Федеральное государственное бюджетное образовательное учреждение высшего образования

«Владимирский государственный университет

имени Александра Григорьевича и Николая Григорьевича Столетовых»

Кафедра информационных систем и программной инженерии

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине   
"Распределенные программные системы"

на тему

Реализация прототипа программной системы «Дистанционное обучение студентов». Подсистема: «Взаимодействие со студентами»

Выполнил: студент группы ПРИ-120 Ермолаев Д. А.

Приняла: преп. кафедры ИСПИ

Проскурина Г.В.

Владимир, 2023

**Аннотация**

В данном курсовом проекте производилась разработка программной системы для автоматизации работы университета по дистанционному обучению.

Проект состоит из 3 основных этапов, включающих в себя основные три этапа, такие как: «Проектирование информационной системы», «Разработка прототипа системы» и «Тестирование системы и подведение итогов».

Первый этап под названием «Проектирование информационной системы», содержащий в себе такие блоки, как: описание предметной области, описание пользователей разрабатываемой подсистемы, описание прецедентов и сущностей системы, динамического моделирования системы, а также моделирования и разработки макета интерфейса отображения информации в системе.

Второй этап под названием «Разработка прототипа системы» состоит из реализации спроектированной системы в виде веб-сервиса с электронным сайтом с набором функционала, реализованном согласно возможностям платформы Java EE, фреймворка Spring Boot и фреймворка Angular.

Третий этап под названием «Тестирование системы и подведение итогов» состоит из нагрузочного тестирования сервиса и выводов по результатам выполнения разработки.

Реализованная система может применяться для таких целей, как, например, создание и редактирование дисциплин, наполнение их группами, лабораторными работами и тестами, а также осуществление взаимодействия пользователей.

Реализованная система спланирована для использования различными видами пользователей, имеющих различный функционал согласно их ролям и возможностям в системе.

Курсовой проект представлен на 46 страницах, основных рисунков - 41, использованных источников – 7, приложений – 1.

**ABSTRACT**

In this course project, a software system was developed to automate the university's work on distance learning.

The project consists of 3 main stages, including the main three stages, such as: "Information system design", "System prototype development" and "System testing and summing up".

The first stage is called "Information system Design", which contains such blocks as: a description of the subject area, a description of the users of the subsystem being developed, a description of the precedents and entities of the system, dynamic modeling of the system, as well as modeling and development of the layout of the interface for displaying information in the system.

The second stage, called "System Prototype Development", consists of the implementation of the designed system in the form of a web service with an electronic website with a set of functionality implemented according to the capabilities of the Java EE platform, the Spring Boot framework and the Angular framework.

The third stage, called "System testing and summing up", consists of load testing of the service and conclusions based on the results of the development.

The implemented system can be used for such purposes as, for example, creating and editing disciplines, filling them with groups, laboratory work and tests, as well as user interaction.

The implemented system is planned for use by different types of users with different functionality according to their roles and capabilities in the system. The implemented system is planned for use by different types of users with different functionality according to their roles and capabilities in the system.

The course project is presented on 46 pages, the main figures - 41, the sources used – 7, appendices – 1.

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc137097514)

[1. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 4](#_Toc137097515)

[1.1. Постановка задачи 4](#_Toc137097516)

[1.2. Описание предметной области 5](#_Toc137097517)

[1.3. Пользователи системы 5](#_Toc137097518)

[1.4. Построение диаграмм 6](#_Toc137097519)

[2. РАЗРАБОТКА ПРОТОТИПА СИСТЕМЫ 21](#_Toc137097520)

[2.1. Разработка backend – части системы 22](#_Toc137097521)

[2.2. Разработка frontend – части системы 25](#_Toc137097522)

[3. ТЕСТИРОВАНИЕ СИСТЕМЫ 33](#_Toc137097523)

[3.1. Нагрузочное тестирование 33](#_Toc137097524)

[Репозиторий проекта 35](#_Toc137097525)

[Заключение 36](#_Toc137097526)

[Список использованных источников 37](#_Toc137097527)

[ПРИЛОЖЕНИЕ А. Код программной системы. 38](#_Toc137097528)

# ВВЕДЕНИЕ

Для начала выполнения разработки прототипа информационной системы «Дистанционное обучение» требуется выполнить следующие шаги по проектированию:

- определить задачи

- описать предметную область

- выделить ее основные функциональные требования, с помощью основных UML-диаграмм описать составы данных и категории рассматриваемой системы

- провести описание основных прецедентов пользователей, сущности системы

- провести динамическое моделирование системы

- осуществить моделирование основных бизнес-процессов в методологии IDEF0 (as-is)

- разработать макет интерфейса системы

- выбрать средства разработки и основные фреймворки, требующиеся для реализации функционала

# ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## Постановка задачи

Целью разработки данного курсового проекта является разработка программной системы «Дистанционное обучение студентов», включающая в себя подсистему хранения данных и позволяющая студентам выполнять различные задания, которые оцениваются лекторами.

Функциональные требования:

* Авторизация и аутентификация;
* Создание справочников студентов;
* Редактирование справочников студентов;
* Создание справочников лекторов;
* Редактирование справочников лекторов;
* Создание различных ролей;
* Создание тестов;
* Редактирование тестов;
* Прохождение тестов;
* Предоставление студентам доступа к курсу;
* Включение студентов в дистанционное прохождение тестов;
* Оценка тестов;
* Вывод результатов тестов;

Нефункциональные требования:

* Система должна выдержать до 100 пользователей параллельно
* Система должна быть расширяемой и масштабируемой
* Системе важна защищённость пользовательских данных

## Описание предметной области

В данной работе рассматривается предметная область «Дистанционное обучение студентов» предназначенная для обучения студентов вне учебного заведения.

Все студенты распределены по группам относительно направлению подготовки. Также они имеют свои личные данные.

Лекторы могут создавать дисциплины, добавлять в них учебные материалы, лабораторные работы, создавать тесты. К этим дисциплинам они могут подключить определенные группы студентов. В свою очередь студенты могут выполнять созданные лектором задания.

Администратор разрабатываемой системы может создавать и редактировать аккаунты для студентов и лекторов, чтобы те впоследствии могли авторизоваться на электронном сайте и пользоваться им.

## Пользователи системы

**Студент** – обучающийся, который может выполнять задания и просматривать лекции.

**Лектор** – преподаватель, имеющий возможность создавать и редактировать дисциплины.

**Администратор**– лицо, администрирующее электронный сайт, а также имеющее возможность создавать и редактировать аккаунты студентов и лекторов.

**Словарь предметной области.**

*Тест* – задание, выполняемое студентом, для проверки теоретических знаний.

Атрибуты: название, дата сдачи, список заданий, баллы.

*Задание –* вопрос, содержащийся в тесте, включающий в себя варианты ответа.

Атрибуты: название, список вариантов ответа.

Вариант ответа – одно из нескольких решений задания.

Атрибуты: название.

*Лабораторная работа* – задание, выполняемое студеном, содержащее в себе методические указания.

Атрибуты:название, дата сдачи, методические указания, баллы.

*Группа* – состав студентов, относящихся к определенному направлению подготовки.

Атрибуты: название, направление подготовки, список дисциплин.

*Дисциплина* – предмет, включающий в себя, лабораторные работы, тесты и методические материалы.

Атрибуты: название,список тестов, лектор, список студентов, лабораторных работ.

*Пользователь* – человек, использующий функционал автоматизированной электронная системы.

Атрибуты: ФИО, логин, пароль, e-mail, дата рождения, номер телефона.

## Построение диаграмм

**Описание основных прецедентов пользователей в системе:**

1. **Название: «Создать дисциплину»**

*Предусловие:* Пользователь авторизован

*Действующее лицо:*Лектор

*Основной поток:*Создание дисциплины

Лектор открывает электронный сайт (представление веб-приложения), переходит на страницу создания дисциплины. Далее может добавить лабораторные работы, тесты и методические указания. Также может предоставить доступ к дисциплине студентам. Нажимает на кнопку создания дисциплины.

Система сохраняет в базу данных созданную дисциплину.

*Альтернативный поток:*Дисциплина, которую лектор хочет создать, уже существует.

Выводится уведомление об этом.

*Постусловие:*После создания дисциплины появляется окно оповещения о создании.

1. **Название: «Вывести результаты теста»**

*Предусловие:* Пользователь авторизован

*Действующее лицо:* Студент

*Основной поток:*Вывод результатов теста

Студент открывает электронный сайт (представление веб-приложения), переходит на страницупройденного теста, на которой выводятся оценка, поставленная лектором (прецедент: «Оценка теста»).

*Альтернативный поток:* Тест не оценён.

В окне, в котором прописывается оценка, указано: «тест не оценён».

1. **Название: «Оценка теста»**

*Предусловие:* Пользователь авторизован, студент должен пройти тест

*Действующее лицо:* Лектор

*Основной поток:*Оценка теста

Лектор открывает электронный сайт (представление веб-приложения), переходит на страницу, на которой отображаются тесты, пройденные студентами, выбранной дисциплины. Проверяет и выставляет оценку.

Система сохраняет в базу данных количество баллов за тест.

*Постусловие:*После выставления оценки появляется окно оповещения об оценивании.

1. **Название: «Создать лектора»**

*Предусловие:* Пользователь авторизован

*Действующее лицо:* Администратор

*Основной поток:*Создание лектора

Администратор открывает электронный сайт (представление веб-приложения), переходит на страницу создания пользователей. Далее он может создать новый аккаунт и выдать ему права лектора.

Система сохраняет в базу данных информацию о созданном лекторе.

*Постусловие:*После создания лектора появляется окно оповещения о создании.

1. **Название: «Авторизоваться на сайте»**

*Предусловие:* Пользователь не авторизован

*Действующее лицо:* Пользователь

*Основной поток:*Авторизация на сайте

Пользователь открывает электронный сайт (представление веб-приложения), переходит на страницуавторизации. Далее вводит логин и пароль.

*Альтернативный поток:*Логин или пароль указаны неверно.

*Постусловие:*После авторизации пользователь получает доступ к возможностям сайта.

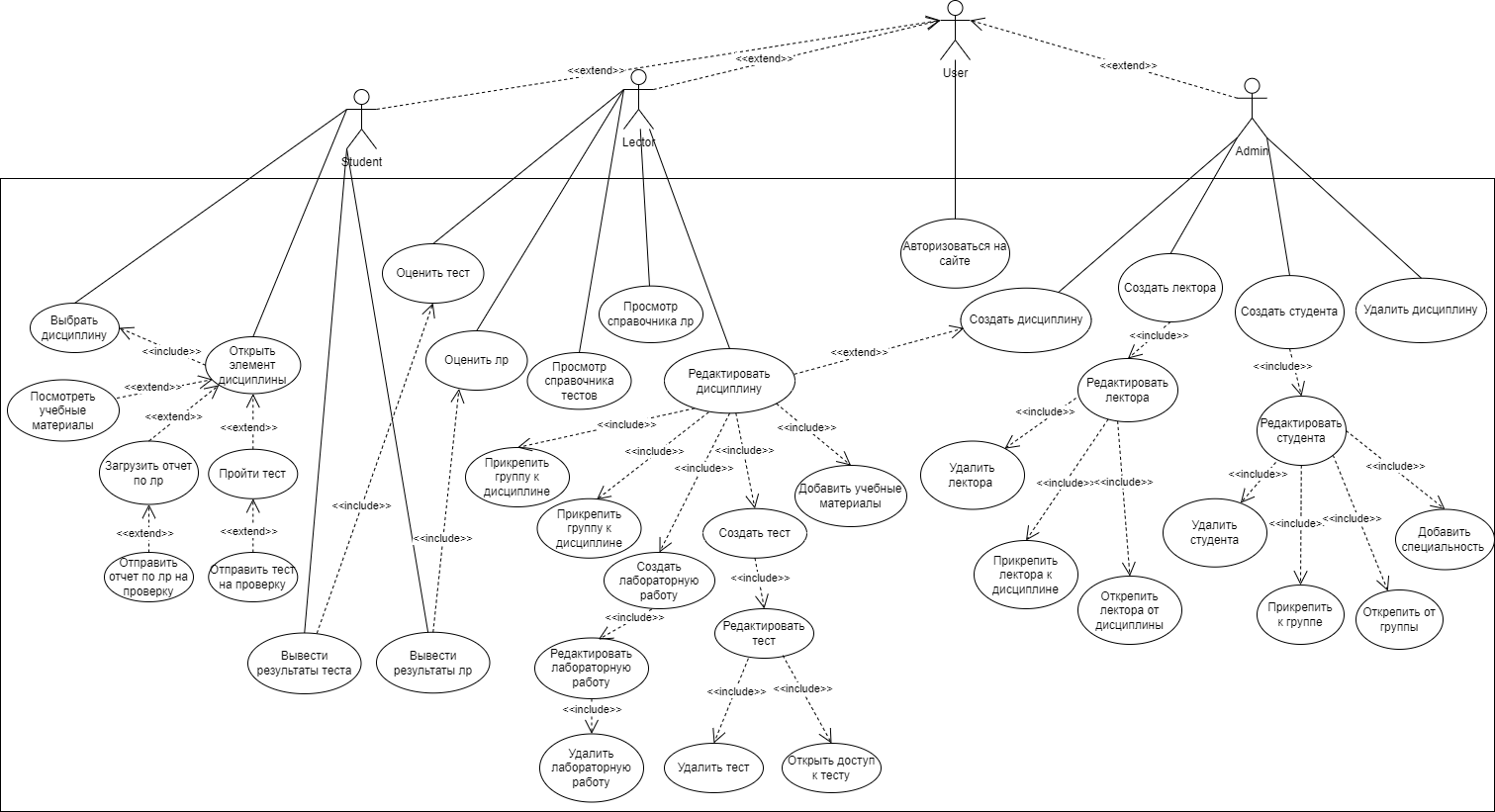


Рисунок 1. Диаграмма прецедентов.

