

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Приложение таск-трекер для планирования и
контроля выполнения практических заданий
студентами**

Студент _____ Капасовский Даниил Михайлович
(подпись, дата) (ФИО полностью)

Руководитель _____ Шибанов С.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____ А.Ю. Козлов
«20» декабря 2024 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Капасовский Даниил Михайлович гр. 21ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Приложение таск-трекер для планирования и контроля выполнения
практических заданий студентами

Тема утверждена приказом ПГУ № _____ от " ____ " _____ 20 ____

3. Руководитель работы к.т.н., доцент кафедры «МО и ПЭВМ» Шибанов С.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: Приложение таск-трекер для планирования и контроля выполнения
практических заданий студентами

2) Среда выполнения: Windows 11

Языковые средства разработки: C#, SQL, UML

Инструментальные средства: Visual Studio, Windows Forms, SQL Server
Management Studio, Supabase, Draw.io

Технологии разработки: ООП, многоуровневая архитектура

3) Основные функции разрабатываемого программного обеспечения:

а) Создание, назначение выполнения предметов, практических заданий, курсовых
работ

б) Создание, назначение выполнения предметов, практических заданий,
курсовых
работ

в) Отправка результатов выполненных работ на проверку

<i>г) Прием, отклонение выполненной работы</i>
<i>д) Визуализация прогресса выполнения заданий</i>
<i>е) Хранение информации об успеваемости студента, группы</i>
<i>ж) Доступ к приложению без выхода в сеть</i>

5. Перечень подлежащих разработке вопросов:

<i>1) Анализ предметной области и требований к системе</i>
<i>2) Проектирование программных средств</i>
<i>3) Реализация программных средств</i>
<i>4) Тестирование программных средств</i>
<i>5) Оформление пояснительной записки</i>

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 125 листов, 31 рисунок, 15 таблиц, 25 использованных источников и 8 приложений.

TASK-ТРЕКЕР, ДЕСКТОПНОЕ ПРИЛОЖЕНИЕ, C#, WINDOWS FORMS, БАЗА ДАННЫХ, SQL, UML, ПРОГРЕСС ВЫПОЛНЕНИЯ, УСПЕВАЕМОСТЬ, ОБУЧАЮЩИЙСЯ, ПРЕПОДАВАТЕЛЬ, ПРАКТИЧЕСКОЕ ЗАДАНИЕ, КУРСОВАЯ РАБОТА, УЧЕБНЫЙ ПРЕДМЕТ.

Объект исследования – приложение для контроля, планирования и отслеживания выполнения практических заданий студентами.

Цель выпускной квалификационной работы – спроектировать и разработать десктопное приложение, предоставляющие студентам и преподавателям платформу для организации электронного образования с визуализацией прогресса выполнения заданий и возможностью работы без выхода в сеть.

Технологии разработки – реализация программных средств ведется в среде разработки Visual Studio на языке программирования C# с использованием технологий Windows Forms. Конструирование базы данных производится в СУБД SQL Server Management Studio 19.

Среда исполнения – персональный компьютер пользователя.

Результаты работы – в ходе выполнения выпускной квалификационной работы было спроектировано, разработано и протестировано десктопное приложение для контроля, назначения и отображения прогресса выполнения практических заданий студента.

					ПГУ 09.03.04 – 08BP211.13 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Приложение таск-трекер для планирования и контроля выполнения практических заданий студентами Пояснительная записка			Лит.	Лист	Листов	
Разраб.		Капасовский Д.М.									
Пров.		Шибанов С.В.							4	125	
Реценз.								Группа 21ВП1			
Н. Контр.		Такташкин Д.В.									
Утв.		Козлов А.Ю.									

