МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет по учебной практике
Тема НИР: Разработка, развёртка и поддержка приложения, написанного на Java с использованием Spring.

Студентка гр. 5304 _____ Фомченко О.А.

Санкт-Петербург
2020

ЗАДАНИЕ
НА НАУЧНО-ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ

Студент Фомченко О.А.
Группа 5304

Тема работы: Разработка, развёртка и поддержка приложения, написанного на Java с использованием Spring.

Содержание пояснительной записки:
«Содержание», «Определения, обозначения и сокращения», «Введение», «Постановка задачи», «Результаты работы в весеннем семестре», «План работы на осенний семестр, «Вывод», «Список литературы»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Сроки выполнения НИР: 05.02.2020 – 02.06.2020

Дата сдачи отчета: 02.06.2020

Дата защиты отчета: 02.06.2020

Студент _____ Фомченко О.А.

Преподаватель _____ Заславский М.М.

АННОТАЦИЯ

В данной работе представлена идея и реализация приложения с использованием Spring и план настройки CI/CD данного приложения.

SUMMARY

This work presents the idea and implementation of Spring application with a plan of integrating CI/CD settings.

СОДЕРЖАНИЕ

ОГЛАВЛЕНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

- **Непрерывная интеграция (Continuous Integration – CI)** — это практика разработки программного обеспечения, которая заключается в слиянии рабочих копий в общую основную ветвь разработки и выполнении частых автоматизированных сборок проекта для решения интеграционных проблем [3];
- **Непрерывная доставка (Continuous delivery – CD)** – является продолжением непрерывной интеграции, дающая возможность быстро и эффективно выпускать новые изменения для своих клиентов [4];
- **Непрерывное развертывание (Continuous deployment – CD)** – идет на один шаг дальше, чем непрерывная доставка. С этой практикой каждое изменение, которое проходит все стадии производственного процесса, передается клиентам [2];
- **Spring** – это платформа для разработки приложений на языке Java, позволяющая быстро создавать web-приложения [1];
- **Travic CI** — распределённый веб-сервис для сборки и тестирования программного обеспечения, использующий GitHub в качестве хостинга исходного кода [10];
- **Docker** — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации. Позволяет «упаковать» приложение со всем его окружением и зависимостями в контейнер, который может быть перенесён на Linux-систему, а также предоставляет среду по управлению контейнерами [5];
- **GitHub** — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки [10];
- **FlyWay –** инструмент для управления миграциями реляционных баз данных [8];
- **SonarQube -** платформа с открытым исходным кодом для непрерывного анализа и измерения качества кода [9].

ВВЕДЕНИЕ

Spring – это самый большой современный фреймворк для разработки web-приложений на Java. Это удобный и мощный инструмент для быстрого и эффективного создания, готового для продакшена, решения.

Но сам фреймворк не дает никаких возможностей для эффективного контроля версий, развертывания и интеграции изменений в уже готовое приложение. Так же, любое, достаточно серьезное приложение разрабатывается целой командой разработчиков, и в целях повышения эффективности, команды используют систему контроля версий, непрерывную интеграцию и непрерывное развертывание.

Система контроля версий – это программное обеспечение, обеспечивающее возможность управления новыми версиями программы, сохранение информации о создателе и предоставляющее возможность откатиться до предыдущей версии, при необходимости.

Непрерывная интеграция (CI) — это процесс автоматизации сборки и тестирования кода каждый раз, когда член команды вносит изменения в управление версиями.

Непрерывное развертывание (CD) можно рассматривать как расширение непрерывной интеграции, и оно представляет собой процесс автоматического развертывания приложения после успешного завершения CI.

Цель CI/CD состоит в том, чтобы минимизировать время между написанием одной новой строки кода во время разработки и началом использования этого нового кода живыми пользователями в производстве.

1. ПОСТАНОВКА ЗАДАЧИ

Целью данной работы является разработка приложения на основе фреймворка Spring, его изучение, а также изучение современных CI/CD инструментов.

Проблема, которой обусловлена эта работа – это то, что ежедневная жизнь Java разработчика заполнена однообразными и повторяющимися задачами, которые отлично автоматизируются, такие как создание и настройка часто используемых сервис-объектов, настройка конфигурации и информационной безопасности приложения, прогон тестов, проверка стиля кода. Именно этим обусловлена актуальность использования Spring и настройки CI/CD приложения.

Поставлены следующие задачи:

- обзор наиболее подходящих модулей и решений;
- разработка web-приложения на их основе;
- тестирование приложения;
- планирование итогового решения.

2. РЕЗУЬТАТЫ РАБОТЫ В ВЕСЕННЕМ СЕМЕСТРЕ

2.1 Основные технологии, использованные при разработке приложения:

Spring - легкий, но мощный фреймворк для разработки web-приложений на Java. При разработке нашего приложения были использованы следующие его модули:

- **Inversion of Control**: контейнер, который предоставляет средства для конфигурации и управления жизненным циклом объекта при помощи рефлекции, создавая и связывая объекты между собой, как показано на рисунке 1.

