eClinic

Technical solution description

Bogdan Sukonnov

T-Systems

Saint-Petersburg 2019

Content.

PROJECT GOALS

Application development for the medical rehabilitation clinic. The application should have cohesive data model, user friendly interface, separate access to different system's part, reliable system.

APPLICATION DESCRIPTION

Web application has three types of user: doctors and nurses. Admin has access to each part of the application and can do everything that the other users do.

Doctors can add and discharge patients, create, edit and cancel prescriptions. Also they have access to any information about patients, prescriptions and events.

Nurses has limited access to the service. After authentication a nurse can only see events, complete and cancel them.

There is an authentication mechanism in the system that control access to the service. Each user in application has access level that defines information for displaying and privileges.

Data of users stores in a reliable database.

USED TECHNOLOGIES

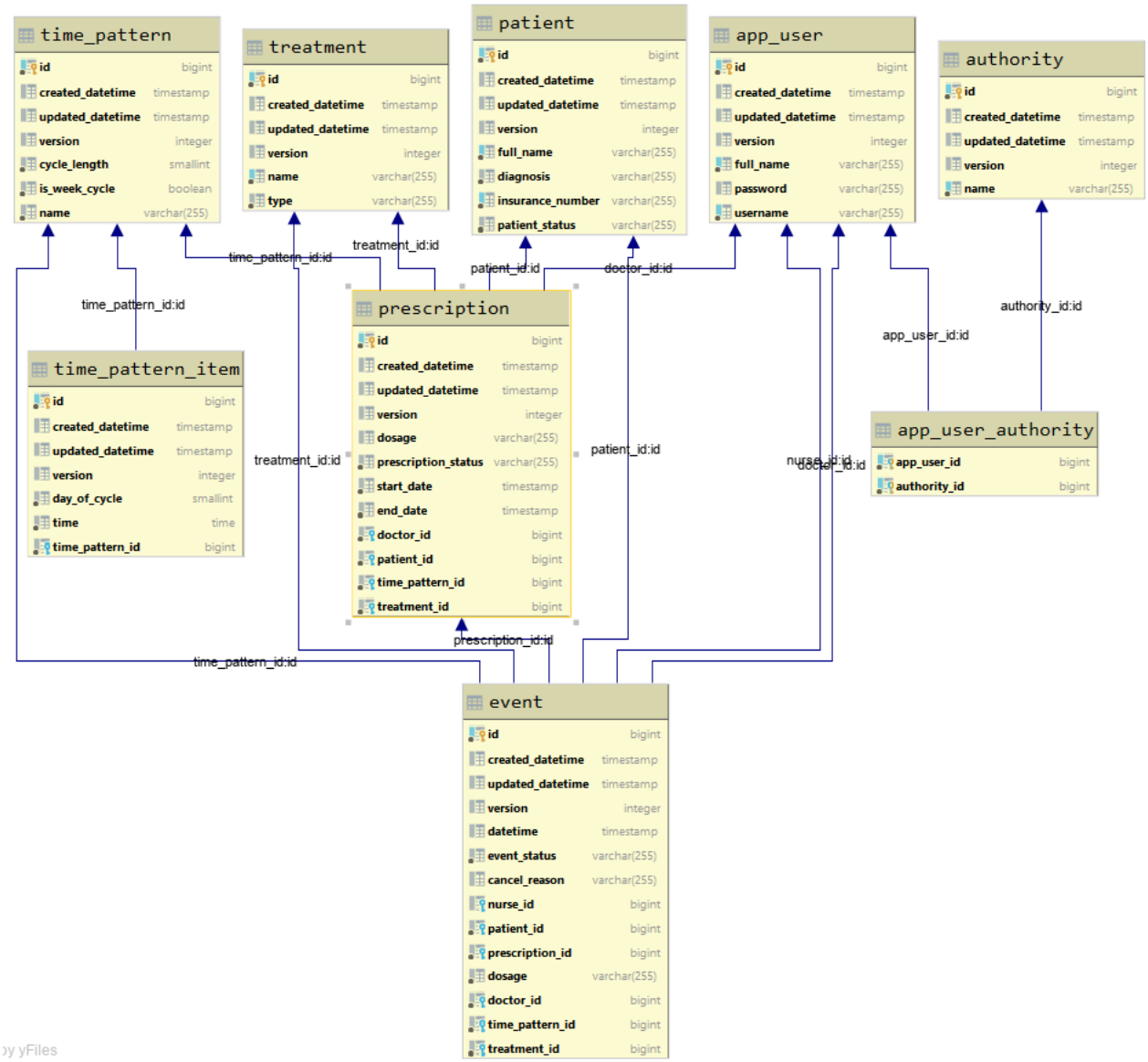
- Instruments:
    1. IDE - IntelliJ IDEA
    2. Maven
    3. Docker