Оглавление

Введение	8
1 Анализ предметной области и планирование разработки	10
1.1 Актуальность применения приложений трекеров задач в электронном обучении	10
1.1.1 Назначение и особенности электронного обучения	10
1.1.2 Задачи отслеживания прогресса	11
1.1.3 Аспекты применения трекеров задач в электронном образовании ..	13
1.2 Анализ предметной области разрабатываемого приложения	14
1.2.1 Назначение разрабатываемого приложения	14
1.2.2 Структурная организация предметной области, действующие лица, их взаимоотношения	15
1.2.3 Анализ бизнес-требований.....	16
1.3 Сравнительный анализ аналогичных приложений.....	17
1.3.1 Цель сравнительного анализа	17
1.3.2 Обзор аналогов	17
1.3.3 Результаты сравнительного анализа.....	20
1.4 Планирование и оценка затрат процесса разработки	22
2 Анализ требований и постановка задач на разработку	25
2.1 Анализ функциональных требований	25
2.2 Анализ требований к данным	29
2.3 Анализ требований к пользовательскому интерфейсу	31
2.4 Анализ требований к архитектуре приложения	33
2.5 Обоснование выбора технологий и средств разработки	34

3 Разработка приложения	37
3.1 Разработка базы данных.....	37
3.2 Разработка пользовательского интерфейса.....	39
3.3 Разработка программного обеспечения	48
3.3.1 Разработка архитектуры программных средств	48
3.3.2 Программное обеспечение слоя представления данных.....	49
3.3.3 Программное обеспечение слоя бизнес-логики.....	51
3.3.4 Программное обеспечение слоя обращения к данным.....	52
3.3.5 Сценарии выполнения обращений	53
4 Оценка качества программного обеспечения	55
4.1 Обзор мероприятий оценки качества программного обеспечения.....	55
4.2 Тестирование методом черного ящика с применением функциональной диграммы и таблицы решений	56
4.3 Попарное тестирование.....	61
4.4 Оценка качества кода с помощью метрик.....	63
Заключение	64
Список использованных источников	66
Приложение А (справочное) Глоссарий	70
Приложение Б (обязательное) Листинг основных классов слоя отображения данных.....	72
Приложение В (обязательное) Листинг основных классов слоя бизнес-логики	88
Приложение Г (обязательное) Листинг основных классов слоя доступа к данным	99
Приложение Д (обязательное) Листинг класса выгрузки данных.....	102

Приложение Е (обязательное) Листинг классов добавления пользователя.....	108
Приложение Ж (рекомендуемое) Атрибуты сущностей концептуальной модели базы данных.....	112
Приложение И (рекомендуемое) Спецификации атрибутов таблиц логической модели базы данных.....	118

Введение

Развитие технических средств в настоящее время приводит к пересмотру отношения общества к образовательной системе. Появилась необходимость в создании более современных способов организации процесса обучения, в том числе с использованием последних достижений в области информационных технологий. В последнее время популярность приобрел термин «электронное обучение» (англ. Electronic Learning, e-Learning) – организация образовательной деятельности с использованием информационных технологий, технических средств, включая телекоммуникационные сети, обеспечивающие передачу по каналам связи необходимой информации, взаимодействие обучающихся и педагогических работников [1].

Появлению новых технических средств в образовательном процессе университетов способствуют несколько факторов, но среди них основными выделяют внешние: потребности студентов и их потенциальных работодателей, необходимость постоянного улучшения в связи с конкуренцией, улучшение технологий и т. д. [2]. Внедрение инструментов для онлайн образования позволит студентам более гибко выстраивать свой учебный процесс, без вреда совмещая его со своей внеучебной деятельностью. Это очень важно для обучающихся, которые совмещают получение образования в ВУЗе с работой, активистской, научной или волонтерской деятельностью.

У электронного обучения есть множество плюсов, среди них основными выделяют:

- доступ к образовательным ресурсам из любого места с выходом в глобальную сеть;
- качественные образовательные материалы, созданные командой специалистов;
- отсутствие необходимости приобретать физические образовательные материалы за деньги;

- возможность гибкого обучения путем разделения содержимого электронного курса на модули;
- адаптация процесса обучения студентом в соответствии со своими потребностями;
- возможность оперативного обновления базы знаний в соответствии с новой информацией;
- определение четких критериев оценивания знаний обучающегося [3].

Таким образом, целью моей выпускной квалификационной работы будет являться разработка приложения, которое обеспечит студентов и преподавателей возможностью взаимодействия друг с другом с использованием современных информационных технологий. Разрабатываемое приложение должно удовлетворять требования потенциальных пользователей в удобстве, мобильности и доступности информации, а также соответствовать современным тенденциям разработки площадок для организации дистанционного обучения.