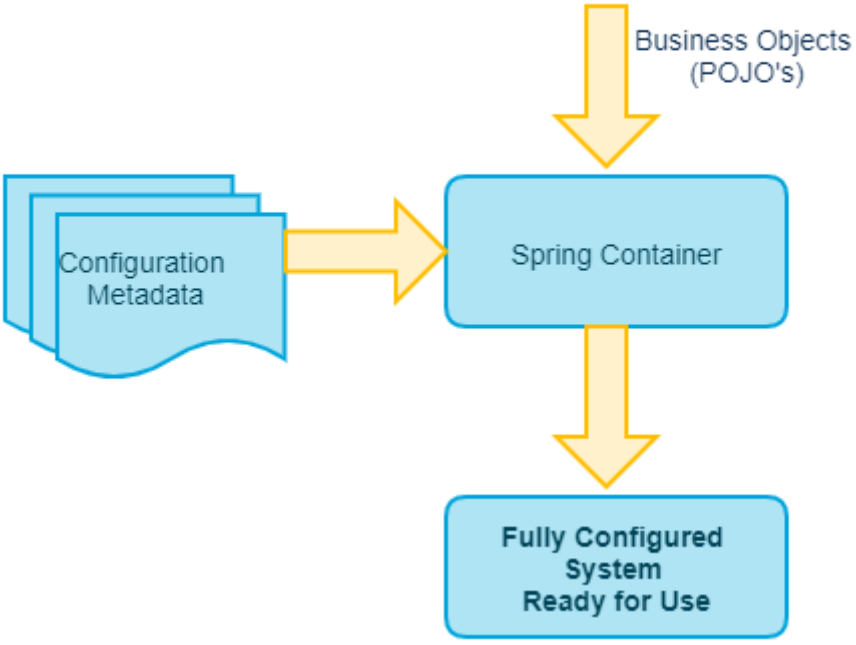

Рисунок 1. Spring Inversion Of Control.

- **Модуль аспектно-ориентированного программирования**: предоставляет возможность реализовать аспектно-ориентированную парадигму в пределах объектно-ориентированного приложения.
- **Модуль доступа к данным JDBC**: система управления реляционными базами данных на Java-платформе.
- Модуль управления транзакциями: координация различных API управления транзакциями и инструментарий настраиваемого управления транзакциями для объектов Java.

- **Модуль Model-View-Controller**: каркас, основанный на HTTP и сервлетах, предоставляющий множество возможностей для расширения, настройки и перераспределения ответственности за каждую конкретную функциональность, основанный на model – объект, предоставляющий данные, controller – класс, отвечающий за функциональность бэка, view – объекты отображения, фронт.