Разработка **диаграммы классов (UMLClassDiagram)** в нотации UML.

Класс – это описание совокупности объектов с общими атрибутами, операциями, отношениями и семантикой.

Выделим такие основные сущности в системе, как:

- **Student** (пользователь, обладающий правами студента)

- **Group** (список студентов, которые учатся по своей специальности)

- **Lector** (пользователь, обладающий правами лектора)

-**LabWork** (задания, имеющие методические указания, выполняемые студентом)

-**Discipline** (предмет, который содержит в себе лабораторные работы, тесты и учебные материалы)

- **Test** (задания по итогам тем(ы), выполняемые студентом)

- **User** (абстрактный пользователь с его данными в системе)

- **Task** (задание, содержащееся в тесте)

- **AnswerOption** (один из вариантов ответа, содержащийся в задании)

Используютсятакиеосновныевиды отношений, как:

* Наследование:

1. User: Student, Lector.

* Композиция:

1. LabWork - Student \* - \*;
2. Group - Student1 - \*;
3. Discipline - Group\* - \*;
4. Discipline – Lector \* - 1;
5. LabWork – Discipline \* - 1;
6. Test – Discipline \* - 1;
7. Test – Student \* - \*;
8. Task – Test \* - 1;
9. AnswerOption – Task \* - 1

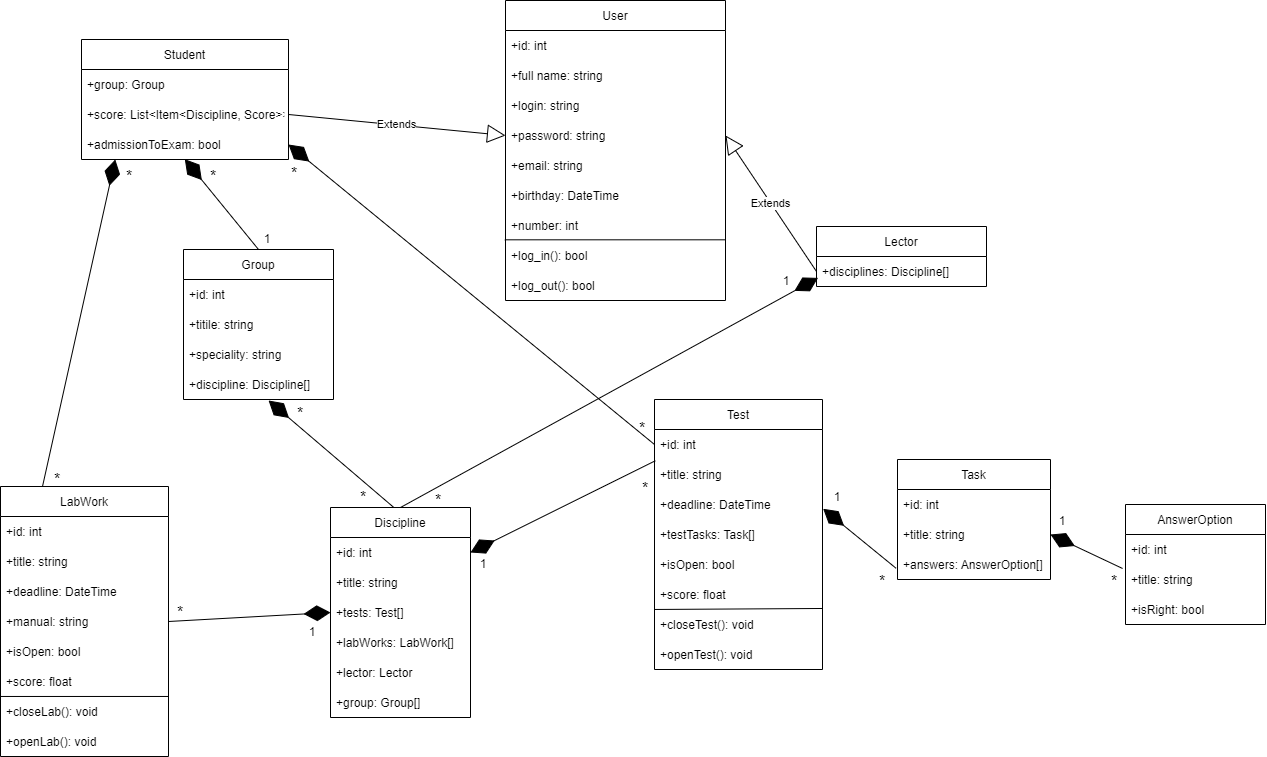
****

Рисунок 2. Диаграмма классов.

Далее был создан набросок будущего интерфейса системы (Рис. 3).

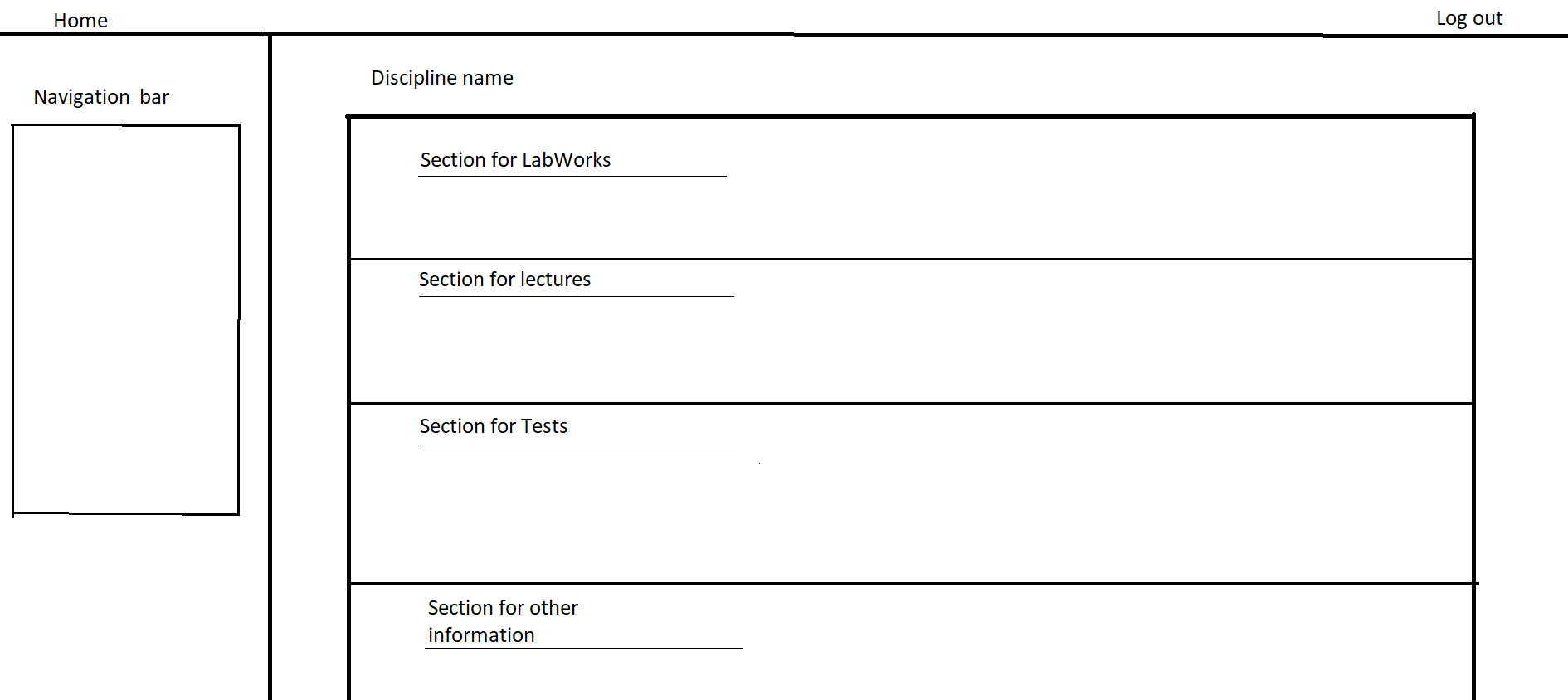


Рисунок 3. Набросок интерфейса.

**Построение диаграммы видов деятельности**

Вид деятельности: Проверка и оценка отчёта по лабораторной работе.

Субъекты: лектор (пользователь), система (Рис. 4).

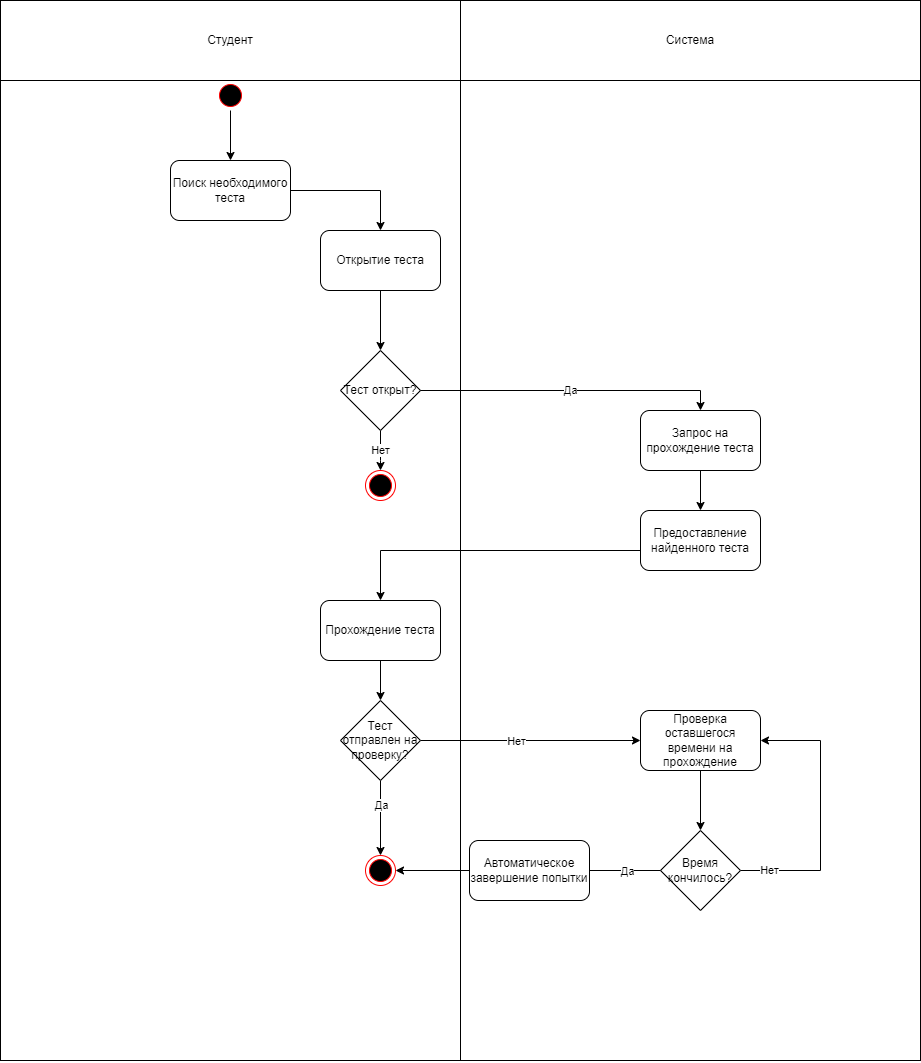


Рисунок 4. Диаграмма видов деятельности для вида деятельности «Проверка и оценка отчёта по лабораторной работе».

**Построение диаграммы последовательностей**

Прецедент: «Оценить тест» Субъекты: лектор (пользователь), система (Рис. 5).

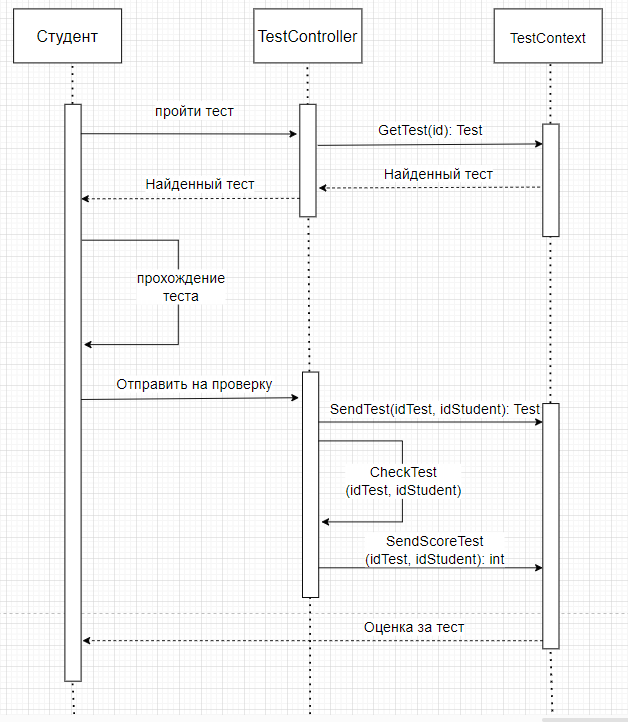


Рисунок 5. Диаграмма последовательностей для прецедента «Оценить тест».

