview and tasks
- Technologies and frameworks
- Additional implemented features
- Database
- Implementation of events generation
- UI
- Services
- DAO
- Tests
- Screenshots
- Build and depoyment
- Patterns
- Sonar statistics
- Improvement and new features

Overview and tasks

A web application with a graphical interface modulates the work of the information system of the rehabilitation center. User can sing up as doctor or as nurse and depending on this, he can perform various operations: the doctor records patients and adds appointments, and the nurse processes events.

Technologies and frameworks

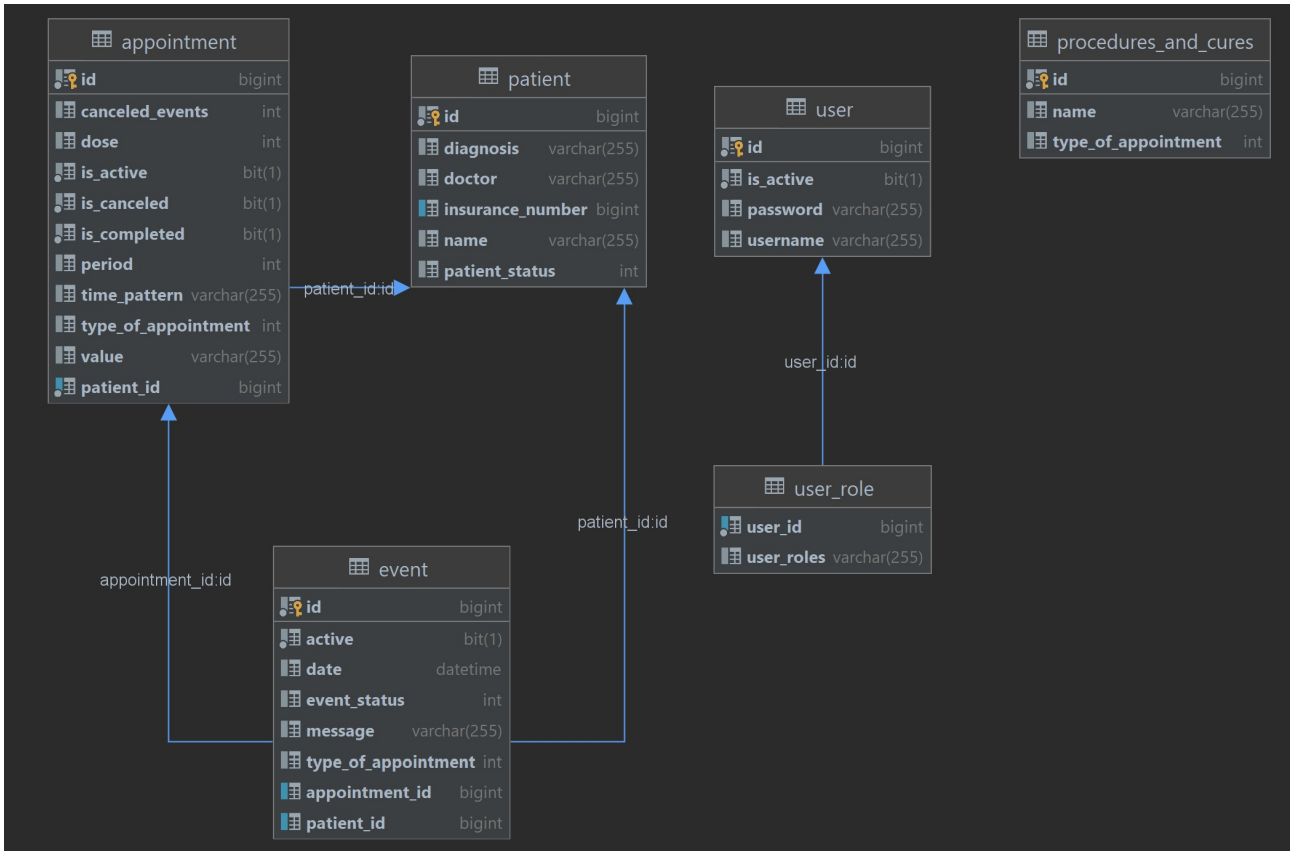
- Spring boot
- Spring Data
- Spring Security
- Maven
- Hibernate
- MySQL
- Thymeleaf
- Apache Active MQ
- JUnit
- Mockito
- Selenium
- Lombok

Additional implemented features

- Administrator panel, where user who signed up as admin can add new procedures and cures through graphical interface. Also he can register new doctors or nurses.