1 Анализ предметной области и планирование разработки

1.1 Актуальность применения приложений трекеров задач в электронном обучении

1.1.1 Назначение и особенности электронного обучения

На сегодняшний день электронным обучением можно считать организацию учебного процесса с использованием различных цифровых средств доставки информации: дисковые носители, сети и т.д, что позволяет решить задачу организации обучения на расстоянии. Однако и для очной формы обучения e-Learinig может найти свое применение, повысив качество и эффективность классического учебного процесса.

К электронному обучению относятся следующие процессы:

- самостоятельную работу обучающихся с электронными учебными материалами, с использованием технических средств обучения;
- получение консультаций, рекомендаций, разъяснений у консультанта (преподавателя) путем дистанционного взаимодействия;
- создание в сети территориально распределённого сообщества обучающихся (социальных сетей, форумов), для совместной виртуальной учебной деятельности;
- своевременное обеспечение обучающихся учебной информацией, предоставление им электронных учебных материалов [1].

Таким образом, разрабатываемое программное средство должно предоставлять возможность обучающимся поддерживать связь с преподавателями для оценивания самостоятельного выполнения заданий, а также предоставлять обучающимся доступ к учебным материалам, причем независимо от наличия выхода в глобальную сеть.

Электронное обучение предоставляет возможности решать проблемы, возникающие при отсутствии в процессе образования информационных

технологий. Наиболее главной проблемой является отсутствие мотивации обучающихся во время учебного процесса.

Традиционное обучение не позволяет обучающимся отслеживать прогресс выполнения назначенных заданий в реальном времени, как правило, эта информация хранится у преподавателя в журнале. Конечно, студент может вести отслеживание прогресса самостоятельно, однако этот процесс сопровождается рядом проблем:

- обучающийся в процессе отслеживания может что-то пропустить, из-за чего его информация будет не совпадать с действительной;
- необходимо постоянно отслеживать, запоминать и вносить изменения в свою систему, что требует от студента концентрации и дисциплины;
- некоторые могут сталкиваться с трудностями при выборе метода отслеживания прогресса;
- иногда зафиксированный прогресс может быть безвозвратно утерян: студент может потерять блокнот с записями, случайно удалить таблицу или диск с данными может сломаться.

Отсутствие визуального индикатора, который мог бы информировать о достигнутом прогрессе и об остатке работы значительно ухудшает мотивацию студентов: они выполнили работу, однако не видят результата, который пока зафиксирован только у преподавателя [4]. Однако эта проблема решается при внедрении информационных технологий, которые могут самостоятельно отслеживать прогресс обучающихся и демонстрировать его в удобном виде.

1.1.2 Задачи отслеживания прогресса

В современных образовательных платформах автоматизированное отслеживание прогресса студентов выступает ключевым фактором повышения мотивации и академической успеваемости. Согласно исследованию Университета Иллинойса, учащиеся, регулярно получавшие обратную связь через автоматизированные системы, демонстрировали рост мотивации и

эффективности обучения на 33 % по сравнению с теми, кому предоставлялись традиционные методы контроля прогресса. Постоянное визуальное отображение процента выполненных модулей или изменение цвета индикатора позволяет студентам получать мгновенную информацию о собственных достижениях и зонах, требующих внимания, что способствует выработке установки на рост и укреплению веры в собственные силы [5].

Демонстрация степени выполнения через шкалу прогресса, значки или процентные индикаторы способствует формированию у обучающихся объективного представления о пройденном материале и оставшихся задачах, что устраняет когнитивную неопределённость и снижает уровень тревожности. При этом ощущение преодоления «контрольных точек» на пути освоения знаний содействует выработке внутренней мотивации продолжать образовательную деятельность и укрепляет чувство самодостаточности [6].

Помимо этого, автоматизированные системы снижают ощущение изоляции, характерное для дистанционного обучения. Ученики, получая мгновенную информацию о своём статусе, испытывают меньший стресс и большую уверенность в собственных силах, что положительно сказывается на их удовлетворённости образовательным процессом и готовности взаимодействовать в онлайн-сообществе [7].

Наконец, автоматизированное отслеживание прогресса снижает ощущение изолированности, характерное для дистанционного обучения: студенты, наблюдая за собственным прогрессом, испытывают меньший уровень тревожности и большую удовлетворённость образовательным процессом.

