Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет инженерно-экономический
Кафедра экономической информатики
Дисциплина «Программирование сетевых приложений»

«К ЗАЩИТЕ ДОПУСТИТЬ»
Руководитель курсового проекта
старший преподаватель
_____ Т.М. Унучек
___.____.2021

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему

РЕЙТИНГОВЫЕ ОЦЕНКИ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ И ПРОГРАММНАЯ ПОДДЕРЖКА МОДУЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЫ В ВУЗЕ

БГУИР КП 1-40 05 01-10 030 ПЗ

Выполнил студент группы 914301
Батян Глеб Павлович

_____
(подпись студента)

Курсовой проект представлен на проверку ___.____.2021

_____
(подпись студента)

Минск 2021

РЕФЕРАТ

БГУИР КП 1-40 05 01 10 030 ПЗ

РЕЙТИНГОВЫЕ ОЦЕНКИ В ОБРАЗОВАТЕЛЬНОМ ПРОЦЕССЕ И ПРОГРАММНАЯ ПОДДЕРЖАКА МОДУЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЫ В ВУЗЕ / Г.П. БАТЯН. – Минск: БГУИР, 2021. – 78 с., чертежей (плакатов) – 5 л. формата А3

ПРОГРАММИРОВАНИЕ ПРИЛОЖЕНИЯ ДЛЯ ПОДДЕРЖКИ МОДУЛЬНО-РЕЙТИНГОВОЙ СИСТЕМЫ В ВУЗЕ С ИСПОЛЬЗОВАНИЕМ АРХИТЕКТУРЫ КЛИЕНТ-СЕРВЕР И ОРГАНИЗАЦИЕЙ ВЗАИМОДЕЙСТВИЯ С БАЗОЙ ДАННЫХ НА ОБЪЕКТНО-ОРИЕНТИРОВАННОМ ЯЗЫКЕ JAVA.

Предмет: приложение.

Объект: методы разработки приложения с использованием языков Java и CSS и стандартной библиотеки пользовательского интерфейса JavaFX.

Цель: разработка приложения для учёта успеваемости студентов с выполнением расширенного функционала.

Методология проведения работы: в процессе решения поставленных задач использованы методики проектирования графического пользовательского интерфейса и алгоритмы, описывающие логику работы приложения университета.

Результаты работы: выполнен анализ литературных источников по теме; изучены основы создания приложений с насыщенным графическим интерфейсом; изучены визуальные инструменты дизайна; рассмотрены особенности реализации системы; осуществлена функциональная часть программы.

Область применения результатов: разработка приложений.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Мы живем в эпоху информационных технологий, когда почти все аспекты жизни были затронуты информационными и коммуникационными технологиями (ИКТ). ИКТ были включены в нашу повседневную жизнь и изменили то, как мы работаем, исследуем, ведем бизнес и общаемся друг с другом. В частности, система результатами студентов является одной из основных информационных систем, которые легко привлекают внимание большинства университетов. Итоговые оценки призваны передать уровень достижений каждого студента в группе. Эти сорта используются для принятия множества решений. Если система выставления оценок не является достаточно надежной и эффективной, то скорее всего возникнет путаница и появятся вопросы у разных лиц университета. Сама по себе обработка результатов оказывается очень обременительным делом, особенно когда она выполняется вручную при большом количестве студентов. Это требует много времени и подвержено ошибкам. Задача одновременно сложная. Однако интересно найти достаточно быстрый и надежный метод обработки оценок учащихся в университетах. В этом проекте была разработана компьютерная программа, позволяющая автоматизировать обработку результатов студентов.

Приложение предназначено для решения проблем, с которыми сталкиваются преподаватели при выставлении оценок учащихся и их управлении и студенты при получении информации об учебном процессе.

Целью является создание приложения для учёта успеваемости студентов с получением быстрого, простого и оперативного доступа к информации учебного процесса.

Главными задачами курсового проекта является:

- рассмотрение особенностей реализации системы;
- решение вопроса доступности данных об успеваемости;
- автоматизация выставления оценок;
- организация взаимодействия людей в структуре университета на основе клиент-серверного приложения;
- разработка функционального графического интерфейса пользователя.

1 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ РЕЙТИНГОВАЯ СИСТЕМА

Современный мир характеризуется разработкой и внедрением информационных технологий во все сферы жизнедеятельности человека. Информационные технологии настолько вжились в реальность, что без них трудно представить современную жизнь. Управление социально-экономическими и технологическими процессами осуществляется за счет использования информационных систем и баз данных. Перед обществом остро стоит вопрос об обеспечении конкурентоспособности национальной системы высшего образования на международном уровне и реформировании системы высшего образования на основе сочетания отечественных традиций с мировым опытом. Это невозможно реализовать без внедрения соответствующих изменений в систему организации работы университета, в частности использования информационной системы управления деятельностью университета, а именно учётом успеваемостью студентов.

В каждом учебном заведение большой поток данных (абитуриенты, студенты, преподаватели, данные об успеваемости и т.п.) и чтобы снизить время обработки информации и облегчить работу сотрудникам заведений создается информационная система, которая может это позволить.

Студенты в течение определенного срока (например, 5 лет) изучают дисциплины в соответствии с учебным планом выбранной специальности. Изучение каждой дисциплины имеет две стадии: приобретение знаний и контроль усвоения знаний. Университет ведет учет изучаемых дисциплин и результатов сдачи экзаменов, зачетов, курсовых работ и прочих видов контроля.

Дисциплины ведут преподаватели вуза. Один преподаватель, как правило, ведет несколько дисциплин, а одну дисциплину могут вести несколько преподавателей. Необходимо отразить в базе данных все дисциплины, которые ведет каждый преподаватель. Качество обучения характеризуется оценками, которые студенты получают во время экзаменационной сессии. Каждый студент изучает много дисциплин и поэтому имеет много оценок. Необходимо вести учет полученных оценок.

Ограничения предметной области:

– срок обучения в институте по всем специальностям одинаковый – 5 лет (10 семестров);

– не учитывается год поступления в вуз конкретного студента. В реальной ситуации год образования новой группы и год поступления

студента в вуз может не совпадать, т.к. студент может уйти в академический отпуск и затем продолжить учиться в группе другого года набора;

– год окончания вуза студентов приравнивается к году выпуска всей группы;

– один студент учится только в одной группе.

Учащиеся: получают быстрый, простой и оперативный доступ к касающейся учебного процесса информации посредством графического интерфейса: информации о собственной успеваемости.

Сотрудники учебного заведения: получают оперативный доступ к учебным планам, графикам, спискам контрольных мероприятий; получают оперативный доступ к информации об обучающихся, их успеваемости. Организовывают автоматизированный сбор статистических данных по контингенту учащихся.

Руководители учебного заведения: уменьшают временные затраты в процессе планирования и управления деятельностью учебного заведения; получают оперативный доступ к информации, сопровождающей учебный процесс для принятия эффективных управленческих решений; повышают эффективность управления образовательным процессом и образовательным заведением в целом.

Для того чтобы выполнить оценку качества освоения студентами учебного материала, пройденного на протяжении семестра, проводится промежуточная аттестация.

Целью деятельности образовательной организации является обеспечение современного качественного образования.

Контроль успеваемости - комплексный процесс, в котором принимают участие учебная часть вуза, кафедры и деканат.

Учебная часть отвечает за формирование образовательных программ, рабочих и семестровых учебных планов, ведение и учет контингента, составление расписания, составление списка учебных дисциплин.

Кафедры проводят учебные занятия и принимают решение о допуске или недопуске студентов, к промежуточной аттестации по дисциплине на основании текущей успеваемости. Сведения о студентах, имеющих задолженность, передаются в деканат соответствующего факультета.

Деканаты осуществляют контроль за проведением промежуточной аттестации, прием и анализ аттестационных ведомостей, поступивших с кафедр, выдачу аттестационных листов на пересдачи, ведение учебных карточек студентов, формирование приказов о движении контингента.

На основании учебных планов для каждого факультета строятся учебные планы. Из учебных планов формируются учебные графики и

расписания, определяется перечень дисциплин для каждого курса и семестра обучения. Каждой дисциплине соответствует форма контроля. Перед зачетно-экзаменационной сессией на основании сведений, поданных кафедрой об успеваемости студентов, издаются приказы о допуске к сессии. По итогам сдачи зачета экзамена в аттестационную ведомость выставляется оценка (зачет) и рейтинг студента по дисциплине. Итоговый документ, поступающий с кафедр в день экзамена/зачета – аттестационная ведомость.

Практическое использование информационных систем в деятельности университета позволяет выделить ряд преимуществ, в частности: увеличение эффективности учебного процесса; управление результатами деятельности; объективность оценки деятельности преподавателей и студентов; доступ к информации о деятельности университета; сокращение объема работы.

Одним из главных функциональных обязанностей университета является учет успеваемости студентов. Студенты и множество дисциплин создают необходимость вести учет за данными, и именно поэтому проблема формирования, а так же постижения объективной системы контроля знаний студентов в образовании является особенно актуальной в наше время. По этой причине придается огромное значение ее решению, в связи с тем, что использование таких систем контроля знаний помогают поддерживать необходимый образовательный уровень студентов.

Входными данными системы являются: оценки, даты сдачи экзаменов, имена студентов, номера групп, специальность, факультет. На выходе выдаются обработанные данные: средний балл по студенту, группе или факультету, процентное соотношение оценок у студента в группе или на факультете, имена. Контроль успеваемости студентов может функционировать отдельно от всей системы, что дает возможность установить и использовать ее независимо, если это необходимо. Контроль успеваемости студентов включает следующие функции:

– ввод, вывод и редактирование информации по информационным объектам;

– сохранение информации, поступившей от системы контроля успеваемости студентов;

– расчет процентного соотношения оценок у студента в группе или на факультете и вывод его в виде таблиц, графиков и диаграмм;

– расчет среднего балла по студенту, группе или факультету;

– формирование данных по студенту, группе или факультету;

– выявление сильнейших и слабейших студентов;

– проверку правильности ввода данных.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

Основная задача заключается в создании приложения для организации работы университета в рамках учёта успеваемости студентов.

Необходимо выполнить следующие задачи:

- рассмотрение особенностей реализации системы;
- решение вопроса доступности данных об успеваемости;
- автоматизация выставления оценок;
- организация взаимодействия людей в структуре университета на основе клиент-серверного приложения;
- разработка функционального графического интерфейса пользователя.

Основным методом решения будет являться создание многоуровневой архитектуры с использованием базовых паттернов для проектирования приложений.

Клиентская часть была выполнена в виде оконного приложения с использованием стандартной библиотеки пользовательского интерфейса JavaFX.

JavaFX – это инструментарий графического интерфейса пользователя следующего поколения. Включает в себя набор графических и мультимедийных API, которые мы можем использовать для создания и развертывания многофункциональных клиентских приложений. JavaFX позволяет разработчикам быстро создавать многофункциональные кроссплатформенные приложения. JavaFX поддерживает современные графические процессоры с аппаратным ускорением графики. JavaFX позволяет разработчикам объединять элементы управления графикой, анимацией и пользовательским интерфейсом в одном программном интерфейсе. Технология JavaFX имеет хорошие перспективы, включая возможность прямого вызова API Java. Поскольку JavaFX Script статически типизирован, он также имеет структурированный код, повторное использование и инкапсуляцию, такую как пакеты, классы, наследование, а также отдельные модули компиляции и распространения. Эти функции позволяют создавать большие программы и управлять ими с помощью технологии JavaFX.

В процессе разработки пользовательского интерфейса использовался Scene Builder. JavaFX Scene Builder представляет собой визуальный инструмент, который позволяет пользователям быстро создавать интерфейс для JavaFX приложений, без кодирования. Пользователи могут перетаскивать

компоненты интерфейса в рабочую область, изменять их свойства, применять таблицы стилей, а код FXML для интерфейса генерируется автоматически в фоновом режиме. Результатом является FXML файл, который может быть объединён с проектом Java путем связывания пользовательского интерфейса к логике.

Серверное приложение реализовано в виде консольного приложения. Доступ к данным СУБД осуществляется через драйвер, предоставляемый производителем СУБД.

Сокеты обеспечивают механизм связи между клиентом и сервером, использующими TCP. Клиентская программа создает сокет на своем конце связи и пытается подключить этот сокет к серверу. Когда соединение установлено, сервер создает объект сокета на своем конце связи. Клиент и сервер теперь могут общаться, записывая и считывая данные с сокета.

Класс java.net.Socket представляет собой сокет, а класс java.net.ServerSocket предоставляет механизм серверной программы для прослушивания клиентов и установления соединений с ними. При установлении соединения TCP между двумя компьютерами с использованием сокетов, выполняются следующие этапы:

– сервер создает экземпляр объекта ServerSocket, определяющий, по какому номеру порта должна происходить связь;

– сервер вызывает метод accept() класса ServerSocket. Этот метод ожидает, пока клиент не подключится к серверу по указанному порту;

– по завершению ожидания сервера клиент создает экземпляр объекта сокета, указывая имя сервера и номер порта подключения;

– конструктор класса Socket осуществляет попытку подключить клиента к указанному серверу и номеру порта. Если связь установлена, у клиента теперь есть объект Socket, способный связываться с сервером;

– на стороне сервера метод accept() возвращает ссылку к новому сокету на сервере, который подключен к клиентскому сокету.

После того, как соединения установлены, связь происходит с использованием потоков входных/выходных данных. Каждый сокет имеет и OutputStream (поток выходных данных), и InputStream (поток входных данных). OutputStream клиента подключен к InputStream сервера, а InputStream клиента подключен к OutputStream сервера. TCP является двусторонним протоколом связи, поэтому данные могут передаваться по обоим потокам одновременно.

В ходе создания проекта применялся пакет MySQL в качестве сервера баз данных. MySQL – это реляционная СУБД. MySQL поддерживает SQL (структурированный язык запросов) и может применяться в качестве SQL-

сервера. Это означает, что общаться с сервером можно на языке SQL: клиент посылает серверу запрос, тот его обрабатывает и отдает клиенту только те данные, которые были получены в результате этого запроса. Тем самым клиенту не требуется выкачивать данные и производить вычисления, как, например, в Microsoft Access.

По функциональной оснащенности и надежности MySQL давно конкурирует с другими известными продуктами. Чаще всего ее используют при разработке веб-решений, что объясняется тесной интеграцией с популярными языками программирования, высокими показателями скорости и доступностью

На MySQL обращают внимание как небольшие компании, так и крупные корпорации. Данная СУБД привлекает своей надежностью и свободным распространением. Если дело касается веб-разработки, то MySQL практически всегда будет лучшим вариантом (с учетом потребностей, специфики проекта и экономической целесообразности).

При построении архитектуры проекта использовался шаблон проектирования MVC. Шаблон MVC выступает в качестве решения, позволяющее отделить 3 проблемы друг от друга: визуальные элементы данные и логику.

MVC расшифровывается как Model-View-Controller. Это принцип построения архитектуры большого приложения, при котором оно разбивается на три части. Первая часть содержит всю бизнес-логику приложения. Такая часть называется Модель (Model). В ней содержится код, который делает все то, для чего приложение создавалось. Эта часть наиболее независимая от остальных. Вторая часть содержит все, что касается отображения данных пользователю. Такая часть называется Вид (View). Именно в ней содержится код, который управляет показом окон, страниц, сообщений и т.д. Третья часть содержит код, который занимается обработкой действий пользователя. Любые действия пользователя, направленные на изменения модели, должны обрабатываться тут. Такая часть называется Controller.

Такой подход позволяет независимо делать три вещи: логику программы (Model), механизм показа всех данных программы пользователю (View), обрабатывать ввод/действия пользователя (Controller).

В контроллере сосредоточен весь код, который принимает решение, что менять в модели в ответ на действия пользователя. Например, пользователь принял решение закрыть программу, тогда надо сохранить данные модели в файл на диск. Или пользователь ввел новые данные. Тогда

надо добавить их в модель, а модель потом уведомит все View об изменении данных, чтобы они отображали только актуальное их состояние.

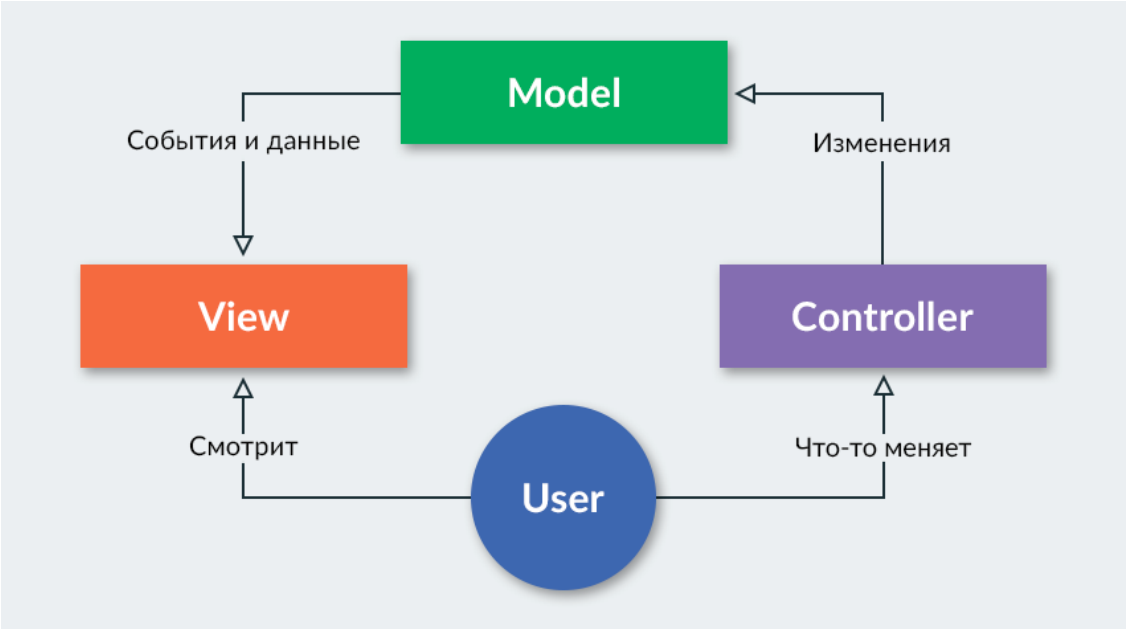

Рисунок 1 – архитектурный шаблон проектирования MVC

Паттерн Builder позволяет создавать различные варианты объекта, избегая загрязнения конструктора. Полезно, когда может быть несколько вариантов объекта. Или когда есть много шагов, вовлеченных в создание объекта.

Паттерн относится к порождающим паттернам. Он предназначен для порождения объектов. Суть его заключается в том, чтобы отделить процесс создания некоторого сложного объекта от его представления. Таким образом, можно получать различные представления объекта, используя один и тот же "технологический" процесс.

Результатом работы паттерна Builder должен быть объект класса. В нашем случае результатом работы паттерна является объект класса StudentO.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

IDEF0 (Integration Definition for Function Modeling) — нотация описания бизнес-процессов. IDEF0 – это метод и язык, являющийся вариантом метода структурного анализа и проектирования (Structural Analysis and Design Technique, SADT). IDEF0 отражает логические отношения между работами, но не позволяет рассматривать временную последовательность. Основными потребителями нотации IDEF0 являются руководители, которым необходимо видеть и понимать взаимосвязь процессов, не вникая в мелочи. Контекстная диаграмма представлена на рисунке 3.1.

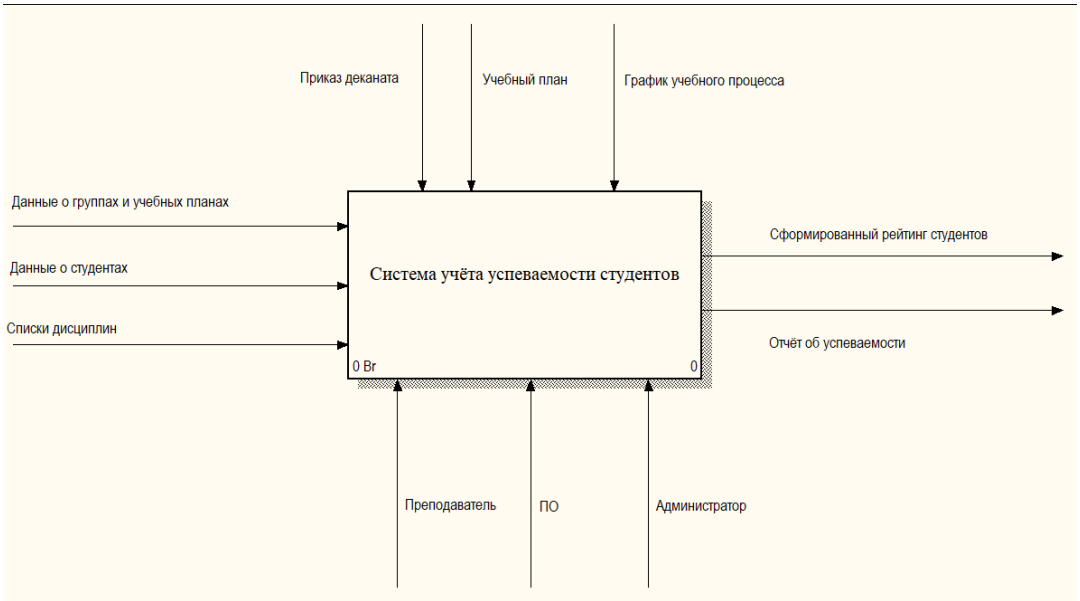

Рисунок 3.1 – Контекстная диаграмма

Контекстная диаграмма — вид IDEF0-диаграммы. Это диаграмма, расположенная на вершине древовидной структуры диаграмм, представляющая собой самое общее описание системы и ее взаимодействие с внешней средой.

Пользователи в лице администратора и преподавателя выступают в качестве механизма процесса с помощью использования программного обеспечения. Входы – данные, которые хранит университет. Выходы – преобразованная информация об успеваемости. Управления являются регламентирующими воздействиями выполнения процесса.

Декомпозиция контекстной диаграммы (рис. 3.2) состоит из трёх блоков, каждый из которых также декомпозирован.