**Построение диаграммы состояний**

Проведём анализ диаграммы состояний. Она отражает жизненный цикл объекта «Тест» (Рис. 6).

Состояния объекта «Тест»:

1. Тест закрыт
2. Тест открыт
3. Тест не пройден
4. Тест проходится студентом
5. Тест проверен

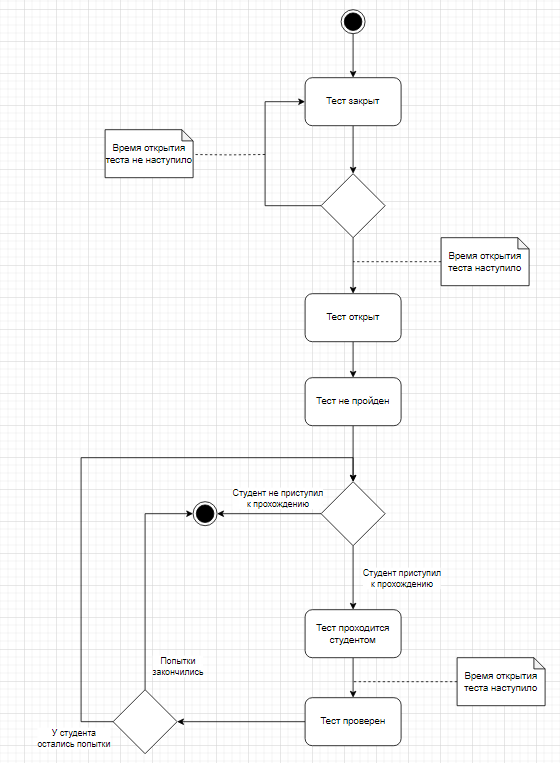


Рисунок 6. Диаграмма состояний объекта «Тест».

**Построение диаграммы развёртывания системы**

Далее построим диаграмму развёртывания системы (Рис. 7).

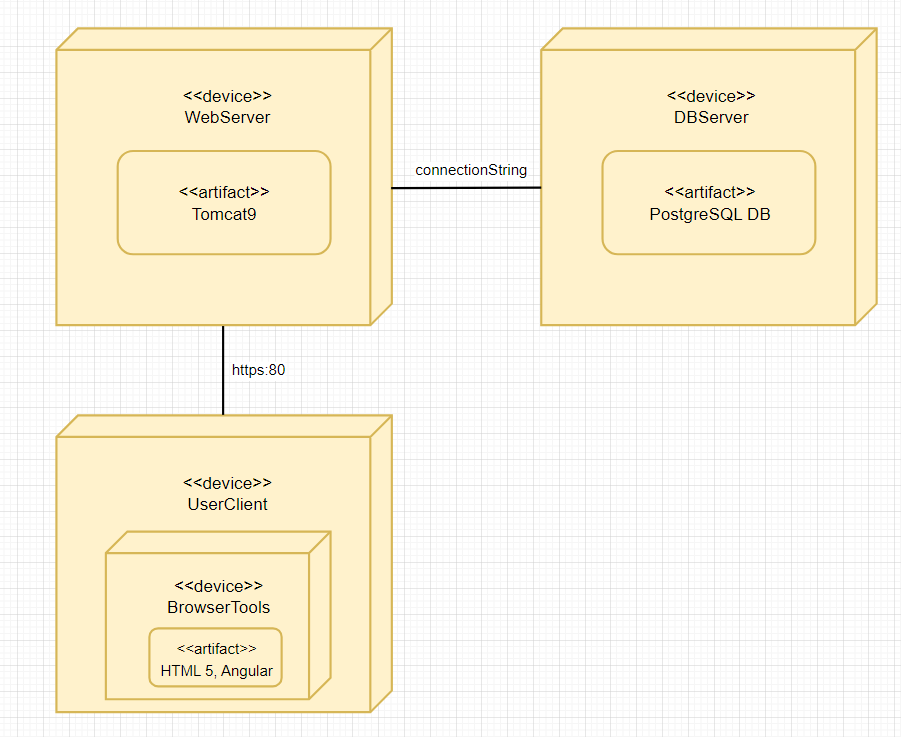


Рисунок 7. Диаграмма развёртывания.

**Построение диаграммы взаимодействия**

Задача формируется после отправки студентом отчёта на проверку. После этого лектор может принять к проверке отчёт, а затем оценить, но может и открепить отчёт по какой-либо причине (например, по просьбе студента, чтобы тот доработал отчёт) (Рис. 8).

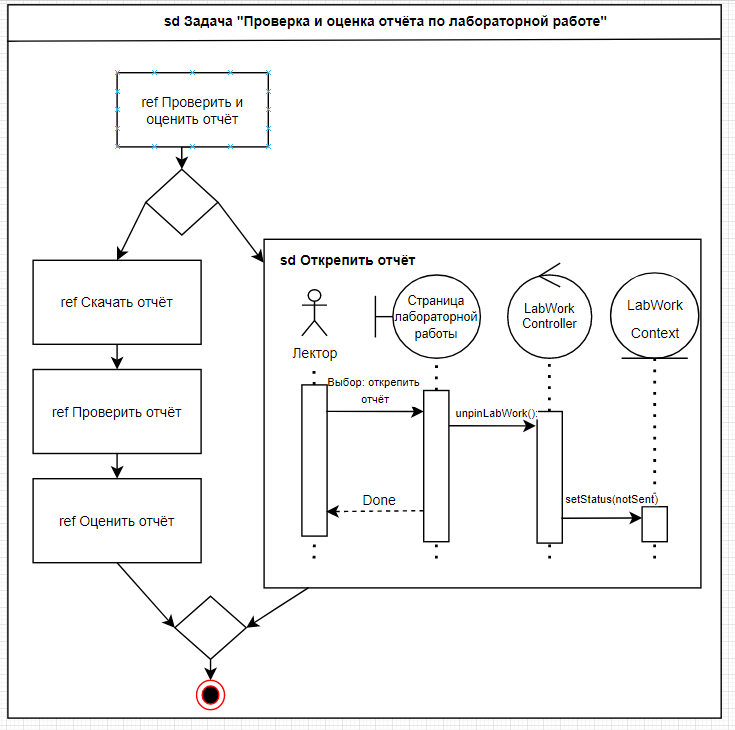


Рисунок 8. Диаграмма взаимодействия задачи "Проверка и оценка отчёта по лабораторной работе".

**Построение диаграммы компонентов системы**

Затем построим диаграмму компонентов системы (Рис. 9).

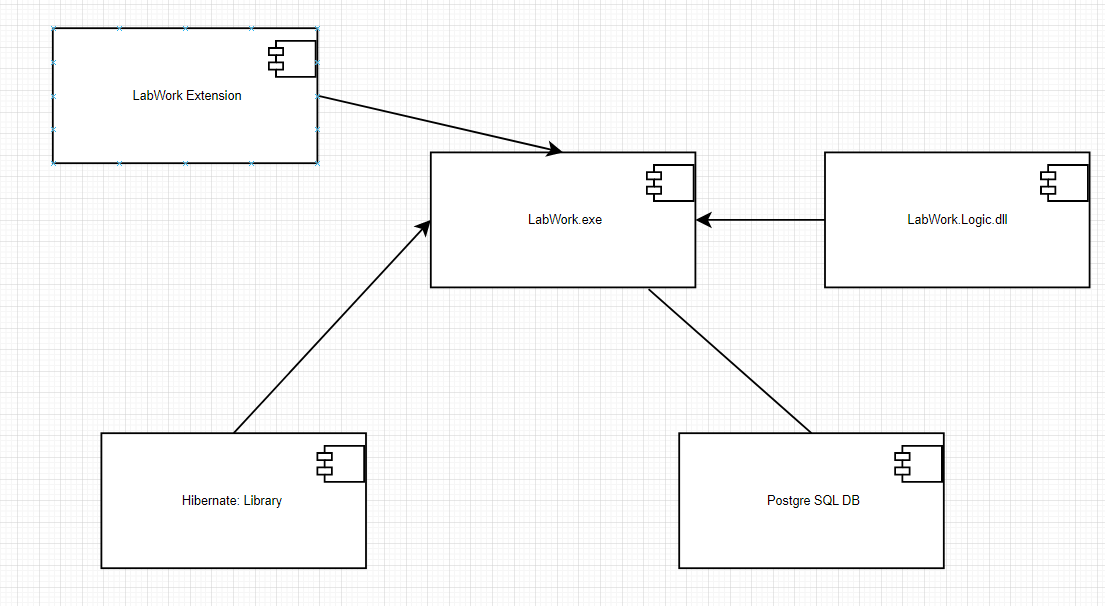


Рисунок 9. Диаграмма компонентов.

**Построение диаграммы IDEF0**

Далее построим диаграмму в нотации IDEF0 с четырьмя уровнями декомпозиции. (Рис. 10 - 14).

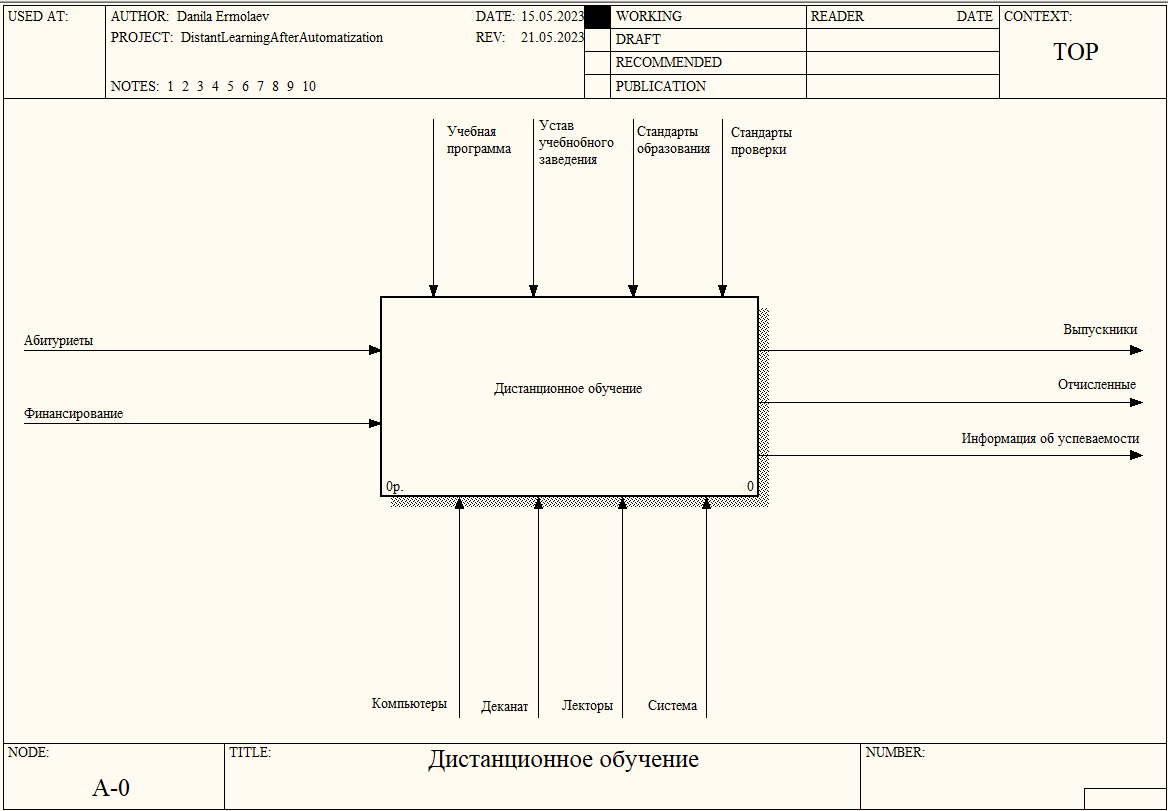


Рисунок 10. Диаграмма модели IDEF0 нулевого уровня.

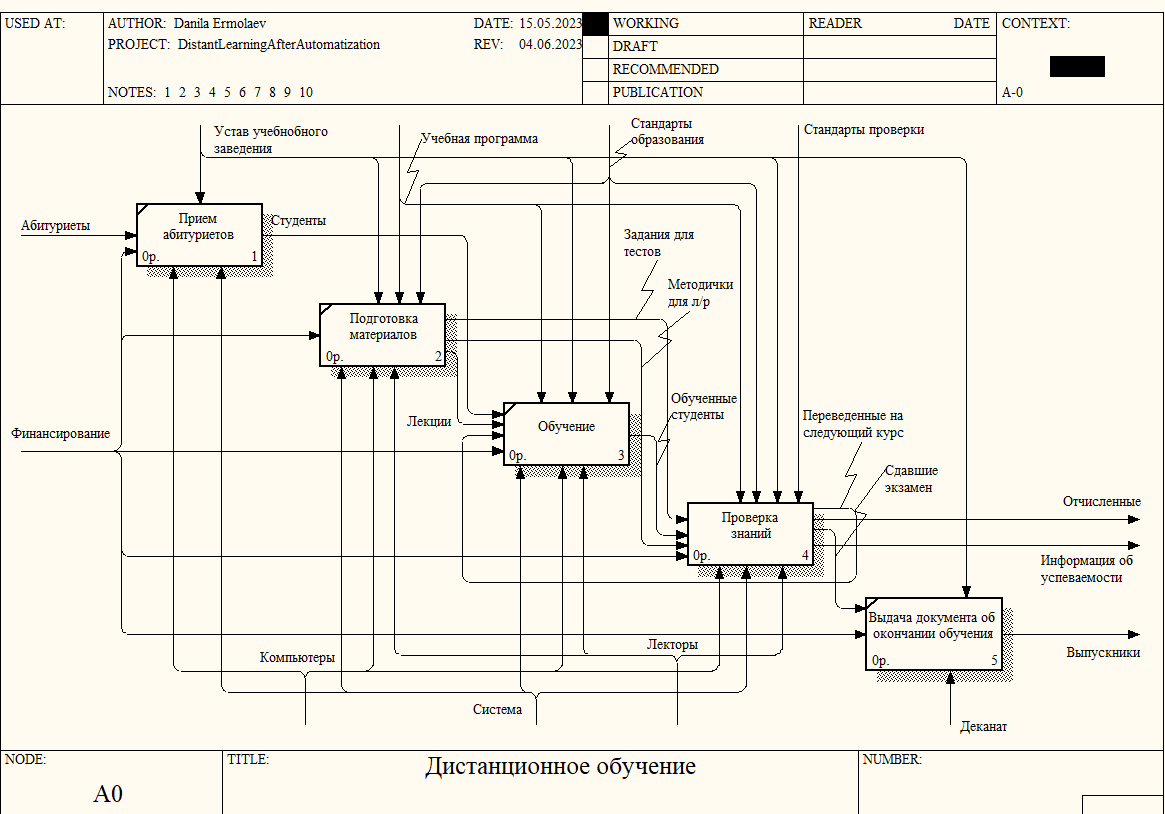


Рисунок 11. Диаграмма модели IDEF0 первого уровня.