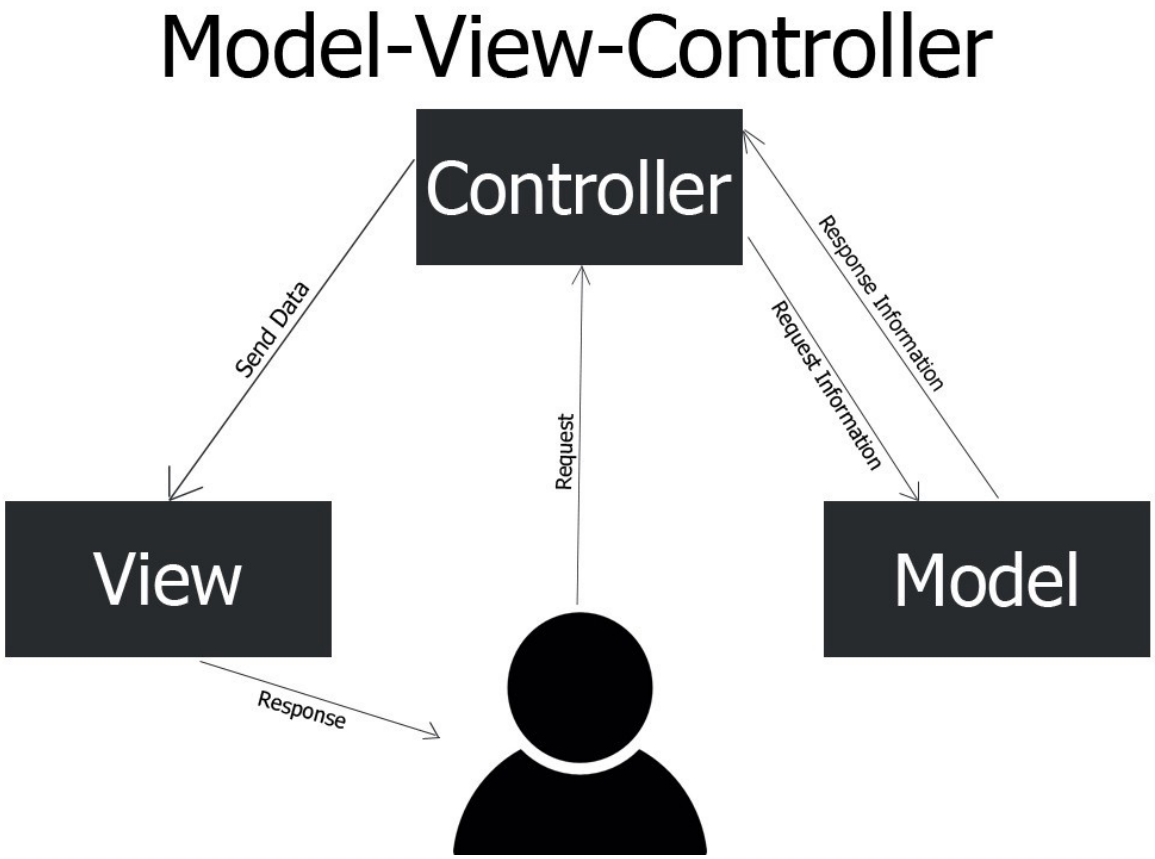

Рисунок 2. MVC

- **Модуль аутентификации и авторизации**: конфигурируемый инструмент аутентификации и авторизации, поддерживающий множество разнообразных протоколов, инструментов, практик, для обеспечения информационной безопасности.
- **Модуль тестирования**: каркас, поддерживающий классы для написания модульных и интеграционных тестов.

Так же были использованы:

- Git и Github, в качестве системы контроля версий;
- PostgreSQL, в качестве реляционной базы данных;
- SonarQube, в качестве системы контроля качества кода;
- FlyWay, в качестве системы миграций баз данных;
- Docker, в качестве контейнера для приложения;
- Freemarker, в качестве JSP фреймворка;
- Bootstrap, в качестве фронт-энд фреймворка;
- Gradle, в качестве сборщика;
- Jackson Formatter, в качестве JSON форматера;
- Jupiter, как инструмент для модульного тестирования;
- Intellij IDEA, в качестве IDE;

2.2 Общее описание приложения.

Приложение представляет из себя инструмент для управления командой разработчиков, тестировщиков и бизнес аналитиков, по методологии Agile, представляющей из себя набор подходов и практик для гибкого и эффективного управления всеми этапами разработки.

Приложение имеет систему регистрации и аутентификации, позволяющая организовать безопасность данных, а также предоставляющая систему ролей в приложении.

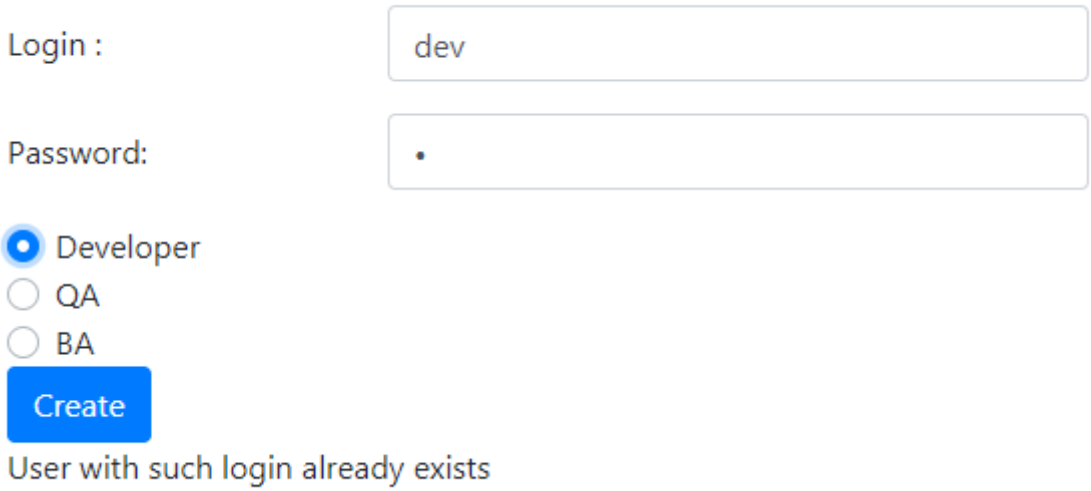

Рисунок 3. Регистрация

Зарегистрированные пользователи, одобренные администратором, могут видеть и создавать тикеты (задания) разных типов (Story, Task или Bug), которые отображаются в таблице с соответствующим статусом (ToDo,

InProgress, InTest, Done). Созданные тикеты можно искать по заданным фильтрам.