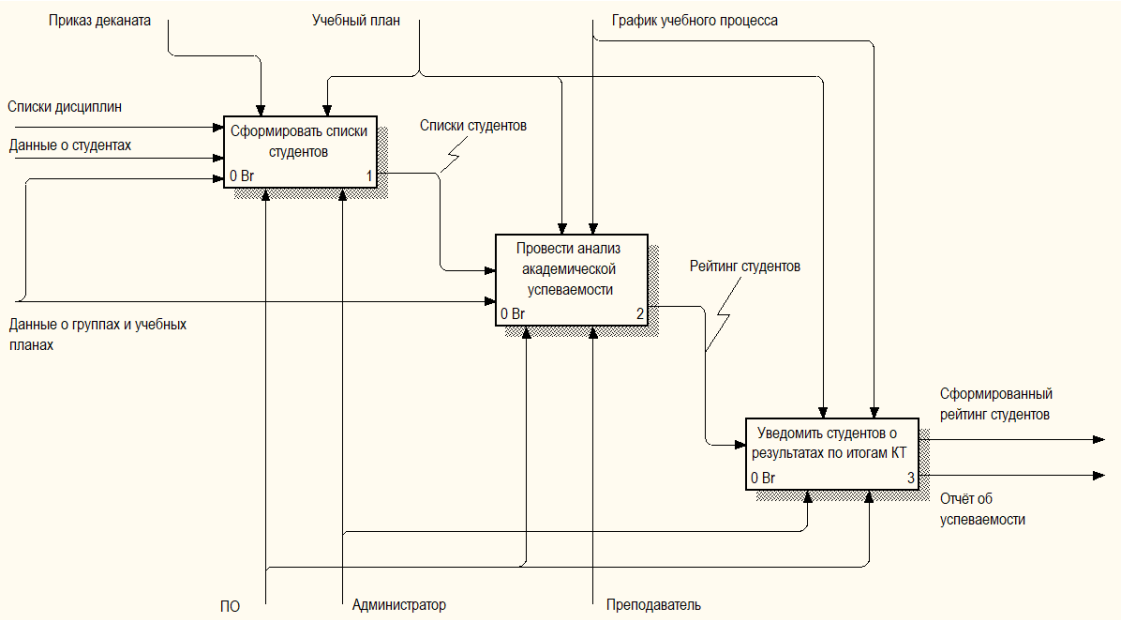

Рисунок 3.2 – Диаграмма декомпозиции контекстной диаграммы

2 уровень декомпозиции работы «Сформировать списки студентов» (рис. 3.3). Состоит из трёх блоков. Результатом работы является получение списков студентов.

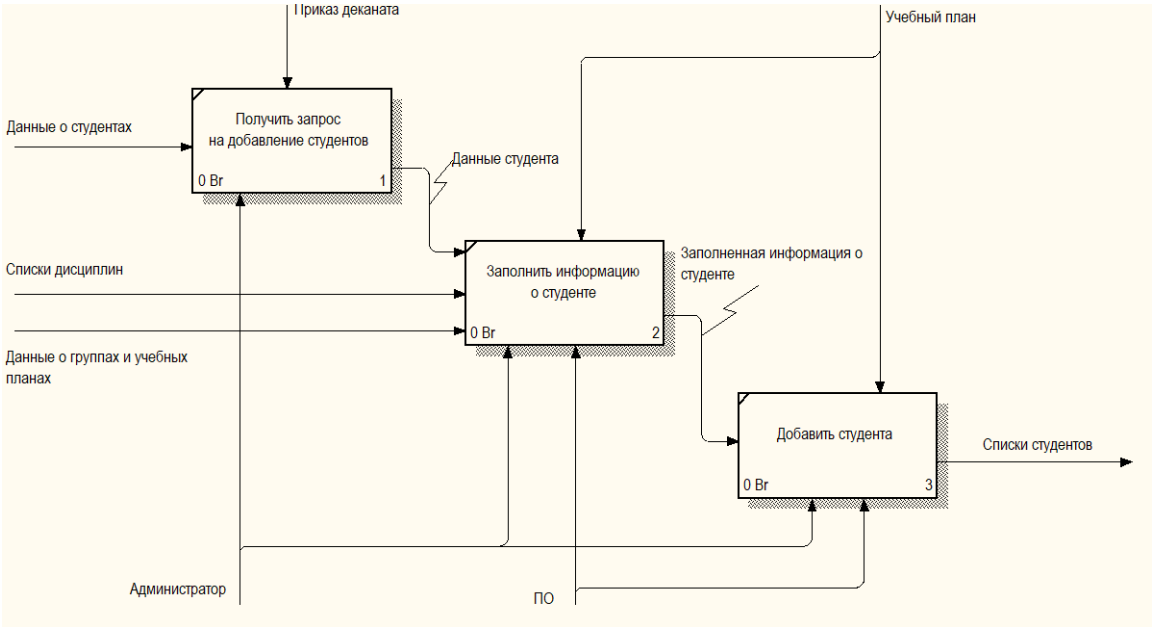

Рисунок 3.3 – Декомпозиция «Сформировать списки студентов»

3 уровень декомпозиции работы «Провести анализ академической деятельности». Состоит из трёх блоков, описывает процесс выставления оценки студенту.

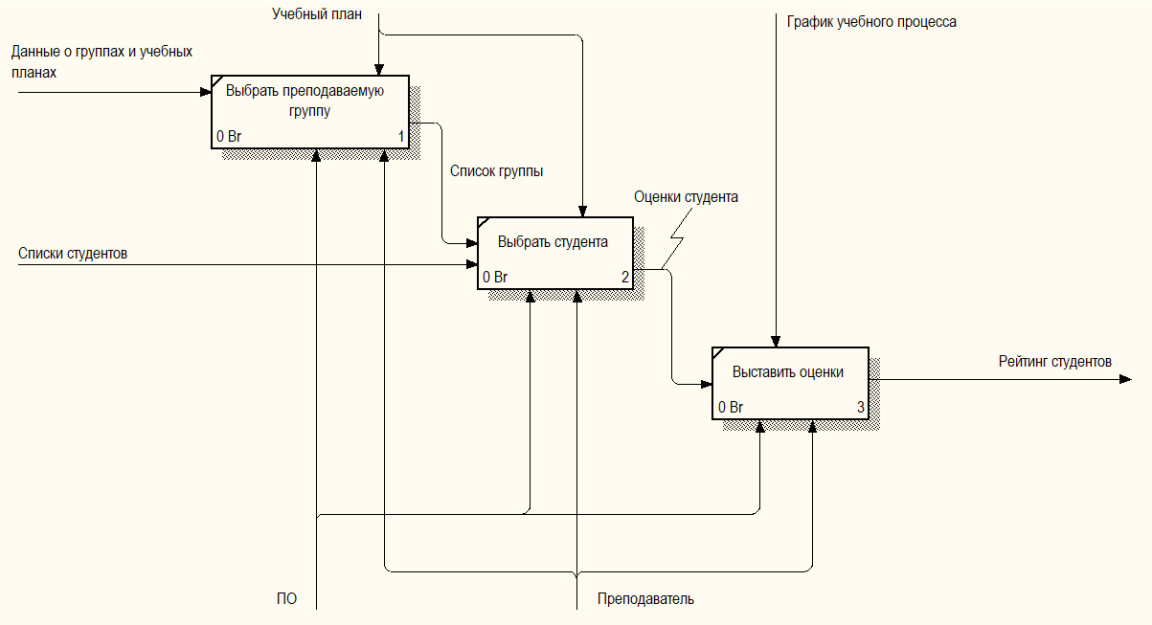

Рисунок 3.4 – Декомпозиция «Провести анализ академической успеваемости»

4 уровень декомпозиции работы «Уведомить студентов о результатах по итогам КТ». Состоит из трёх блоков, описывает анализ успеваемости студентов.

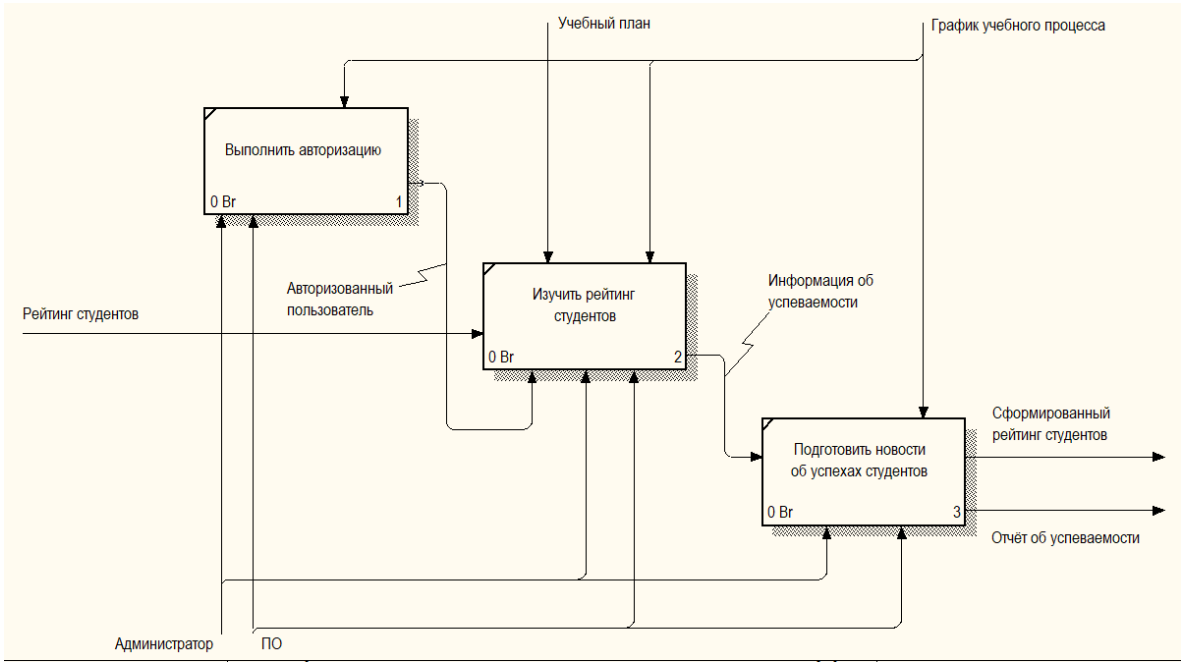

Рисунок 3.5 – Декомпозиция «Уведомить студентов о результатах по итогам КТ»

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЁ ОПИСАНИЕ

В данном курсовом проекте рассматриваются задачи, выполняемые учебной частью ВУЗа. Информационная система (рис. 4.1) обеспечивает: хранение информации о студентах, учет процессов, связанных с обучением студентов, данные о преподавателях, учебных дисциплинах и т.д. Каждый поступивший или уже учащийся в данном учебном заведении и так же преподаватели заносятся в информационную систему. Это способствует удобству обработки данных, уменьшению времени поиска определенных данных. Промежуточная аттестация студентов осуществляется в соответствии с учебными планами по направлениям и специальностям подготовки в форме экзаменов и зачетов по учебным дисциплинам и практикам.

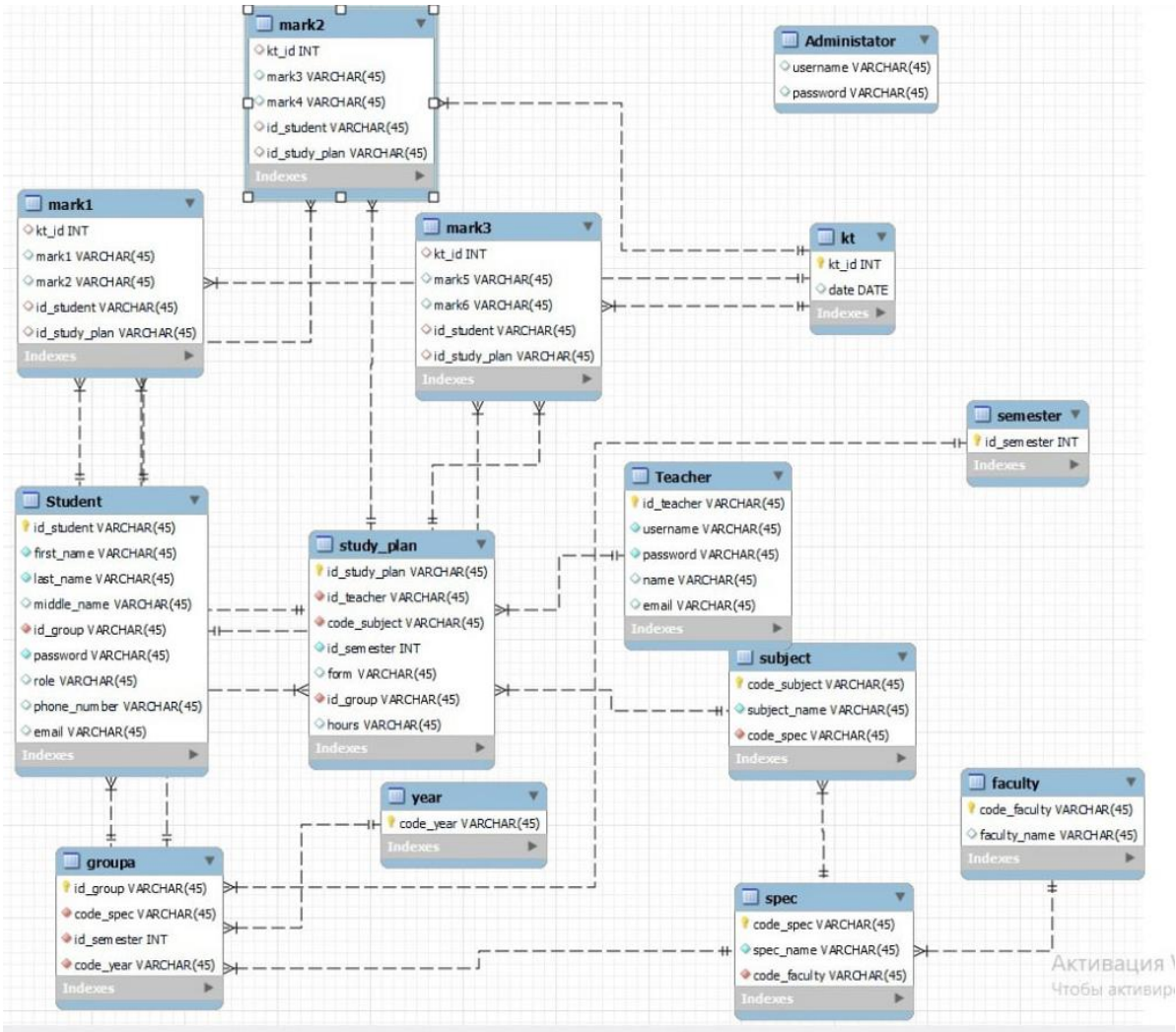

Рисунок 4.1 – Информационная модель системы

Информационная модель представляет собой совокупность описаний информационных процессов, происходящих в вузе. Создание ИМ заключается в выделении сущностей и определении их атрибутов.

Для максимально подробного описания структуры университета понадобилось создать 14 таблиц.

Сущности «Student», «Teacher» , «Administrator» хранят основную информацию о студенте, учителе и администраторе соответственно.

Сущности «Year», «Spec», «Faculty», «Semester», «Groupa» описывают структурную особенность университета.

Сущности «Subject», «Study_plan», «Mark1», «Mark2», «Mark3», «Kt» составляют логику учебного плана.

Отношения один ко многим: «Faculty» – «Spec», «Spec» – «Subject», «Spec» – «Groupa», «Groupa» – «Student», «Year» – «Goupa», «Semester» – «Goupa», «Year » – «Goupa», «Student» – «Mark1», «Student» – «Mark2», «Student» – «Mark3», «Study_plan» – «Mark1», «Study_plan» – «Mark2», «Study_plan» – «Mark3», «Kt» – «Mark1», «Kt» – «Mark2», «Kt» – «Mark3», «Kt» – «Mark1», «Teacher» – «Study_plan», «Subject» – «Study_plan», «Groupa» – «Study_plan».

Важной особенностью системы является наличие таблицы с информацией о контрольных точках, в соответствии с которыми анализируется успеваемость студентов на определённом отрезке времени.

Наличие большого числа связей позволять создавать запросы для получения всех необходимых данных. Преподаватели могут просматривать преподаваемые ими предметы и группы с оценками студентов. Учащиеся в свою очередь получают информацию о своих предметах в текущем семестре, составу группы и оценкам. Всем пользователям доступны сведения о рейтинге студентов, исходя из выбора года обучения, факультета и специальности. Администратор имеет возможность изменять все необходимые ему данные.

Нормализация – это процесс организации данных в базе данных. Это включает создание таблиц и установление связей между этими таблицами в соответствии с правилами, предназначенными как для защиты данных, так и для того, чтобы сделать базу данных более гибкой за счет устранения избыточности и непоследовательной зависимости.

Существует несколько правил нормализации базы данных. Каждое правило называется "нормальной формой". Если первое правило соблюдается, база данных, как сообщается, находится в "первой нормальной форме". Если соблюдаются первые три правила, база данных

рассматривается как "третья нормальная форма". Третья нормальная форма считается наивысшим уровнем, необходимым для большинства приложений.

Третья нормальная форма (3NF) предполагает, что каждый столбец, не являющийся ключом, должен зависеть только от первичного ключа.

В каждой из таблиц столбцы, не являющиеся ключевыми, зависят только от первичного ключа. У таблиц нет функциональных зависимостей между неключевыми столбцами. Например, в таблице «Faculty» для каждого значения столбца code_faculty точно определено значение столбца faculty_name.

Информационная модель было разработана в унифицированном визуальном инструменте для архитекторов, разработчиков и администраторов баз данных MySQL Workbench.

MySQL Workbench позволяет администраторам баз данных, разработчикам или архитекторам данных визуально проектировать, моделировать, создавать и управлять базами данных. Он включает в себя все, что необходимо специалисту по моделированию данных для создания сложных моделей ER, прямого и обратного проектирования, а также предоставляет ключевые функции для выполнения сложных задач управления изменениями и документирования, которые обычно требуют много времени и усилий.

MySQL Workbench предоставляет визуальную консоль для простого администрирования сред MySQL и улучшения обзора баз данных. Разработчики и администраторы баз данных могут использовать визуальные инструменты для настройки серверов, администрирования пользователей, выполнения резервного копирования и восстановления, проверки данных аудита и просмотра состояния базы данных.

MySQL Workbench предоставляет полное, простое в использовании решение для миграции Microsoft SQL Server, Microsoft Access, Sybase ASE, PostreSQL и других таблиц, объектов и данных СУБД в MySQL. Разработчики и администраторы баз данных могут быстро и легко преобразовать существующие приложения для работы в MySQL как на Windows, так и на других платформах. Миграция также поддерживает переход с более ранних версий MySQL на последние версии.

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ

Языки визуального моделирования – формализованные наборы графических символов и правила построения из них визуальных моделей.

UML — унифицированный язык моделирования. Это графический язык моделирования, предназначенный для спецификации, визуализации, проектирования и документирования всех артефактов, создаваемых при разработке программных средств.

Спецификация UML не должна зависеть от языков программирования, описание языка должно включать в себя семантический базис

Структурные диаграммы отражают элементы, из которых состоит система. Поведенческие модели описывают процессы, протекающие в системе.

Диаграмма вариантов использования.

На диаграммах вариантов использования (рис. 5.1) отображается взаимодействие между вариантами использования, представляющими функции системы, и действующими лицами, представляющими людей или системы, получающие или передающие информацию в данную систему. Из диаграмм вариантов использования можно получить довольно много информации о системе. Этот тип диаграмм описывает общую функциональность системы. Пользователи, менеджеры проектов, аналитики, разработчики, специалисты по контролю качества и все, кого интересует система в целом, могут, изучая диаграммы вариантов использования, понять, что система должна делать.

Вариант использования применяется для спецификации общих особенностей поведения системы или другой сущности без рассмотрения ее внутренней структуры (в проекте выполняется получение необходимой информации об учебном процессе, выполнение авторизации, изменение основных данных).

Актер – это внешняя по отношению к моделируемой системе сущность, которая взаимодействует с системой и использует ее функциональные возможности для решения определенных задач. При этом актеры служат для обозначения согласованного множества ролей, которые могут играть пользователи в процессе взаимодействия с проектируемой системой. На диаграмме представлено 3 типа пользователей: администратор, студент и учитель.