Technologies:

1. Java 8
2. Spring
3. JSP
4. JPA
5. Hibernate
6. JavaScript

7. jQuery
8. JSF
9. REST
10. Spring security
11. ActiveMQ
12. Bootstrap4
13. PostgreSQL
14. EJB
15. Junit
16. Mockito
17. Jenkins
18. SonarQube
19. Tomcat
20. WildFly
21. Log4j2
22. Flyway

DATABASE MODEL



SYSTEM INFRASTRUCTURE

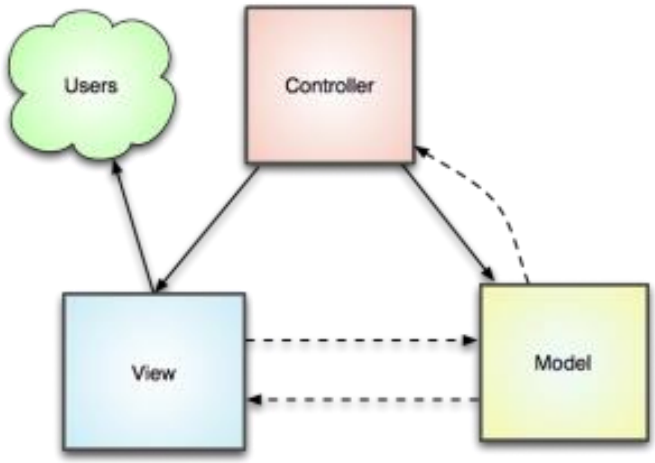
- Front-end (for doctors and nurses):
  1) Web-page structure - HTML, JSP
  2) Page-design – Bootstrap4
  3) Dynamic contents: JavaScript, Jquery

- Back-end (server based level):
  1) Application server - Tomcat
  2) Database - PostgreSQL
  3) Server logic - Spring Framework

- Client board application:
  1) Web-pages - JSF
  2) JMS - ActiveMQ
  3) Application server - WildFly
  4) Server logic - EJB
  5) WS - REST

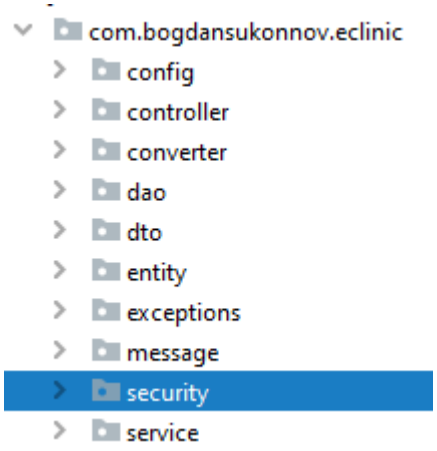
SYSTEM ARCHITECTURE

Architecture of server-based part presented by MVC - design pattern.