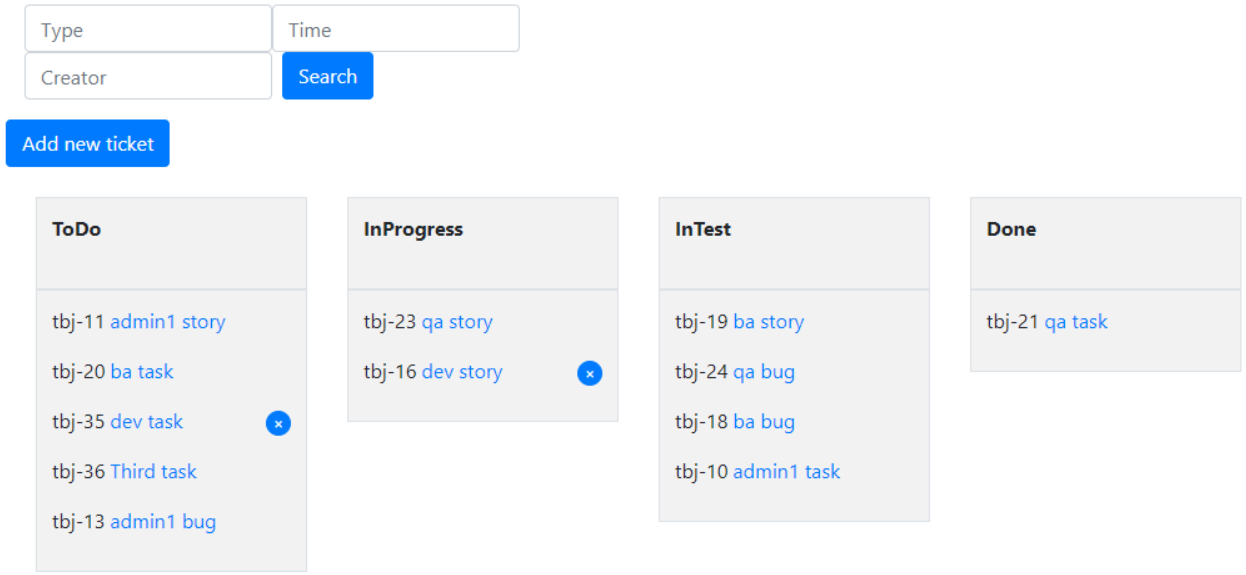

Рисунок 4. Главная страница приложения

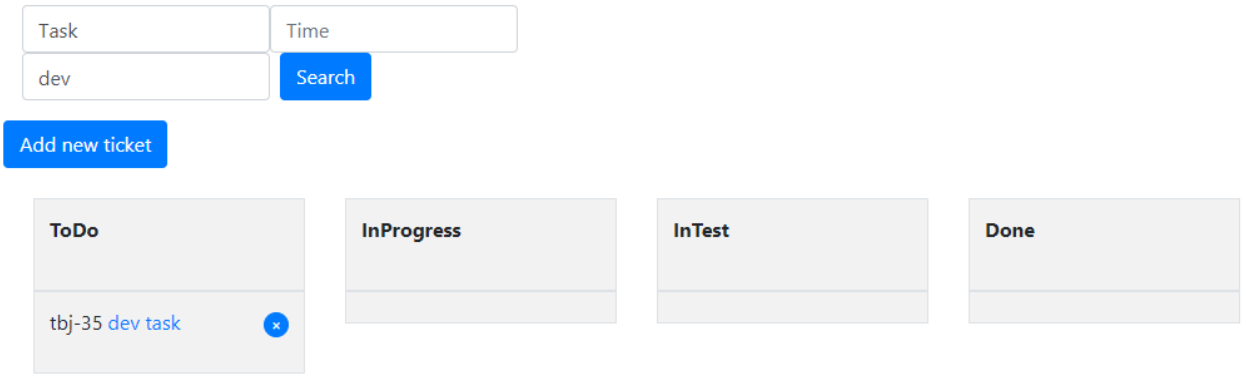

Рисунок 5. Применение фильтра

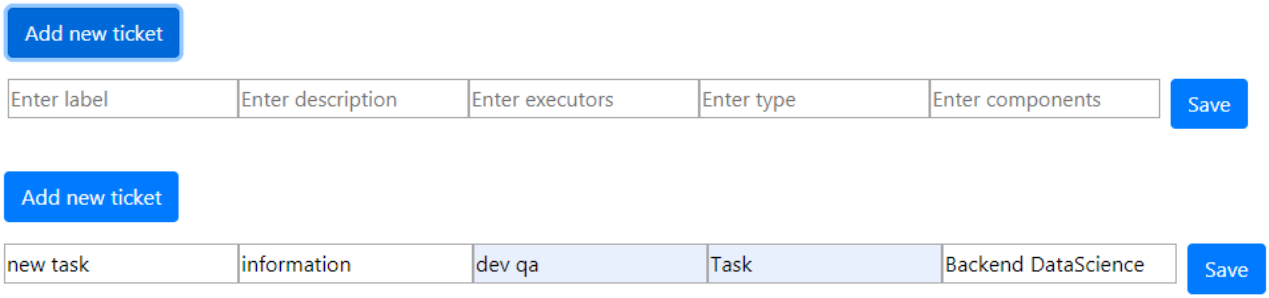

Рисунок 6. Форма добавления тикета

Каждый пользователь может переходить на страницу тикета, дабы посмотреть информацию о нем или перевести его в следующее состояние, в случае, если пользователь является назначенным исполнителем данного тикета.

