

**Final project:**

**«Student Rapid Testing System»**

**Developer: Svitlana Berezhna**

**Trainer: Maxym Liashenko**

## 1 Option description

**Option 12. Student Rapid Testing System.** The student is registered by e-mail, to which his **Success** is tied and to which messages about the test results will be sent. The system contains a catalog of **Tests** on themes, an authorized **Student** can take tests. At the end of the test, a form should be displayed on the page showing student errors. All data on academic performance and courses completed are displayed on the **Admin** page as a summary table for all **Students**.

## 2 General requirements

**It should to build a web application that supports the following functionality:**

1. Based on the entities of the subject area, create **classes** that describe them.
2. **Classes** and **methods** should have names reflecting their functionality and should be competently structured by packages.
3. Information about the subject area should stored in the **database**, for access use the **JDBC API** using the connection pool - standard or developed independently. **MySQL** is recommended as a **DBMS**.
4. The **application** should support work with the **Cyrillic alphabet** (be multilingual), including when storing information in the **database**.
5. Code should be **documented**.
6. The application should be covered by **Unit tests**.

7. When developing business logic, use **sessions** and **filters**, and process **events** in the system using **Log4j**.
8. In the application you should implement **Pagination**, **Transaction** depending on your project.
9. Using **servlets** and **JSP**, implement the functionality proposed in the statement of the specific task.
10. Use **JSTL** library in **JSP** pages.
11. The application must correctly respond to errors and exceptions of various kinds (User should never see **stack-trace** on the **front-end** side).
12. The application should implement the **Authorization** and **Authentication** system.

### **3 List of used technologies, libraries and frameworks**

#### **3.1 Spring project:**

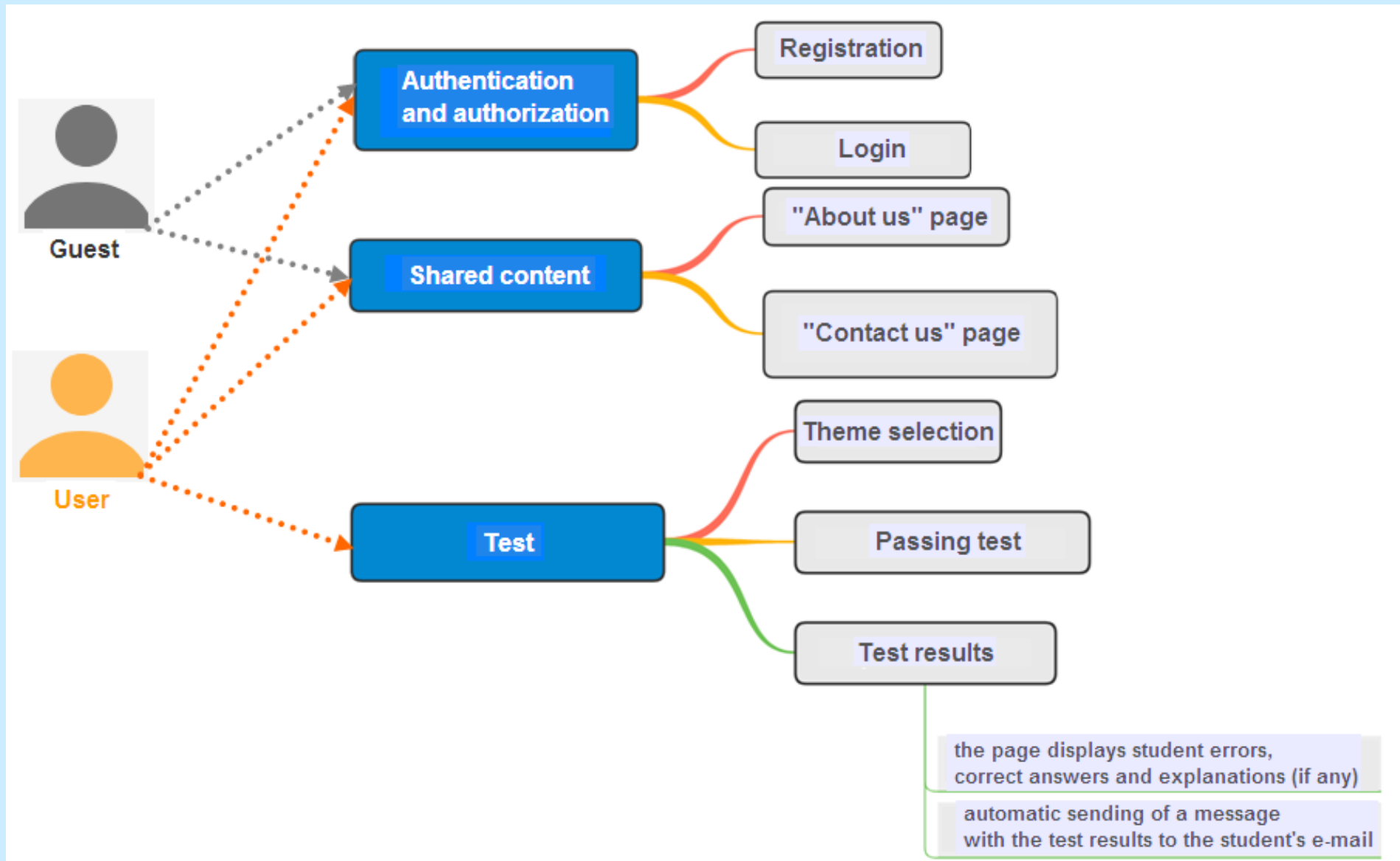
- Java 8;
- IntelliJ IDEA software development environment;
- Spring Framework;
- Apache Tomcat;
- Lombok;
- for sending email via SMTP protocol:
  - Java Mail API (javax.mail.jar, smtp.jar libraries)
  - Java Activation Framework (JAF) - javax.activation.jar library
- MySQL;
- JDBC;
- JSP + JSTL;
- taglibs tag library;
- Apache Maven framework for automating project build

