Финальный проект:
«Система Быстрого Тестирования Студентов»
(«Student Rapid Testing System»)

Разработчик: Светлана Бережная
Тренер: Максим Ляшенко

1 Описание варианта

Вариант 12. Система **Быстрого Тестирования Студентов**. **Студент** регистрируется е-мейлом, к нему привязывается его **Успешность** и на него будут приходить сообщения о результате тестов. В системе существует каталог **Тестов** по темам, авторизованный **Студент** может проходить тесты. В конце теста должна на странице отобразиться форма, где показано ошибки студента. Все данные об успеваемости и пройденных курсах отображаются на странице **Администратора** как сводная таблица по всем **Студентам**.

2 Общие требования

Необходимо построить веб-приложение, поддерживающее следующую функциональность:

1. На основе сущностей предметной области создать **классы,** их описывающие.
2. **Классы** и **методы** должны иметь отражающую их функциональность названия и должны быть грамотно структурированы по пакетам.
3. Информацию о предметной области хранить в **БД**, для доступа использовать **API JDBC** с использованием пула соединений - стандартного или разработанного самостоятельно. В качестве **СУБД** рекомендуется **MySQL**.
4. **Приложение** должно поддерживать работу с **кириллицей** (быть многоязычным), в том числе и при хранении информации в **БД**.

5. Код должен быть **документирован**.
6. Приложение должно быть покрыто **Юнит-тестами**.
7. При разработке бизнес логики использовать **сессии** и **фильтры**, и **события** в системе обрабатывать с помощью **Log4j**.
8. В приложении необходимо реализовать **Pagination**, **Transaction** в зависимости от Вашего проекта.
9. Используя **сервлеты** и **JSP**, реализовать функциональности, предложенные в постановке конкретной задачи.
10. В страницах **JSP** применять библиотеку **JSTL.**
11. Приложение должно корректно реагировать на ошибки и исключения разного рода (Пользователь никогда не должен видеть **stack-trace** на стороне **front-end**).
12. В приложении должна быть реализована система **Авторизации** и **Аутентификации.**

3 Список использованных технологий, библиотек и фреймворков

3.1 Спринг проект:

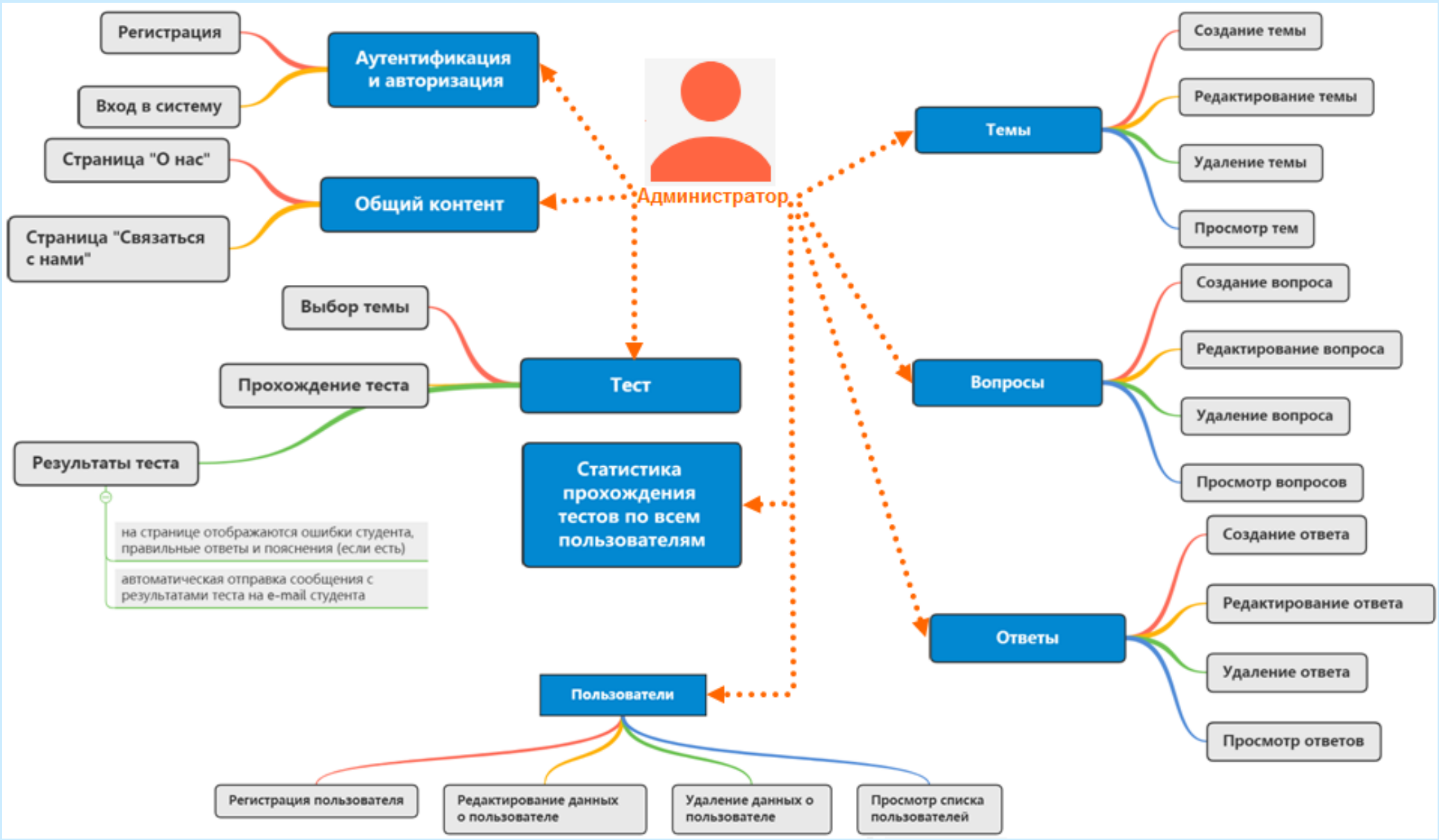
- Java 8;
- среда разработки ПО IntelliJ IDEA;
- Spring Framework;
- Apache Tomcat;
- Lombok;
- для отправки эл. почты по протоколу SMTP:
  - Java Mail API (библиотеки javax.mail.jar, smtp.jar)
  - Java Activation Framework (JAF) - библиотека javax.activation.jar
- MySQL;
- JDBC;
- JSP + JSTL;
- библиотека тегов taglibs;
- фреймворк Apache Maven для автоматизации сборки проекта