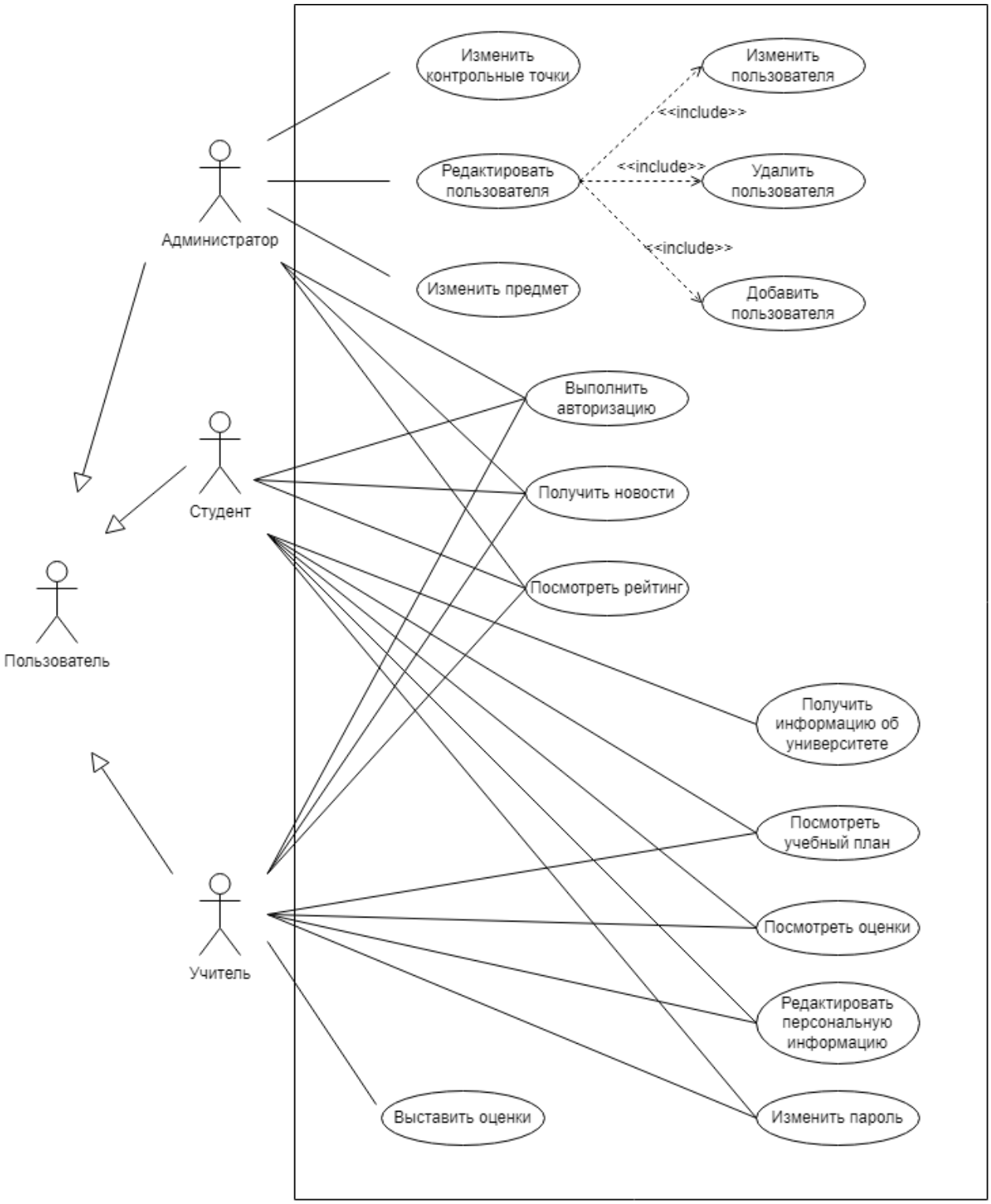

Рисунок 5.1 – Диаграмма вариантов использования

Диаграмма классов.

Диаграмма классов являет собой набор статических, декларативных элементов модели. Она дает наиболее полное и развернутое представление о связях в программном коде, функциональности и информации об отдельных классах. Приложения генерируются зачастую именно с диаграммы классов.

Диаграммы классов являются центральным звеном методологии объектно-ориентированных анализа и проектирования. Диаграмма классов показывает классы и их отношения, тем самым представляя логический аспект проекта. Отдельная диаграмма классов представляет определенный ракурс структуры классов. На стадии анализа диаграммы классов используются, чтобы выделить общие роли и обязанности сущностей, обеспечивающих требуемое поведение системы. На стадии проектирования диаграммы классов используются, чтобы передать структуру классов, формирующих архитектуру системы. На диаграммах классов изображаются атрибуты классов, операции и ограничения, которые накладываются на связи между объектами.

На первой диаграмме (рис. 5.2) представлена диаграмма классов пакета Controller. В пакете находятся классы, которые описывают представление для пользователей.

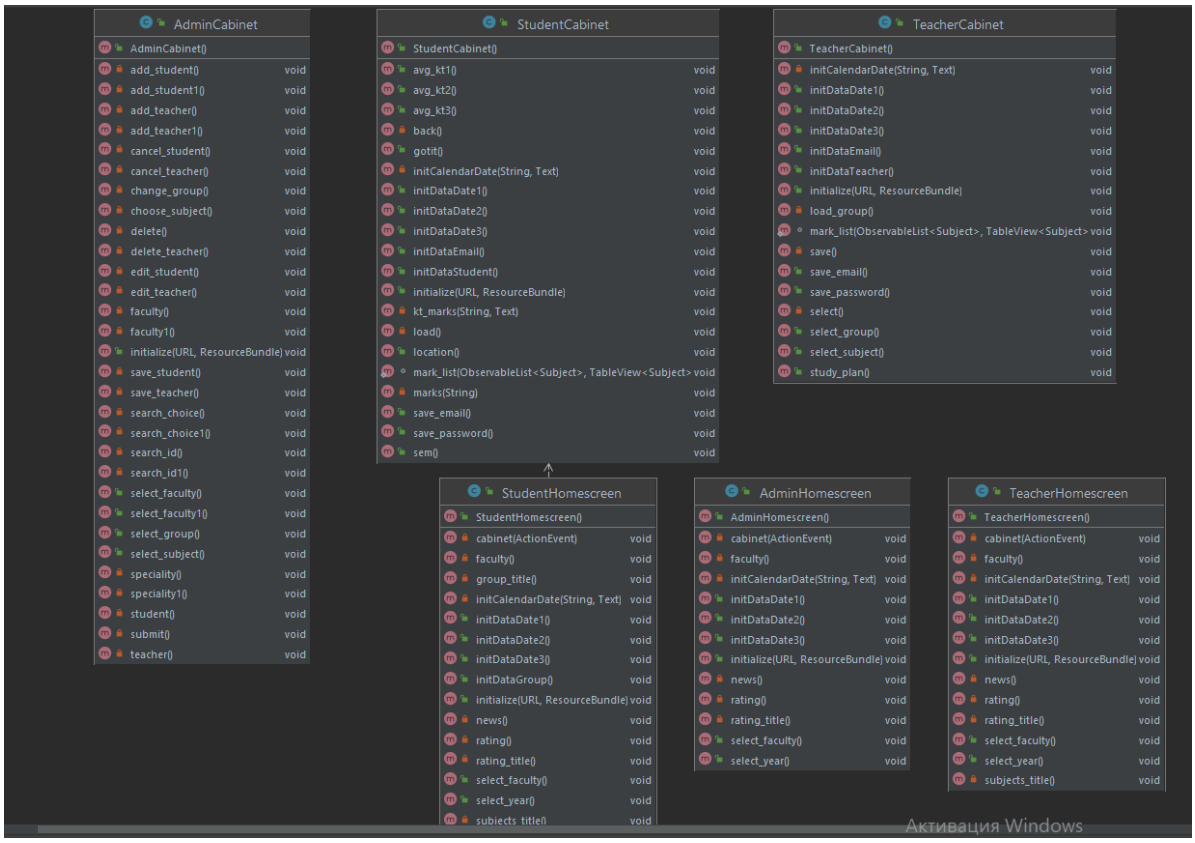

Рисунок 5.2 – Диаграмма классов пакета Controller

На второй диаграмме (рис. 5.3) представлена диаграмма классов пакета Server. Описывается установление соединения, функции, выполняемые на сервере, теги команд, обработка запросов со стороны клиента, связи между классами.

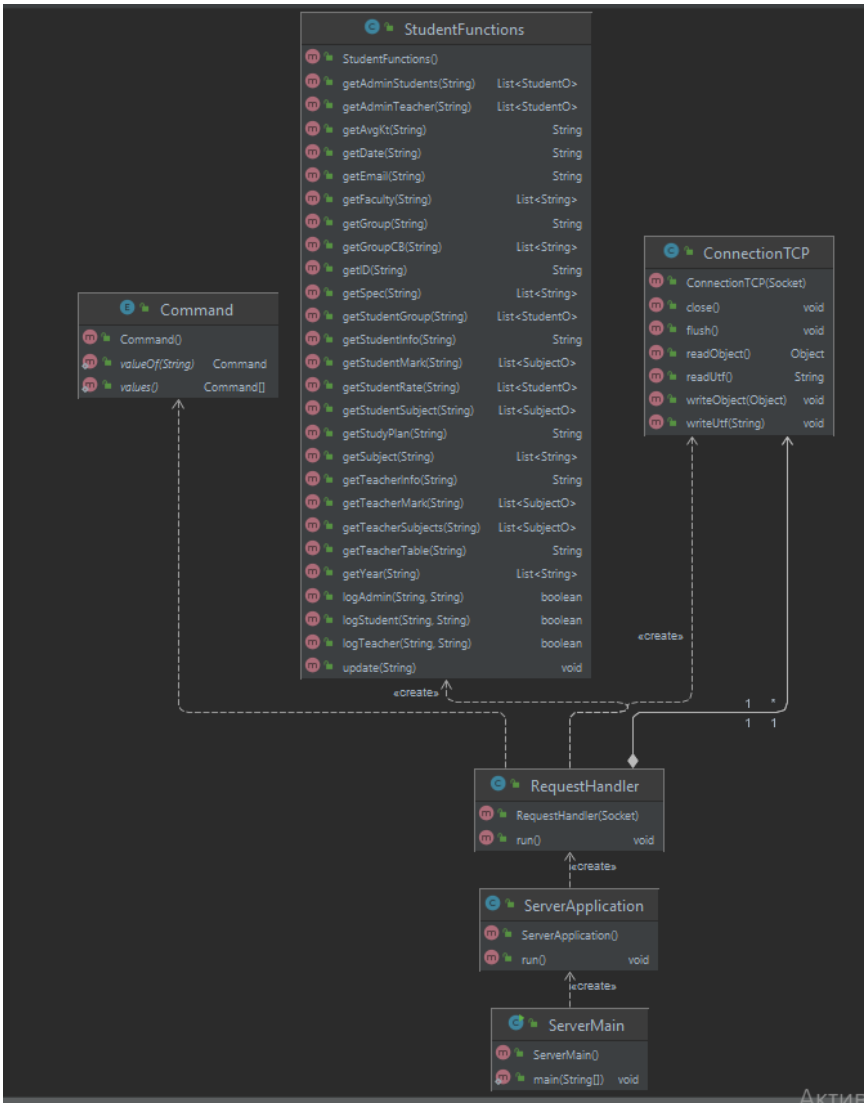

Рисунок 5.3 – Диаграмма классов пакета Server

Диаграмма состояний.

Диаграмма состояний (state machine diagrams) – это известная технология описания поведения системы. В том или ином виде диаграмма состояний существует с 1960 года, и на заре объектно-ориентированного программирования они применялись для представления поведения системы.

В языке UML состоянием (state) называют период в жизни объекта, на протяжении которого он удовлетворяет какому-то условию, выполняет определенную деятельность или ожидает некоторого события. Состояние изображается как закругленный прямоугольник, обычно включающий его имя и подсостояния.

На диаграмме состояния (рис. 5.4) описывается поведение и последовательность добавления оценок преподавателем в его личном кабинете.

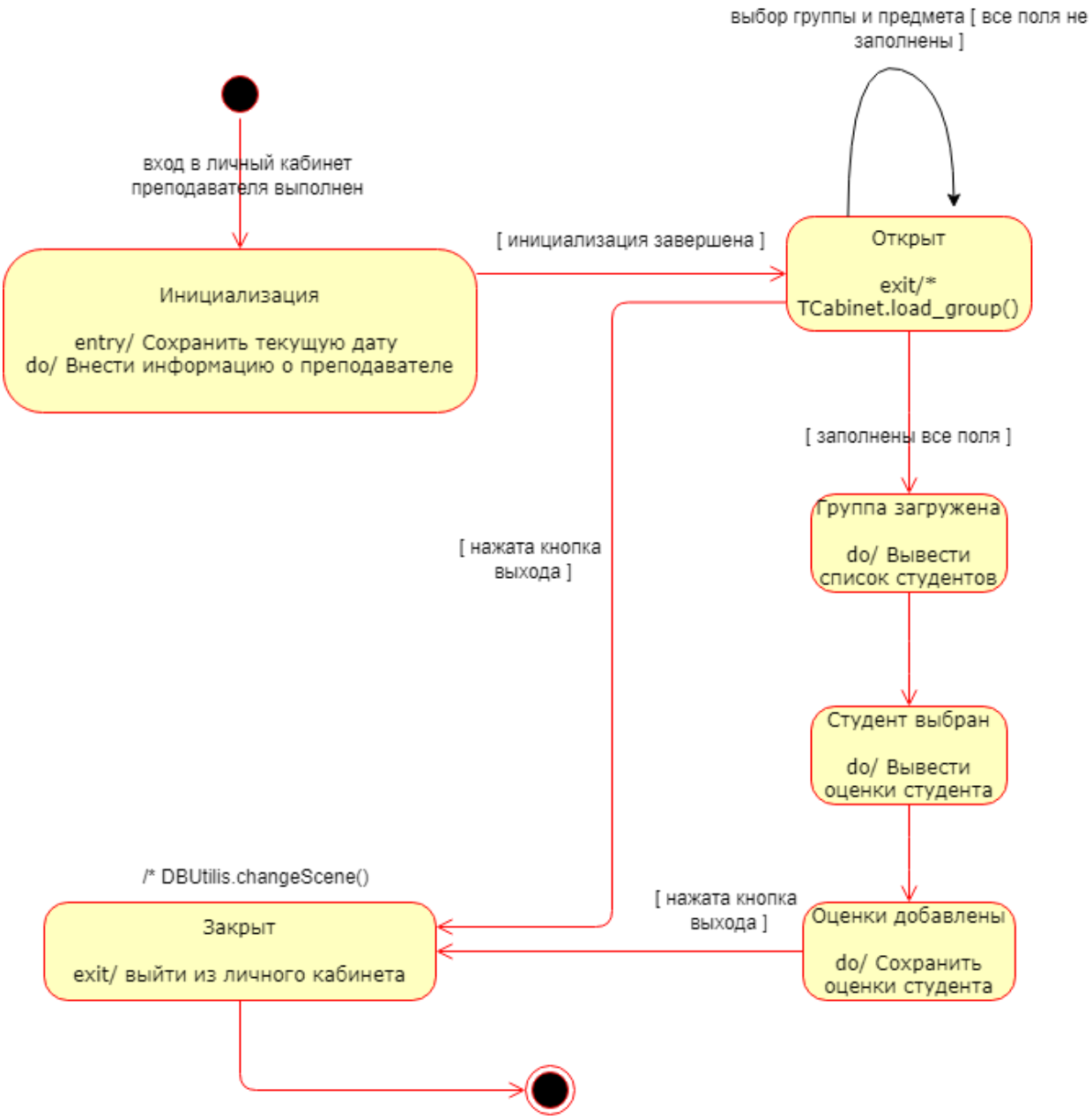

Рисунок 5.4 – Диаграмма состояний выставления оценок

Диаграмма последовательности.

Диаграмма последовательности действий (рис. 5.5) отображает взаимодействие объектов, упорядоченное по времени. На ней показаны объекты и классы, используемые в сценарии, и последовательность сообщений, которыми обмениваются объекты, для выполнения сценария. Диаграммы последовательности действий обычно соответствуют реализациям прецедентов в логическом представлении системы.

При создании диаграмм моделируется логика в основе сложной процедуры, функции или операции; анализируется, как объекты и

компоненты взаимодействуют между собой в ходе выполнения процесса; выявляются возможности оптимизации.

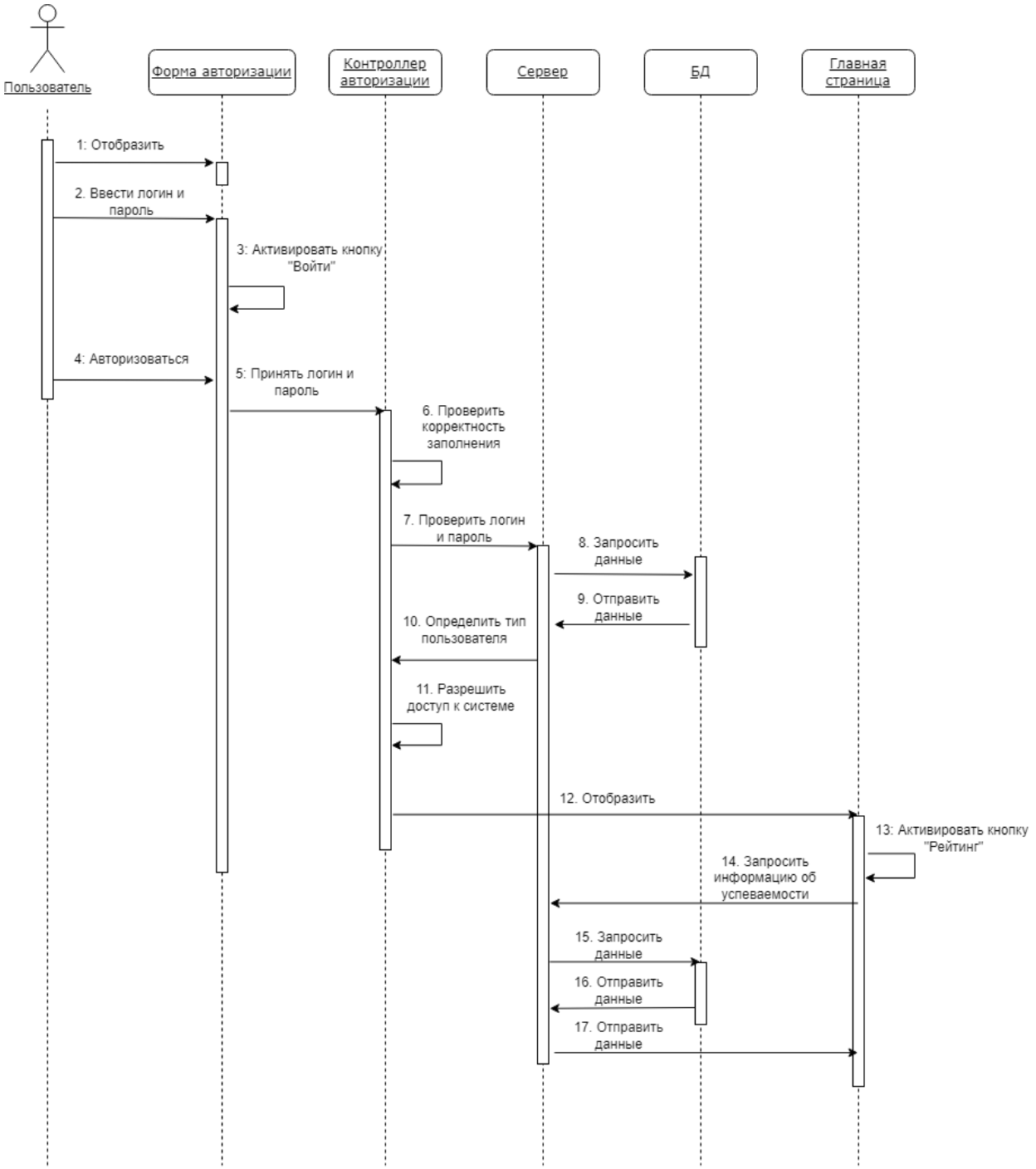

Рисунок 5.5 – Диаграмма последовательности просмотра рейтинга

Пользователь взаимодействует с графическим интерфейсом, клиентское приложение взаимодействует с сервером, происходит обмен данными. Сервер также взаимодействует с базой данных для получения

информации из информационной системы. Обработанные данные передаются обратно клиенту.

Диаграмма компонентов.

На диаграмме компонентов (рис. 5.6) изображаются зависимости между компонентами программы: компоненты исходного кода, бинарного кода, а также выполнимые компоненты. Программный модуль также можно представить в виде компонента. Некоторые компоненты существуют во время компиляции, другие - во время сборки, а прочие - во время выполнения или даже в нескольких различных периодах времени. Некоторые компоненты имеют значение только во время компиляции (compile-only components). В таком случае, компонент, существующий во время выполнения, будет являться выполнимой программой. У диаграммы компонентов есть только описательная форма, формы с использованием экземпляров у нее нет. Чтобы изобразить экземпляры компонентов, нужно строить диаграмму развертывания.

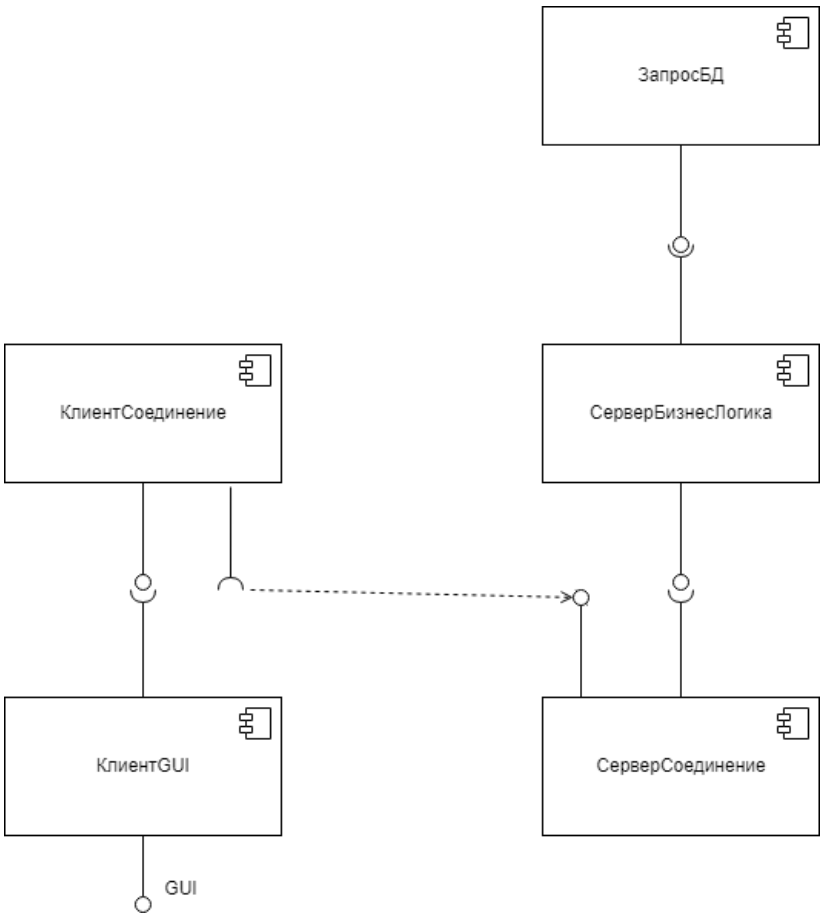

Рисунок 5.6 – Диаграмма компонентов

Диаграмма развертывания.