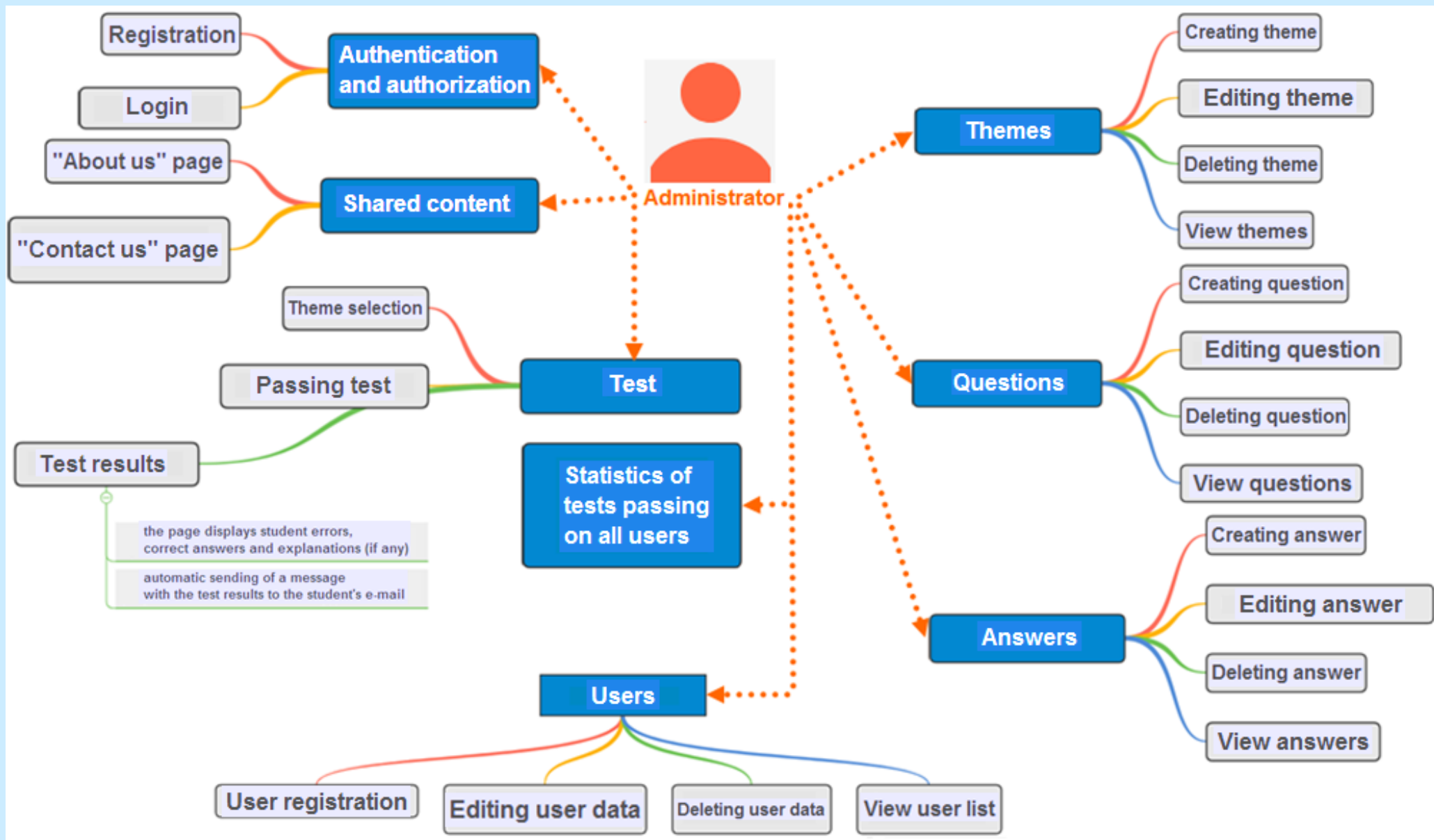
### **3 List of used technologies, libraries and frameworks**