3 Список использованных технологий, библиотек и фреймворков

3.2 Сервлетный проект:

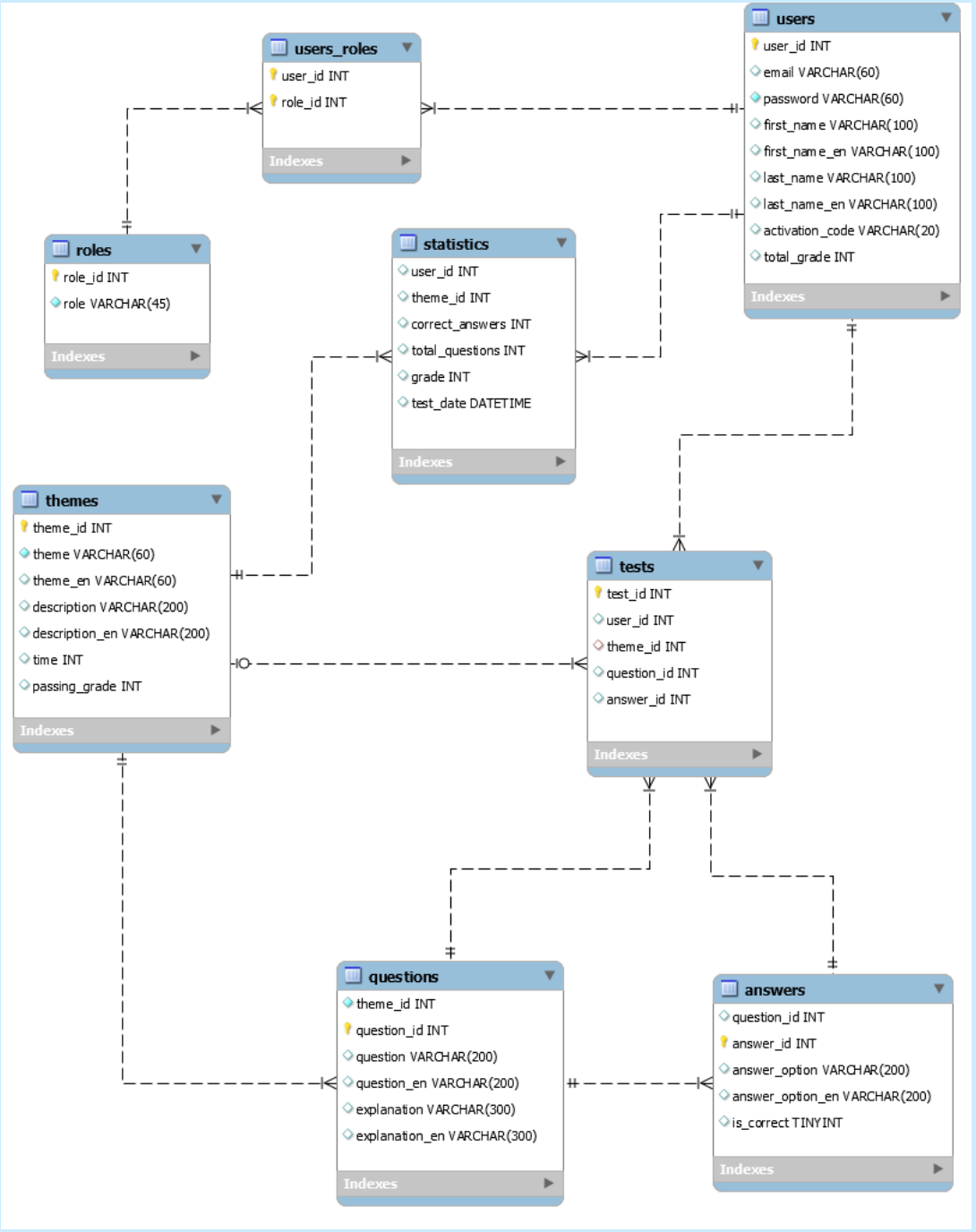
- Java 8;
- среда разработки ПО IntelliJ IDEA;
- Servlet + Filters;
- Apache Tomcat;
- для отправки эл. почты по протоколу SMTP:
  - Java Mail API (библиотеки javax.mail.jar, smtp.jar)
  - Java Activation Framework (JAF) - библиотека javax.activation.jar
- MySQL;
- JDBC;
- JSP + JSTL;
- библиотека тегов taglibs;
- JUnit;
- фреймворк Apache Maven для автоматизации сборки проекта