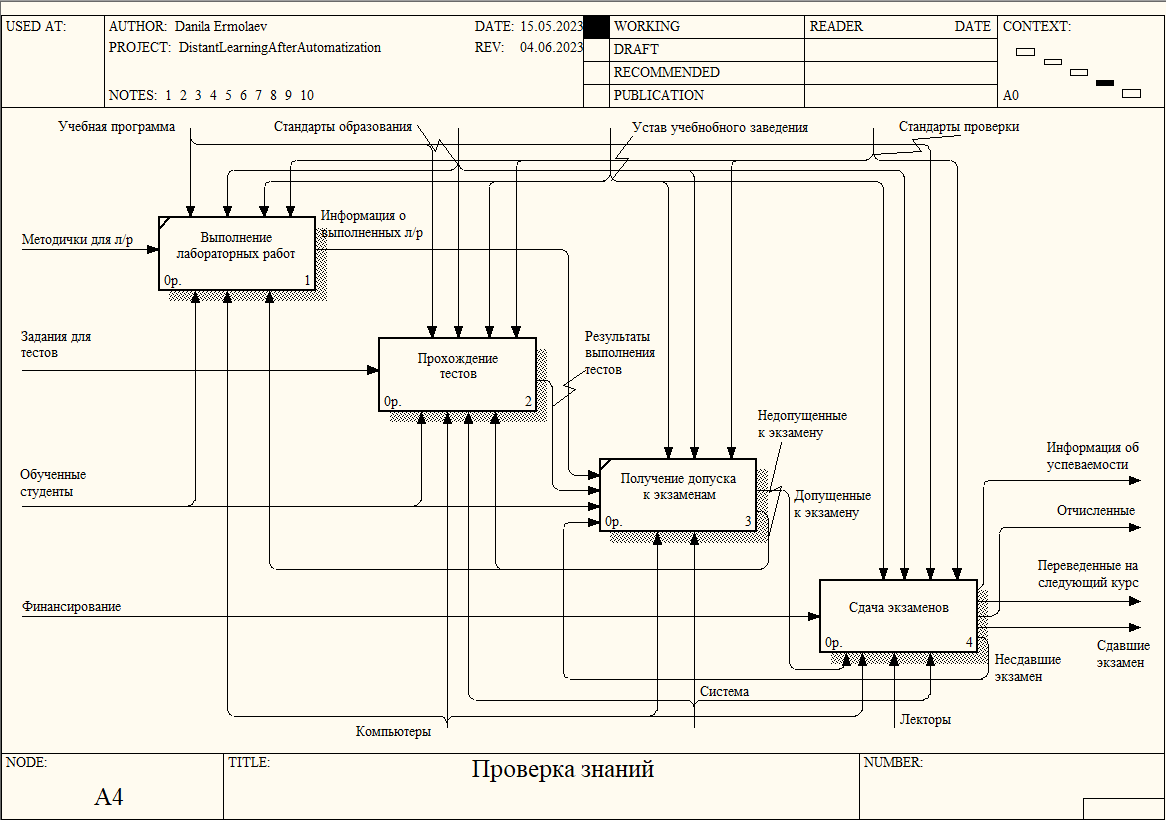


Рисунок 12. Диаграмма модели IDEF0 второго уровня.

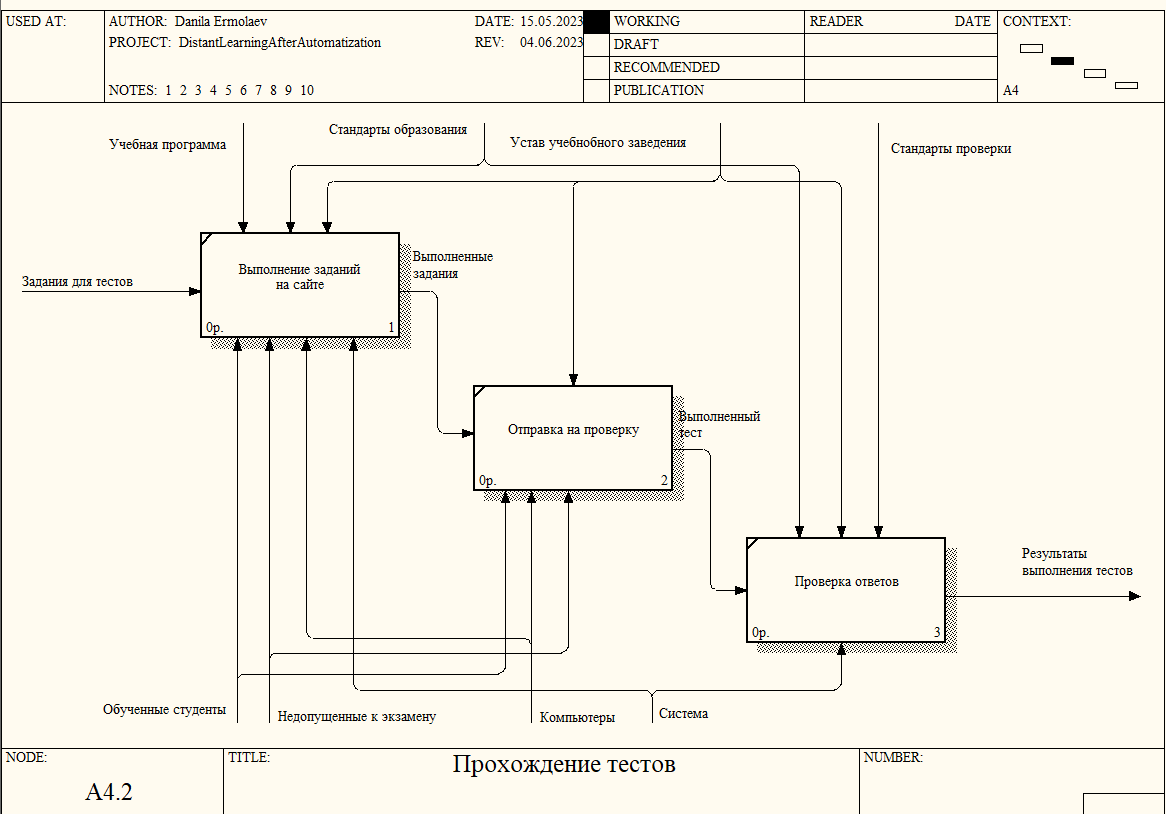


Рисунок 13. Диаграмма модели IDEF0 третьего уровня.

**Построение ER – диаграммы**

Далее построим ER – диаграмму, отражающую связи таблиц в базе данных (Рис. 14).

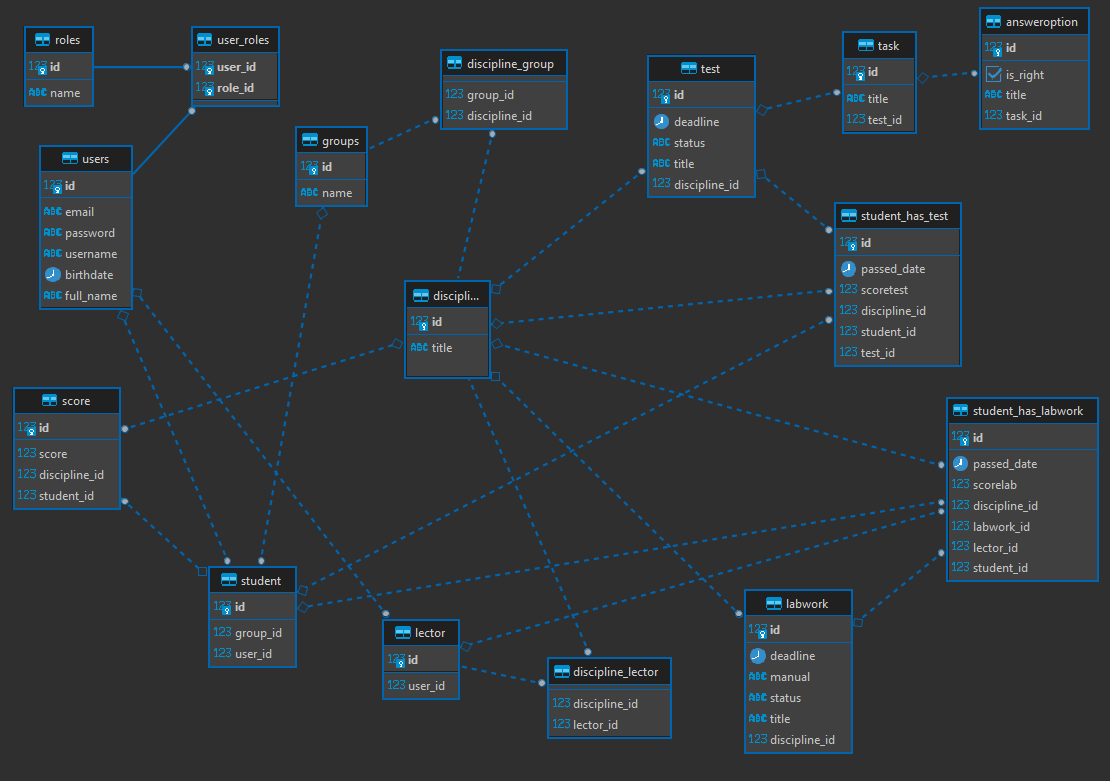


Рисунок 14. ER –диаграмма.

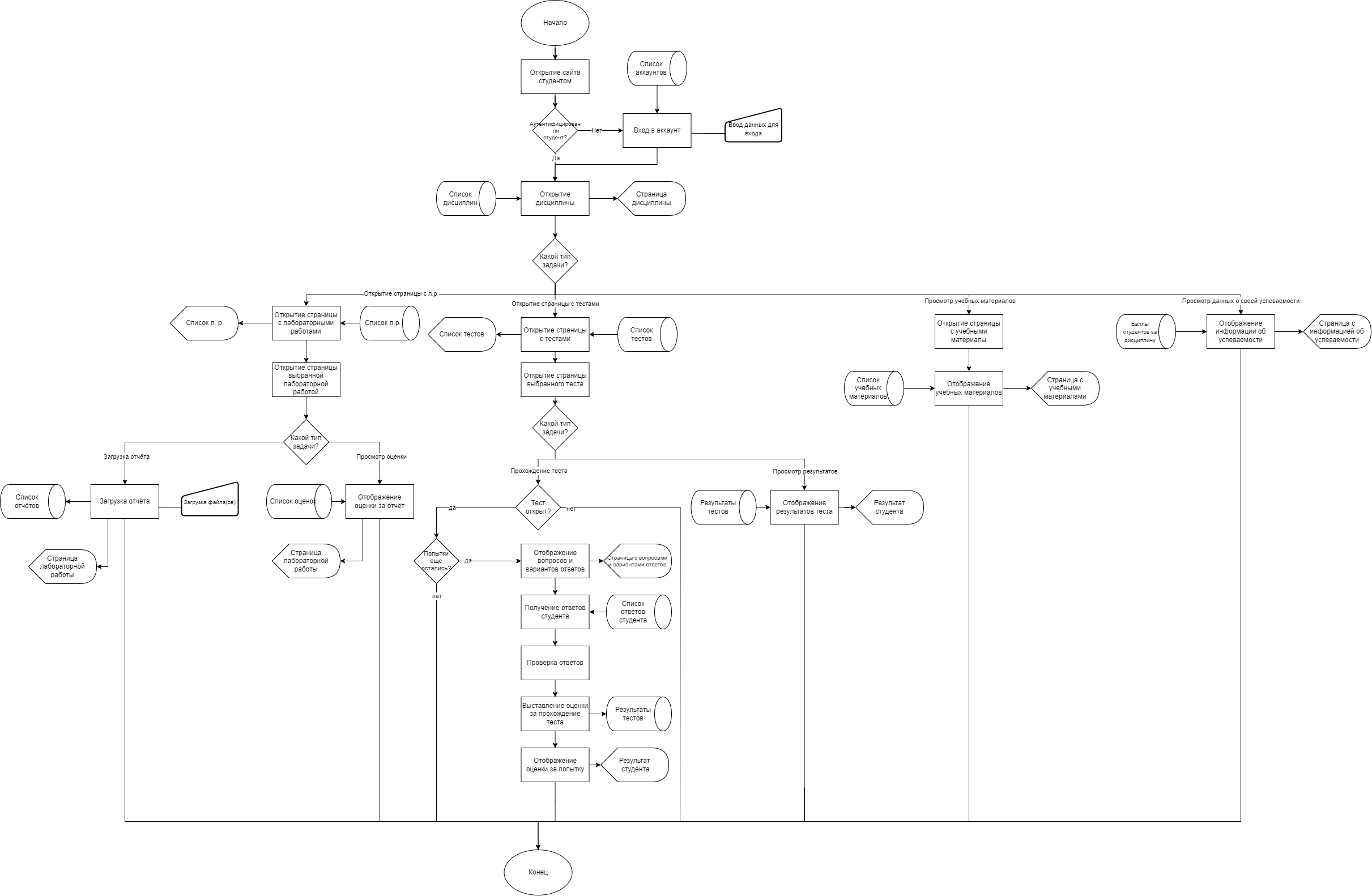


Рисунок 15. Схема работы системы (ГОСТ 19.701-90).