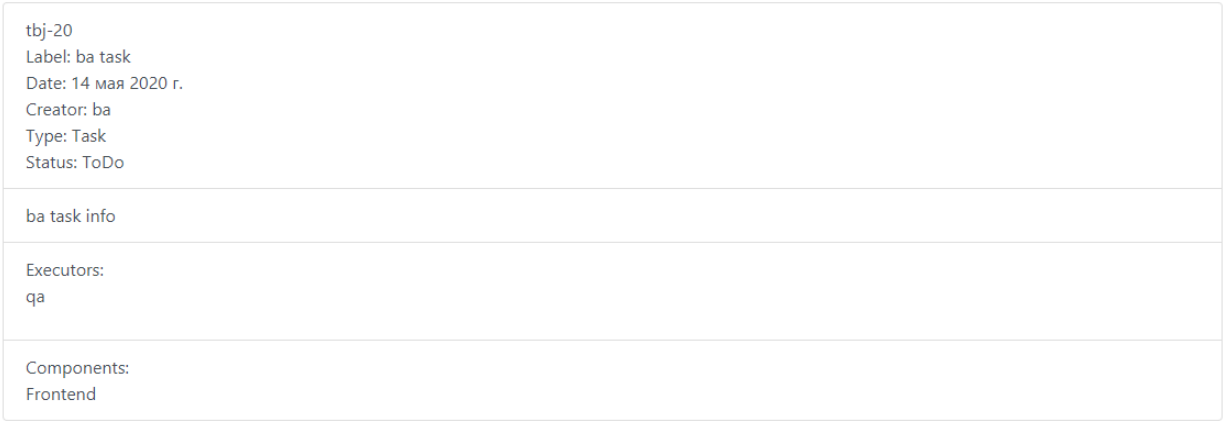

Рисунок 7. Страница тикета

Если пользователь является создателем тикета, он может изменять часть его содержимого (изменение описания, добавление и удаление исполнителей или сабтикетов, в случае, если тикет типа Story).

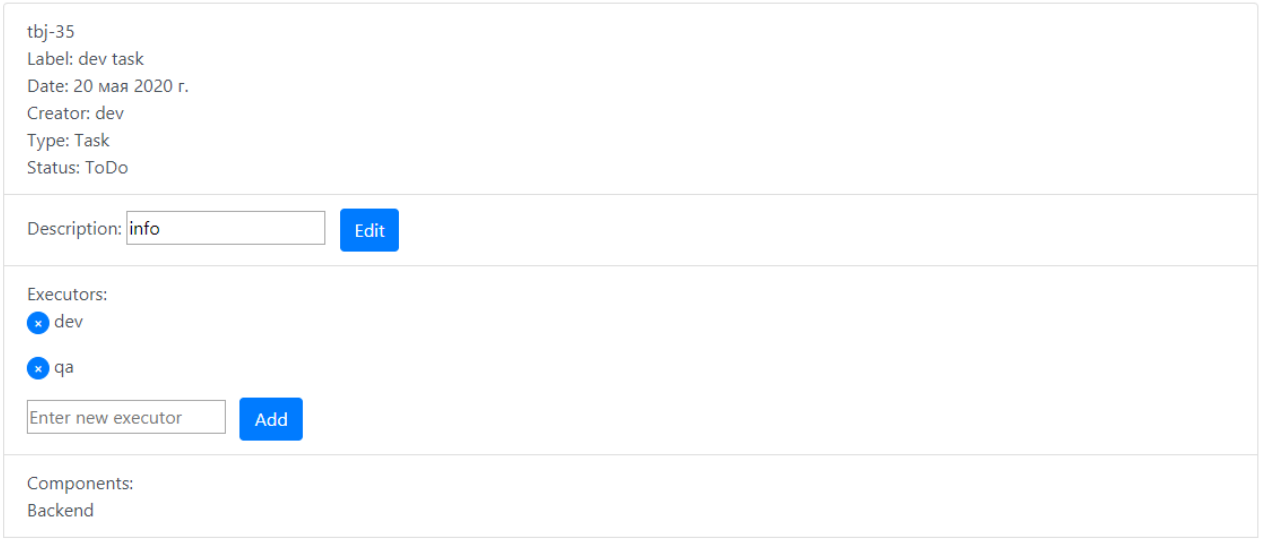

Рисунок 8. Форма редактирования тикета

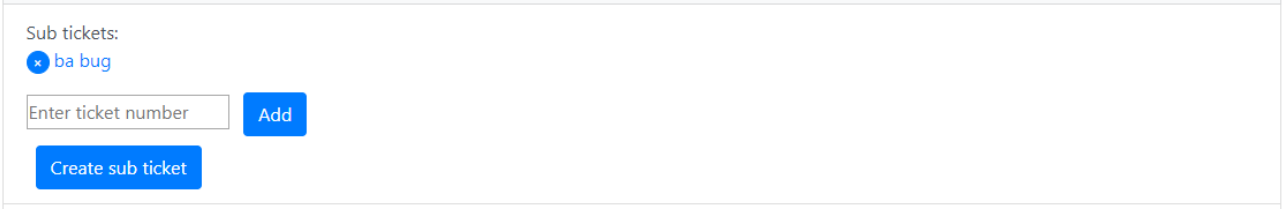

Рисунок 9. Форма добавления и удаления сабтикета для Story

В случае, если тикет находится в статусе InTest, изменять его статус, а то есть, либо заново открыть, либо переводить в Done, либо создавать по нему новый тикет-bug, могут только пользователи, имеющие роли QA или BA (тестировщики или бизнес аналитики).