4 Диаграмма вариантов использования (Use case диаграмма)


Регистрация
Вход в систему
Аутентификация и авторизация
Страница "О нас"
Страница "Связаться с нами"
Общий контент
Администратор
Выбор темы
Прохождение теста
Тест
Результаты теста
на странице отображаются ошибки студента, правильные ответы и пояснения (если есть)
автоматическая отправка сообщения с результатами теста на e-mail студента
Статистика прохождения тестов по всем пользователям
Темы
Создание темы
Редактирование темы
Удаление темы
Просмотр тем
Вопросы
Создание вопроса
Редактирование вопроса
Удаление вопроса
Просмотр вопросов
Ответы
Создание ответа
Редактирование ответа
Удаление ответа
Просмотр ответов
Пользователи
Регистрация пользователя
Редактирование данных о пользователе
Удаление данных о пользователе
Просмотр списка пользователей

5 EER-модель Базы Данных MySQL «testing_system»

База Данных **«testing_system»** имеет следующие структуру данных и связи:



6 Архитектура проектов

MVC (Model – View - Controller) - принцип построения архитектуры приложения, при котором оно разбивается на три части Model, View, Controller; с точки зрения Java-программы, это три группы классов:

а) у каждой группы есть свое назначение;
б) связи между классами одной группы довольно сильные;
в) связи между группами довольно слабые;
г) способы взаимодействия групп довольно сильно регламентированы.

Первая часть **Модель (Model)** содержит бизнес-логику приложения, это самая независимая часть системы. В ней содержится код, реализующий функционал приложения, т.е то, для чего приложение создавалось.

Вторая часть **Вид (View)** содержит код, реализующий отображение данных пользователю (управление показом страниц, окон, сообщений и т.д.). Основное ограничение вида – вид не может менять модель.

Третья часть **Контроллер (Controller)** содержит код, обрабатывающий ввод/действия пользователя, направленные на изменение модели. Основное ограничение контроллера – он не реализует отображение данных.

Такой подход позволяет легко расширять, независимо разрабатывать и тестировать три части: логику программы (**Model**), механизм показа данных приложения пользователю (**View**), обрабатывать ввод/действия пользователя (**Controller**).