# РАЗРАБОТКА ПРОТОТИПА СИСТЕМЫ

После проектирования системы по дистанционному обучению требуется выполнить реализацию основных компонентов, таких как:

- База данных системы

- Сервис системы, представляющий набор контроллеров и зависимостей логических компонентов, благодаря взаимодействию которых происходит работа системы.

- Электронный сайт для системы, который будет открываться в веб-браузере и доступен для просмотра пользователям системы. Он будет состоять из набора представлений, сгенерированных на основе созданных моделей объектов системы, а также стандартной HTML – верстки (с использованием CSS и стандартных скриптов библиотеки Bootstrap).

В процесс реализации была использована такая среда программирования, как IntelliJIDEA 2021 и платформа JavaEE. Разработка будет вестись с использованием фреймворка SpringBoot.

Основой для решения будет являться создание проекта MVC с веб-архитектурой, представляющего сервис системы.

Дополнительно в директории проекта будут созданы папки, в которые будут добавляться такие классы, как:

- Модели системы, на основе которых будет создаваться база данных

- Логика системы, к которой обращаются контроллеры (сервисы)

- Репозитории моделей системы, к которым обращаются методы сервисов

## Разработка backend – части системы

Начнем с создания базы данных. В процессе разработки использовался такой фрэймфорк, как Hibernate (а конкретно – JavaPersitenseAPI), а также PostgreSQLServer – СУБД для созданной БД.

Использованный подход – CodeFirst (то есть базы данных сгенерировалась после кодирования основных зависимостей её компонентов) – на рисунке 14 представлены созданные таблицы в базе данных (показанные в программе DBeaver)

Все настройки конфигурации были произведены в файле application.properties, расположенном в ресурсах приложения (на рисунке 16):

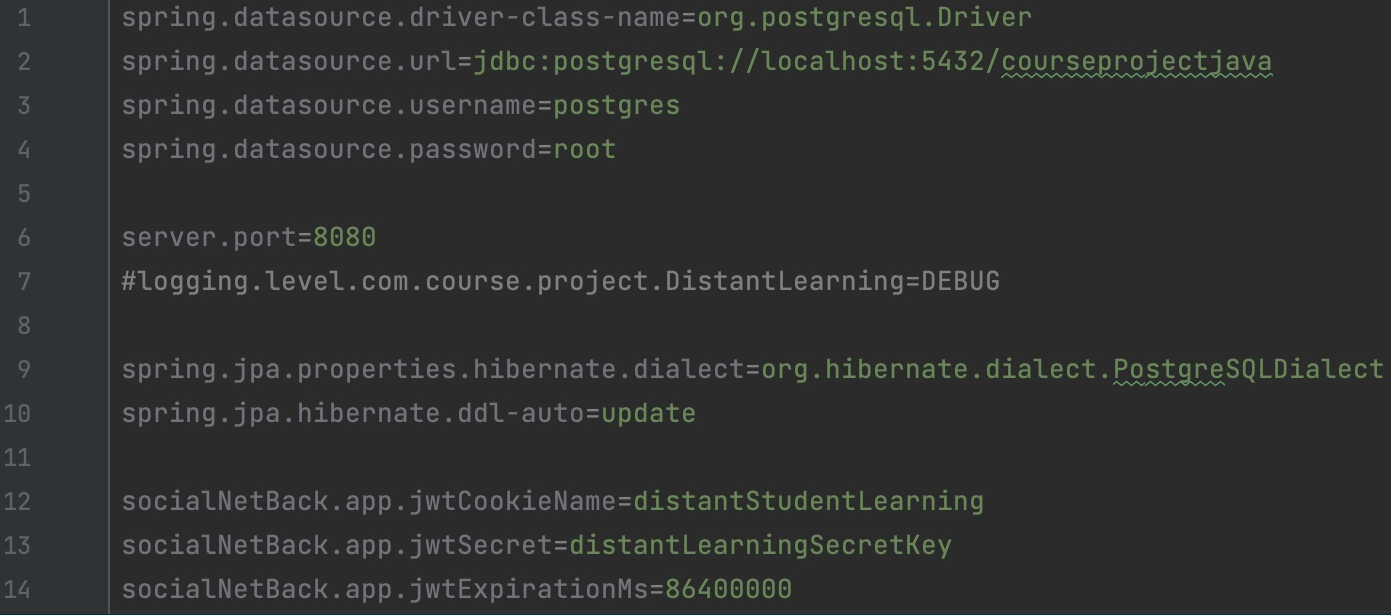


Рисунок 16. Конфигурационный файл веб-приложения Spring.

Далее были созданы модели (представлены на рисунке 17).

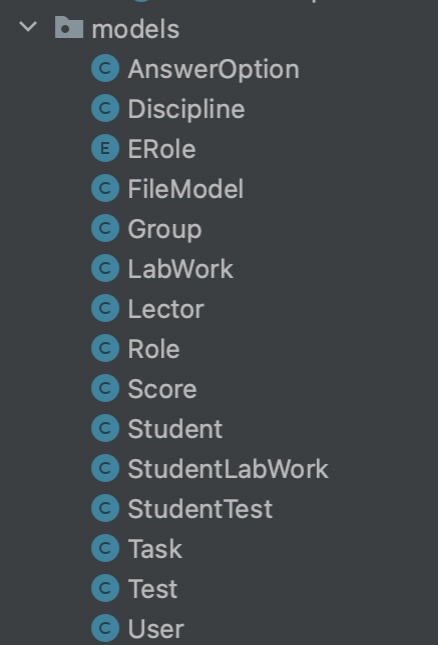


Рисунок 17. Созданные модели.

А затем для моделей были созданы репозитории (представлены на рисунке 18).

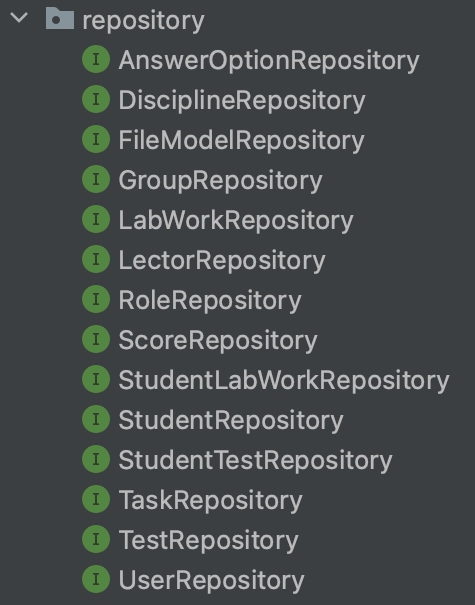


Рисунок 18. Созданные репозитории для моделей.

Далее были созданы три сервиса, представленные на рисунке 19, которые взаимодействуют с ранее созданными репозиториями, реализуя некую логику, например, выполнение CRUD операций, поиск объекта по id, оценка теста и так далее. Код реализации test service представлен в приложении А.



Рисунок 19. Файлы с сервисами.

Затем были созданы контроллеры, представленные на рисунке 20 (выделено синим). Они в свою очередь содержат методы, которые используют созданные ранее сервисы и которые вызываются в результате получения запросов от пользователя.

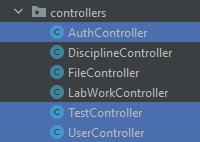


Рисунок 20. Файлы с контроллерами.

## Разработка frontend – части системы

Далее была произведена разработка внешнего вида системы, а именно создания электронного сайта, который состоит из отдельных страниц (или views – представлений). Для разработки веб-страниц использовался фреймворк Angular.

Сначала были созданы модели (представлены на рисунке 21).

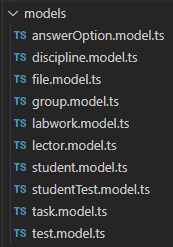


Рисунок 21. Модели.

Далее были созданы сервисы, содержащие методы, которые отправляют запросы на сервер (рисунок 22, выделено синим).

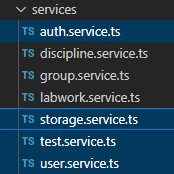


Рисунок 22. Сервисы.

Были также созданы компоненты, каждый из которых состоит из трёх файлов. Главным файлом является файл .html, содержащий код самой страницы, которую видит пользователь и с которой взаимодействует, далее идёт файл .css, определяющий стили страниц, и последним файлом является файл .ts, содержащий методы, которые обращаются к созданным ранее сервисам и которые реализуют логику взаимодействия пользователя с системой.

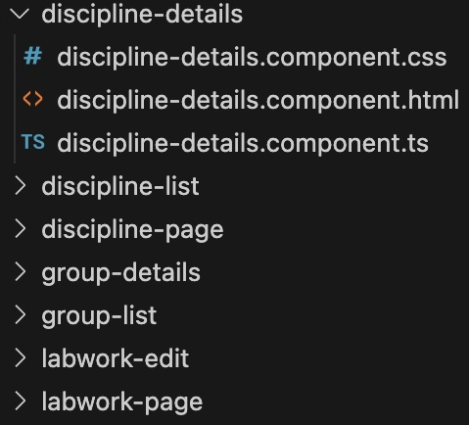


Рисунок 23. Компоненты.

На рисунках 24-35 показана работа электронного сайта:

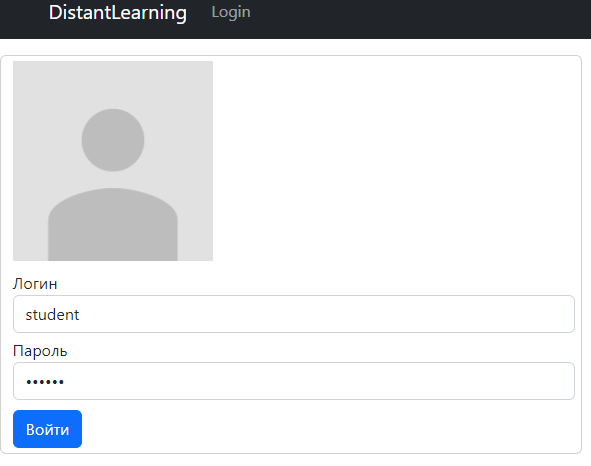


Рисунок 24. Авторизация.

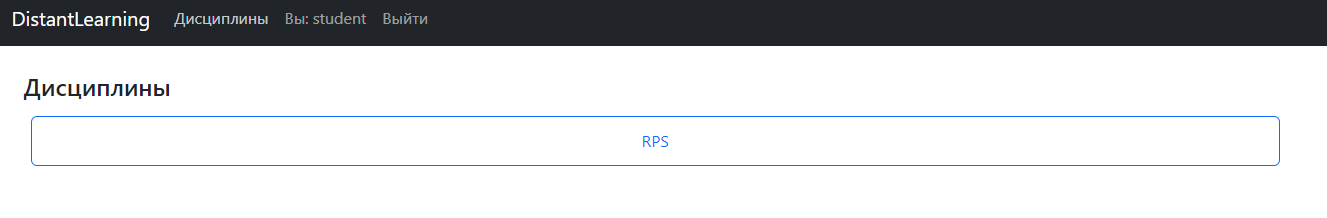


Рисунок. 25. Список дисциплин (за студента).

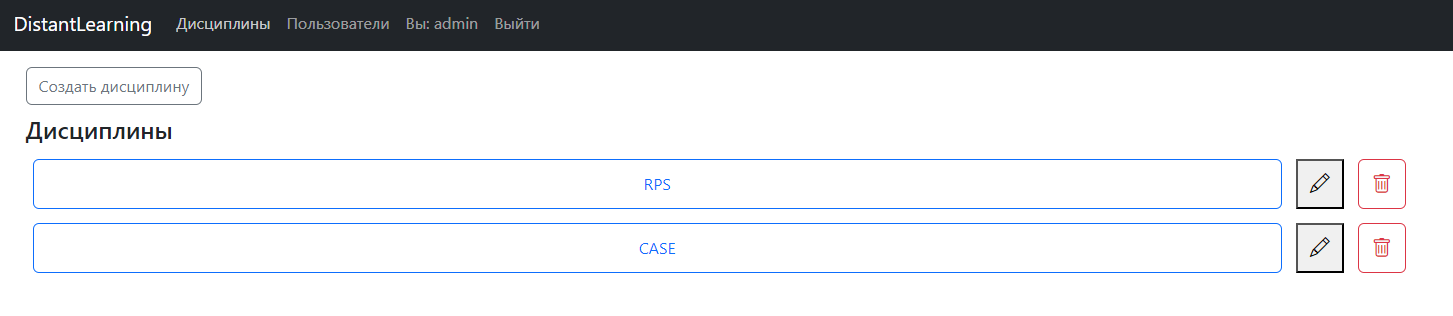


Рисунок 26. Список дисциплин (за администратора).