После диаграммы компонентов разработчики переходят к созданию диаграммы развертывания (рис. 5.7). С помощью диаграммы развертывания моделируются узлы (аппаратные средства) и артефакты (программные средства), которые будут развернуты на них. Диаграмма развертывания не является подробным описанием аппаратной части ИС. Данный тип диаграмм предназначен для моделирования оборудования, связанного с ИС, на среднем и (или) высоком уровне абстракции.

Узел (Node) — один из основных элементов диаграммы развертывания. Узел является аппаратным элементом ИС и представляет собой либо техническое устройство, либо вычислительный ресурс. Узлом может быть автоматизированное рабочее место, сеть, процессор, видеокамера наблюдения, сканер штрих-кодов и т.п.

Второй элемент— это артефакт {Artifact). Артефакты в диаграмме развертывания близки но смыслу к компонентам. Однако в ряде случаев для большей наглядности требуется показать их именно в виде компонентов.

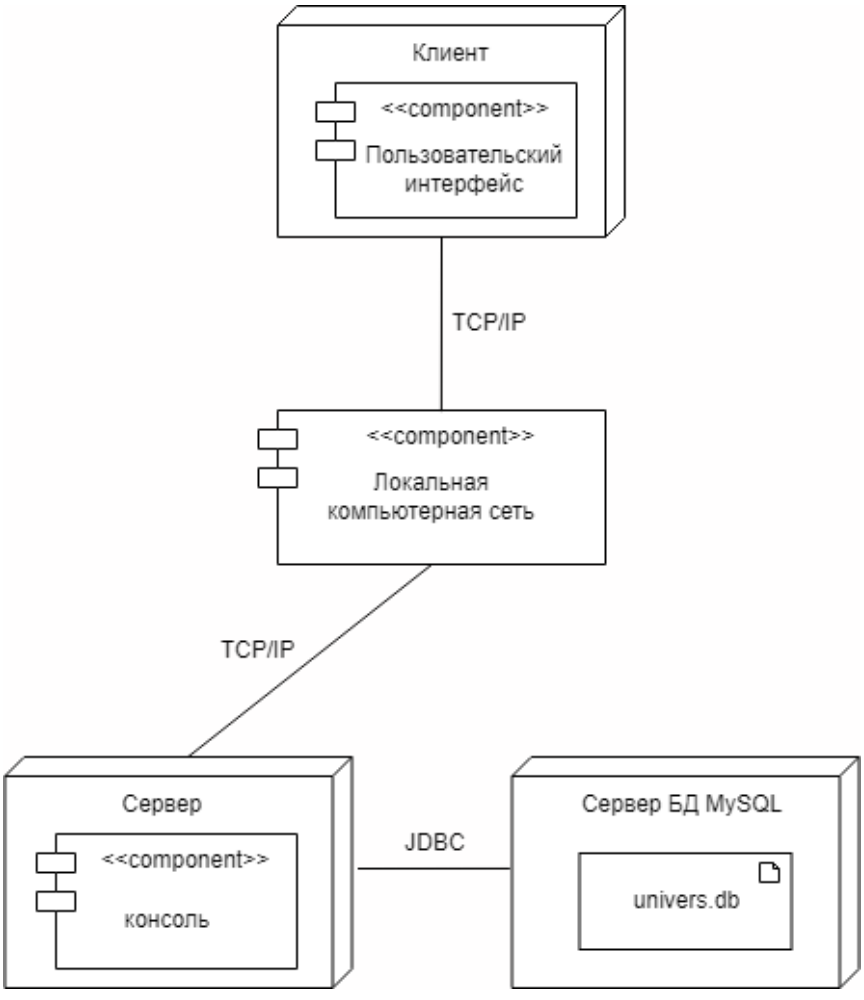

Рисунок 5.7 – Диаграмма развертывания

6. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске приложения пользователь попадает на страницу авторизации (рис. 6.1). Возможность регистрации отсутствует, так как в системе университета добавлением пользователя занимается исключительно администратор. В окне выбирается тип пользователя и вводятся логин и пароль.

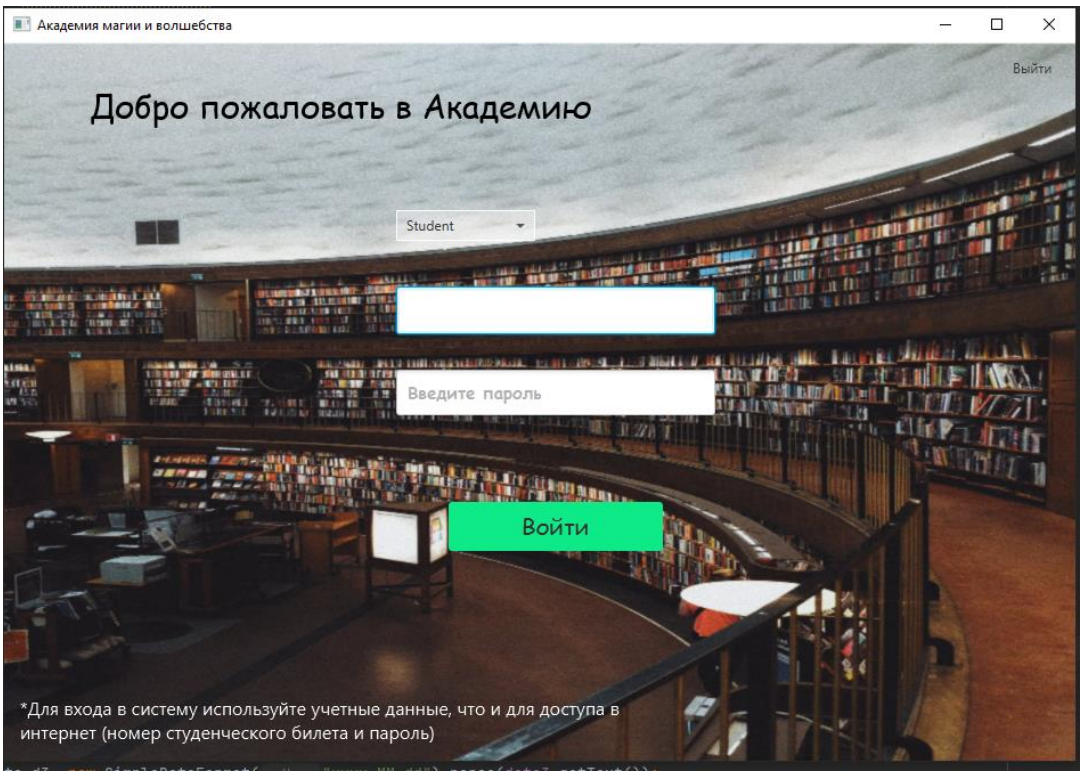

Рисунок 6.1 – Окно авторизации

После успешной авторизации приложение перенаправляет пользователя на главную страницу. Представление главной страницы зависит от типа пользователя (рис. 6.2). Тип пользователя влияет на предоставление необходимой информации и возможности перехода в свой личный кабинет.

Главная страница будет рассматриваться с точки зрения студента. Вкладки “Новости” (рис. 6.3) и “Рейтинг” (рис. 6.4) доступны всем пользователям. Для просмотра рейтинга студентов необходимо ввести год поступления, факультет и специальность студентов. Студент может получить информацию о своей группе (рис. 6.5) и предметам в текущем семестре (рис. 6.6). На своей главной странице учитель может получить информацию о преподаваемых им предметах. Администратору доступны только заголовки “Новости” и “Рейтинг”.

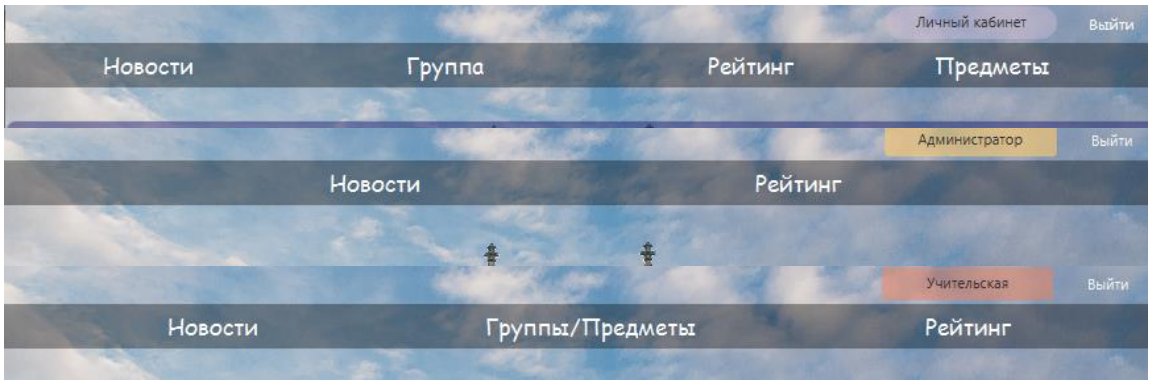

Рисунок 6.2 – Заголовки главной страницы пользователей

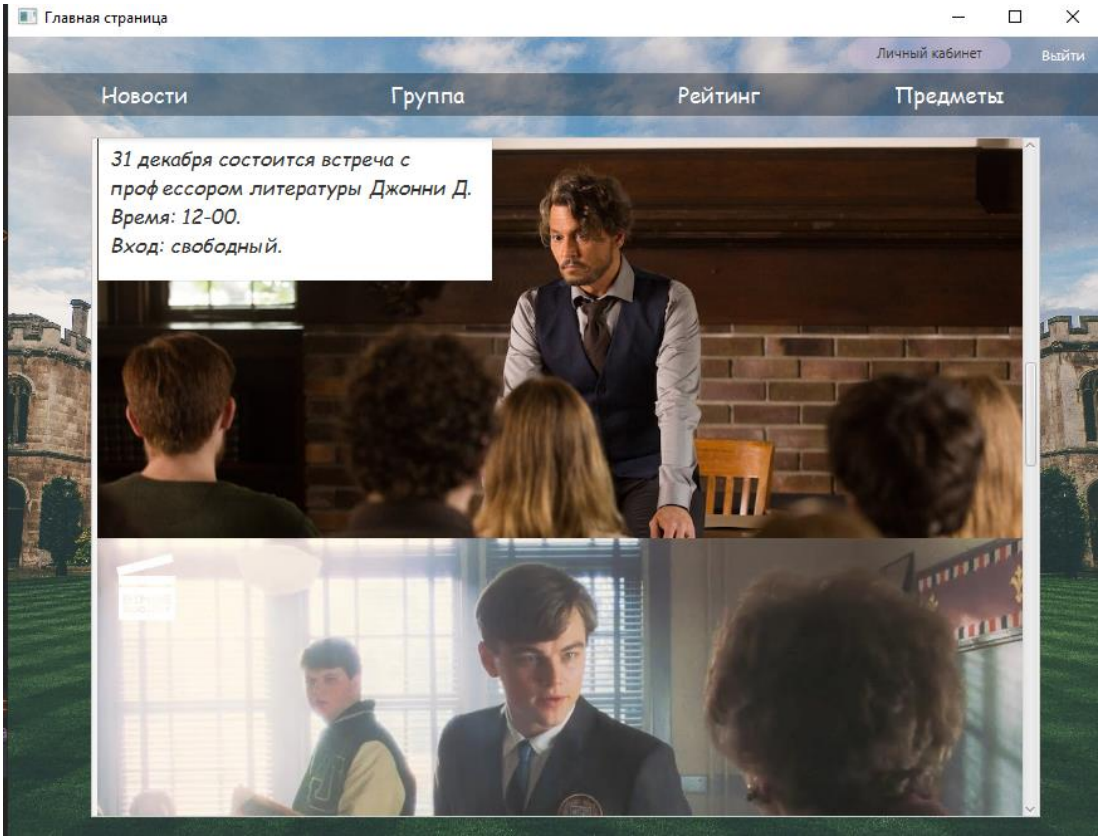

Рисунок 6.3 – Вкладка “Новости”

Визуализация таблиц выполнена с добавлением CSS-файла для стилизации и адаптации представления данных в приложении. Были применены данные о влиянии цвета на психофизиологию человека. Данные отображаются жёлтым цветом, но несильно ярким. Немного тусклый фон таблицы снижает утомление глаз для пользователя.

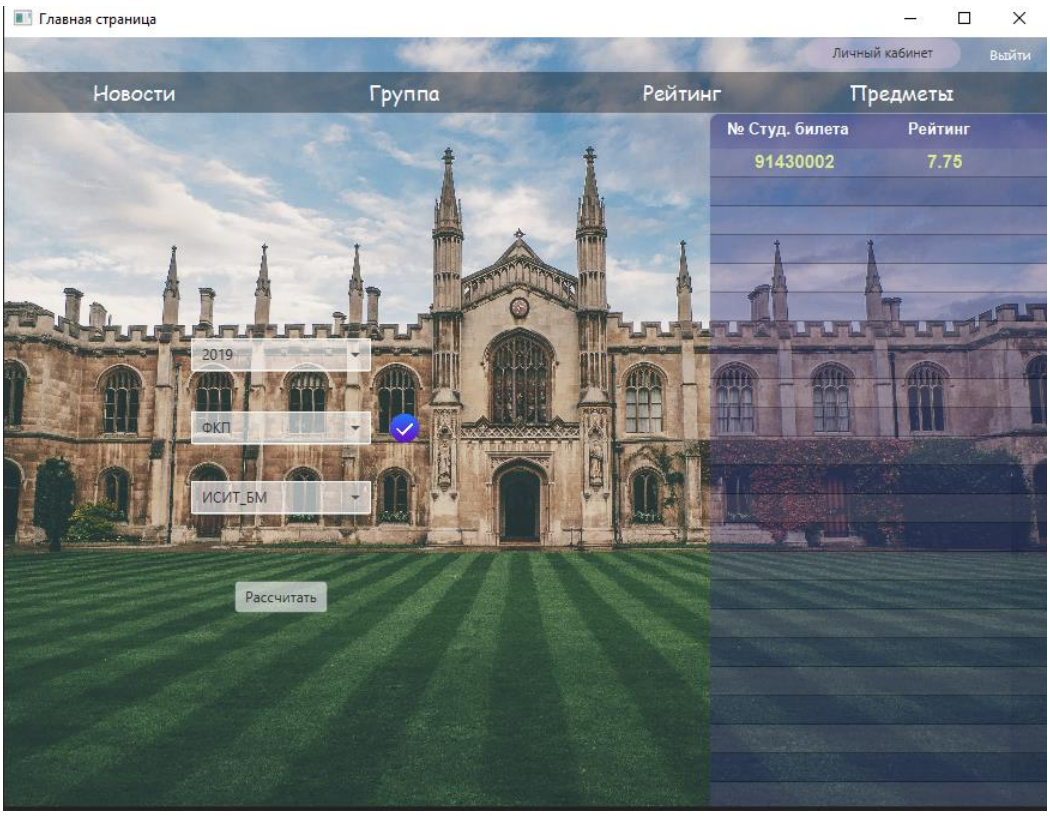

Рисунок 6.4 – Вкладка “Рейтинг”

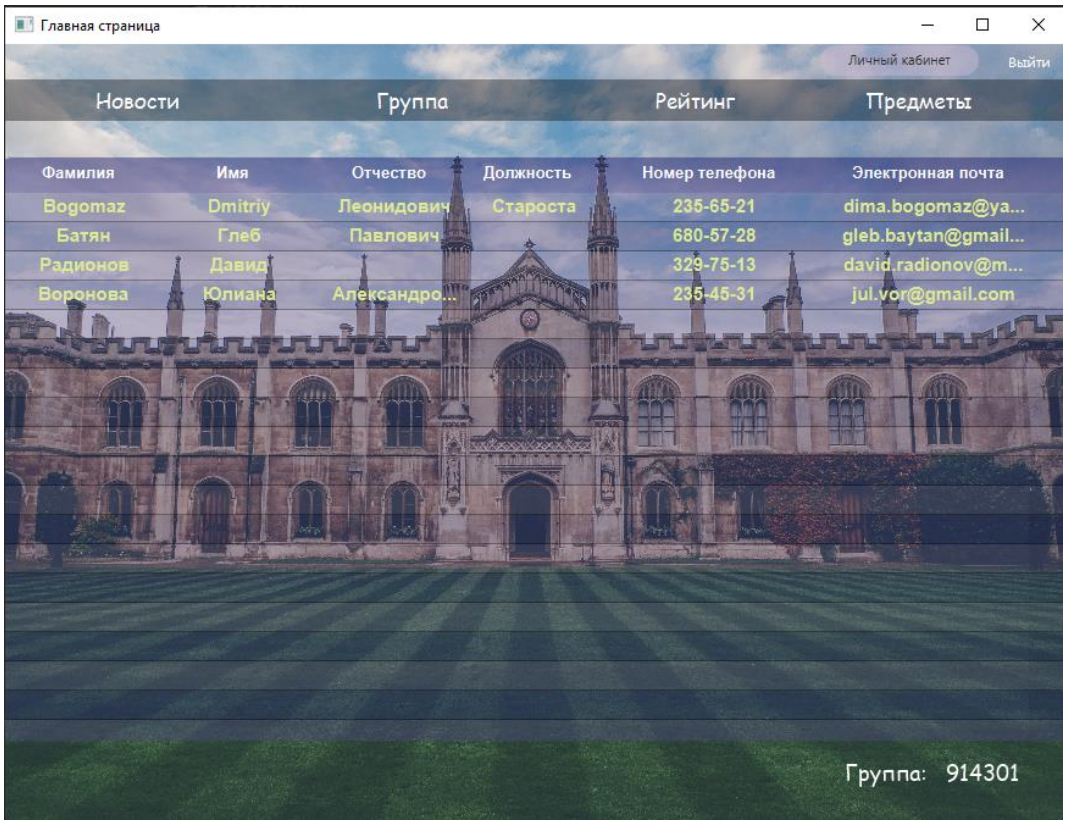

Рисунок 6.5 – Вкладка “Группа”

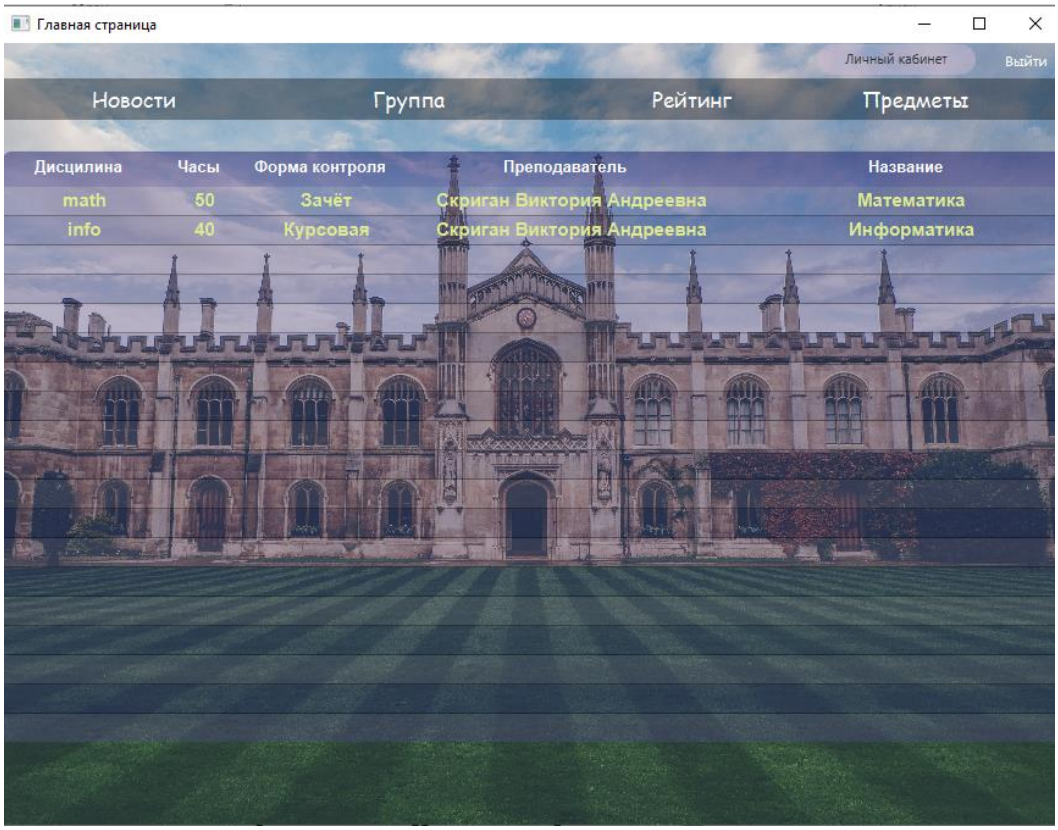

Рисунок 6.6 – Вкладка “Предметы”

В личном кабинете (рис. 6.7, рис. 6.8) преподаватель может изменить почту и пароль, при этом нужно ввести текущий пароль и повторить новый. Учитель выбирает номер группы и предмет, выставляет оценки нужному студенту.