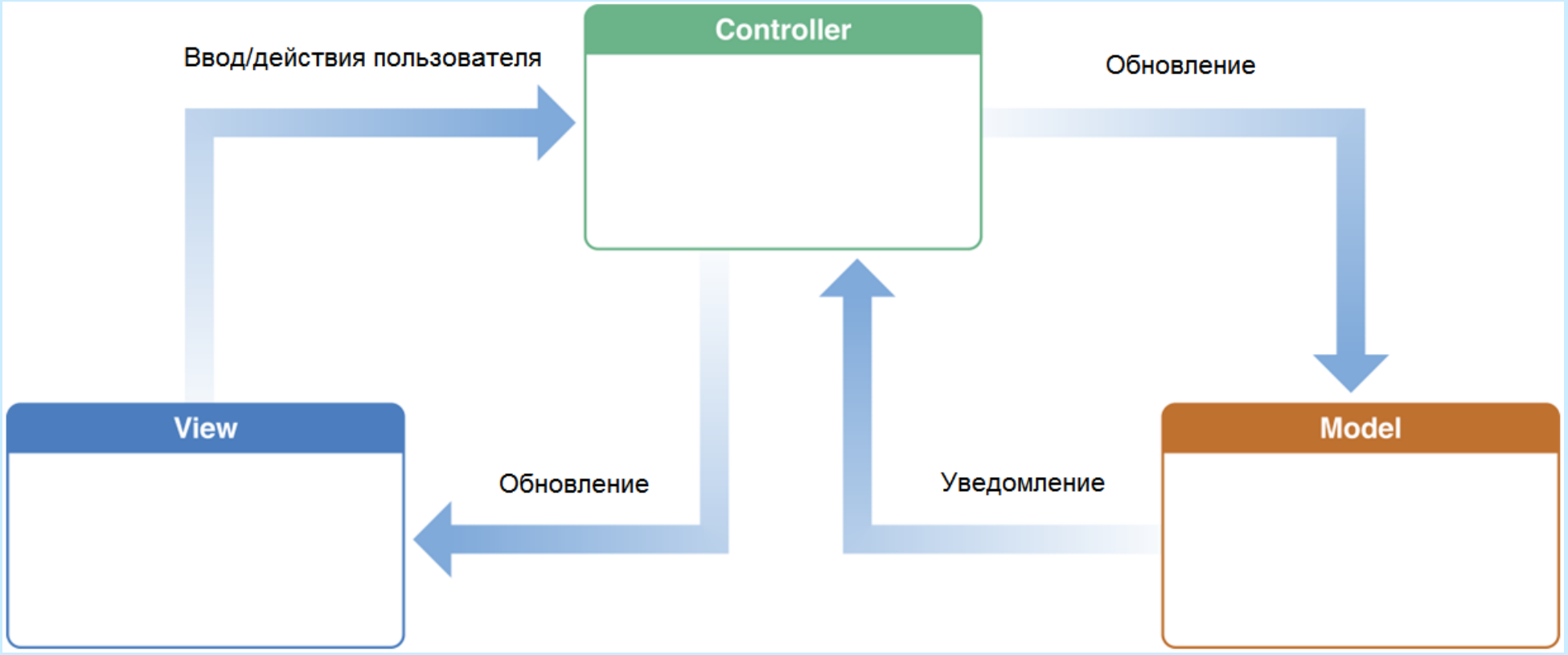
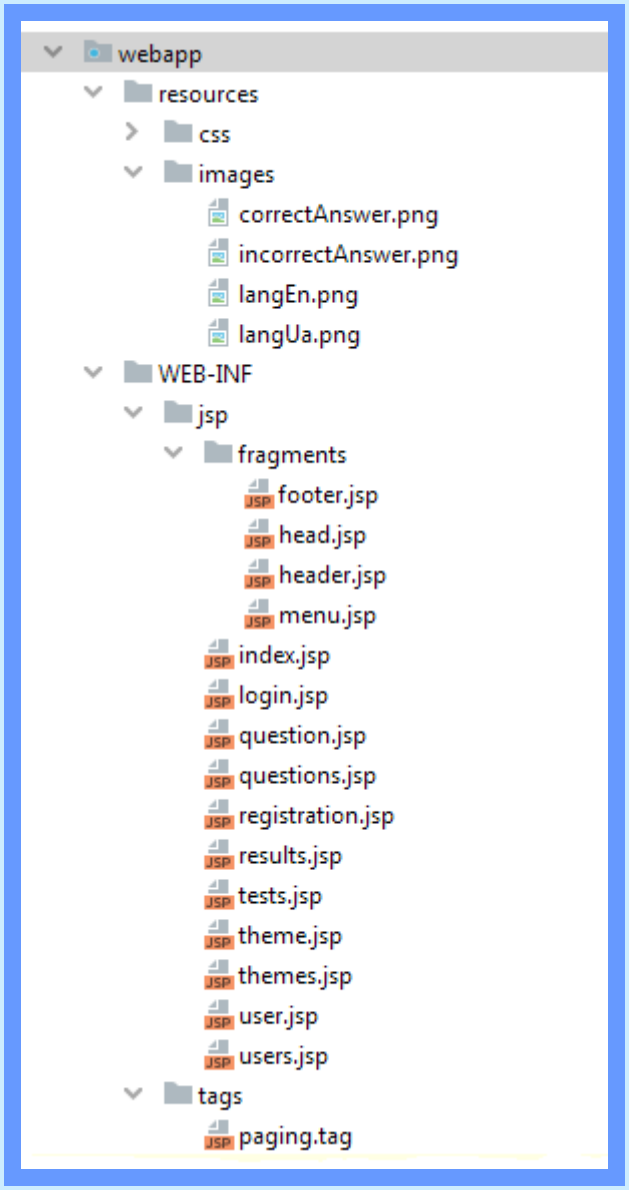
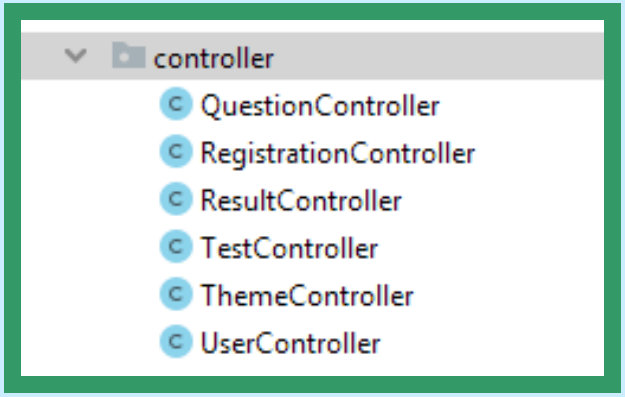