Таким образом, для удовлетворения выявленных потребностей обучающихся разрабатываемое в рамках выпускной квалификационной работы приложение для электронного обучения должно обеспечивать сохранение, обработку и предоставление информации о прогрессе обучающегося в удобном и легком для восприятия виде.

1.1.3 Аспекты применения трекеров задач в электронном образовании

Для обеспечения работы перечисленных процессов необходима платформа, которая бы предоставляла инструменты для организации комфортного для всех участников учебного пространства с возможностью создания, назначения и приема практических заданий с отслеживанием выполненного объема работ. С подобной задачей столкнулись разработчики программного обеспечения, которые, как правило, работают над одним продуктом в командах, у каждой из которых определена своя зона ответственности. Для удовлетворения запроса на систему управления и наведения порядка в рабочем процессе были созданы специальные приложения – таск-менеджеры или таск-трекеры.

Task Tracker (таск трекер) – это инструмент для управления задачами, который помогает организовать и отслеживать выполнение различных задач и проектов. Он используется как в личных целях, так и в профессиональной среде для повышения продуктивности и эффективности [8].

Таск-трекеры позволяют организовывать процесс работы с задачами, или в случае разрабатываемого в рамках выпускной квалификационной работы приложения – с практическими заданиями. Программа может быть представлена в различных форматах: десктопное приложение, веб-приложение или мобильное приложение, что идеально подходит для обозначенных выше требований для организации электронного обучения. Кроме организации учебного процесса таск-трекеры также могут решать и другие необходимые для онлайн образования задачи, такие как:

- контроль выполнения назначенных практических заданий путем задания сроков выполнения;
- работа с большим количеством данных, которые требуют вычисления (подсчет процента выполнения заданий и среднего балла в группе студентов);

- автоматическое формирование документации на основе результатов вычислений.

Таким образом, в ходе анализа предметной области были определены главные цели, которые необходимо решить в рамках выполняемой работы с помощью выделенных и проанализированных инструментов. Для определения основных требований к разрабатываемому приложению-трекеру прогресса будут проанализированы аналогичные системы.

1.2 Анализ предметной области разрабатываемого приложения

1.2.1 Назначение разрабатываемого приложения

По итогам анализа предметной области можно определить назначение разрабатываемого программного средства:

- приложение должно предоставить студентам и преподавателям удобную для использования платформу, на которой пользователи могут обмениваться информацией и поддерживать связь друг с другом;
- приложение должно предоставлять пользователям возможность получать доступ к информации без выхода в сеть;
- приложение должно предоставлять студентам доступ к информации, которая касается образовательного процесса: список необходимых для выполнения практических заданий с отображением прогресса выполнения в удобном для просмотра виде, текущее количество баллов, крайние сроки сдачи работ и т. д.;
- приложение должно предоставлять преподавателям возможность просматривать статистику студентов и предметов в удобном для просмотра виде: количество баллов всех студентов группы и их текущие оценки, прогресс выполнения заданий индивидуального студента группы, прогресс выполнения заданий в рамках созданного преподавателем предмета и т. д.;
- преподаватель должен иметь возможность генерировать отчеты на основе сформированной информации.

Задачи, сформулированные в результате проведения анализа предметной области в дальнейшем позволят определить бизнес-требования, которые будут учитываться при реализации разрабатываемого в рамках выпускной квалификационной работы приложения таск-трекера для планирования и отслеживания выполнения практических заданий студентами.

1.2.2 Структурная организация предметной области, действующие лица, их взаимоотношения

В результате определения назначения разрабатываемых программных средств можно выделить основных действующих лиц:

- преподаватель;
- студент;
- администратор.

Основной задачей преподавателя должно быть создание практических заданий, сгруппированных в учебные дисциплины – предметы. Преподаватель должен иметь возможность просматривать статистику и информацию о созданных предметах и студентах группы. Также преподаватель должен иметь возможность генерировать и выгружать отчеты.

Студент должен иметь возможность просматривать назначенные для выполнения задания, сгруппированные в предметы, с просмотром прогресса. Для выполнения задания студент должен иметь возможность отправлять выполненную работу преподавателю. Кроме того, студент может просмотреть текущие баллы и оценки по предметам.