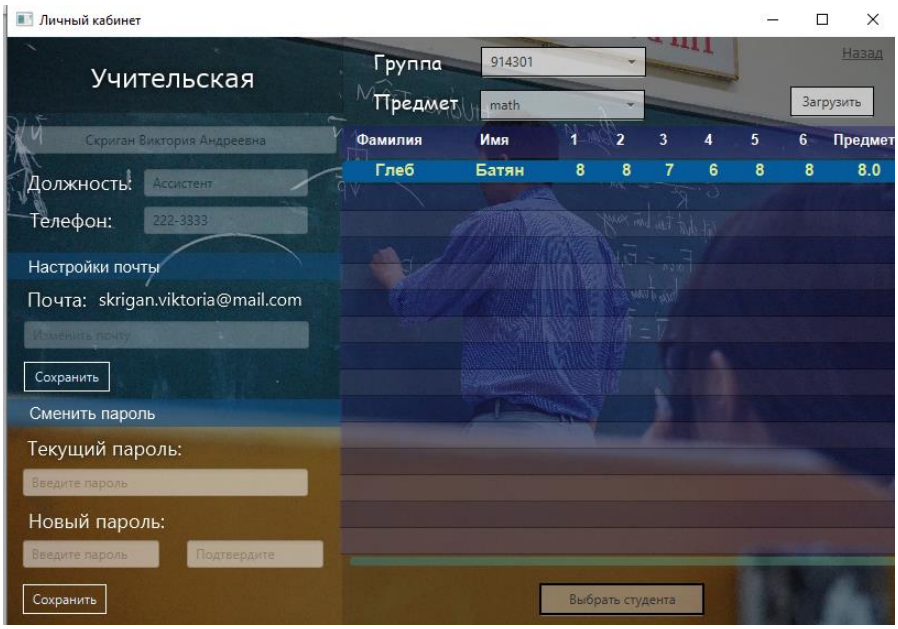

Рисунок 6.7 – Личный кабинет

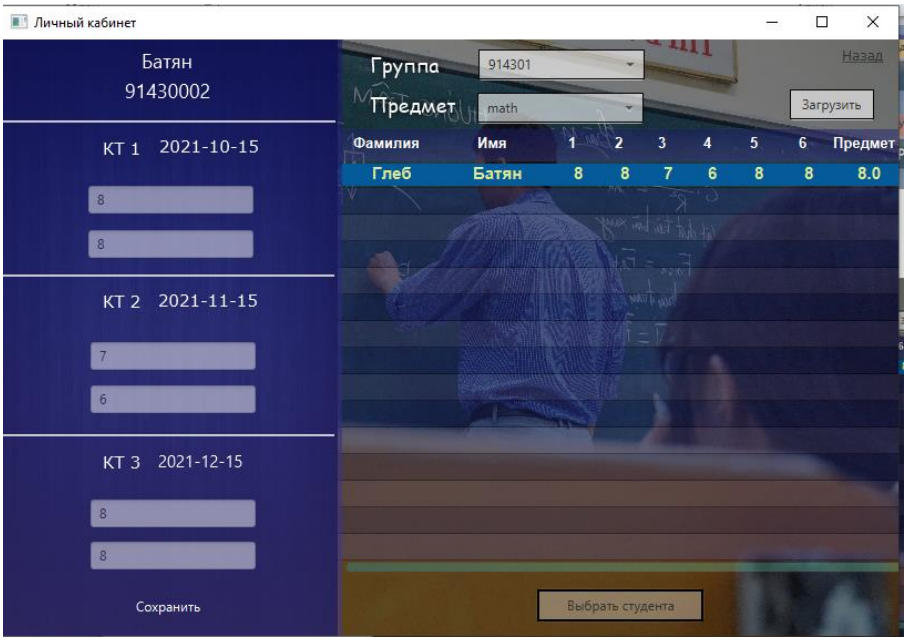

Рисунок 6.8 – Оценки студентов

В личном кабинете студента (рис. 6.9, рис. 6.10, рис. 6.11) обучающийся также может изменить почту и пароль. У студента есть возможность посмотреть информацию о дополнительных возможностях, которые предлагает университет. Во вкладке “Рейтинг” предоставляется информация о текущих предметах и оценках с формирование рейтинга по предмету. По каждой контрольной точке рассчитывается средний балл. Средняя оценка по КТ рассчитывается, если текущая дата преодолела необходимую отметку.

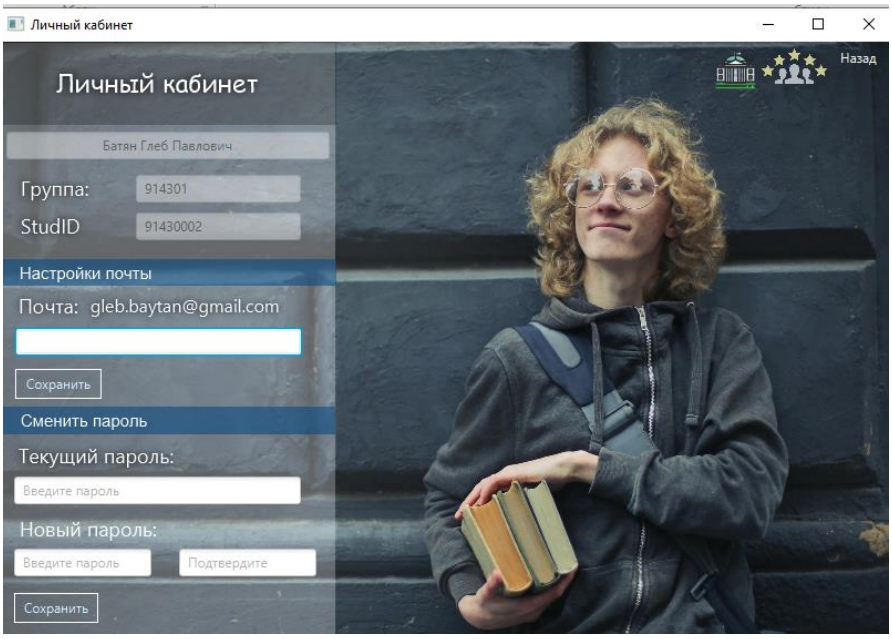

Рисунок 6.9 – Личный кабинет студента

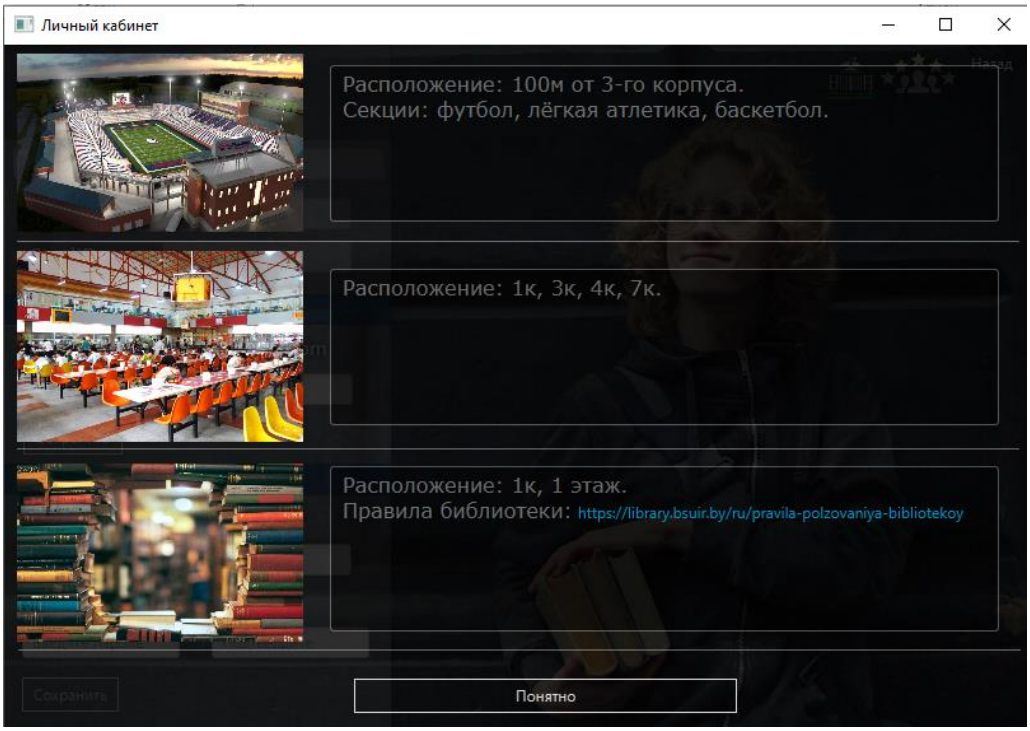

Рисунок 6.10 – Заведения университета

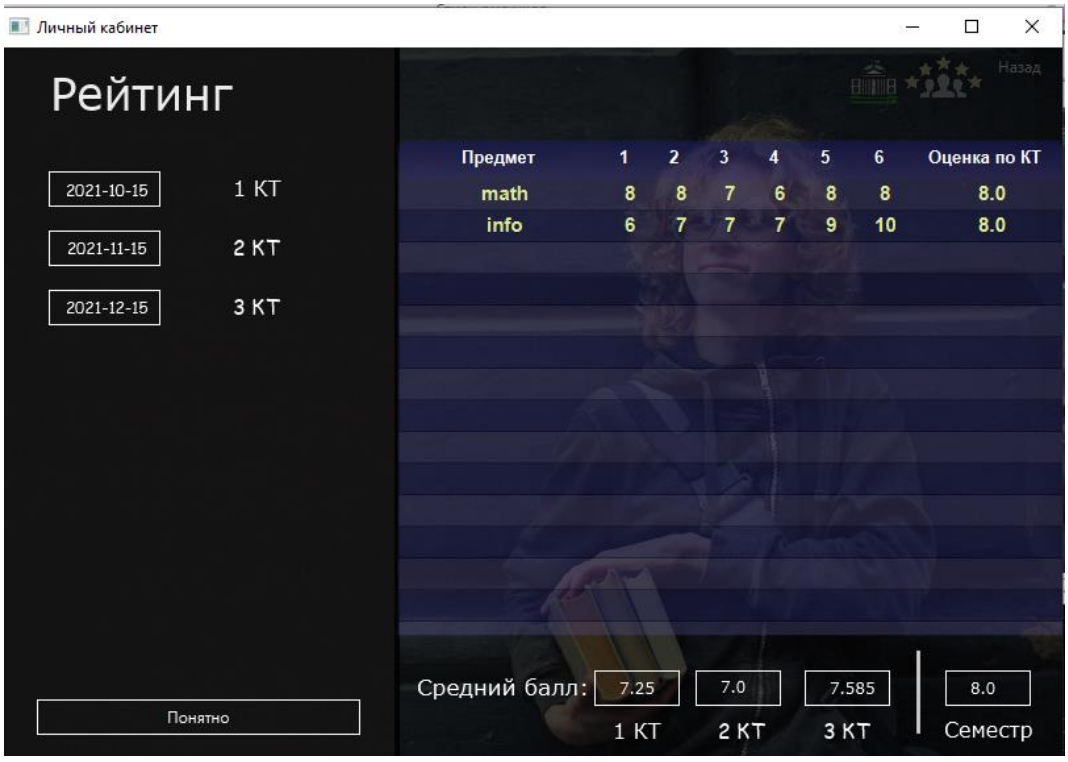

Рисунок 6.11 – Успеваемость студента

Кабинет администратора (рис. 6.12, рис. 6.13) предлагает возможность изменения информации о студентах и преподавателях, а именно добавление, удаление, изменение данных, изменение данных. Доступен поиск по студенческому id для студента и по id для преподавателя.

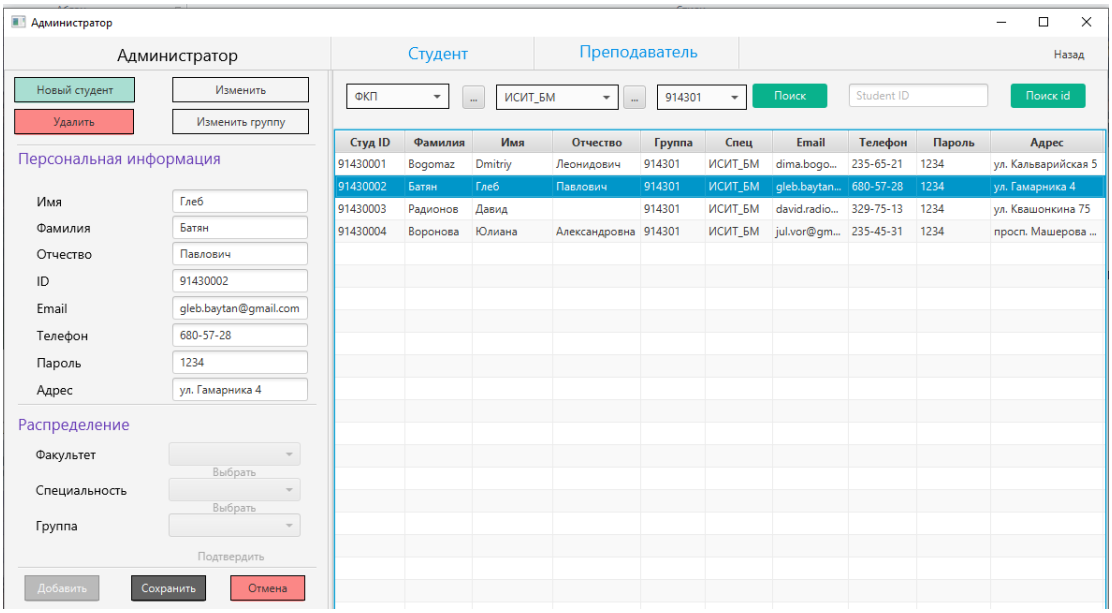

Рисунок 6.12 – Данные студентов

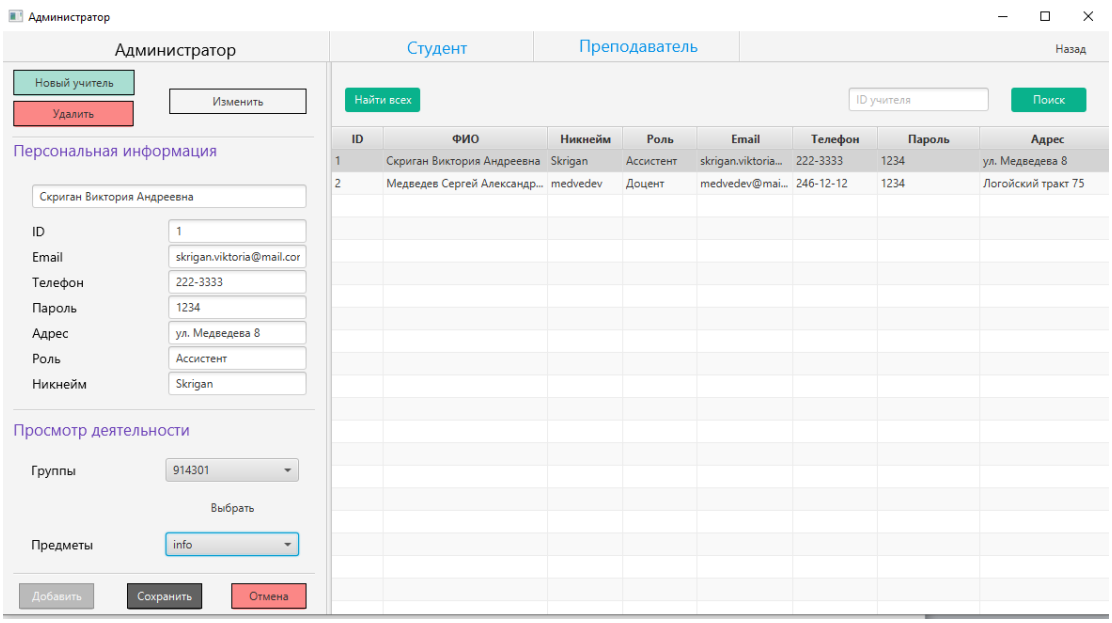

Рисунок 6.13 – Данные преподавателей

7 РЕЗУЛЬТАТ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

В приложении предусмотрена обработка неправильного ввода с помощью оповещения. Оповещение является частью JavaFX и является подклассом класса Dialog. Предупреждения – это некоторые предопределенные диалоговые окна, которые используются для отображения некоторой информации пользователю. Оповещения в основном относятся к конкретным типам оповещений. Валидация добавлена в формах авторизации (рис. 7.1), личных кабинетах студента и преподавателя (рис. 7.2, рис. 7.3). Присутствует множественное блокирование текстовых полей (рис. 7.4) для правильной организации работы пользователя.

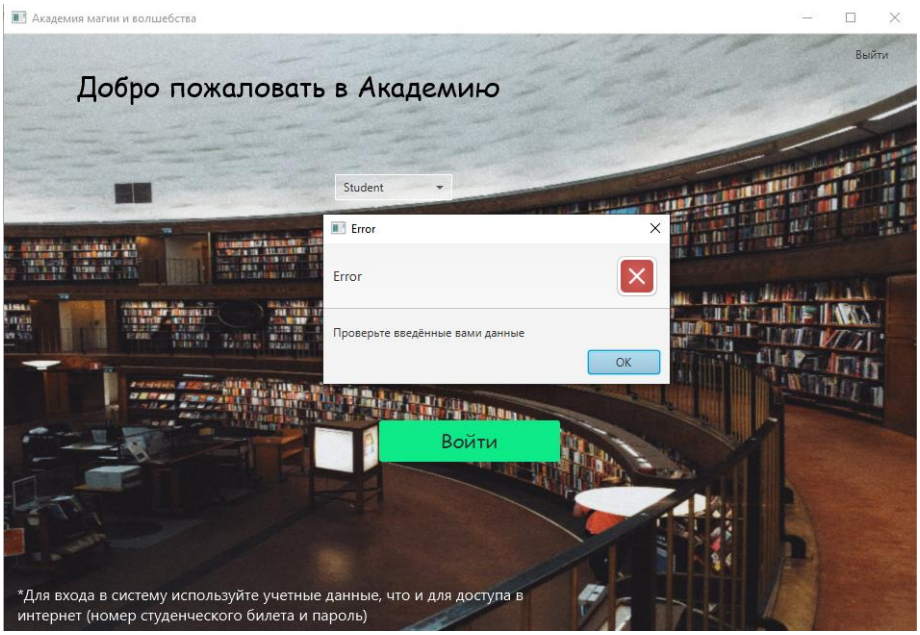

Рисунок 7.1 – Авторизация

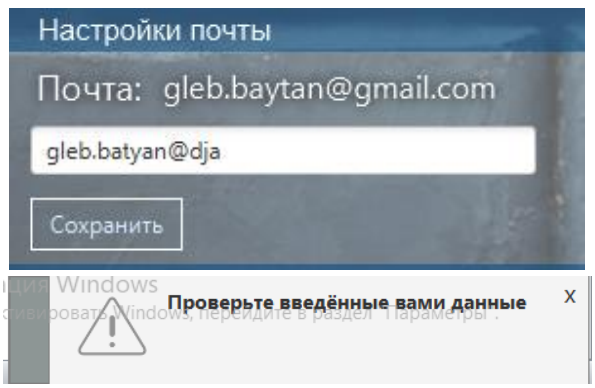

Рисунок 7.2 – Попытка изменения почты

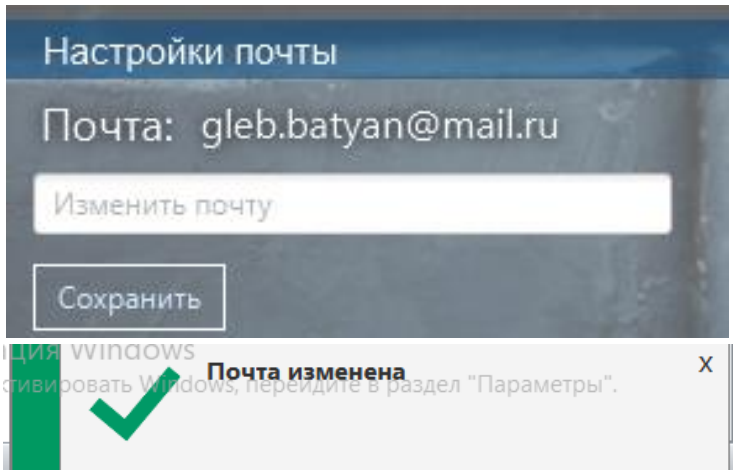

Рисунок 7.3 – Успешное изменение почты

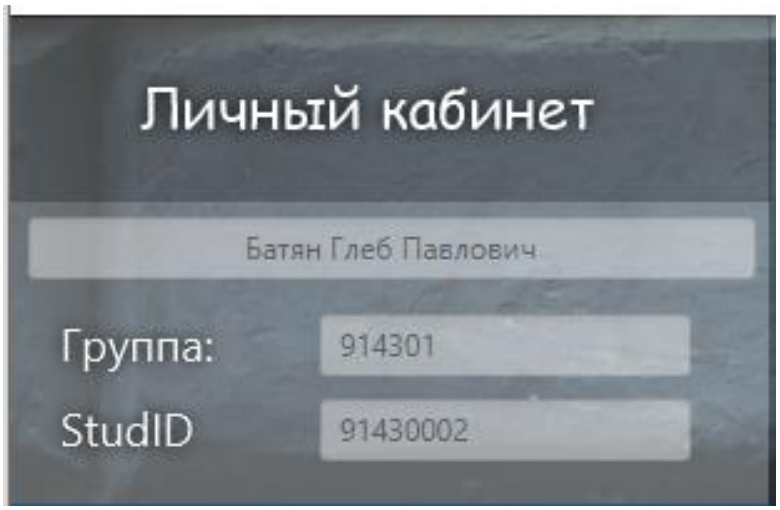

Рисунок 7.4 – Блокировка полей

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано приложение для учёта успеваемости студентов с получением быстрого, простого и оперативного доступа к информации учебного процесса.

Для реализации необходимого функционала приложения использованы особенности объектно-ориентированного программирования, набор параметров форматирования CSS и язык программирования Java. Изучены инструменты для разработки приложений с насыщенным графическим интерфейсом.

По мере работы с проектом были созданы логическая и физическая модели базы данных, установлены связи, созданы запросы на языке SQL и в конструкторе запросов, были сделаны отчёты по запросам, созданы таблицы и заполнены тестовыми данными, формы просмотра, редактирования и ввода данных.

В ходе выполнения курсового проекта были выполнены поставленные задачи, а именно:

- рассмотрение особенностей реализации системы;
- решение вопроса доступности данных об успеваемости;
- автоматизация выставления оценок;
- организация взаимодействия людей в структуре университета на основе клиент-серверного приложения;
- разработка функционального графического интерфейса пользователя.

Практически были применены различные методы проектирования информационных систем, выработаны навыки работы с базами данных, усвоены приёмы работы с СУБД MySQL и языком SQL. В ходе работы над курсовым проектом было разработано пользовательское приложение, которое позволяет работать с базой данных, осуществлять подачу запросов на выборку данных в соответствии с заданием. В ходе разработки пользовательского приложения, были на практике продемонстрированы возможности языка SQL на возможность сложной выборки, условной выборки и использования вложенных запросов.