Рисунок 10. Форма редактирования BA и QA в InTest

Удалять тикеты могут только их создатели, но до того, как они попали в Done.

Также, в приложении присутствует система администрирования. Пользователи с ролью администратора имеют полные права и доступы, и могут удалять и редактировать абсолютно любые тикеты. Кроме того, у администратора есть отдельная страница со списком всех зарегистрированных пользователей (кроме администраторов), где он может перейти на страницу конкретного пользователя и переименовать, изменить пароль или роль, или полностью заблокировать пользователя.

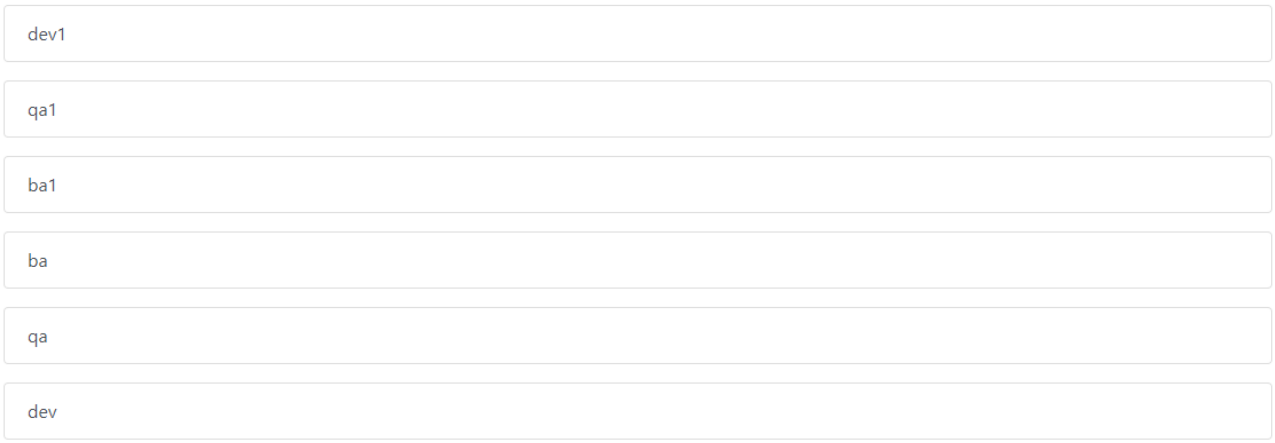

Рисунок 11. Список зарегистрированных пользователей

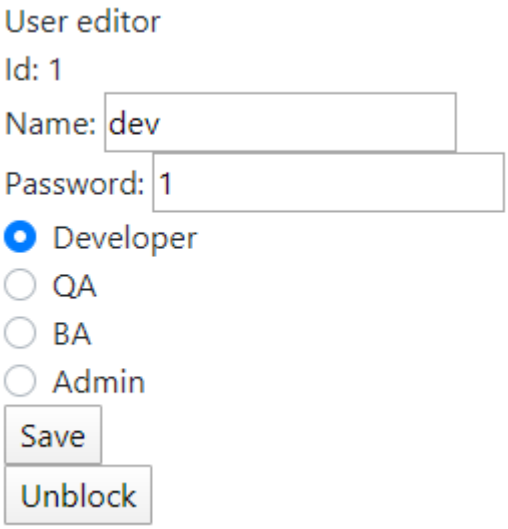

Рисунок 12. Форма управления пользователем, пользователь заблокирован

Именно для блокировки пользователей и была добавлена поддержка АОП в наше приложение, так как при блокировке, пользователя просто так не выбивает из сессии, пока он не разлогиниться. В целях своевременного выбивания пользователя и была добавлена кастомная аннотация NonBlockedCheck, которая в случае блокировки пользователя моментально выбивает его из сессии, а в дальнейшем, система не пускает пользователя обратно, даже при вводе верных данных, пока пользователь не будет разблокирован.

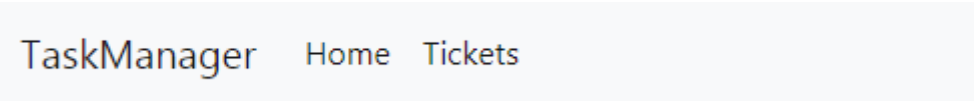

Рисунок 13. Навигационная панель обычного пользователя

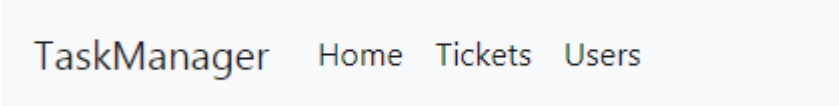

Рисунок 14. Навигационная панель администратора

2.3 Структура приложения.

Приложение написано с использованием Spring MVC, и поэтому логика завязана на контроллерах. Тем не менее вся бизнес логика вынесена в сервисы, фронт вынесен во view в формате ftlh. Взаимодействие с базой данных производится через DAO (Data Access Object) объекты. Данные из базы данных накладываются на model-объекты при помощи специальных классов mapper'ов.