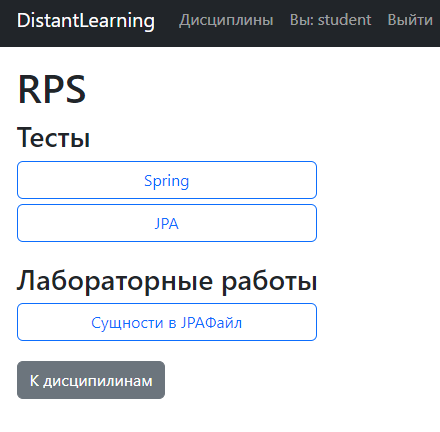


Рисунок. 27. Страница выбранной дисциплины.

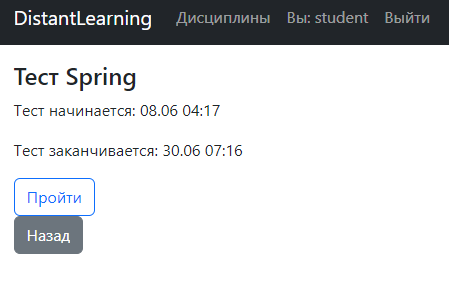


Рисунок. 28. Страница выбранного теста.

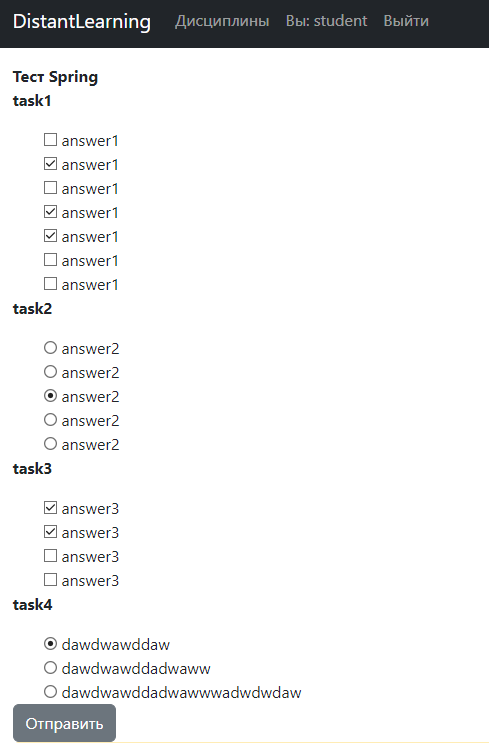


Рисунок. 29. Страница прохождения теста.

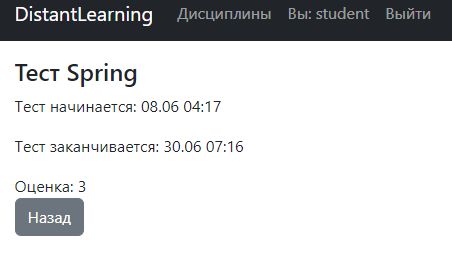


Рисунок 30. Отображение оценки (за студента).

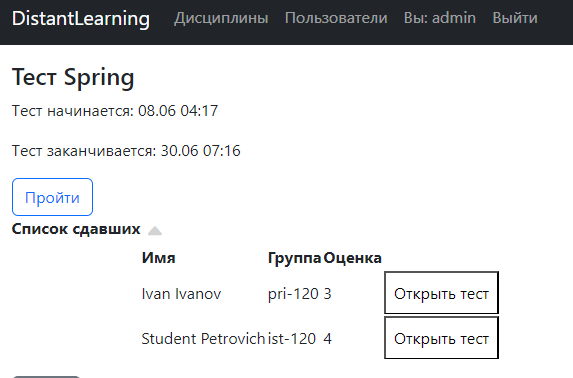


Рисунок 31. Отображение оценки (за лектора).

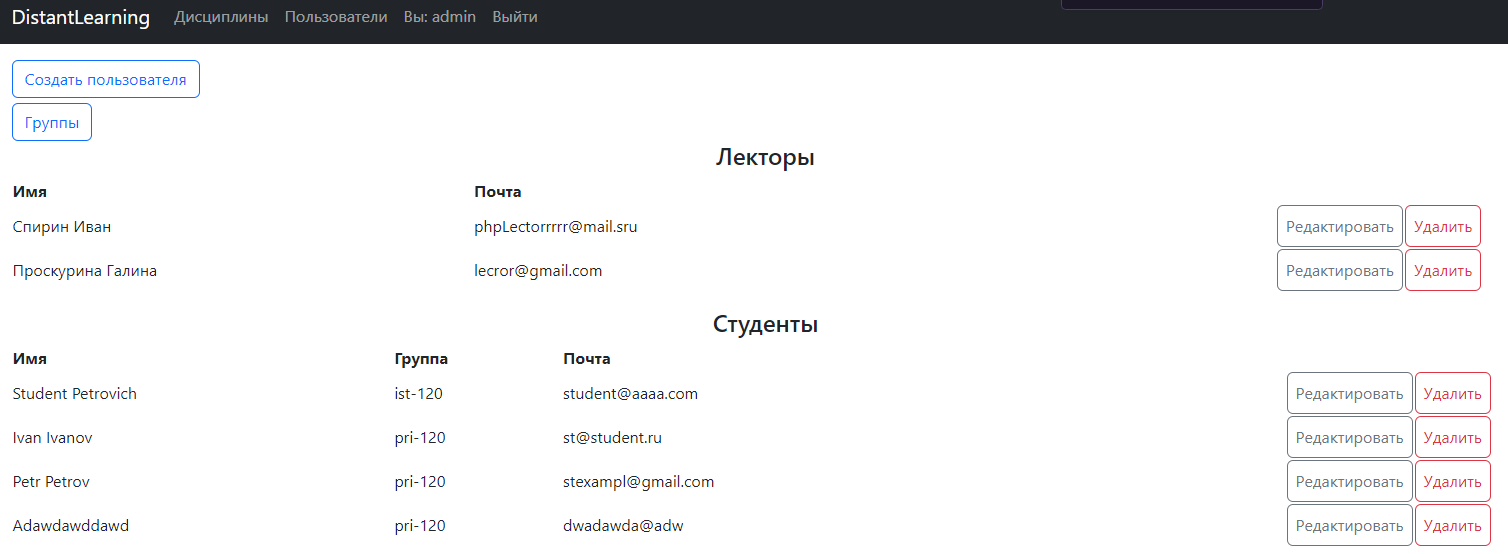


Рисунок 32. Страница вывода пользователей (за администратора).

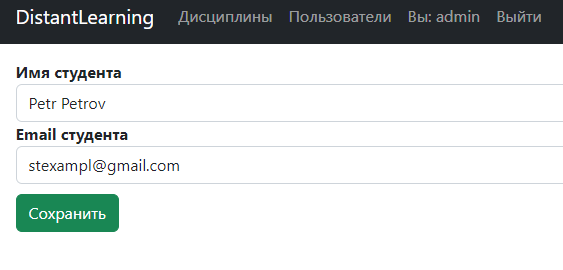


Рисунок 33. Страница Редактирования пользователей (за администратора).

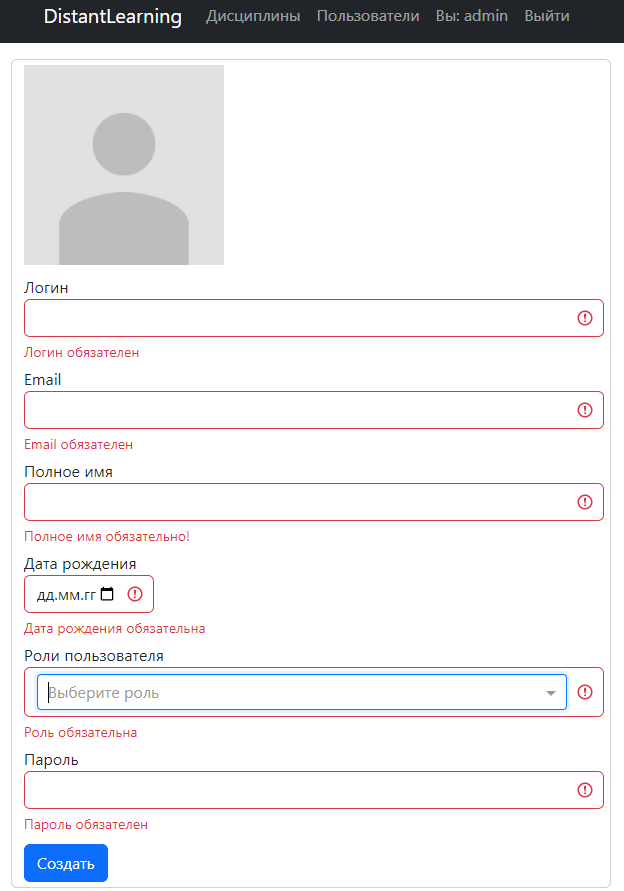


Рисунок 34. Страница создания пользователей (за администратора).

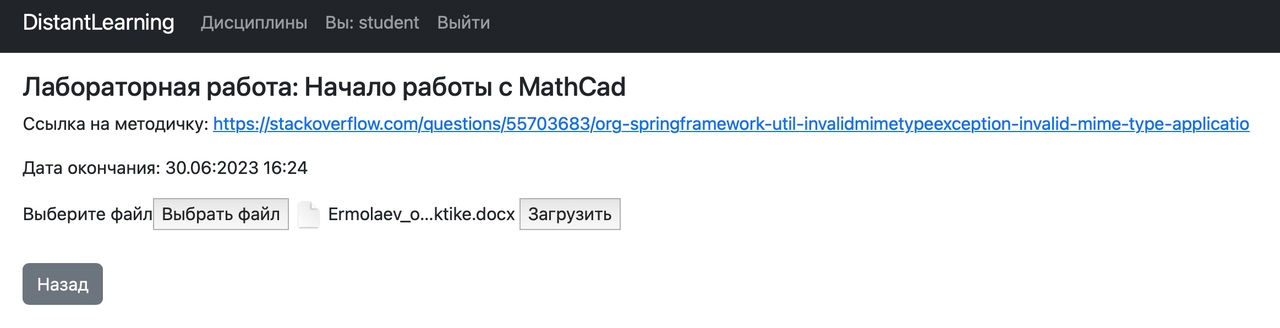


Рисунок 35. Страница с загрузкой отчета по лабораторной работе.

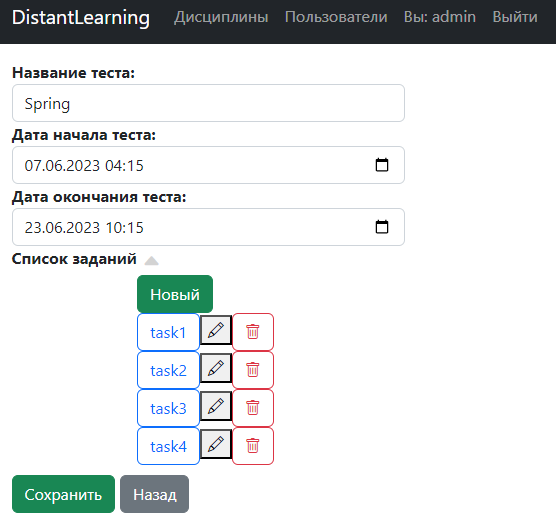


Рисунок 36. Форма создания теста.

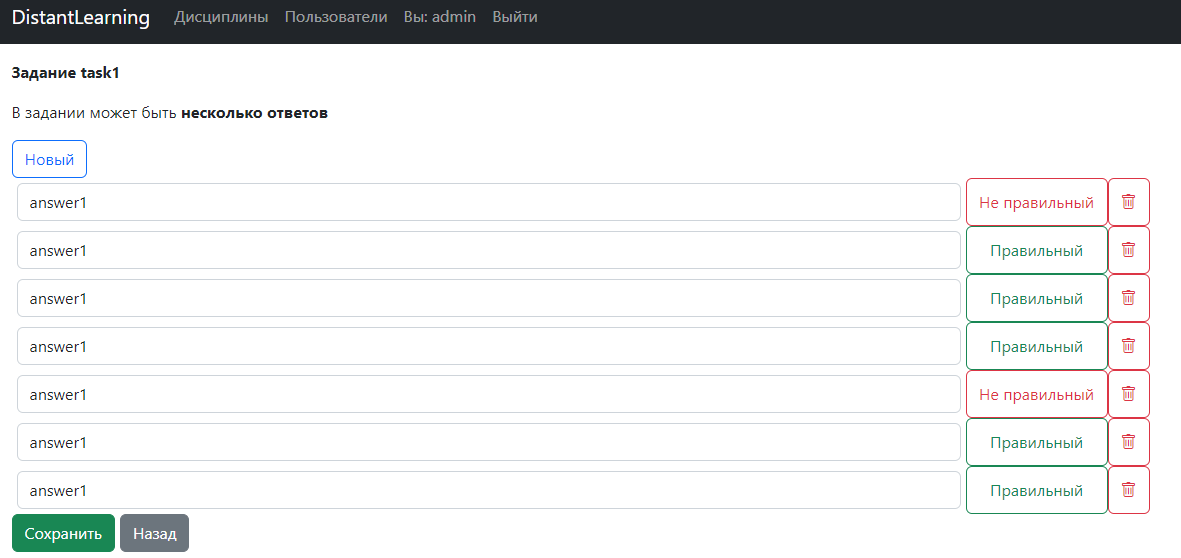


Рисунок 37. Форма добавления и редактирования ответов на вопрос.