.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Н.З. Емельянова, Т.Л. Партыка, И.И. Попов. Проектирование информационных систем. – М.: Форум, 2009. – 432 с.

[2] В.А. Гвоздева. Автоматизированные информационные технологии и системы. – М.: Форум, Инфра-М, 2011. – 544 с.

[3] К.Н. Мезенцев. Автоматизированные информационные системы. – М.: Академия, 2010. – 176 с.

[4] Громов, А.И. Управление бизнес-процессами: современные методы. монография / А.И. Громов, А. Фляйшман, В. Шмидт. – Люберцы: Юрайт, 2016. – 367 с.

[6] Эккель, Брюс Философия Java / Брюс Эккель. – М.: Питер, 2016. – 809 с.

[7] Савитч, Уолтер Язык Java. Курс программирования / Уолтер Савитч. – М.: Вильямс, 2015. – 928 с.

[8] Дэвид Макфарланд. Новая большая книга CSS. – М.: Питер, 2018. – 720 с.

[9] Макнейл П. Веб-дизайн. Книга идей веб-разработчика / П. Макнейл. – СПб.: Питер, 2017. – 480 с.

[10] Петроченков А., Новиков Е. Идеальный Landing Page. Создаем продающие веб-страницы. – СПб.: Питер, 2017. – 320 с.

[11] Кайт Т. Эффективное проектирование приложений Oracle / Кайт Т. - М.: «ЛОРИ», 2008. – 656 с.

[12] Арнолд К., Гослинг Д., Холмс Д. Язык программирования Java. 3-е издание. – М.: Издательский дом «Вильямс», 2001.

[13] Т.Ф. Старовойтова, А.Н. Лавренов. Информационные системы в бизнесе. – М.: Академия управления при Президенте Республики Беларусь, 2012. – 150 с.

ПРИЛОЖЕНИЕ А
(обязательное)
Листинг основных элементов

```
package Client.StudentController;

import Server.Command;
import Client.DBUtilis;
import Client.Entity.Student;
import Client.Entity.StudentO;
import Client.Entity.Subject;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;

import java.io.IOException;
import java.net.URL;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;

import static Client.LoginController.username;
import static Client.LoginController.connectionTCP;

public class StudentHomescreen implements Initializable {

    @FXML
    private Button btn_logout;
    @FXML
    private AnchorPane news_anchor;
    @FXML
    private AnchorPane rating_anchor;
    @FXML
    private AnchorPane group_anchor;
    @FXML
    private AnchorPane subject_anchor;
    @FXML
    private TableView<Student> rate_tableview;
    @FXML
    private TableView<Student> group_tableview;
    @FXML
    private TableView<Subject> subject_tableview;
    @FXML
    private TableColumn<Student, String>  stid_column;
    @FXML
    private TableColumn<Student, Double>  avg_column;
    @FXML
    private TableColumn<Student, String>  name_column;
    @FXML
    private TableColumn<Student, String>  lastname_column;
```

```
    @FXML
    private TableColumn<Student, String>  middlename_column;
    @FXML
    private TableColumn<Student, String>  role_column;
    @FXML
    private TableColumn<Student, String>  phone_column;
    @FXML
    private TableColumn<Student, String>  email_column;
    @FXML
    private TableColumn<Subject, String>  discipline_column;
    @FXML
    private TableColumn<Subject, String>  hours_column;
    @FXML
    private TableColumn<Subject, String>  form_column;
    @FXML
    private TableColumn<Subject, String>  teacher_column;
    @FXML
    private TableColumn<Subject, String>  namesub_column;
    @FXML
    private ComboBox<String> year;
    @FXML
    private ComboBox<String> faculty;
    @FXML
    private ComboBox<String> spec;
    @FXML
    Text tName;
    @FXML
    Text date1;
    @FXML
    Text date2;
    @FXML
    Text date3;


    public static String yea = null;
    public static String facult;
    public static String sp = null;


    private final  ObservableList<Student> data =
FXCollections.observableArrayList();
    private final  ObservableList<Subject> sub =
FXCollections.observableArrayList();
    private final  ObservableList<String> string_faculty =
FXCollections.observableArrayList();
    private final  ObservableList<String> string_year =
FXCollections.observableArrayList();
    private final  ObservableList<String> string_spec =
FXCollections.observableArrayList();

    @FXML
    private void rating() throws ParseException {
        try {
            initDataDate1();
            initDataDate2();
            initDataDate3();
        } catch (IOException e) {
            e.printStackTrace();
        }
```

```
        yea = year.getValue().trim();
         sp = spec.getValue().trim();
          facult = faculty.getValue().trim();

          Date now = new Date();
          Date mark1 = new SimpleDateFormat("yyyy-MM-
dd").parse(date1.getText());
          Date mark2 = new SimpleDateFormat("yyyy-MM-
dd").parse(date2.getText());
          Date mark3 = new SimpleDateFormat("yyyy-MM-
dd").parse(date3.getText());

        String selectStmt = "SELECT student.id_student,
AVG(ROUND(((mark1+mark2)/2),2)) as mark_avg\n" +
                "                 FROM mark1\n" +
                "                 JOIN student ON
mark1.id_student=student.id_student \n" +
                "                 JOIN groupa ON
student.id_group=groupa.id_group \n" +
                "                JOIN spec ON
groupa.code_spec=spec.code_spec \n" +
                "                 JOIN faculty ON
spec.code_faculty=faculty.code_faculty \n" +
                "                 JOIN year ON
groupa.code_year=year.code_year \n" +
                "                WHERE
faculty.code_faculty='"+facult+"' AND spec.code_spec='"+sp+"' AND
year.code_year='"+yea+"' \n" +
                "                 GROUP BY id_student";
        String selectStmt1 = "SELECT student.id_student,
AVG(ROUND(((mark1.mark1+mark1.mark2+mark2.mark3+mark2.mark4)/4),2))
as mark_avg\n" +
                "                 FROM mark1\n" +
                "                 JOIN mark2 ON
mark1.id_student=mark2.id_student \n" +
                "                 JOIN student ON
mark1.id_student=student.id_student \n" +
                "                 JOIN groupa ON
student.id_group=groupa.id_group \n" +
                "                JOIN spec ON
groupa.code_spec=spec.code_spec \n" +
                "                 JOIN faculty ON
spec.code_faculty=faculty.code_faculty \n" +
                "                 JOIN year ON
groupa.code_year=year.code_year \n" +
                "                  WHERE
faculty.code_faculty='"+facult+"' AND spec.code_spec='"+sp+"' AND
year.code_year='"+yea+"'\n" +
                "                 GROUP BY id_student";
        String selectStmt2 = "SELECT student.id_student,
ROUND(AVG(ROUND(((mark1.mark1+mark1.mark2+mark2.mark3+mark2.mark4+mar
k3.mark5+mark3.mark6 )/6),0)),2) as mark_avg\n" +
                "                 FROM mark1\n" +
                "                 JOIN mark2 ON
mark1.id_student=mark2.id_student \n" +
                "                 JOIN mark3 ON
mark1.id_student=mark3.id_student \n" +
                "                 JOIN student ON
mark1.id_student=student.id_student \n" +
```

```
                "                  JOIN groupa ON
student.id_group=groupa.id_group \n" +
                "                 JOIN spec ON
groupa.code_spec=spec.code_spec \n" +
                "                  JOIN faculty ON
spec.code_faculty=faculty.code_faculty \n" +
                "                  JOIN year ON
groupa.code_year=year.code_year \n" +
                "                   WHERE
faculty.code_faculty='"+facult+"' AND spec.code_spec='"+sp+"' AND
year.code_year='"+yea+"'\n" +
                "                  GROUP BY id_student;";

        data.clear();
        if( now.after(mark1) && now.before(mark2)) {
            connectionTCP.writeUtf(selectStmt);

        } else if (now.after(mark2) && now.before(mark3)) {
            connectionTCP.writeUtf(selectStmt1);

        } else if (now.after(mark3)){
            connectionTCP.writeUtf(selectStmt2);
                    }
        connectionTCP.writeObject(Command.READ);
        connectionTCP.flush();

        List<StudentO> students  = (List<StudentO>)
connectionTCP.readObject();
        for (StudentO student : students) {
            Student e = new Student(student);
            data.add(e);
        }

        rate_tableview.setItems(data);
        }


    public void select_year() {
            String selectStmt = "select code_year from year";
            string_year.clear();
            connectionTCP.writeUtf(selectStmt);
            connectionTCP.writeObject(Command.READYEAR);
            connectionTCP.flush();
            List<String> faculty  = (List<String>)
connectionTCP.readObject();
            string_year.addAll(faculty);
            year.setItems(string_year);
    }


    public void select_faculty() {
            String selectStmt = "select code_faculty from faculty";
            string_faculty.clear();
            connectionTCP.writeUtf(selectStmt);
            connectionTCP.writeObject(Command.READFACULTY);
            connectionTCP.flush();
            List<String> facul  = (List<String>)
connectionTCP.readObject();
            string_faculty.addAll(facul);
```

```
            faculty.setItems(string_faculty);
    }

    @FXML
    private void faculty() {
            String selectStmt = "select code_spec from spec WHERE
code_faculty='"+faculty.getValue()+"'";
            string_spec.clear();
            connectionTCP.writeUtf(selectStmt);
            connectionTCP.writeObject(Command.READSPEC);
            connectionTCP.flush();
            List<String> faculty  = (List<String>)
connectionTCP.readObject();
            string_spec.addAll(faculty);
            spec.setItems(string_spec);
    }

    @FXML
    private void rating_title() {
        rate_tableview.getItems().clear();
        news_anchor.setVisible(false);
        rating_anchor.setVisible(true);
        group_anchor.setVisible(false);
        subject_anchor.setVisible(false);
        string_spec.clear();
        select_faculty();
        select_year();
    }


    @FXML
    private void group_title() {
        news_anchor.setVisible(false);
        rating_anchor.setVisible(false);
        subject_anchor.setVisible(false);
        group_anchor.setVisible(true);
        initDataGroup();

        String selectStmt ="SELECT first_name, last_name,
middle_name, role, phone_number, email FROM student WHERE id_group
='"+tName.getText()+"'";

        data.clear();
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READGROUP);
        connectionTCP.flush();
        List<StudentO> group  = (List<StudentO>)
connectionTCP.readObject();
        for (StudentO studentO : group) {
            Student e = new Student(studentO);
            data.add(e);
        }
        group_tableview.setItems(data);
    }


    public void initDataGroup() {
        String selectStmt = "SELECT id_group FROM student WHERE
id_student = '" + username + "'";
```

```
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.SELECTGROUP);
        connectionTCP.flush();
        String group = (String) connectionTCP.readObject();
        tName.setText(group);
    }

    public void initDataDate1() throws IOException {
        String selectStmt = "SELECT date FROM kt WHERE kt_id=1";
        initCalendarDate(selectStmt, date1);
    }
    public void initDataDate2() throws IOException {
        String selectStmt = "SELECT date FROM kt WHERE kt_id=2";
        initCalendarDate(selectStmt, date2);
    }
    public void initDataDate3() throws IOException {
        String selectStmt = "SELECT date FROM kt WHERE kt_id=3";
        initCalendarDate(selectStmt, date3);
    }

    private void initCalendarDate(String selectStmt, Text date) {

        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READDATE);
        connectionTCP.flush();
        String date_kt = (String) connectionTCP.readObject();
            date.setText(date_kt);
        }


    @FXML
    private void cabinet(ActionEvent event){
        DBUtilis.changeScene(event,"student_cabinet.fxml", "Личный
кабинет", username);
    }


    @FXML
    private void subjects_title()  {
        news_anchor.setVisible(false);
        rating_anchor.setVisible(false);
        group_anchor.setVisible(false);
        subject_anchor.setVisible(true);

        String selectStmt ="SELECT
study_plan.code_subject,study_plan.hours,study_plan.form,
teacher.name,subject.subject_name\n" +
                "FROM mark1 \n" +
                "JOIN student ON
mark1.id_student=student.id_student\n" +
                "JOIN study_plan ON
mark1.id_study_plan=study_plan.id_study_plan\n" +
                "JOIN teacher ON
study_plan.id_teacher=teacher.id_teacher\n" +
                "JOIN subject ON
study_plan.code_subject=subject.code_subject\n" +
                "WHERE student.id_student ='"+ username+"'";

        sub.clear();
```

```
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READSUB);
        connectionTCP.flush();
        StudentCabinet.mark_list(sub, subject_tableview);
    }


    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {

        stid_column.setCellValueFactory(cellData ->
cellData.getValue().stIdProperty());
        avg_column.setCellValueFactory(cellData ->
cellData.getValue().avgProperty().asObject());
        name_column.setCellValueFactory(cellData ->
cellData.getValue().firstNameProperty());
        lastname_column.setCellValueFactory(cellData ->
cellData.getValue().lastNameProperty());
        middlename_column.setCellValueFactory(cellData ->
cellData.getValue().middleNameProperty());
        role_column.setCellValueFactory(cellData ->
cellData.getValue().roleProperty());
        phone_column.setCellValueFactory(cellData ->
cellData.getValue().phoneNumberProperty());
        email_column.setCellValueFactory(cellData ->
cellData.getValue().emailProperty());
        discipline_column.setCellValueFactory(cellData ->
cellData.getValue().disciplineProperty());
        hours_column.setCellValueFactory(cellData ->
cellData.getValue().hoursProperty());
        form_column.setCellValueFactory(cellData ->
cellData.getValue().formProperty());
        teacher_column.setCellValueFactory(cellData ->
cellData.getValue().teacherProperty());
        namesub_column.setCellValueFactory(cellData ->
cellData.getValue().nameProperty());



        btn_logout.setOnAction(event -> {
            String a ="";
            connectionTCP.writeUtf(a);
            connectionTCP.writeObject(Command.EXIT);
            connectionTCP.close();
            DBUtilis.changeScene(event,"login.fxml", "Академия магии
и волшебства", null);
        });
    }
    @FXML
    private void news() {
        news_anchor.setVisible(true);
        rating_anchor.setVisible(false);
        group_anchor.setVisible(false);
        subject_anchor.setVisible(false);
    }
}
```

```
package Client.TeacherController;

import Server.Command;
import Client.DBUtilis;
import Client.Entity.Subject;
import Client.Entity.SubjectO;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.*;
import javafx.scene.layout.AnchorPane;
import javafx.scene.text.Text;
import javafx.util.Duration;
import tray.notification.NotificationType;
import tray.notification.TrayNotification;
import java.net.URL;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.ResourceBundle;
import java.util.StringTokenizer;

import static Client.LoginController.*;
import static Client.LoginController.connectionTCP;

public class TeacherCabinet implements Initializable {

    @FXML
    private Button btn_logout;
    @FXML
    Text name;
    @FXML
    Text id;
    @FXML
    Text date1;
    @FXML
    Text date2;
    @FXML
    Text date3;
    @FXML
    Text email_text;
    @FXML
    TextField name_text;
    @FXML
    TextField email_TextField;
    @FXML
    TextField old_password;
    @FXML
```

```
TextField mark1;
@FXML
TextField mark2;
@FXML
TextField mark3;
@FXML
TextField mark4;
@FXML
TextField mark5;
@FXML
TextField mark6;
@FXML
PasswordField new_password;
@FXML
PasswordField new1_password;
@FXML
Text kt1;
@FXML
Text sp;
@FXML
Text kt2;
@FXML
Text kt3;
@FXML
Text sem;
@FXML
TextField role_text;
@FXML
TextField phone_text;
@FXML
ComboBox group_combobox;
@FXML
ComboBox subj;

@FXML
private TableColumn<Subject, String>  stid_column;
@FXML
private TableColumn<Subject, Double>  avg_column;
@FXML
private TableColumn<Subject, String>  name_column;
@FXML
private TableColumn<Subject, String>  lastname_column;
@FXML
private TableColumn<Subject, Integer>  mark1_column;
@FXML
private TableColumn<Subject, Integer>  mark2_column;
@FXML
private TableColumn<Subject, Integer>  mark3_column;
@FXML
private TableColumn<Subject, Integer>  mark4_column;
```

```
    @FXML
    private TableColumn<Subject, Integer>  mark5_column;
    @FXML
    private TableColumn<Subject, Integer>  mark6_column;
    @FXML
    private TableView<Subject> group_tableview;
    @FXML
    private AnchorPane mark_anchor;

    private final  ObservableList<Subject> sub = FXCollections.observableArrayList();
    private final  ObservableList<String> string_group = FXCollections.observableArrayList();
    private final  ObservableList<String> string_subject = FXCollections.observableArrayList();


    public void study_plan() {
        String selectStmt = "SELECT id_study_plan\n" +
                "from study_plan\n" +
                "JOIN teacher ON study_plan.id_teacher=teacher.id_teacher\n" +
                "where id_group='"+group_combobox.getValue()+"' AND teacher.username='"+username+"'
AND code_subject='"+subj.getValue()+"'";
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READSTUDYPLAN);
        connectionTCP.flush();
        String stud_plan= (String) connectionTCP.readObject();
        sp.setText(stud_plan);
    }

    public void select_group() {
        String selectStmt = "SELECT study_plan.id_group\n" +
                "            FROM study_plan \n" +
                "            JOIN teacher ON study_plan.id_teacher=teacher.id_teacher\n" +
                "\t\t\t\tWHERE teacher.username ='"+username+"'\n" +
                "            group by study_plan.id_group";
        string_group.clear();
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READGROUPCB);
        connectionTCP.flush();
        List<String> facul  = (List<String>) connectionTCP.readObject();
        string_group.addAll(facul);
        group_combobox.setItems(string_group);

    }
    public void select_subject() {

        String selectStmt = "SELECT study_plan.code_subject\n" +
                "            FROM study_plan \n" +
                "            JOIN teacher ON study_plan.id_teacher=teacher.id_teacher\n" +
                "\t\t\t\tWHERE teacher.username ='"+username+"'\n" +
                "            group by study_plan.code_subject";
        string_subject.clear();
```

```
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READSUBJECTCB);
        connectionTCP.flush();
        List<String> subject  = (List<String>) connectionTCP.readObject();
        string_subject.addAll(subject);
        subj.setItems(string_subject);
    }

    @FXML
    private void load_group() throws ParseException {
        mark_anchor.setVisible(false);
        Date now = new Date();
        Date d1=new SimpleDateFormat("yyyy-MM-dd").parse(date1.getText());
        Date d2 =new SimpleDateFormat("yyyy-MM-dd").parse(date2.getText());
        Date d3 =new SimpleDateFormat("yyyy-MM-dd").parse(date3.getText());
        String selectStmt = "SELECT student.last_name,student.first_name, mark1.mark1, mark1.mark2,
mark2.mark3,mark2.mark4,mark3.mark5,mark3.mark6, ROUND(((mark1.mark1+mark1.mark2)/2),0) as
mark_avg, student.id_student\n" +
                "          FROM mark1\n" +
                "            JOIN mark2 ON mark1.id_study_plan=mark2.id_study_plan\n" +
                "            JOIN mark3 ON mark1.id_study_plan=mark3.id_study_plan\n" +
                "            JOIN student ON mark1.id_student=student.id_student\n" +
                "            JOIN groupa ON student.id_group=groupa.id_group\n" +
                "            JOIN study_plan ON mark1.id_study_plan=study_plan.id_study_plan\n" +
                "          WHERE groupa.id_group='"+group_combobox.getValue()+"' AND
study_plan.code_subject='"+subj.getValue()+"'\n" +
                "          GROUP BY  study_plan.id_study_plan";

        String selectStmt1 = "SELECT student.last_name,student.first_name, mark1.mark1, mark1.mark2,
mark2.mark3,mark2.mark4,mark3.mark5,mark3.mark6,
ROUND(((mark1.mark1+mark1.mark2+mark2.mark3+mark2.mark4)/4),0) as mark_avg,
student.id_student\n" +
        "          FROM mark1\n" +
                "            JOIN mark2 ON mark1.id_study_plan=mark2.id_study_plan\n" +
                "            JOIN mark3 ON mark1.id_study_plan=mark3.id_study_plan\n" +
                "            JOIN student ON mark1.id_student=student.id_student" +
                "            JOIN groupa ON student.id_group=groupa.id_group\n" +
                "            JOIN study_plan ON mark1.id_study_plan=study_plan.id_study_plan\n" +
                "          WHERE groupa.id_group='"+group_combobox.getValue()+"' AND
study_plan.code_subject='"+subj.getValue()+"'\n" +
                "          GROUP BY  study_plan.id_study_plan";

        String selectStmt2 = "SELECT student.last_name,student.first_name, mark1.mark1, mark1.mark2,
mark2.mark3,mark2.mark4,mark3.mark5,mark3.mark6,
ROUND(((mark1.mark1+mark1.mark2+mark2.mark3+mark2.mark4+mark3.mark5+mark3.mark6)/6),0)
as mark_avg,student.id_student\n" +
        "          FROM mark1\n" +
                "            JOIN mark2 ON mark1.id_study_plan=mark2.id_study_plan\n" +
                "            JOIN mark3 ON mark1.id_study_plan=mark3.id_study_plan\n" +
                "            JOIN student ON mark1.id_student=student.id_student\n" +
```

```
            "            JOIN groupa ON student.id_group=groupa.id_group\n" +
            "            JOIN study_plan ON mark1.id_study_plan=study_plan.id_study_plan\n" +
            "           WHERE groupa.id_group='"+group_combobox.getValue()+"' AND
study_plan.code_subject='"+subj.getValue()+"'\n" +
            "           GROUP BY  study_plan.id_study_plan";

      sub.clear();
      if( now.after(d1) && now.before(d2)) {
         connectionTCP.writeUtf(selectStmt);
         connectionTCP.writeObject(Command.TEACHERMARK);
         mark_list(sub, group_tableview);
      } else if (now.after(d2) && now.before(d3)) {
         connectionTCP.writeUtf(selectStmt1);
         connectionTCP.writeObject(Command.TEACHERMARK);
         mark_list(sub, group_tableview);
      } else if (now.after(d3)){
         connectionTCP.writeUtf(selectStmt2);
         connectionTCP.writeObject(Command.TEACHERMARK);
         mark_list(sub, group_tableview);
      }
      study_plan();
   }

   static void mark_list(ObservableList<Subject> sub, TableView<Subject> mark_tableview) {
      List<SubjectO> subjects  = (List<SubjectO>) connectionTCP.readObject();
      for (SubjectO subject : subjects) {
         Subject w = new Subject(subject);
         sub.add(w);
      }
      mark_tableview.setItems(sub);
   }

   @FXML
   private void select() {
      name.setText(group_tableview.getSelectionModel().getSelectedItem().getFirstName());
      id.setText(group_tableview.getSelectionModel().getSelectedItem().getSt_Id());
      mark_anchor.setVisible(true);
      String selectStmt = "SELECT mark1.mark1,mark1.mark2, mark2.mark3, mark2.mark4, mark3.mark5,
mark3.mark6\n" +
            "FROM mark1\n" +
            "JOIN mark2 ON mark1.id_study_plan=mark2.id_study_plan\n" +
            "JOIN mark3 ON mark1.id_study_plan=mark3.id_study_plan\n" +
            "JOIN study_plan ON mark1.id_study_plan=study_plan.id_study_plan\n" +
            "WHERE mark1.id_student='"+id.getText()+"' AND study_plan.id_study_plan='"+sp.getText()
+"'\n" +
            "group by mark1.id_student\n";

      connectionTCP.writeUtf(selectStmt);
      connectionTCP.writeObject(Command.TEACHERTABLE);
      connectionTCP.flush();
```

```
   String input = (String) connectionTCP.readObject();
   StringTokenizer st = new StringTokenizer(input);
        String m1 = st.nextToken();
        String m2 = st.nextToken();
        String m3 = st.nextToken();
        String m4 = st.nextToken();
        String m5 = st.nextToken();
        String m6 = st.nextToken();
        mark1.setText(m1);
        mark2.setText(m2);
        mark3.setText(m3);
        mark4.setText(m4);
        mark5.setText(m5);
        mark6.setText(m6);
}

@FXML
private void save() throws ParseException {
   String updateStmt =
        "UPDATE mark1\n" +
             "SET mark1='"+mark1.getText()+"', mark2='"+mark2.getText()+"'\n" +
             "WHERE id_study_plan='"+sp.getText()+"';\n";
   String updateStmt1 = "UPDATE mark2\n"  +
        "          SET mark3='"+mark3.getText()+"', mark4='"+mark4.getText()+"'\n" +
        "          WHERE id_study_plan='"+sp.getText()+"'";
   String updateStmt2 = "UPDATE mark3\n"  +
        "          SET mark5='"+mark5.getText()+"', mark6='"+mark6.getText()+"'\n" +
        "          WHERE id_study_plan='"+sp.getText()+"'";

   connectionTCP.writeUtf(updateStmt);
   connectionTCP.writeObject(Command.UPDATE);
   connectionTCP.writeUtf(updateStmt1);
   connectionTCP.writeObject(Command.UPDATE);
   connectionTCP.writeUtf(updateStmt2);
   connectionTCP.writeObject(Command.UPDATE);
   mark_anchor.setVisible(false);
   load_group();
}


public void initDataDate1() {
   String selectStmt = "SELECT date FROM kt WHERE kt_id=1";
   initCalendarDate(selectStmt, date1);
}
public void initDataDate2() {
   String selectStmt = "SELECT date FROM kt WHERE kt_id=2";
   initCalendarDate(selectStmt, date2);
}
public void initDataDate3() {
```

```
        String selectStmt = "SELECT date FROM kt WHERE kt_id=3";
        initCalendarDate(selectStmt, date3);
    }
    private void initCalendarDate(String selectStmt, Text date) {
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READDATE);
        connectionTCP.flush();
        String date_kt = (String) connectionTCP.readObject();
        date.setText(date_kt);
    }


    public void initDataEmail() {
        String selectStmt = "SELECT email FROM teacher WHERE username = '" + username + "'";
        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.READEMAIL);
        connectionTCP.flush();
        String email= (String) connectionTCP.readObject();
        email_text.setText(email);
    }

    public void initDataTeacher() {
        String selectStmt = "SELECT name,role,phone,email FROM teacher WHERE username = '" +
username + "'";

        connectionTCP.writeUtf(selectStmt);
        connectionTCP.writeObject(Command.TEACHERINFO);
        connectionTCP.flush();

            String input = (String) connectionTCP.readObject();
            StringTokenizer st = new StringTokenizer(input);
            String name = st.nextToken();
            String name1 = st.nextToken();
            String name2 = st.nextToken();
            String role = st.nextToken();
            String phone = st.nextToken();
            String email = st.nextToken();
            name_text.setText(name+" "+name1+" "+name2);
            role_text.setText(role);
            phone_text.setText(phone);
            email_text.setText(email);
        }

    public void save_password() {
        String updateStmt = "UPDATE teacher SET password ='" + new_password.getText() + "' WHERE
username ='" + username+"'";
        if (old_password.getText().equals(password) && !new_password.getText().equals("")&&
new_password.getText().matches("^(?=.*[0-9])(?=.*[a-z])(?=\\S+$).{8,20}$") &&
new_password.getText().equals(new1_password.getText())) {
```

```
            connectionTCP.writeUtf(updateStmt);
            connectionTCP.writeObject(Command.UPDATE);
            password = new_password.getText();

            old_password.clear();
            new_password.clear();
            new1_password.clear();

            NotificationType notificationType = NotificationType.SUCCESS;
            TrayNotification tray = new TrayNotification();
            tray.setTitle("Пароль успешно изменён");
            tray.setNotificationType(notificationType);
            tray.showAndDismiss(Duration.millis(3000));
        }
        else {
            NotificationType notificationType = NotificationType.NOTICE;
            TrayNotification tray = new TrayNotification();
            tray.setTitle("Проверьте введённые данные");
            tray.setMessage("Кол-во символов: 8-20. Наличие одной цифры\n и одной буквы с нижним
индексом!");
            tray.setNotificationType(notificationType);
            tray.showAndDismiss(Duration.millis(3000));
        }
    }

    public void save_email() {
        String updateStmt = "UPDATE teacher SET email ='"  + email_TextField.getText() +"' WHERE
username ='" + username+"'";
        if (email_TextField.getText().matches("^[\\w!#$%&'*+/=?`{|}~^-]+(?:\\.[\\w!#$%&'*+/=?`{|}~^-
]+)*@(?:[a-zA-Z0-9-]+\\.)+[a-zA-Z]{2,6}")) {

            connectionTCP.writeUtf(updateStmt);
            connectionTCP.writeObject(Command.UPDATE);
            email_TextField.clear();

            NotificationType notificationType = NotificationType.SUCCESS;
            TrayNotification tray = new TrayNotification();
            tray.setTitle("Почта изменена");
            tray.setNotificationType(notificationType);
            tray.showAndDismiss(Duration.millis(3000));
            initDataEmail();
        }
        else {
            NotificationType notificationType = NotificationType.NOTICE;
            TrayNotification tray = new TrayNotification();
            tray.setTitle("Проверьте введённые вами данные");
            tray.setNotificationType(notificationType);
            tray.showAndDismiss(Duration.millis(3000));
        }
    }
```

```
    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        stid_column.setCellValueFactory(cellData -> cellData.getValue().stIdProperty());
        avg_column.setCellValueFactory(cellData -> cellData.getValue().avgProperty().asObject());
        name_column.setCellValueFactory(cellData -> cellData.getValue().firstNameProperty());
        lastname_column.setCellValueFactory(cellData -> cellData.getValue().lastNameProperty());
        mark1_column.setCellValueFactory(cellData -> cellData.getValue().mark1Property().asObject());
        mark3_column.setCellValueFactory(cellData -> cellData.getValue().mark3Property().asObject());
        mark4_column.setCellValueFactory(cellData -> cellData.getValue().mark4Property().asObject());
        mark5_column.setCellValueFactory(cellData -> cellData.getValue().mark5Property().asObject());
        mark6_column.setCellValueFactory(cellData -> cellData.getValue().mark6Property().asObject());
        mark2_column.setCellValueFactory(cellData -> cellData.getValue().mark2Property().asObject());
        select_group();
        select_subject();
        initDataDate1();
        initDataDate2();
        initDataDate3();
        initDataTeacher();

        btn_logout.setOnAction(event -> DBUtilis.changeScene(event,"teacher_homescreen.fxml",
"Главная страница", username));
    }
}
```

```
package Client

import javax.sql.rowset.*;

import javafx.event.ActionEvent;

import javafx.fxml.FXMLLoader;

import javafx.scene.Node;

import javafx.scene.Parent;

import javafx.scene.Scene;

import javafx.stage.Stage;

import java.io.IOException;

import java.sql.*;

import java.util.Objects;
```

```
public class DBUtilis {

   public static void changeScene(ActionEvent event,String fxmlFile, String title, String username){
      Parent root = null;
      if (username != null) {
         try {
            FXMLLoader loader = new FXMLLoader(DBUtilis.class.getResource(fxmlFile));
            root = loader.load();
         } catch (IOException e) {
            e.printStackTrace();
         }
      } else {
            try {
               root = FXMLLoader.load(Objects.requireNonNull(DBUtilis.class.getResource(fxmlFile)));
            } catch (IOException e) {
               e.printStackTrace();
            }
         }
      Stage stage = (Stage) ((Node) event.getSource()).getScene().getWindow();
         stage.setTitle(title);
      assert root != null;
      stage.setScene(new Scene(root));
         stage.show();
      }

   private static Connection connection = null;



   public static void dbConnect() throws SQLException, ClassNotFoundException {
      try {
         Class.forName("com.mysql.cj.jdbc.Driver");
         connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/univers", "root",
"Gleb210719762001");
      } catch (ClassNotFoundException e) {
         e.printStackTrace();
      }
      }


   public static void dbDisconnect() throws SQLException {
      if (connection != null && !connection.isClosed()) {
         connection.close();
      }
   }

   public static ResultSet dbExecuteQuery(String queryStmt) throws SQLException,
ClassNotFoundException {

      Statement stmt = null;
```

```
    ResultSet resultSet = null;
    CachedRowSet crs;
    try {
        dbConnect();
        stmt = connection.createStatement();

        resultSet = stmt.executeQuery(queryStmt);
        //CachedRowSet имплементация
        //Чтобы предотвратить ошибку "java.sql.SQLRecoverableException: Closed Connection: next"
        crs = RowSetProvider.newFactory().createCachedRowSet();
        crs.populate(resultSet);
    } catch (SQLException e) {
        System.out.println("Возникла проблема при executeQuery операции: " + e);
        throw e;
    } finally {
        if (resultSet != null) {
            //Close resultSet
            resultSet.close();
        }
        if (stmt != null) {
            //Close Statement
            stmt.close();
        }
        //Close connection
        dbDisconnect();
    }
    //Return CachedRowSet
    return crs;
}

public static void dbExecuteUpdate(String sqlStmt) throws SQLException, ClassNotFoundException {
    Statement stmt = null;
    try {
        dbConnect();
        stmt = connection.createStatement();
        stmt.executeUpdate(sqlStmt);
    } catch (SQLException e) {
        System.out.println("Возникла проблема при executeUpdate операции: " + e);
        throw e;
    } finally {
        if (stmt != null) {
            stmt.close();
        }
        dbDisconnect();
    }
}


 public static boolean logInAdmin (String username, String password){
     PreparedStatement preparedStatement = null;
```

```
        ResultSet resultSet = null;
        boolean log = false;
        try{
            dbConnect();
            preparedStatement = connection.prepareStatement("SELECT password FROM administrator
where username = ?");
            preparedStatement.setString(1, username);
            resultSet = preparedStatement.executeQuery();


            if (!resultSet.isBeforeFirst()){
                log = false;
            } else{
                while (resultSet.next()){
                    String retrievedPassword = resultSet.getString("password");
                    if (retrievedPassword.equals(password))
                        log = true;
                }
            }
        } catch (SQLException | ClassNotFoundException e){
            e.printStackTrace();
        } finally {
            try_catch(preparedStatement, resultSet);
        }
        return log;
    }

    private static void try_catch(PreparedStatement preparedStatement, ResultSet resultSet) {
        if (resultSet != null){
            try {
                resultSet.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (preparedStatement != null){
            try {
                preparedStatement.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
        if (connection != null) {
            try {
                connection.close();
            } catch (SQLException e){
                e.printStackTrace();
            }
        }
    }
```

```
  public static boolean logInTeacher (String username, String password){
    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    boolean log = false;
    try{
      dbConnect();
      preparedStatement = connection.prepareStatement("SELECT password FROM teacher where
username = ?");
      preparedStatement.setString(1, username);
      resultSet = preparedStatement.executeQuery();
      System.out.println(preparedStatement);

      if (!resultSet.isBeforeFirst()){
        log = false;
      } else{
        while (resultSet.next()){
          String retrievedPassword = resultSet.getString("password");
          if (retrievedPassword.equals(password))
            log = true;
        }
      }
    } catch (SQLException | ClassNotFoundException e){
      e.printStackTrace();
    } finally {
      try_catch(preparedStatement, resultSet);
    }
    return log;
  }


  public static boolean logInStudent (String username, String password){

    PreparedStatement preparedStatement = null;
    ResultSet resultSet = null;
    boolean log = false;
    try{
      dbConnect();
      preparedStatement = connection.prepareStatement("SELECT password FROM student where
id_student = ?");
      preparedStatement.setString(1, username);
      resultSet = preparedStatement.executeQuery();

      if (!resultSet.isBeforeFirst()){
       log = false;
      } else{
        while (resultSet.next()){
          String retrievedPassword = resultSet.getString("password");
          if (retrievedPassword.equals(password))
            log = true;
```

```
            }
        }
    } catch (SQLException | ClassNotFoundException e){
        e.printStackTrace();
    } finally {
        try_catch(preparedStatement, resultSet);
    }
    return log;
}
}
```

```
package Client;
import Server.Command;
import Server.ConnectionTCP;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;
import java.io.IOException;
import java.net.Socket;
import java.net.URL;
import java.util.ResourceBundle;

public class LoginController implements Initializable {

    @FXML
    private PasswordField txt_password;

    @FXML
    private TextField txt_username;
    @FXML
    private ChoiceBox cbUser;

    public static ConnectionTCP connectionTCP ;
    public static String username = null;
    public static String password = null;


    @FXML
    private void loginButtonClick(ActionEvent event) {
        boolean aa;
        username = txt_username.getText().trim();
        password = txt_password.getText();
        String userType = cbUser.getValue().toString().trim();
        switch (userType) {
            case "Admin" -> {
```

```
            connectionTCP.writeUtf(username+" "+password);
            connectionTCP.writeObject(Command.LOGINADMIN);
            aa = (boolean) connectionTCP.readObject();
            if (aa) {
                DBUtilis.changeScene(event,"admin_homescreen.fxml", "Главная страница", username);
                System.out.println("Добро пожаловать!");
            }
            else{
                System.out.println("Авторизация не удалась.");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("Проверьте введённые вами данные");
                alert.show();
            }
        }
        case "Student" -> {
            connectionTCP.writeUtf(username+" "+password);
            connectionTCP.writeObject(Command.LOGINSTUDENT);
            aa = (boolean) connectionTCP.readObject();

            if (aa) {
                DBUtilis.changeScene(event, "student_homescreen.fxml", "Главная страница", username);
                System.out.println("Добро пожаловать!");
            }
            else{
                System.out.println("Авторизация не удалась.");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("Проверьте введённые вами данные");
                alert.show();
            }
        }
        case "Teacher" -> {
            connectionTCP.writeUtf(username+" "+password);
            connectionTCP.writeObject(Command.LOGINTEACHER);
            aa = (boolean) connectionTCP.readObject();

            if (aa) {
              DBUtilis.changeScene(event, "teacher_homescreen.fxml", "Главная страница", username);
                System.out.println("Добро пожаловать!");
            }
            else{
                System.out.println("Авторизация не удалась.");
                Alert alert = new Alert(Alert.AlertType.ERROR);
                alert.setContentText("Проверьте введённые вами данные");
                alert.show();
            }
        }
    }
}

@FXML
```

```
    private void exit(){
        String a ="";
        connectionTCP.writeUtf(a);
        connectionTCP.writeObject(Command.EXIT);
        connectionTCP.close();
        System.exit(0);
    }

    @Override
    public void initialize(URL url, ResourceBundle resourceBundle) {
        try {
            connectionTCP = new ConnectionTCP(new Socket("localhost", 3636));
        } catch (IOException e) {
            e.printStackTrace();
            System.exit(-1);
        }

    }
}


package
Server;
                import Client.Entity.StudentO;
                import Client.Entity.SubjectO;
                import java.io.IOException;
                import java.net.Socket;
                import java.sql.SQLException;
                import java.util.ArrayList;
                import java.util.List;
                import java.util.StringTokenizer;
                public class RequestHandler implements Runnable {
                    private final ConnectionTCP connectionTCP;
                    private ArrayList<ConnectionTCP> clients;
                    public RequestHandler(Socket socket) {
                        connectionTCP = new ConnectionTCP(socket);//сокет соединения с клиентом
                    }
                    @Override
                    public void run() {
                        StudentFunctions server_execute = new StudentFunctions();
                        while (true) {
                            String aaa = null;
                            try {
                                aaa = connectionTCP.readUtf();
                            } catch (IOException e) {
                                e.printStackTrace();
                            }
                            Command command = (Command) connectionTCP.readObject();
                            System.out.println(command);
```

```
                switch (command) {
                    case READ -> {
                        List<StudentO> students = null;
                        try {
                            System.out.println(aaa);
                            students = server_execute.getStudentRate(aaa);
                        } catch (SQLException | ClassNotFoundException | IOException e) {
                            e.printStackTrace();
                        }
                        connectionTCP.writeObject(students);
                    }
                    case ADMINSTUDENTS -> {
                        List<StudentO> students = null;
                        try {
                            System.out.println(aaa);
                            students = server_execute.getAdminStudents(aaa);
                        } catch (SQLException | ClassNotFoundException | IOException e) {
                            e.printStackTrace();
                        }
                        connectionTCP.writeObject(students);
                    }
```