Все sql-запросы находятся в специальных query классах, которые берут их из yml-файлов.

Основные настройки находятся в классах config.

Структура приложения показана на следующей UML-диаграмме:

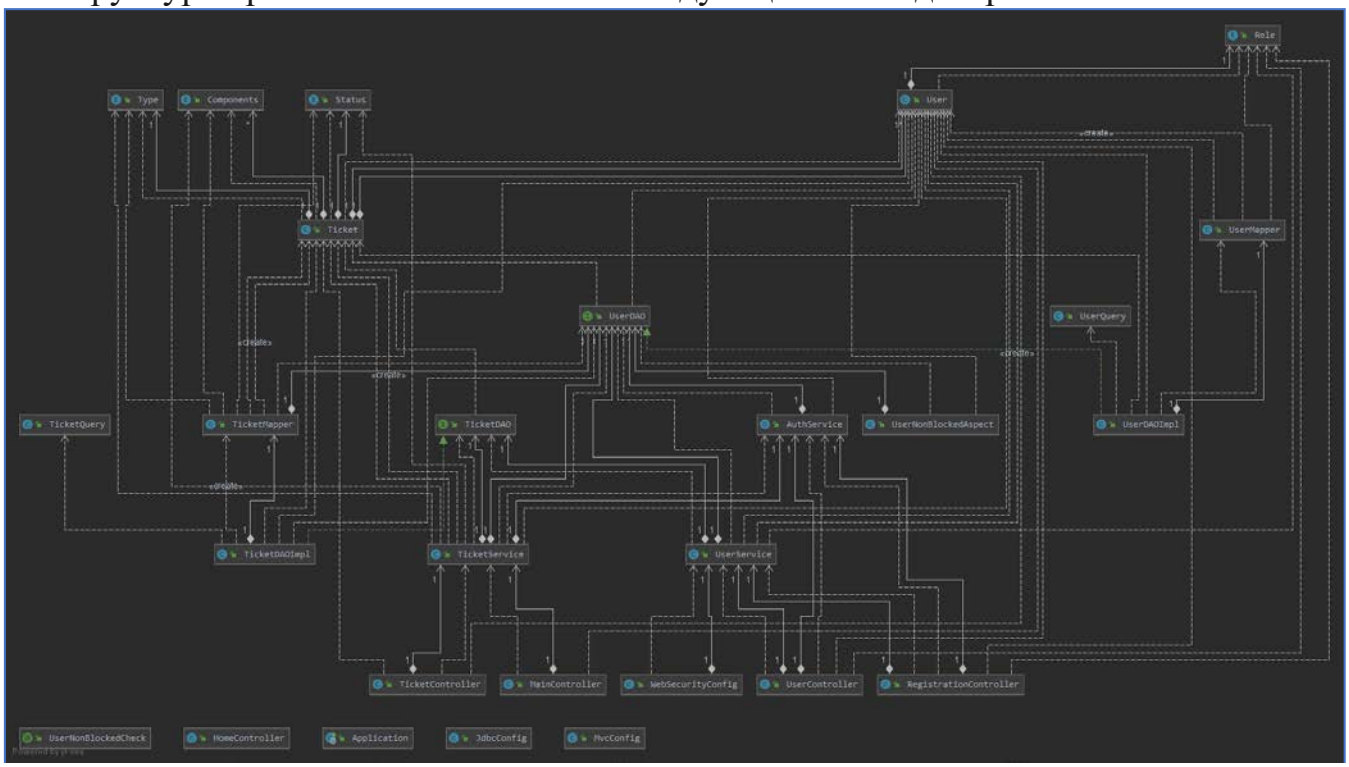

Рисунок 15. UML диаграмма.

Исходный код находится на github, ссылка в приложении А.

2.4 Тестирование приложения.

Приложение было протестировано при помощи интеграционных (с использованием тестовой базы данных) и модульных тестов с покрытием кода более 95%. Все тесты завершены успешно, приложение работает правильно.

Также, приложение было протестировано при помощи SonarQube, результаты проверки положительны и показаны ниже.