#### **3.2 Servlet project:**

- Java 8;
- IntelliJ IDEA software development environment;
- Servlet + Filters;
- Apache Tomcat;
- for sending email via SMTP protocol:
  - Java Mail API (javax.mail.jar, smtp.jar libraries)
  - Java Activation Framework (JAF) - javax.activation.jar library
- MySQL;
- JDBC;
- JSP + JSTL;
- taglibs tag library;
- JUnit;
- Apache Maven framework for automating project build

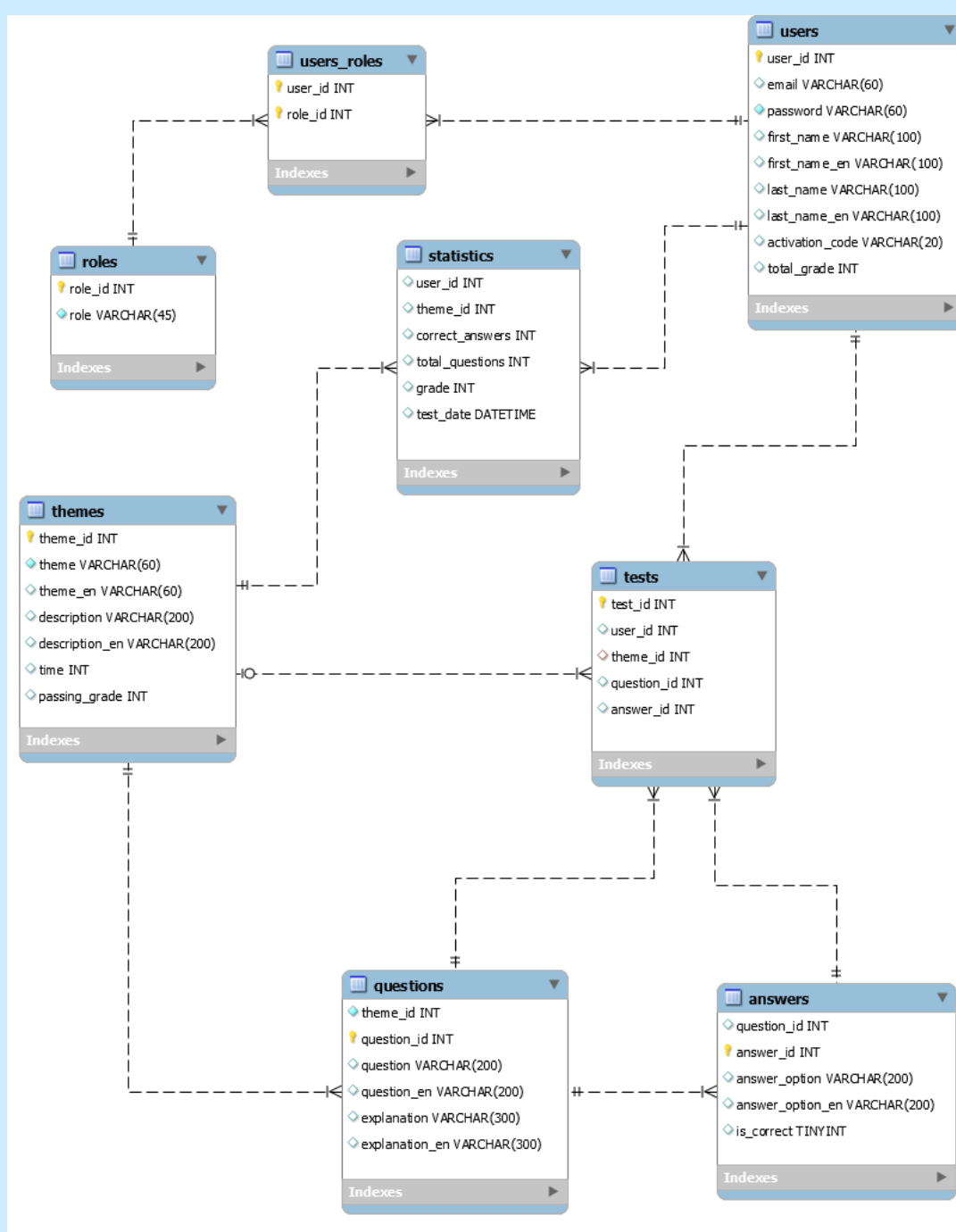
## 4 Use Case Diagram





## 5 EER model of MySQL database «testing\_system»

The «testing\_system» database has the following data and communication structure:



## 6 Projects architecture

**MVC** (**Model** – **View** - **Controller**) - the principle of building the architecture of the application, in which it is divided into three parts Model, View, Controller; in terms of a Java program, these are three groups of classes:

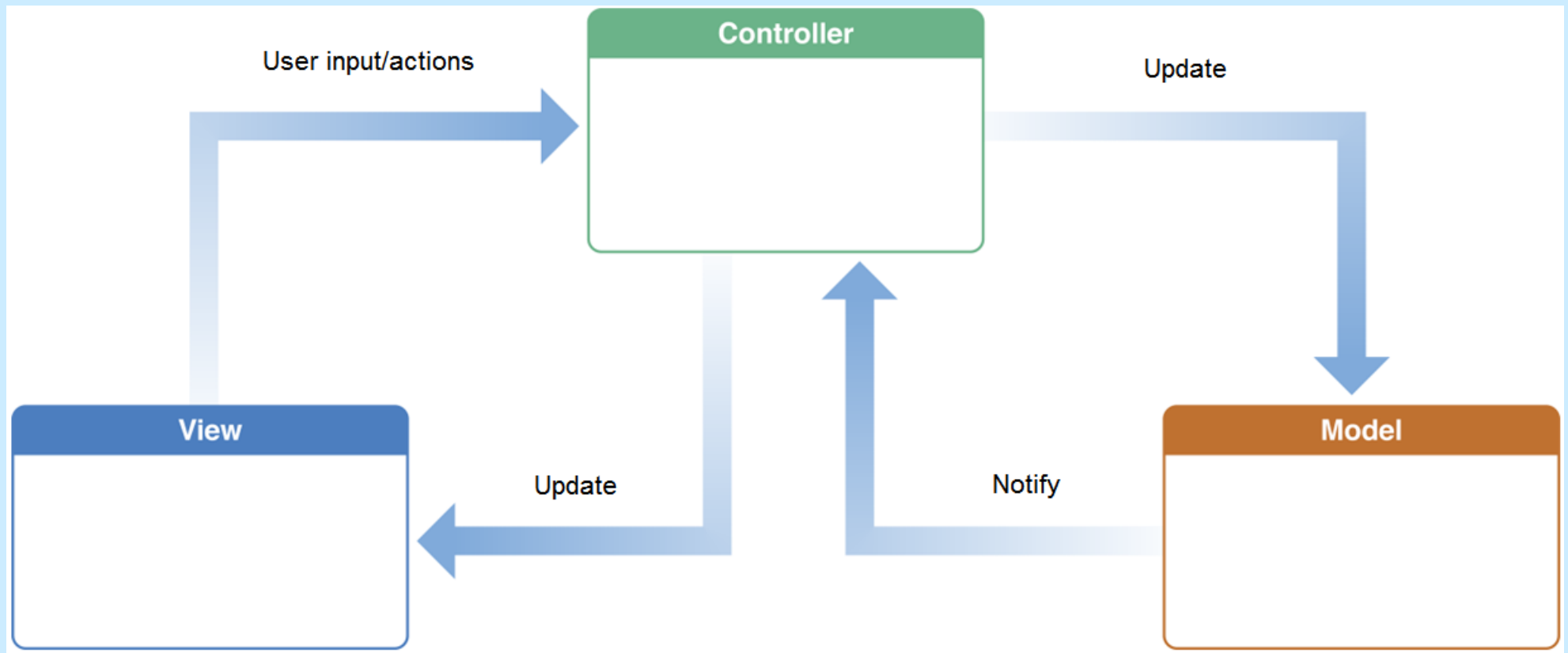
- a) each group has its own purpose;
- б) the connections between the classes of one group are quite strong;
- в) connections between groups are rather weak;
- г) group interaction methods are pretty much regulated.

The first part (**Model**) contains the business logic of the application, it is the most independent part of the system. It contains code that implements the functionality of the application, i.e. what the application was created for.

The second part (**View**) contains a code that implements the display of data to the user (control the display of pages, windows, messages, etc.). The main limitation of the view is that the view cannot change the model.