……

```
public class
StudentFunctions{
//
                    public List<StudentO> getStudentRate(String aaa) throws SQLException,
                ClassNotFoundException, IOException {
                        List<StudentO> students = new ArrayList<>();
                        try {
                            ResultSet rs = DBUtilis.dbExecuteQuery(aaa);
                            while (rs.next()) {
                                String st_id = rs.getString("id_student");
                                double avg= rs.getDouble("mark_avg");
                                StudentO student = new StudentO(st_id, avg);
                                students.add(student);
                            }
                        } catch (SQLException | ClassNotFoundException e){
                            System.out.println("Error occurred while getting students information from
                DB.\n" + e);
                            e.printStackTrace();
                        }
                        return students;
                    }
                    public boolean logTeacher(String username, String password){
                        return DBUtilis.logInTeacher(username,password);
                    }
```

```
    public boolean logAdmin(String username, String password){
        return DBUtilis.logInAdmin(username,password);
    }
    public boolean logStudent(String username, String password){
        return DBUtilis.logInStudent(username,password);
    }
    public List<StudentO> getAdminStudents(String aaa) throws SQLException,
ClassNotFoundException, IOException {
        List<StudentO> students = new ArrayList<>();
        try {
            ResultSet rs = DBUtilis.dbExecuteQuery(aaa);
            while (rs.next()) {
                String st_id = rs.getString("id_student");
                String first_name = rs.getString("first_name");
                String last_name = rs.getString("last_name");
                String middle_name = rs.getString("middle_name");
                int group = rs.getInt("id_group");
                String spec = rs.getString("code_spec");
                String email = rs.getString("email");
                String phone_number = rs.getString("phone_number");
                String password = rs.getString("password");
                String address = rs.getString("address");
                StudentO student = new StudentO(st_id,
first_name,last_name,middle_name,group,spec,email,phone_number,password,a
ddress);
                students.add(student);
            }
        } catch (SQLException | ClassNotFoundException e){
            System.out.println("Error occurred while getting students information from
DB.\n" + e);
            e.printStackTrace();
        }
        return students;
    }
```