# ТЕСТИРОВАНИЕ СИСТЕМЫ

## Нагрузочное тестирование

Для тестирования производительности SpringBoot веб-приложения было использовано программное обеспечение под названием Jmeter 5.

В ходе проведения нагрузочного тестирования попробуем повышать число одновременных запросов с авторизацией к приложению, поступающих от разных пользователей (в интервале – от 100 до 1000 пользователей). Если не произойдет закрытия приложения и получения результата запроса наподобие «Error 500», то важнейшее функциональное требование к выдерживанию нагрузки будет выполнено.

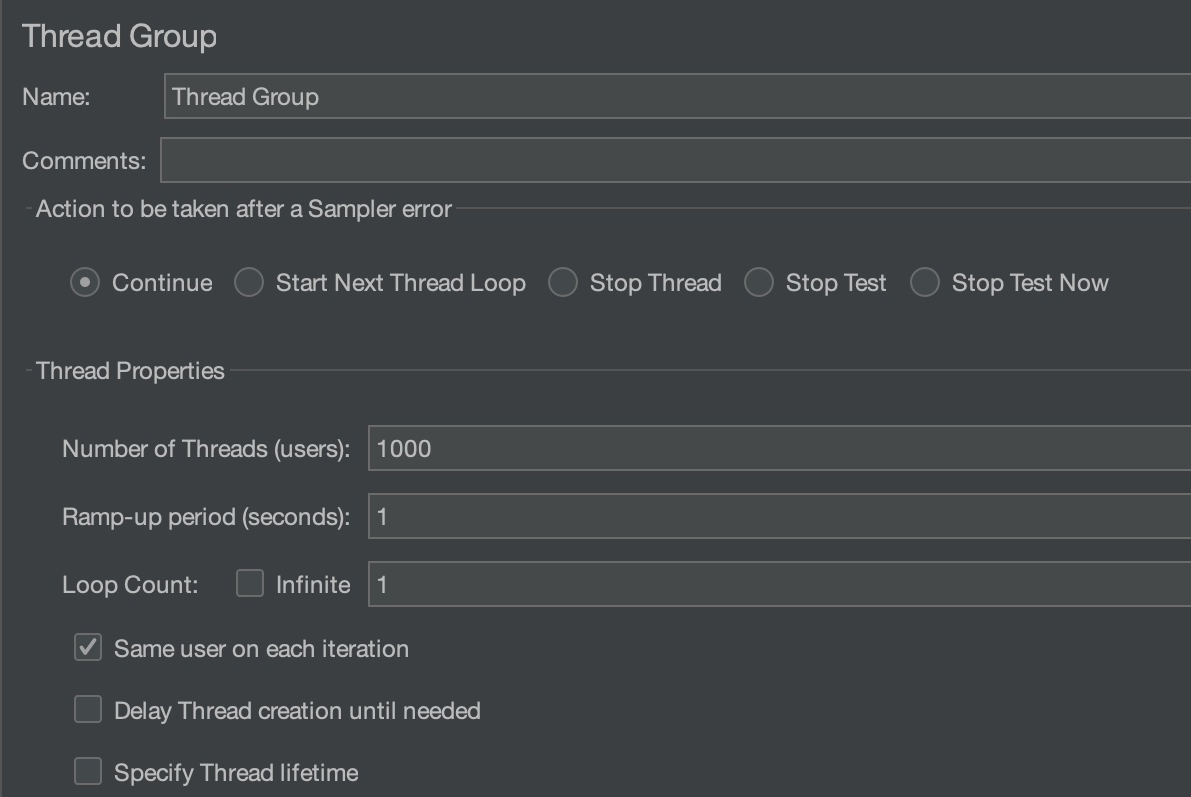


Рисунок 38. Создание теста на 1000 пользователей.

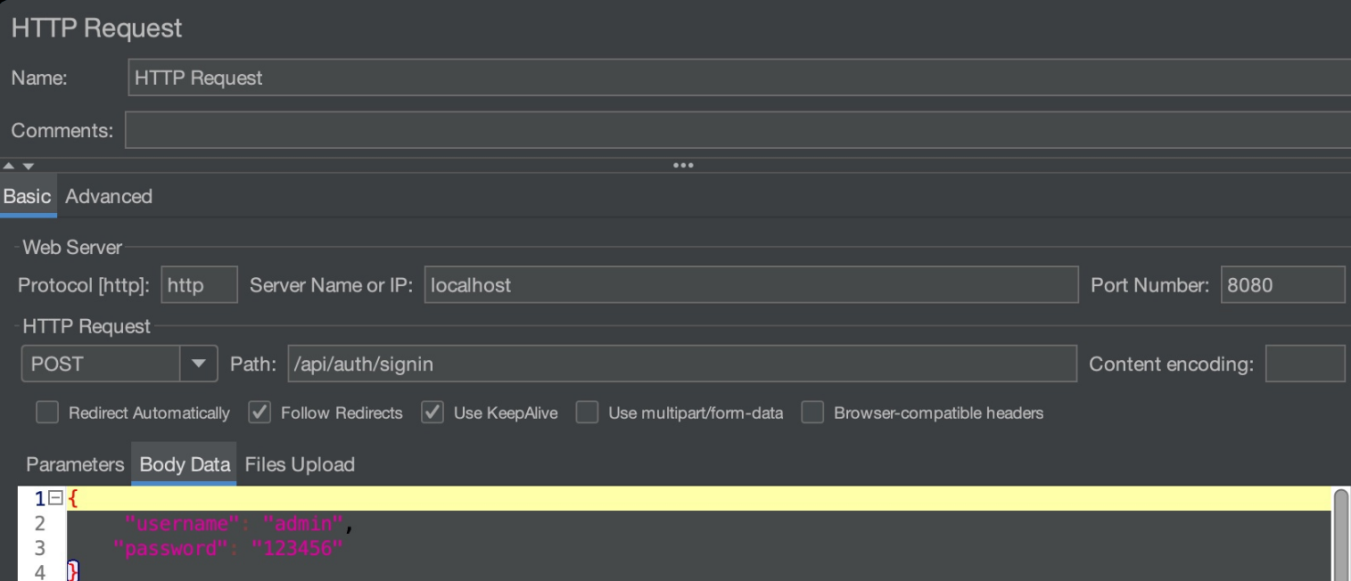


Рисунок 39. Первый POST-запрос (авторизация).

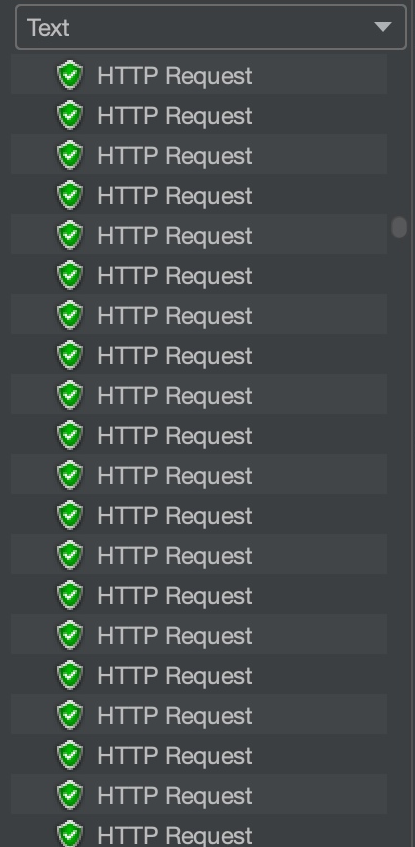


Рисунок 40. Результат POST-запроса для 1000 пользователей.

В результате можно сказать, что приложение выдержало тесты, не было закрыто во время них и тем самым успешно прошло нагрузочное тестирование.

## Репозиторий проекта

Вся разработка велась с использованием системы контроли версий git и сервиса для хранения удаленных репозиториев Github, на котором размещен код моей программной системы по ссылке: https://github.com/javavlsu/course-work-team-777-kazakov-ermolaev/tree/front

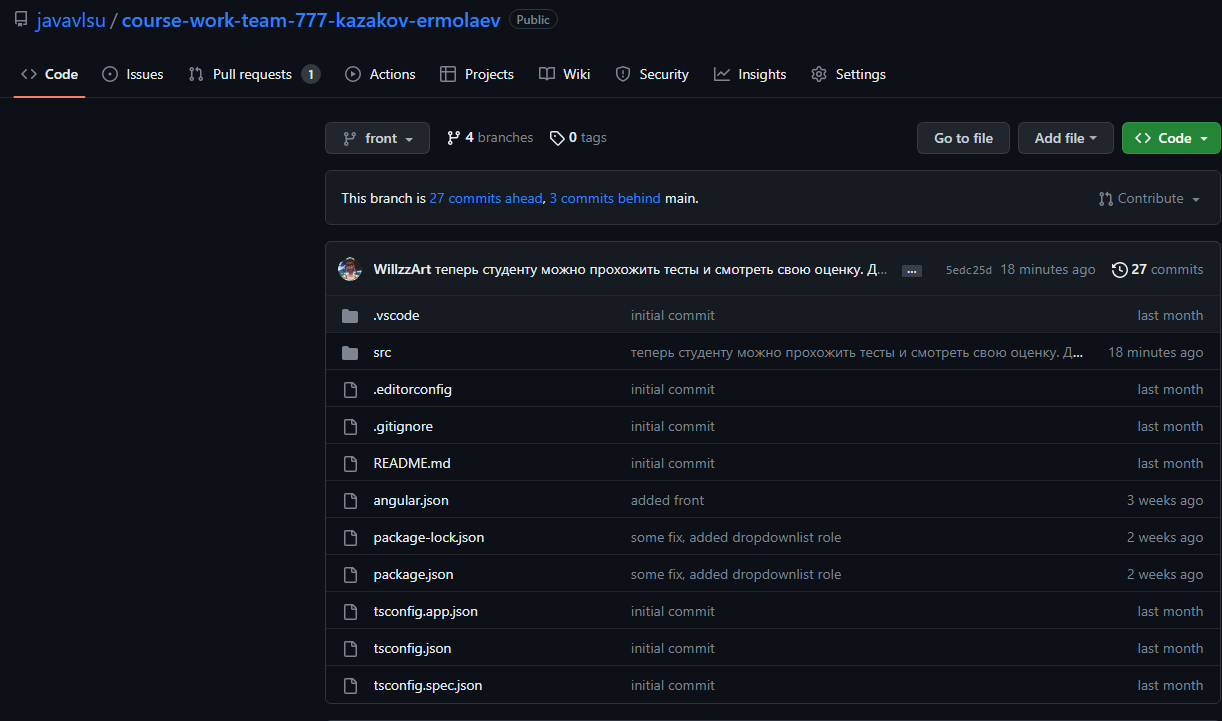


Рисунок 41. Репозиторий проекта на GitHub.

# Заключение

В ходе выполнения курсового проекта была разработана программная система для автоматизации работы университета, занимающегося дистанционным обучением. По итогу был получен рабочий протип данной программно-информационной системы, который соответствует заявленным функциональным и нефункциональным требованиям), а также обладает электронным сайтом с хорошим дизайном и рабочим понятным функционалом.

# Список использованных источников

1. Будилов, В. А. Интернет-программирование на Java / В. А. Будилов – СПб.: БХВ-Петербург, 2003.

1. Дейтел, Х. М. Технологии программирования на Java 2 / Х. М. Дей-тел, П. Дж. Дейтел, С. И. Сантри – М.: ООО "Бином-Пресс", 2003.

3. Вершинин, М. М. Java 2 EnterpriseEdition. Технологии проектиро-вания и разработки / М. М. Вершинин, Е. Б. Иванова– СПб.: БХВ-Петербург, 2003

1. METANIT.COM – Сайт о программировании.

URL: <https://metanit.com/sharp/>(Дата обращения: 27.05.2023).

Режим доступа – открытый электронный ресурс, вход в сети Интернет по URL.

1. Habr - Сайт о программировании. [Электронный ресурс] URL: <https://habr.com/ru/hub/csharp/page3/>(Дата обращения: 28.05.2023).

Режим доступа – открытый электронный ресурс, вход в сети Интернет по URL.

1. BootstrapDocs - Библиотека для HTML, CSSиJSS.

URL: <https://getbootstrap.com/> (Дата обращения: 28.05.2023).

Режим доступа – открытый электронный ресурс, вход в сети Интернет по URL.

1. SpringTutorial - Уроки и примеры работы с SpringBootфреймворком. URL: <https://www.baeldung.com/spring-tutorial> (Датаобращения: 28.05.2023).

Режим доступа – открытый электронный ресурс, вход в сети Интернет по URL.