- New section 'Discharged patients' where doctor can return to treatment patients who have been discharged without need to add them again.
- Doctor now can track appointment progress and see appointment events statuses by clicking on appointments title.
- When events marked as 'PLANNED' end, appointment marks as completed and doctor can see number of canceled events.
- When doctor clicks to 'Discharge patient' he see all unfinished appointments(if they exist) and discharge confirmation button.

Database



Diagram

- Patient includes 'name', 'patient status', 'doctor' and unique field 'insurance number'. In one-to-many relation with 'Event' and 'Appointment'

- Appointment includes 'canceled events'(number), 'dose'(for procedures always equals zero), 'isActive', 'isCompleted', 'isCanceled', 'isCompleted', 'period', 'time pattern', 'type of

appointment' and 'value'. In many-to-one relation with Patient and one-to-many relation with Event

- Event includes 'active', 'date', 'event status', 'message'(if canceled), 'type of appointment'. In many-to-one relations with Appointment and Patient.

- User includes 'isActive', 'username', 'password' in one-to-many with user roles

- Procedures and cures includes 'name' and 'type of appointment'

Implementation of events generation

Method 'createEvents' invokes when user creates appointment. It generates date list focusing on information about appointment (which user inputs through UI) and parameter 'pastEvents'. Method uses static method 'generateDateList' for generating dates focusing on weekdays. Checking whether the date corresponds to one of the days of the week starts from tomorrow, until the moment when the required number of events are created. If user creates appointment - 'pastEvents' always equals to zero. In other case, when user updates appointment, method 'createEvents' also invokes and 'pastEvents' equals to number of all events, which status is not 'Planned'.

UI

This is a web application and user interface is done with html, using Bootstrap 5 for CSS.

Services

Application has a service layer, that contains:

- AppointmentService – for working with appointments
- DispatcherService – for send messages to queue. Application sends list of today events in JSON format every time, when user change appointments or events

- EventService – for working with events
- PatientService – for working with patients
- ProceduresAndCuresService – for working with items in ProceduresAndCures table
- ReceiverService – for receive message from queue and invoke send message from DispatcherService
- UserService – for working with users(doctors and nurses)

DAO

DAO layer has entities: AppointmentRepository, EventRepository, PatientRepository, ProceduresAndCuresRepository and UserRepository which implement CRUD methods. EventRepository also extends from PagingAndSortingRepository for pagination.

Tests

Application has unit tests:

- AppointmentServiceImplUnitTests – tests correct work of AppointmentService methods(information sets properly after appointment creation, updating properly etc.)
- EventServiceImplUnitTests – tests correct work of EventService methods(events generating, finding recent and today events, completing , canceling, finding by patient)
- PatientServiceImplUnitTests – tests correct work of PatientService method(patient create, discharge, checking if the patient still has unfinished appointments)
- ProceduresAndCuresImplUnitTests – tests correct work of ProceduresAndCuresService(adding items)
- UserServiceImplUnitTests – tests correct work of UserService(creating user and finding all user with doctor roles)

Application has three integration tests:

- AppointmentServiceImplTest
- EventServiceImplTest

- PatientServiceImplTest

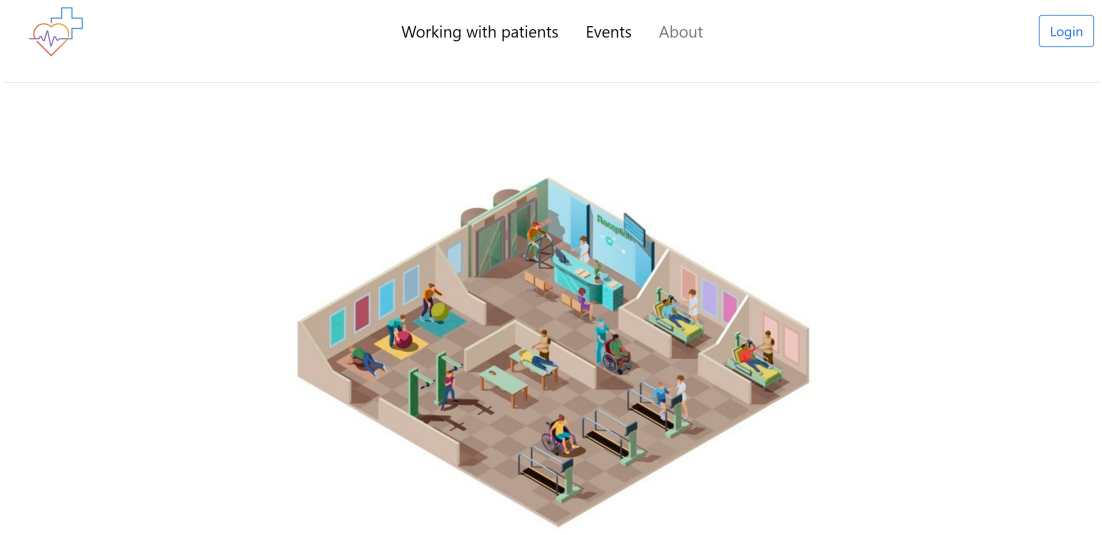
  They work out scenarios for working with entities in the database

  Application has UI tests

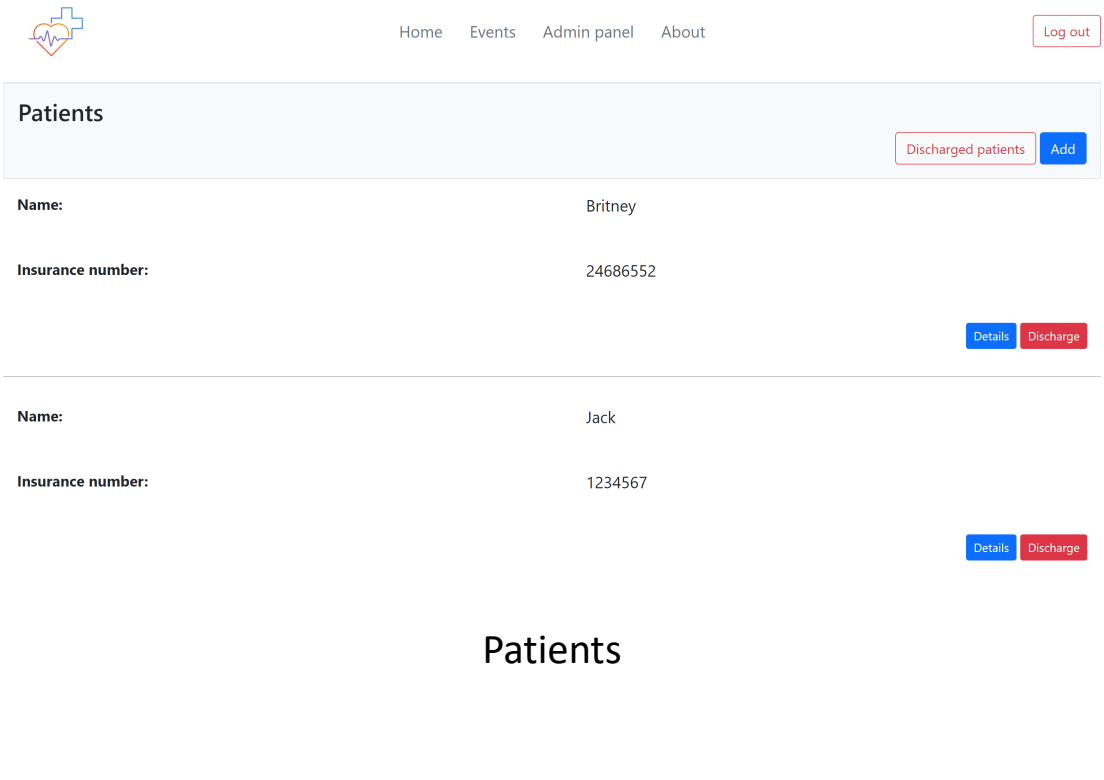
- SeleniumTest

They check the correct display of elements in different parts when authorizing with different roles