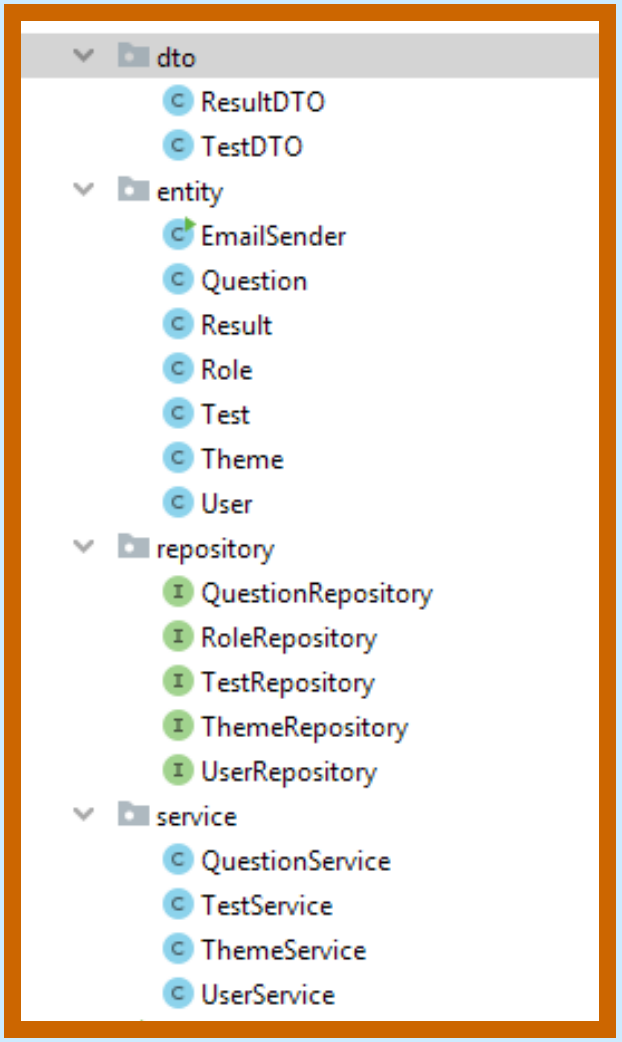


Схема работы объектов в MVC

6.1 Архитектура MVC Спринг проекта







6.2 Архитектура MVC Сервлетного проекта