ПРИЛОЖЕНИЕ А. Код программной системы.

package com.course.project.DistantLearning.service;  
  
import com.course.project.DistantLearning.dto.request.UpdateAnswer;  
import com.course.project.DistantLearning.dto.response.AnswerOptionResponse;  
import com.course.project.DistantLearning.dto.response.StudentResponse;  
import com.course.project.DistantLearning.models.\*;  
import com.course.project.DistantLearning.repository.\*;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.\*;  
  
@Service  
public class TestService {  
 @Autowired  
 TestRepository testRepository;  
  
 @Autowired  
 TaskRepository taskRepository;  
  
 @Autowired  
 DisciplineService disciplineService;  
  
 @Autowired  
 AnswerOptionRepository answerOptionRepository;  
  
 @Autowired  
 StudentTestRepository studentTestRepository;  
  
 @Autowired  
 StudentRepository studentRepository;  
  
 @Autowired  
 ScoreRepository scoreRepository;  
  
 public Boolean testExistsByTitle(String title) { return testRepository.existsByTitle(title); }  
  
 public List<Test> getTestByDiscipline (Long idDiscipline) {  
 List<Test> tests = new ArrayList<>();  
 Optional<Discipline> discipline = disciplineService.getDisciplineById(idDiscipline);  
 discipline.ifPresent(value -> tests.addAll(testRepository.findByDiscipline(value)));  
  
 return tests.stream().sorted(Comparator.*comparingLong*(Test::getId)).toList();  
  
 }  
  
 public Optional<Test> getTestById(Long idTest) {  
 Optional<Test> test = testRepository.findById(idTest);  
 if (test.isPresent()) {  
 Test \_test = test.get();  
 if (\_test.getStatus().equals("close") && \_test.getDateStart().before(new Date())) {  
 if (\_test.getDeadline().after(new Date())) {  
 \_test.setStatus("open");  
 testRepository.save(\_test);  
 }  
 }  
  
 if (\_test.getStatus().equals("open") && \_test.getDeadline().before(new Date())) {  
 \_test.setStatus("close");  
 testRepository.save(\_test);  
 }  
 }  
 return testRepository.findById(idTest);  
 }  
  
 public boolean createTest(Long idDiscipline, Test test) {  
 Optional<Discipline> discipline = disciplineService.getDisciplineById(idDiscipline);  
 if (discipline.isPresent()) {  
 test.setDiscipline(discipline.get());  
 test.setStatus("close");  
 testRepository.save(test);  
 return true;  
 }  
 return false;  
  
 }  
  
 public boolean updateTest(Long idTest, Test test) {  
 Optional<Test> testData = testRepository.findById(idTest);  
  
 if (testData.isPresent()) {  
 Test \_test = testData.get();  
 \_test.setTitle(test.getTitle());  
 \_test.setDateStart(test.getDateStart());  
 \_test.setDeadline(test.getDeadline());  
 testRepository.save(\_test);  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
  
 public void deleteTest(Long idTest) { testRepository.deleteById(idTest); }  
  
 public Boolean taskExistsByTitle(String title) { return taskRepository.existsByTitle(title); }  
  
  
  
 public List<Task> getTaskByTest(Long idTest) {  
 List<Task> tasks = new ArrayList<>();  
 Optional<Test> test = testRepository.findById(idTest);  
 test.ifPresent(value -> tasks.addAll(taskRepository.findByTest(value)));  
 return tasks.stream().sorted(Comparator.*comparingLong*(Task::getId)).toList();  
 }  
  
 public Optional<Task> getTaskById(Long idTask) { return taskRepository.findById(idTask); }  
  
 public boolean createTask(Long idTest, Task task) {  
 Optional<Test> test = testRepository.findById(idTest);  
 if (test.isPresent()) {  
 task.setTest(test.get());  
 taskRepository.save(task);  
 return true;  
 }  
 return false;  
 }  
  
 public boolean updateTask(Long idTask, Task task) {  
 Optional<Task> taskData = taskRepository.findById(idTask);  
  
 if (taskData.isPresent()) {  
 Task \_task = taskData.get();  
 \_task.setTitle(task.getTitle());  
 \_task.setType(task.getType());  
 taskRepository.save(\_task);  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
  
 public void deleteTask(Long idTask) { taskRepository.deleteById(idTask); }  
  
  
  
 public Boolean answerOptionExistsByTitle(String title) { return answerOptionRepository.existsByTitle(title); }  
  
 public List<AnswerOptionResponse> geyAllAnswerOptionInTest(Long idTest) {  
 List<AnswerOption> answerOptions = new ArrayList<>();  
 List<AnswerOptionResponse> answerOptionResponses = new ArrayList<>();  
 Optional<Test> test = testRepository.findById(idTest);  
  
 if (test.isPresent()) {  
 List<Task> tasks = taskRepository.findByTest(test.get());  
 for (var task: tasks) {  
 answerOptions.addAll(answerOptionRepository.findByTask(task));  
 }  
 }  
  
 for (var answer: answerOptions) {  
 var answerResponse = new AnswerOptionResponse();  
  
 answerResponse.setId(answer.getId());  
 answerResponse.setTitle(answer.getTitle());  
 answerResponse.setIdTask(answer.getTask().getId());  
 if (answer.getRight()) {  
 answerResponse.setIsRight("true");  
 } else {  
 answerResponse.setIsRight("false");  
 }  
 answerOptionResponses.add(answerResponse);  
  
 }  
 return answerOptionResponses;  
 }  
  
 public List<AnswerOptionResponse> getAnswerOptionByTask(Long idTask) {  
 List<AnswerOption> answerOptions = new ArrayList<>();  
 List<AnswerOptionResponse> answerOptionResponses = new ArrayList<>();  
  
 Optional<Task> task = taskRepository.findById(idTask);  
 task.ifPresent(value -> answerOptions.addAll(answerOptionRepository.findByTask(value)));  
  
 for(var answer: answerOptions) {  
 var answerResponse = new AnswerOptionResponse();  
  
 answerResponse.setId(answer.getId());  
 answerResponse.setTitle(answer.getTitle());  
 if (answer.getRight()) {  
 answerResponse.setIsRight("true");  
 } else {  
 answerResponse.setIsRight("false");  
 }  
 answerOptionResponses.add(answerResponse);  
 }  
  
 return answerOptionResponses.stream().sorted(Comparator.*comparingLong*(AnswerOptionResponse::getId)).toList();  
 }  
  
 public AnswerOptionResponse getAnswerOptionById(Long idAnswerOption) {  
 Optional<AnswerOption> answerOption = answerOptionRepository.findById(idAnswerOption);  
 AnswerOptionResponse answerOptionResponse = new AnswerOptionResponse();  
 if (answerOption.isPresent()) {  
 answerOptionResponse.setId(answerOption.get().getId());  
 answerOptionResponse.setTitle(answerOption.get().getTitle());  
 if (answerOption.get().getRight()) {  
 answerOptionResponse.setIsRight("true");  
 } else {  
 answerOptionResponse.setIsRight("false");  
 }  
 }  
 return answerOptionResponse;  
 }  
  
 public boolean createAnswerOption(Long idTask, AnswerOptionResponse answerOptionResponse) {  
 Optional<Task> task = taskRepository.findById(idTask);  
 if (task.isPresent()) {  
 AnswerOption answerOption = new AnswerOption();  
 answerOption.setTitle(answerOptionResponse.getTitle());  
 answerOption.setRight(Objects.*equals*(answerOptionResponse.getIsRight(), "true"));  
 answerOption.setTask(task.get());  
 answerOptionRepository.save(answerOption);  
 return true;  
 }  
 return false;  
 }  
  
 public boolean isOnlyOneAnswerUpdate(UpdateAnswer updateAnswer) {  
 Optional<AnswerOptionResponse> answerOptionResponse = updateAnswer.getAnswers().stream().findAny();  
 if (answerOptionResponse.isPresent()) {  
 Task task = answerOptionRepository.findById(answerOptionResponse.get().getId()).get().getTask();  
 if (Objects.*equals*(task.getType(), "radio")) {  
 var i = 0;  
 for(var answer: updateAnswer.getAnswers()) {  
 if (Objects.*equals*(answer.getIsRight(), "true"))  
 i++;  
 }  
 return i < 2 & i > 0;  
 }  
 return true;  
 }  
 return false;  
 }  
  
 public boolean isOnlyOneAnswerCreate(Long idTask, AnswerOptionResponse answerOptionResponse) {  
 Optional<Task> task = taskRepository.findById(idTask);  
 if (task.isPresent()) {  
 if (Objects.*equals*(task.get().getType(), "radio")) {  
 if (Objects.*equals*(answerOptionResponse.getIsRight(), "true")) {  
 for (var answer: answerOptionRepository.findByTask(task.get())) {  
 if (answer.getRight())  
 return false;  
 }  
 }  
 }  
 return true;  
 }  
 return true;  
 }  
  
 public boolean updateAnswerOption(UpdateAnswer updateAnswer) {  
 List<AnswerOptionResponse> answerOptionResponses = updateAnswer.getAnswers();  
  
 if (!answerOptionResponses.isEmpty()) {  
 for (var answer: answerOptionResponses) {  
 var answerOption = answerOptionRepository.findById(answer.getId());  
 if(answerOption.isPresent()) {  
 var \_answerOption = answerOption.get();  
 \_answerOption.setTitle(answer.getTitle());  
 \_answerOption.setRight(Objects.*equals*(answer.getIsRight(), "true"));  
 answerOptionRepository.save(\_answerOption);  
 }  
 }  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
  
 public void deleteAnswerOption(Long idAnswerOption) { answerOptionRepository.deleteById(idAnswerOption); }  
  
  
 public boolean checkStudentAnswer(User user, Long idTest, UpdateAnswer updateAnswer) {  
 try {  
 List<AnswerOption> answerOptions = new ArrayList<>();  
 List<Task> tasks = new ArrayList<>();  
 testRepository.findById(idTest).ifPresent(value -> tasks.addAll(taskRepository.findByTest(value)));  
  
 for (var answer: updateAnswer.getAnswers()) {  
 answerOptionRepository.findById(answer.getId()).ifPresent(answerOptions::add);  
 }  
  
 var score = 0;  
 for (var task: tasks) {  
 for(var answer: answerOptions) {  
 if (answer.getTask().getId().equals(task.getId())) {  
 if (Objects.*equals*(task.getType(), "radio")) {  
 if (answer.getRight()) {  
 score++;  
 break;  
 }  
 }  
 else {  
 var i = 0;  
 if (answer.getRight()) i++;  
 else i--;  
  
 if (i < 0) i = 0;  
  
 score += i;  
 }  
 }  
 }  
 }  
  
 Optional<Student> student = studentRepository.findByUser(user);  
 Optional<Test> test = testRepository.findById(idTest);  
 if (student.isPresent() && test.isPresent()) {  
 Optional<Score> scoreEntity = scoreRepository.findByStudentAndDiscipline(student.get(), test.get().getDiscipline());  
 Optional<StudentTest> studentTestData = studentTestRepository.findByStudentAndTest(student.get(), test.get());  
 if (scoreEntity.isPresent()) {  
 Score \_scoreEntity = scoreEntity.get();  
 if (studentTestData.isEmpty()) {  
 StudentTest studentTest = new StudentTest();  
 studentTest.setScoreTest(score);  
 studentTest.setStudent(student.get());  
 studentTest.setTest(test.get());  
 studentTest.setDiscipline(test.get().getDiscipline());  
 studentTest.setPassedDate(new Date());  
 studentTestRepository.save(studentTest);  
  
 \_scoreEntity.setScore(scoreEntity.get().getScore() + studentTest.getScoreTest());  
 } else {  
 StudentTest studentTest = studentTestData.get();  
 studentTest.setScoreTest(score);  
 studentTest.setPassedDate(new Date());  
 studentTestRepository.save(studentTest);  
  
 \_scoreEntity.setScore(scoreEntity.get().getScore() - (studentTestData.get().getScoreTest() - studentTest.getScoreTest()));  
 }  
 scoreRepository.save(\_scoreEntity);  
 }  
 }  
 return true;  
 } catch (Exception e) {  
 return false;  
 }  
 }  
  
 public Optional<StudentTest> getStudentScoreForTest(Long idTest, User user) {  
 Optional<Student> student = studentRepository.findByUser(user);  
 Optional<Test> test = testRepository.findById(idTest);  
 if (student.isPresent() && test.isPresent()) {  
 return studentTestRepository.findByStudentAndTest(student.get(), test.get());  
 }  
  
 return Optional.*empty*();  
 }  
  
 public List<StudentResponse> getScoresForTest(Long idTest) {  
 List<StudentResponse> studentResponseList = new ArrayList<>();  
 Optional<Test> test = testRepository.findById(idTest);  
 if (test.isPresent()) {  
 for (var student: studentRepository.findAll()) {  
 Optional<StudentTest> studentTest = studentTestRepository.findByStudentAndTest(student, test.get());  
 if (studentTest.isPresent()) {  
 StudentResponse studentResponse = new StudentResponse();  
 studentResponse.setId(student.getId());  
 studentResponse.setName(student.getUser().getFullName());  
 studentResponse.setGroupName(student.getGroup().getName());  
 studentResponse.setEmail(student.getUser().getEmail());  
 studentResponse.setScoreTest(studentTest.get().getScoreTest());  
 studentResponseList.add(studentResponse);  
 }  
 }  
 }  
 return studentResponseList;  
 }  
  
 public void getMoreChance(StudentResponse studentResponse, Long idTest, Long idDiscipline) {  
 Optional<Student> student = studentRepository.findById(studentResponse.getId());  
 Optional<Test> test = testRepository.findById(idTest);  
 Optional<Discipline> discipline = disciplineService.getDisciplineById(idDiscipline);  
  
 if (student.isPresent() & test.isPresent() & discipline.isPresent()) {  
 Optional<Score> scoreData = scoreRepository.findByStudentAndDiscipline(student.get(), discipline.get());  
 Optional<StudentTest> studentTest = studentTestRepository.findByStudentAndTest(student.get(), test.get());  
  
 if (scoreData.isPresent() && studentTest.isPresent()) {  
 Score score = scoreData.get();  
 score.setScore(scoreData.get().getScore() - studentTest.get().getScoreTest());  
 scoreRepository.save(score);  
 studentTestRepository.deleteById(studentTest.get().getId());  
 }  
 }  
 }  
}