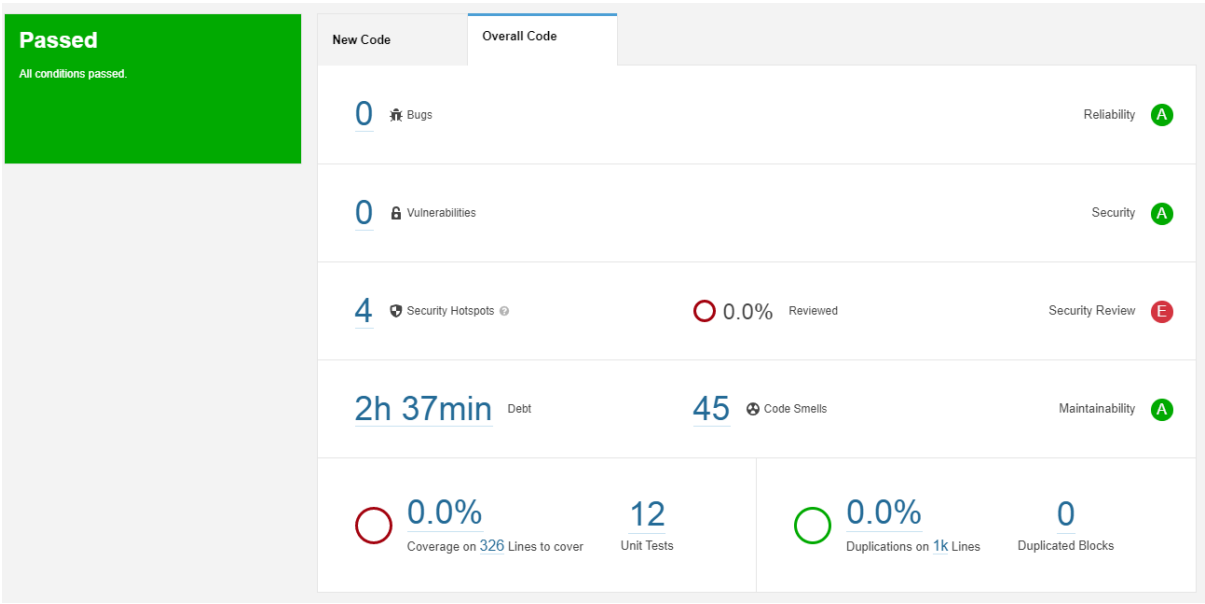

Рисунок 16. Результаты проверки SonarQube

Параметр Coverage не учитывается, так как для проверки покрытия тестами SonarQube необходимо установить дополнительный плагин, что не было произведено, так как внутренняя проверка покрытия Intellij IDEA работает вполне корректно и показывает 95%.

Security Hotspots, это параметр, который требует ручной проверки, в результате которой, уязвимости не были обнаружены.

2.5 **Обзор подходящих CI/CD инструментов**

Система контроля версий является одной из основных частей CI/CD. В качестве VCS (Version Control System) был выбран Git, в качестве удаленного репозитория – GitHub.

В качестве инструмента для собирания и тестирования запушенного в репозиторий кода планируется выбрать Travis CI, так как это удобный, распространенный и легкий в интеграции инструмент.

Для части CD планируется использовать Docker и DocketHub, но на данный момент на машине нет возможности запустить Docker, по причине чего, данный этап планируется перенести на осенний семестр.

В дальнейшем, в качестве сервиса для статического анализа кода как части CI/CD, планируется использовать CodeCov.

Выпуск приложения планируется производить при помощи Docker image и PaaS (Platform as a Service) Heroku.

3. ПЛАН РАБОТЫ НА ОСЕННИЙ СЕМЕСТР

План работы на весенний семестр расписан в таблице 1.

Таблица 1. План работы на весенний семестр

№ п/п	Наименование работ	Срок выполнения
1	Углубленное изучение предметной области и инструментов, детализация поставленных задач, обоснование актуальности разработки	03.09 – 29.09
2	Выбор метода достижения поставленных задач, реализация возможности использования Docker	01.10-15.10
3	Расширение функционала приложения и его подготовка к выпуску	16.10-15.11
4	Настройка сервера, настройка приложения для конкретного сервера, реализация выбранного решения	15.11-31.11
5	Подведение результатов работы и оформление НИР	01.12-31.12

ВЫВОД

В результате работы в осеннем семестре было разработано приложение с использованием Spring, с описанной выше функциональностью, которая планируется расширяться в следующей итерации. Существующий функционал был протестирован на правильность логики и взаимодействия с базой данных.

Также, были изучены основные инструменты CI/CD, подходящие к данному приложению и изучены способы их дальнейшей интеграции с проектом.

СПИСОК ЛИТЕРАТУРЫ

1. Spring documentation // . URL : https://spring.io/docs.
2. Applying CI/CD With Spring Boot // . URL : https://www.baeldung.com/spring-boot-ci-cd
3. Continuous Integration для новичков // Хабр. URL: https://habr.com/ru/post/352282/ (дата обращения: 14.12.2019).
4. Applying CI/CD to Java Apps Using Spring Boot .URL: https://dzone.com/articles/applying-cicd-to-java-apps-using-spring-boot.
5. Микросервисная архитектура, Spring Cloud и Docker. URL: https://habr.com/ru/post/280786/.
6. Знакомимся со Spring Framework // . URL: https://www.kv.by/archive/index2009291108.htm.
7. Spring Boot // . URL: https://spring-projects.ru/projects/spring-boot/.
8. FlyWay //. URL: https://flywaydb.org/documentation/plugins/springboot
9. SonarQube //. URL: https://docs.sonarqube.org/latest/
10. Travis CI: автоматическая загрузка собранных модулей на GitHub // . URL : https://habr.com/ru/post/338126/

ПРИЛОЖЕНИЕ А

Исходный код приложения на github:

https://github.com/legiq/trackingBoard/tree/demo