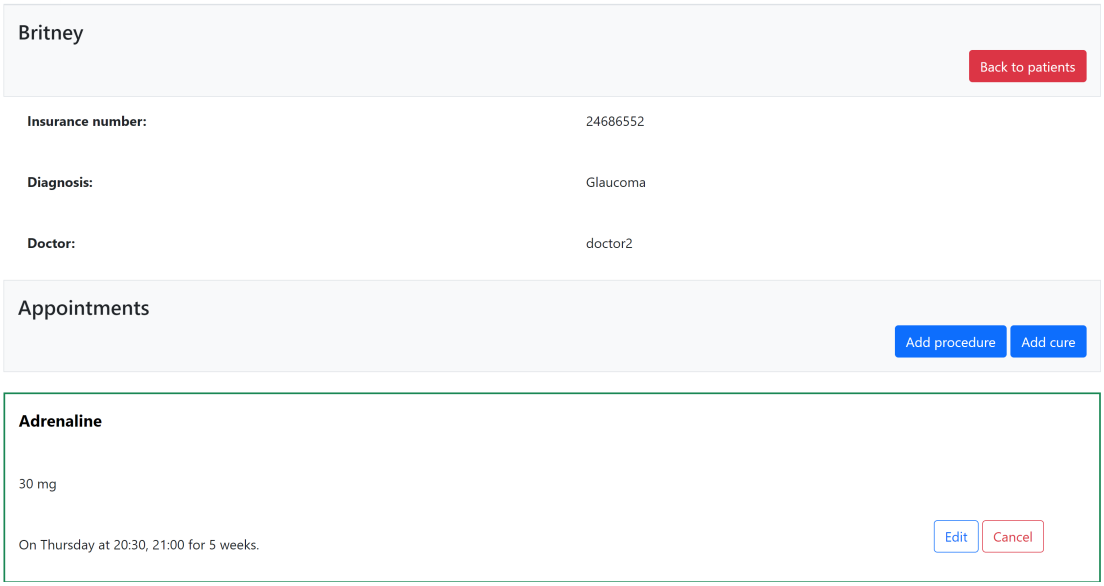
Screenshots



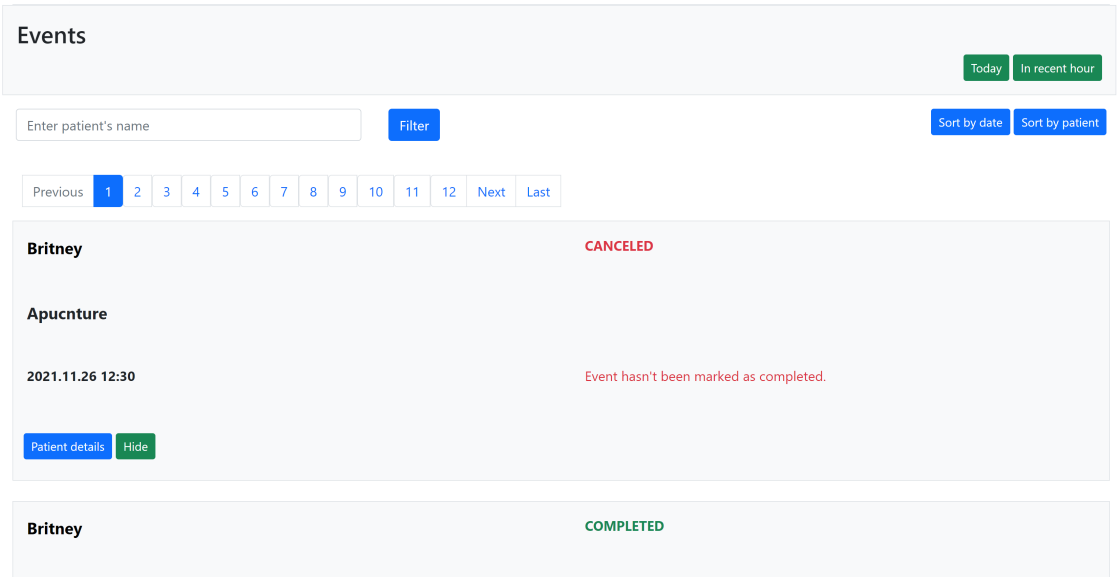
Main page



Patients



Patient details

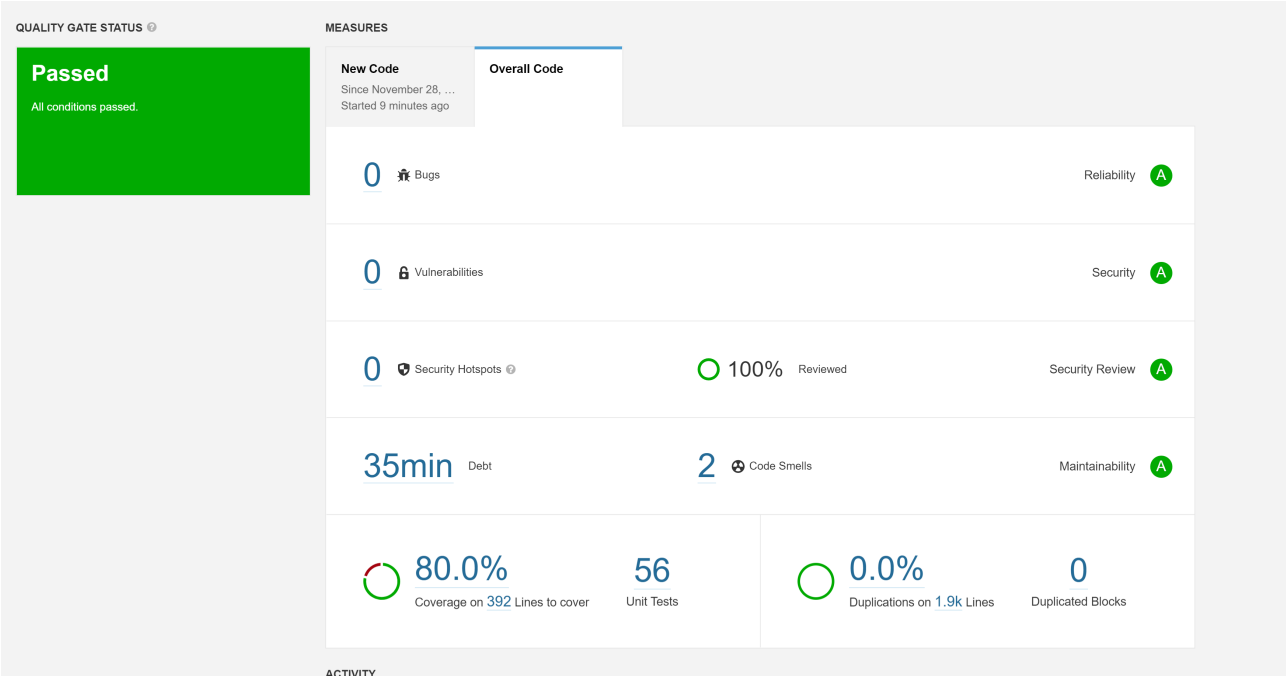


Events

Build and deployment

Application built with Maven and deployed on embedded Tomcat, provided by Spring Boot

Sonar statistics



Improvement and new features

In new versions I would like to add more flexible appointment settings, with editing and tracking events, information and contraindications for procedures and medications, information about diagnoses and their connection with procedures and medications, the ability to authorize patients with viewing the status of their treatment, and also the use of a more reliable password encoder.