ПРИЛОЖЕНИЕ Б
(обязательное)
Графический материал

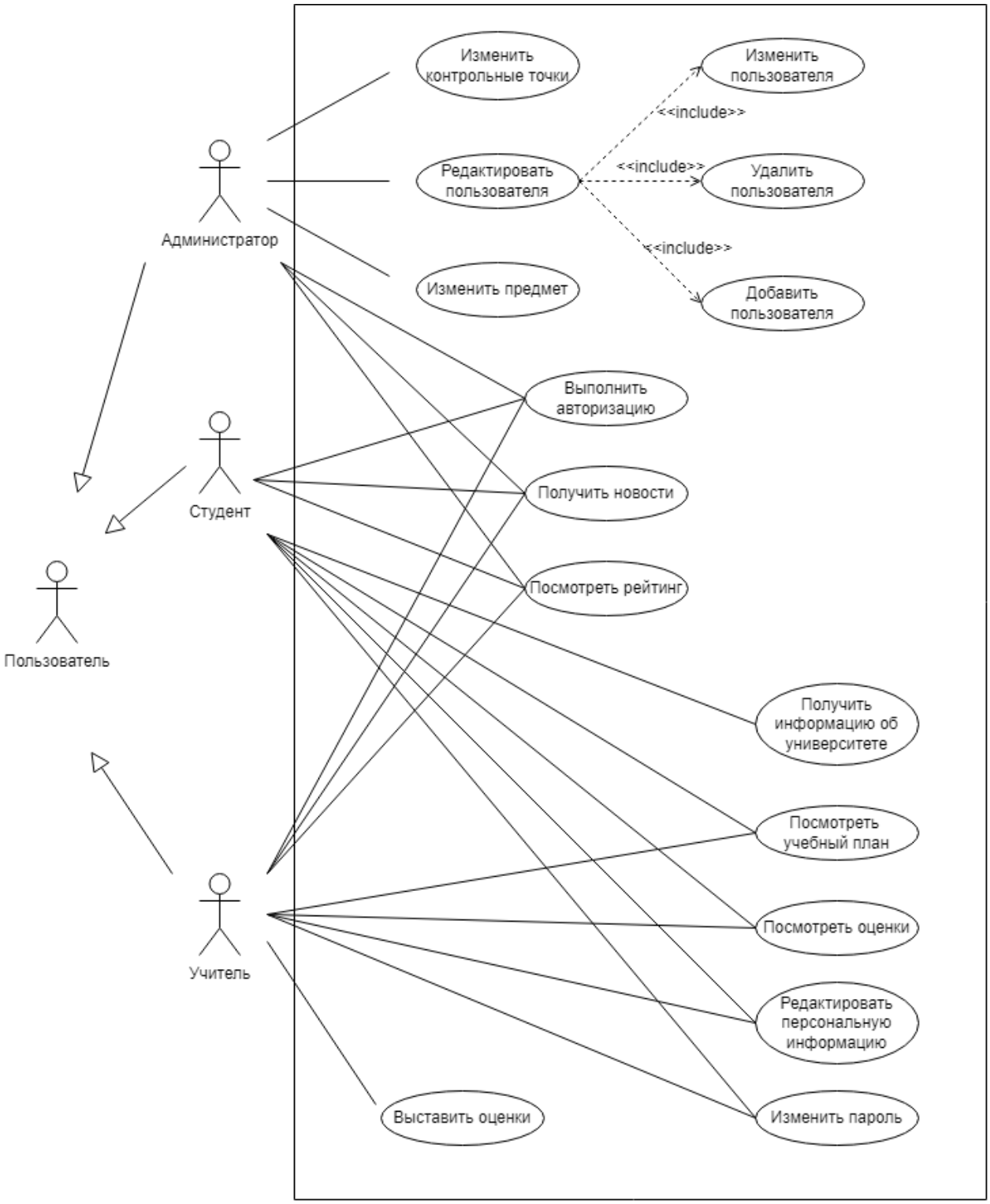

Рисунок В.1 – Диаграмма вариантов использования

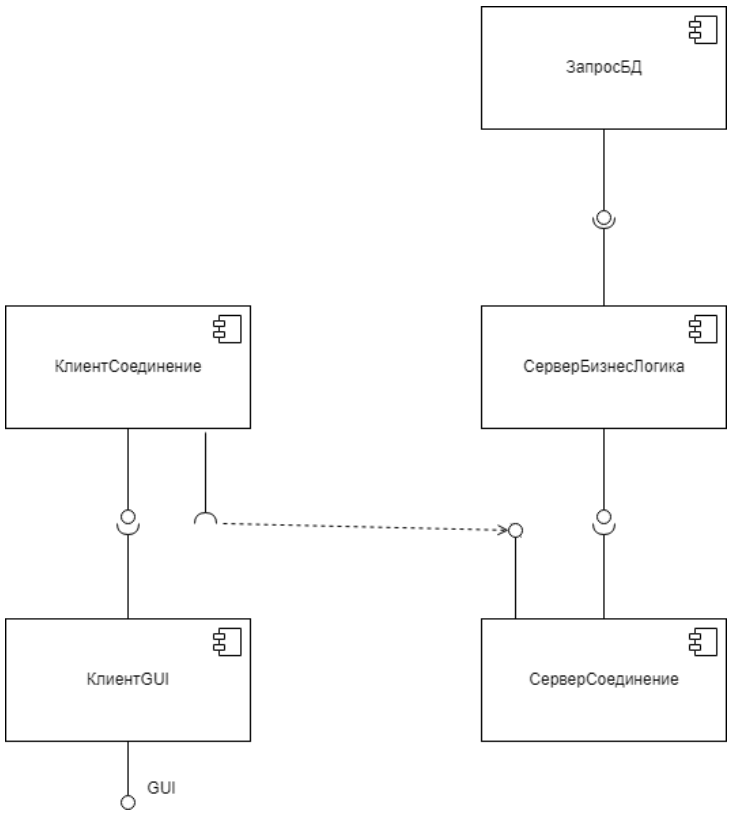

Рисунок В.2 – Диаграмма компонентов

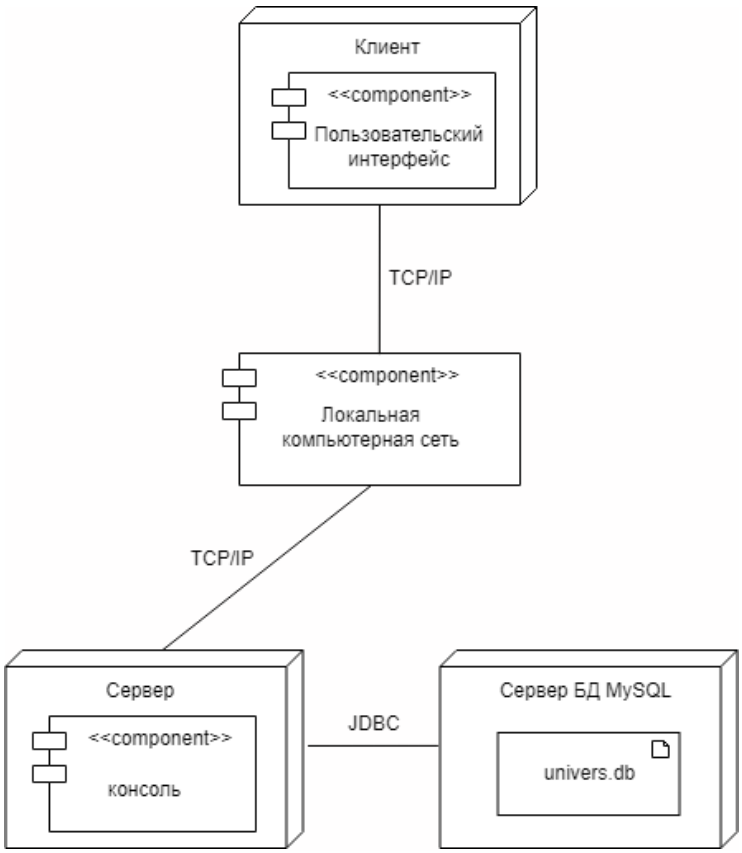

Рисунок В.3 – Диаграмма размещения

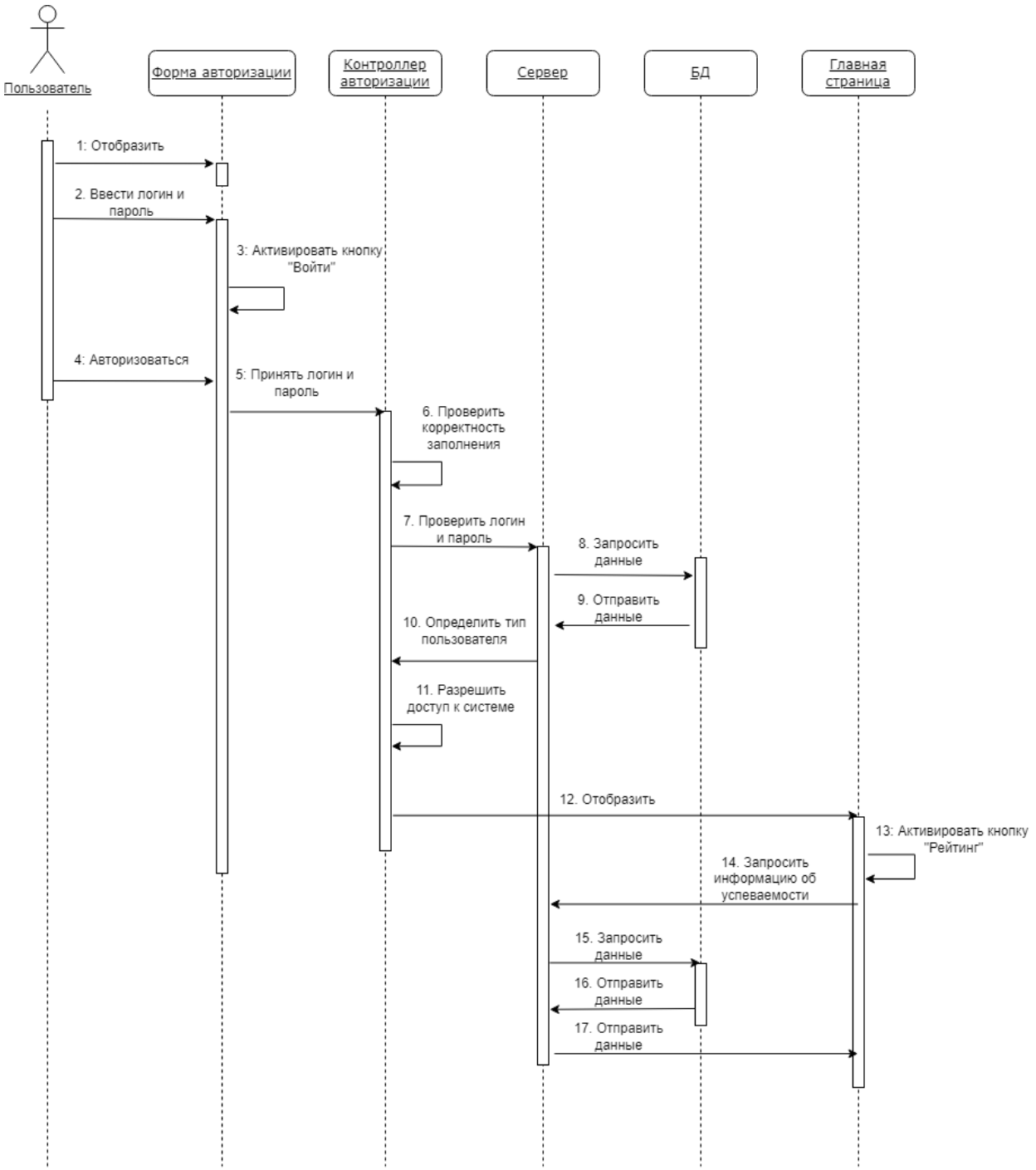

Рисунок В.4 – Диаграмма последовательности

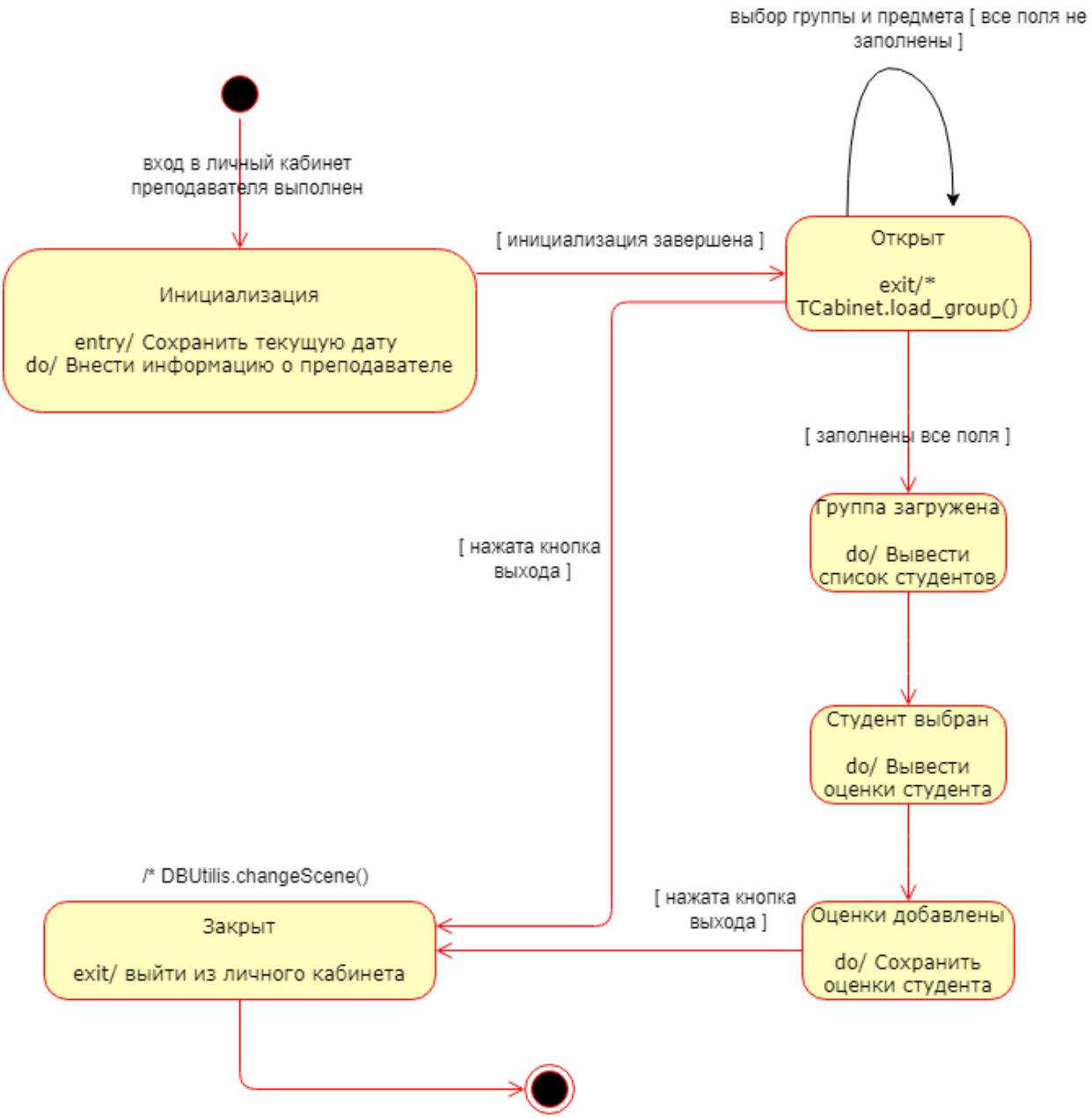

Рисунок В.5 – Диаграмма состояния

ПРИЛОЖЕНИЕ В
(обязательное)
Листинг скрипта базы данных

```
-- MySQL Script generated by MySQL Workbench

-- Wed Nov 24 21:47:46 2021

-- Model: New Model    Version: 1.0

-- MySQL Workbench Forward Engineering


SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DAT
E,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';


-- -----------------------------------------------------

-- Schema univers

-- -----------------------------------------------------


-- -----------------------------------------------------

-- Schema univers

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `univers` DEFAULT CHARACTER SET utf8 COLLATE utf8_danish_ci ;

USE `univers` ;


-- -----------------------------------------------------

-- Table `univers`.`faculty`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `univers`.`faculty` (

  `code_faculty` VARCHAR(45) NOT NULL,

  `faculty_name` VARCHAR(45) NULL DEFAULT NULL,

  PRIMARY KEY (`code_faculty`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8

COLLATE = utf8_danish_ci;
```

```
-- -----------------------------------------------------
-- Table `univers`.`spec`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`spec` (
  `code_spec` VARCHAR(45) NOT NULL,
  `spec_name` VARCHAR(45) NOT NULL,
  `code_faculty` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`code_spec`),
  INDEX `faculty_spec_idx` (`code_faculty` ASC) VISIBLE,
  CONSTRAINT `faculty_spec`
    FOREIGN KEY (`code_faculty`)
    REFERENCES `univers`.`faculty` (`code_faculty`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_danish_ci;


-- -----------------------------------------------------
-- Table `univers`.`semester`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`semester` (
  `id_semester` INT NOT NULL,
  PRIMARY KEY (`id_semester`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `univers`.`year`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`year` (
  `code_year` VARCHAR(45) NOT NULL,
```

```
  PRIMARY KEY (`code_year`))
ENGINE = InnoDB;



-- -----------------------------------------------------
-- Table `univers`.`groupa`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`groupa` (
  `id_group` VARCHAR(45) NOT NULL,
  `code_spec` VARCHAR(45) NOT NULL,
  `id_semester` INT NOT NULL,
  `code_year` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id_group`),
  INDEX `spec_idx` (`code_spec` ASC) VISIBLE,
  INDEX `sem_idx` (`id_semester` ASC) VISIBLE,
  INDEX `year_idx` (`code_year` ASC) VISIBLE,
  CONSTRAINT `spec`
    FOREIGN KEY (`code_spec`)
    REFERENCES `univers`.`spec` (`code_spec`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `sem`
    FOREIGN KEY (`id_semester`)
    REFERENCES `univers`.`semester` (`id_semester`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `year`
    FOREIGN KEY (`code_year`)
    REFERENCES `univers`.`year` (`code_year`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_danish_ci;
```

```
-- -----------------------------------------------------
-- Table `univers`.`Student`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`Student` (
  `id_student` VARCHAR(45) NOT NULL,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `middle_name` VARCHAR(45) NULL,
  `id_group` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `role` VARCHAR(45) NULL,
  `phone_number` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  PRIMARY KEY (`id_student`),
  INDEX `group_idx` (`id_group` ASC) VISIBLE,
  UNIQUE INDEX `id_student_UNIQUE` (`id_student` ASC) VISIBLE,
  CONSTRAINT `group`
    FOREIGN KEY (`id_group`)
    REFERENCES `univers`.`groupa` (`id_group`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_danish_ci;


-- -----------------------------------------------------
-- Table `univers`.`subject`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`subject` (
  `code_subject` VARCHAR(45) NOT NULL,
  `subject_name` VARCHAR(45) NOT NULL,
```

```
  `code_spec` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`code_subject`),
  INDEX `sub_spec_idx` (`code_spec` ASC) VISIBLE,
  CONSTRAINT `sub_spec`
    FOREIGN KEY (`code_spec`)
    REFERENCES `univers`.`spec` (`code_spec`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8
COLLATE = utf8_danish_ci;


-- -----------------------------------------------------
-- Table `univers`.`Administator`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`Administator` (
  `username` VARCHAR(45) NULL,
  `password` VARCHAR(45) NULL)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `univers`.`Teacher`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`Teacher` (
  `id_teacher` VARCHAR(45) NOT NULL,
  `username` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `name` VARCHAR(45) NULL,
  `email` VARCHAR(45) NULL,
  PRIMARY KEY (`id_teacher`))
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `univers`.`study_plan`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`study_plan` (
  `id_study_plan` VARCHAR(45) NOT NULL,
  `id_teacher` VARCHAR(45) NOT NULL,
  `code_subject` VARCHAR(45) NOT NULL,
  `id_semester` INT NOT NULL,
  `form` VARCHAR(45) NULL,
  `id_group` VARCHAR(45) NOT NULL,
  `hours` VARCHAR(45) NULL,
  PRIMARY KEY (`id_study_plan`),
  INDEX `teach_idx` (`id_teacher` ASC) VISIBLE,
  INDEX `subject_plan_idx` (`code_subject` ASC) VISIBLE,
  INDEX `group_idx` (`id_group` ASC) VISIBLE,
  CONSTRAINT `teach`
    FOREIGN KEY (`id_teacher`)
    REFERENCES `univers`.`Teacher` (`id_teacher`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `subject_plan`
    FOREIGN KEY (`code_subject`)
    REFERENCES `univers`.`subject` (`code_subject`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `gruppa`
    FOREIGN KEY (`id_group`)
    REFERENCES `univers`.`groupa` (`id_group`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `univers`.`kt`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`kt` (
  `kt_id` INT NOT NULL,
  `date` DATE NULL,
  PRIMARY KEY (`kt_id`))
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `univers`.`mark1`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`mark1` (
  `kt_id` INT NULL,
  `mark1` VARCHAR(45) NULL,
  `mark2` VARCHAR(45) NULL,
  `id_student` VARCHAR(45) NULL,
  `id_study_plan` VARCHAR(45) NULL,
  INDEX `stud_idx` (`id_student` ASC) VISIBLE,
  INDEX `sp_idx` (`id_study_plan` ASC) VISIBLE,
  INDEX `kt2_idx` (`kt_id` ASC) VISIBLE,
  CONSTRAINT `stud`
    FOREIGN KEY (`id_student`)
    REFERENCES `univers`.`Student` (`id_student`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `sp`
    FOREIGN KEY (`id_study_plan`)
    REFERENCES `univers`.`study_plan` (`id_study_plan`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `kt2`
    FOREIGN KEY (`kt_id`)
```

```
    REFERENCES `univers`.`kt` (`kt_id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION)
ENGINE = InnoDB;


-- -----------------------------------------------------
-- Table `univers`.`mark2`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`mark2` (
  `kt_id` INT NULL,
  `mark3` VARCHAR(45) NULL,
  `mark4` VARCHAR(45) NULL,
  `id_student` VARCHAR(45) NULL,
  `id_study_plan` VARCHAR(45) NULL,
  INDEX `stud1_idx` (`id_student` ASC) VISIBLE,
  INDEX `sp1_idx` (`id_study_plan` ASC) VISIBLE,
  INDEX `kt1_idx` (`kt_id` ASC) VISIBLE,
  CONSTRAINT `stud1`
    FOREIGN KEY (`id_student`)
    REFERENCES `univers`.`Student` (`id_student`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `sp1`
    FOREIGN KEY (`id_study_plan`)
    REFERENCES `univers`.`study_plan` (`id_study_plan`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `kt1`
    FOREIGN KEY (`kt_id`)
    REFERENCES `univers`.`kt` (`kt_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;
```

```
-- -----------------------------------------------------
-- Table `univers`.`mark3`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `univers`.`mark3` (
  `kt_id` INT NULL,
  `mark5` VARCHAR(45) NULL,
  `mark6` VARCHAR(45) NULL,
  `id_student` VARCHAR(45) NULL,
  `id_study_plan` VARCHAR(45) NULL,
  INDEX `stud2_idx` (`id_student` ASC) VISIBLE,
  INDEX `sp2_idx` (`id_study_plan` ASC) VISIBLE,
  INDEX `kt_idx` (`kt_id` ASC) VISIBLE,
  CONSTRAINT `stud2`
    FOREIGN KEY (`id_student`)
    REFERENCES `univers`.`Student` (`id_student`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `sp2`
    FOREIGN KEY (`id_study_plan`)
    REFERENCES `univers`.`study_plan` (`id_study_plan`)
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  CONSTRAINT `kt`
    FOREIGN KEY (`kt_id`)
    REFERENCES `univers`.`kt` (`kt_id`)
    ON DELETE CASCADE
    ON UPDATE CASCADE)
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

ПРИЛОЖЕНИЕ Г
Ведомость документов