Одним из основных пользователей можно выделить администратора, хотя он и не принимает участие в образовательном процессе, а скорее обеспечивает взаимодействие остальных пользователей. Его задачей является регистрация новых студентов в системе, создание студенческих групп и закрепление их за куратором.

Для указанных действующих лиц должен быть разработан функционал, отвечающий за выполнение обозначенных задач.

1.2.3 Анализ бизнес-требований

Для удовлетворения потребностей пользователей разрабатываемое в рамках выпускной квалификационной работы приложение для отслеживания прогресса выполнения практических заданий студентами должно предоставлять возможность пользователям выполнять следующие действия:

- приложение должно предоставлять возможность пользователям регистрироваться и объединяться в группы с помощью администратора, а также в подгруппы для совместного выполнения заданий;
- создаваемые практические задания должны содержать всю необходимую для выполнения работы информацию: само задание, требования, сроки сдачи и т. д.;
- преподаватель должен иметь возможность делегировать выполнение созданных заданий пользователям-студентам, назначение должно осуществляться различными способами: индивидуальному студенту, всей группе студентов, объединению нескольких студентов;
- студент должен отправлять выполненные задания на проверку, а преподаватель может проверить отправленную работу и либо отметить задание как выполненное, либо отправить студенту обратную связь с требованием внести изменения;
- отображение прогресса выполнения практических заданий – соотношения количества выполненных заданий ко всем назначенным заданиям как индивидуального студента, так и всей группы в удобном для восприятия виде;
- приложение должно хранить, обновлять и отображать информацию об оценках каждого пользователя, пользователь должен иметь возможность просматривать информацию о своей успеваемости: оценки за каждый предмет, средний балл за все время;
- преподаватель должен иметь возможность просматривать информацию об успеваемости целой группы студентов;

– приложение должно предоставлять возможность преподавателям выгружать отображаемые на странице пользователя данные в таблицу.

Выделенные бизнес-требования в дальнейшем будут преобразованы в функциональные требования, реализованы и протестированы в разрабатываемом в рамках выпускной квалификационной работы приложении.

1.3 Сравнительный анализ аналогичных приложений

1.3.1 Цель сравнительного анализа

Сравнительный анализ средств назначения и контроля выполнения заданий проводится с целью изучения существующих способов удовлетворения требований, поставленных в предыдущих пунктах. Кроме того, анализ позволит определить:

- сильные стороны подобных средств, которые обязательно должны присутствовать в разрабатываемом приложении;
- слабые стороны, от добавления которых следует воздержаться;
- функционал, который отсутствует у существующих средств и может сделать разрабатываемое приложение уникальным и конкурентоспособным.

Для анализа будут выбраны некоторые представители и будет произведено их сравнение по нескольким параметрам. Результат анализа позволит составить более точные требования к разрабатываемому приложению.

1.3.2 Обзор аналогов

В настоящее время существует множество различных средств для создания и отслеживания выполнения задач, каждый со своими уникальными инструментами и функционалом. Для анализа были выбраны следующие системы:

- Asana;
- Moodle;

- Trello;
- Meteor;
- Megaplan.

Выбранные представители являются наиболее популярными решениями с подробным описанием функционала и большим числом пользователей.

Asana – онлайн-сервис для командной работы над проектами, отслеживанием задач, выстраивания коммуникации в команде и обмена документами. Asana включает в себя многофункциональный набор инструментов, позволяющий вести проекты не отвлекаясь на сторонние сервисы.

Для работы с сервисом организации необходимо создать проект и назначить к нему исполнителей. Проект в Asana можно просматривать в виде списка задач или в виде канбан-доски. Канбан-доска позволяет организовать работу над проектом, сотрудники будут видеть дела, которые необходимо сделать и их приоритетность. Задачи в Asana – основной элемент оптимизации рабочего процесса в команде. Они позволяют поэтапно двигаться к достижению целей организации. К задачам можно добавить описание, выставить их приоритет, прикрепить файлы, назначить сроки выполнения и исполнителей, добавить теги [9].

Moodle (Modular Object-Oriented Dynamic Learning Environment) – система управления образовательными электронными курсами, также известная как система управления обучением Moodle или виртуальная обучающая среда Moodle. Представляет собой свободное веб-приложение, предоставляющее возможность создавать сайты для онлайн-обучения. Основными возможностями Moodle являются:

- создание онлайн-курсов;
- тестирование обучающихся;
- база знаний;
- статистика обучения для преподавателей [10].