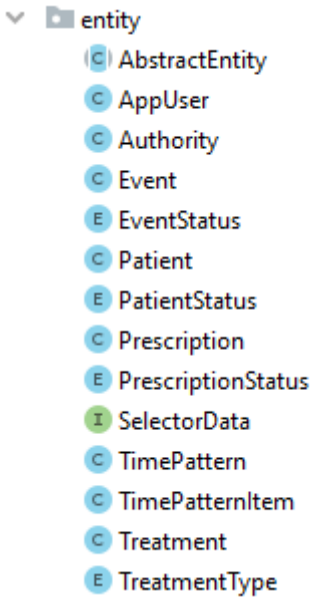


Class structure

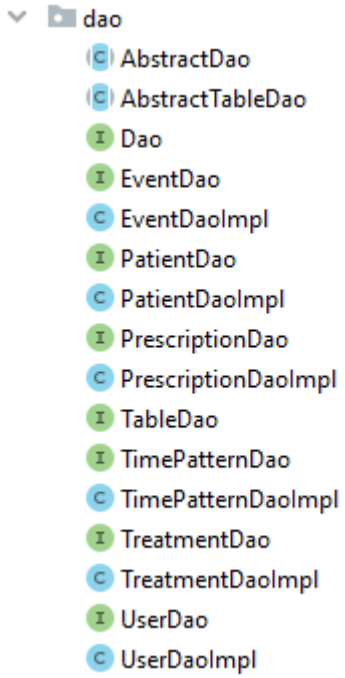
According MVC-pattern application has next structure:



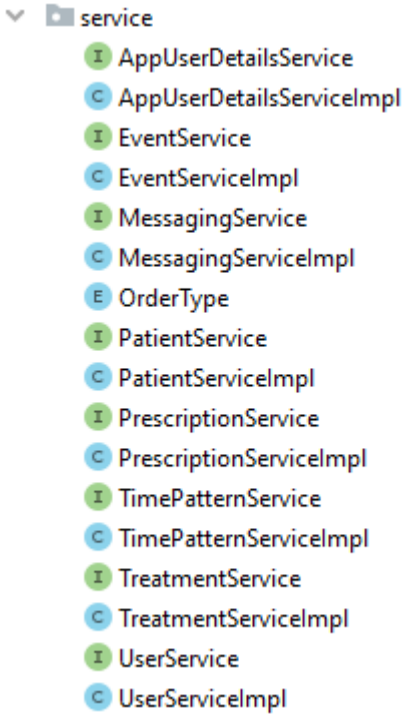
Model level:



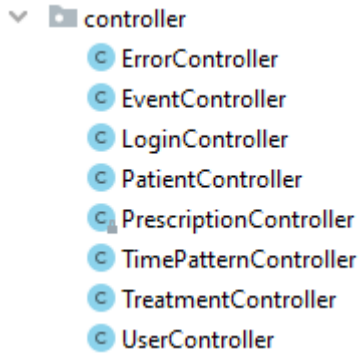
Model-service level:



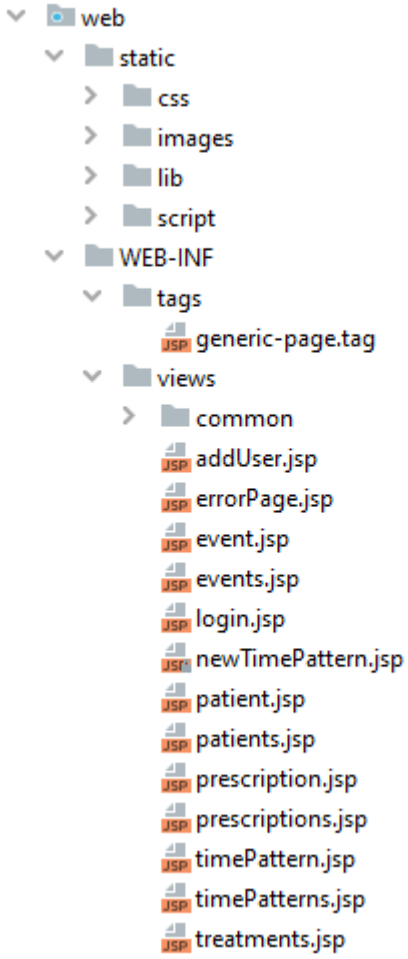
Service level:



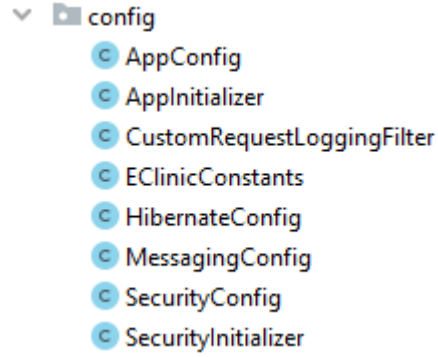
View-service level:



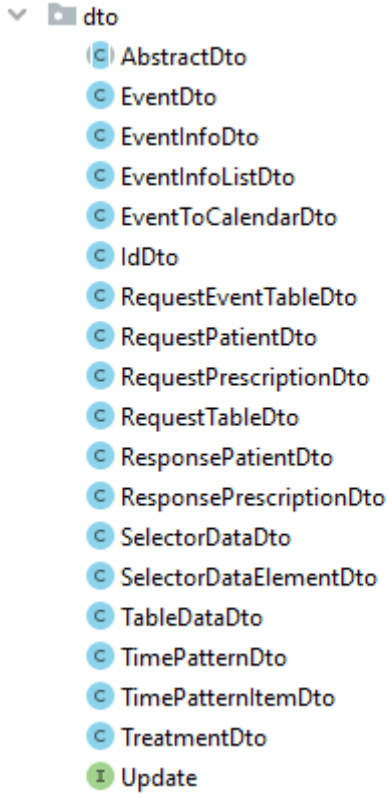
View level:



Configuration:


config
AppConfig
AppInitializer
CustomRequestLoggingFilter
EClinicConstants
HibernateConfig
MessagingConfig
SecurityConfig
SecurityInitializer

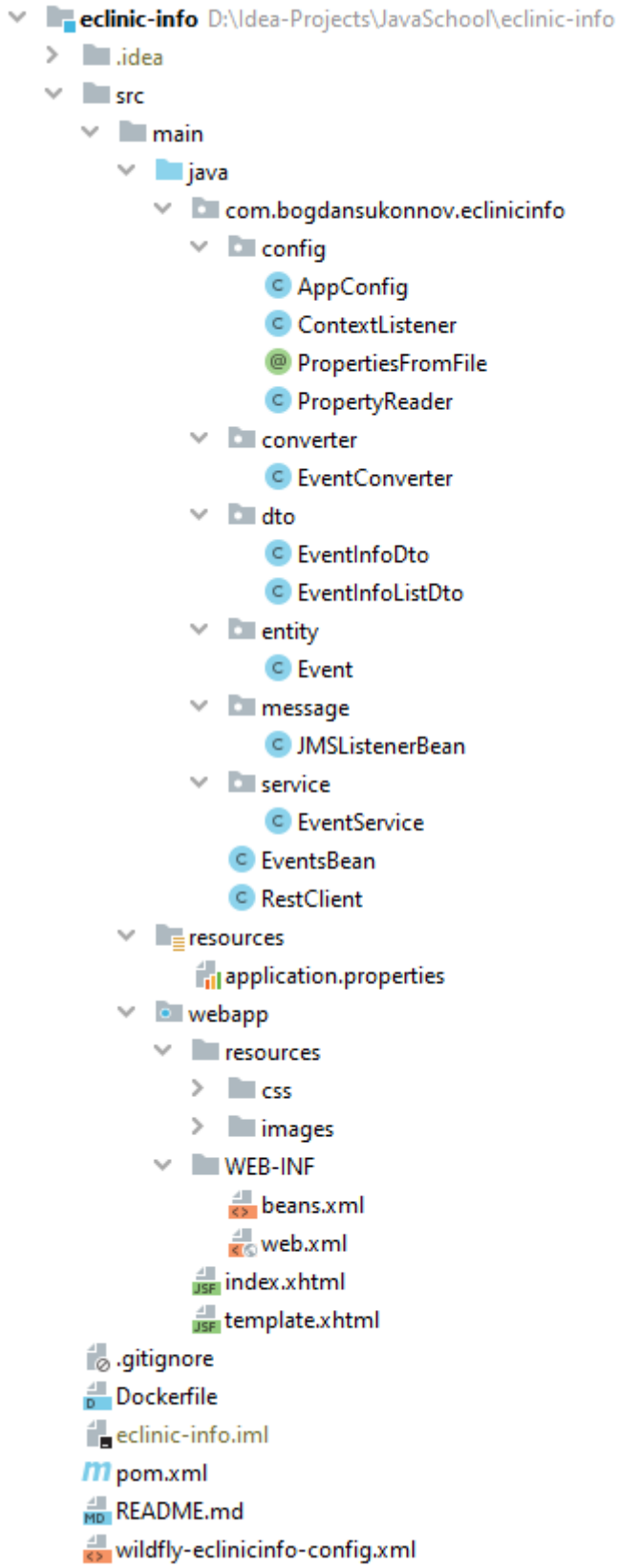
DTO:


dto
AbstractDto
EventDto
EventInfoDto
EventInfoListDto
EventToCalendarDto
IdDto
RequestEventTableDto
RequestPatientDto
RequestPrescriptionDto
RequestTableDto
ResponsePatientDto
ResponsePrescriptionDto
SelectorDataDto
SelectorDataElementDto
TableDataDto
TimePatternDto
TimePatternItemDto
TreatmentDto
Update

Rest client applications:

1.

2.


eclinic-info D:\Idea-Projects\JavaSchool\eclinic-info
.idea
src
main
java
com.bogdansukonnov.eclinicinfo
config
AppConfig
ContextListener
PropertiesFromFile
PropertyReader
converter
EventConverter
dto
EventInfoDto
EventInfoListDto
entity
Event
message
JMSListenerBean
service
EventService
EventsBean
RestClient
resources
application.properties
webapp
resources
css
images
WEB-INF
beans.xml
web.xml
index.xhtml
template.xhtml
.gitignore
Dockerfile
eclinic-info.iml
pom.xml
README.md
wildfly-eclinicinfo-config.xml

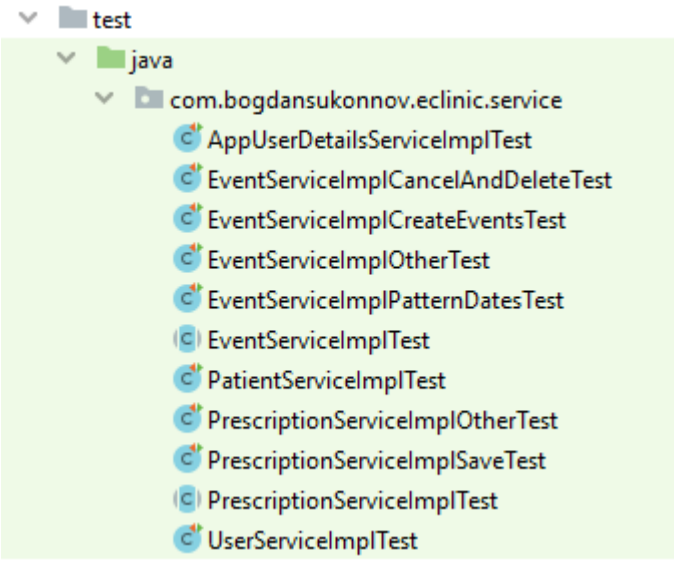
ADDITIONAL FEATURES

1. Docker. All parts of the application are stored in docker containers and starts up by docker-compose command

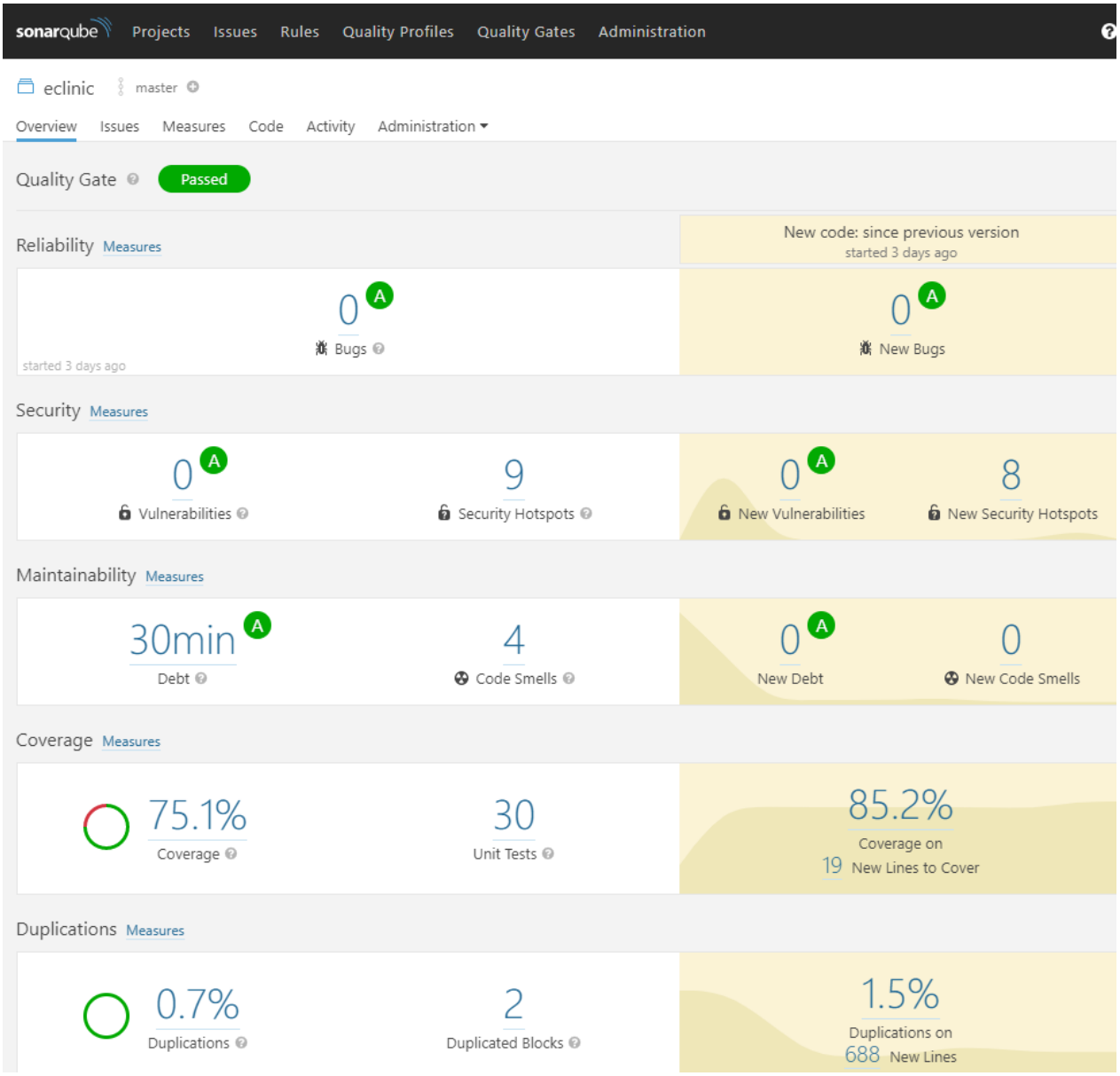
2. Jenkins + SonarCube. Jenkins starts build automatically after the push to bitbucket repository and check code with SonarScanner plugin.

CODE QUALITY

Tests:



Sonar report:



BUILD AND DEPLOY

To build the application run the command in a terminal *mvn clean install flyway:migrate*

Docker-compose up command is a part of the install maven goal.

FUTURE IMPROVEMENT

1. Refactoring and optimization code.
2. Adding new functionality