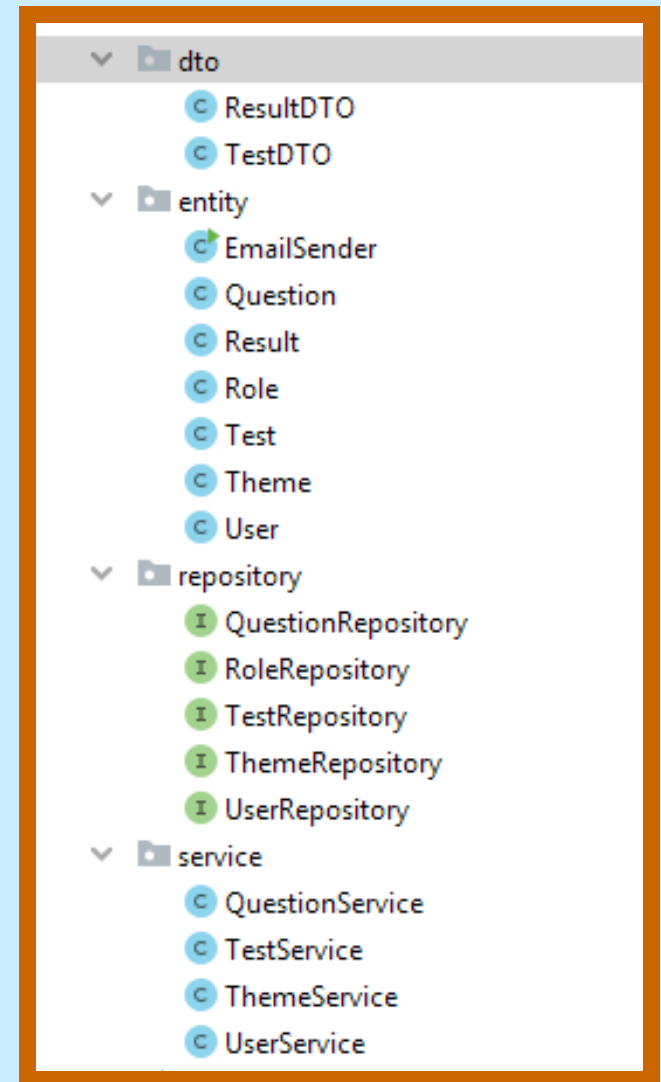
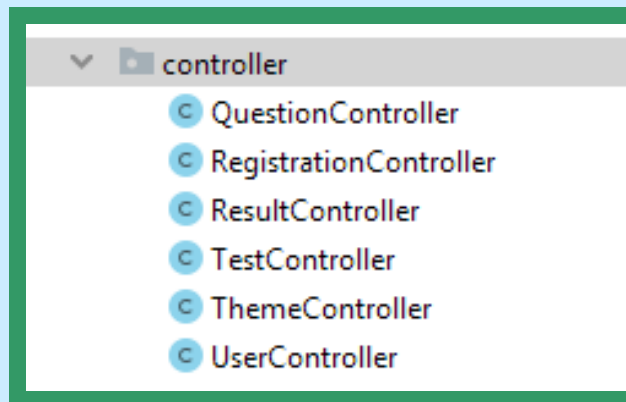
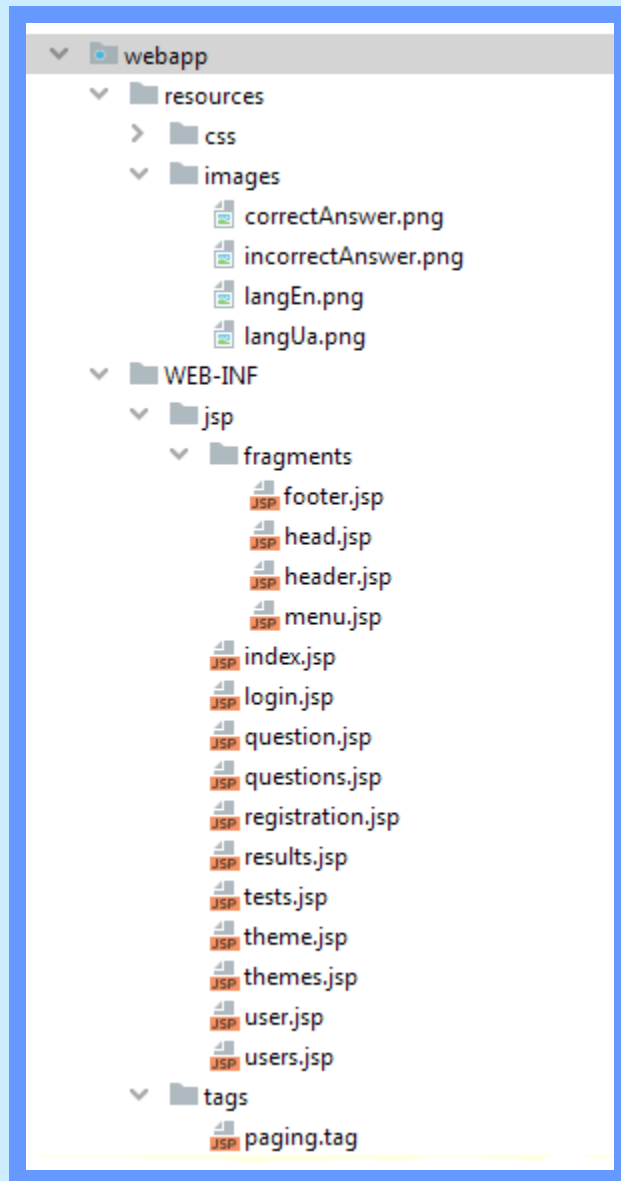
The third part (**Controller**) contains a code that processes user input/actions aimed at changing the model. The main limitation of the controller is that it does not implement data mapping.

This approach allows to easily expand, independently develop and test three parts: the program logic (**Model**), the mechanism for displaying application data to the user (**View**), and process user input/actions (**Controller**).



The scheme of work of objects in MVC

## 6.1 MVC Spring Project Architecture



## 6.2 MVC Servlet Project Architecture