Trello – Это облачный сервис для управления небольшими командами и проектами. Сервис работает по принципам методологии Kanban, которая является частью философии Agile. Каждая задача – отдельная карточка, и карточки перемещаются по доскам (колонкам) в зависимости от стадии работы.

Работа в Trello условно бесплатна. У бесплатной версии ограниченная функциональность, но её вполне хватает для управления небольшими проектами и командами. Платный доступ стоит 10 долларов в месяц за одного пользователя при оплате подписки на год. То есть управление небольшим отделом в десять человек в платной версии будет стоить 100 долларов в месяц [11].

METEOR – это российская система управления проектами и задачами, разработанная компанией «Автоматон». Изначально созданная для внутренних нужд, она эволюционировала в полнофункциональный таск-трекер, подходящий для команд любого размера.

Основные возможности:

- поддержка различных представлений, включая доски (по статусам, исполнителям, версиям), списки и диаграммы Ганта;
- поддержка методологий Scrum, Kanban и Waterfall, что делает систему универсальной для различных бизнес-процессов;
- расширенные возможности: тайм-трекинг, управление спринтами, настройку прав доступа, автоматизацию процессов с помощью триггеров и т. д.;
- мощный опенсорсный бэкенд OpenProject, что обеспечивает стабильность и масштабируемость системы;
- бесплатный тариф без ограничений на небольшое количество пользователей или проектов [12].

CRM Мегаплан – мощная система для управления бизнесом с расширенными возможностями в области проектов и заказов. Продукт позволяет автоматизировать внутренние и внешние процессы, наладить взаимоотношение с клиентами, а также управлять и координировать персонал.

Инструментарий направлен на организацию труда, оптимизацию рутинных обязанностей и автоматизацию бизнес-процессов.

Основной инструментарий направлен на проектную работу и постановку заданий, что позволяет полностью организовать бизнес-процессы предприятия с большим штатом [13].

1.3.3 Результаты сравнительного анализа

Для анализа приведённых программных средств были выбраны следующие критерии:

- автономная работа – возможность пользоваться функционалом приложения без выхода в сеть;
- создание заданий – возможность добавлять новое задание для дальнейшего назначения;
- назначение исполнителей – возможность приписывать созданное задание выбранному пользователю для дальнейшего выполнения;
- установка сроков сдачи работы – возможность назначать срок, в течение которого задание должно быть выполнено исполнителем;
- отправка задания на проверку – возможность для пользователя отправлять файл с выполненным заданием на проверку;
- просмотр статистики исполнителей – возможность всем пользователям просматривать оценки, полученные за выполнение заданий;
- отображение прогресса выполнения – возможность всем пользователям просматривать соотношение выполненных заданий ко всем назначенным заданиям;
- генерация документации – возможность генерировать и выгружать документацию на основе полученных данных.

По приведенным критериям был произведен сравнительный анализ рассмотренных программных средств. Результаты сравнительного анализа аналогов представлены в таблице 1.

Таблица 1 – Результаты сравнительного анализа программных средств

Функционал	Moodle	Asana	Trello	Megaplan	Meteor
Автономная работа	Есть возможность получить доступ к курсам с мобильного приложения	В мобильном приложении	Есть оффлайн режим с синхронизацией при входе в сеть	Коробочная версия для работы в локальной сети	Нет
Создание заданий	Возможность создавать курсы с темами, лекциями, файлами, тестами и т. д.	Есть	Есть	Есть	Есть
Назначение исполнителей	Нет	Есть	Есть	Есть	Есть
Установка сроков сдачи работы	Есть	Есть	Есть	Есть	Есть
Отправка задания на проверку	Есть	Есть возможность добавлять комментарии	Есть возможность добавлять комментарии	Есть возможность добавлять комментарии	Нет
Отображение прогресса выполнения	Нет	Есть возможность просматривать прогресс достижения цели	Есть	Нет	Есть прогресс времени выполнения
Просмотр статистики исполнителей	Только для преподавателей	Есть	Нет	Есть	Есть возможность отслеживать затраченное время и ресурсы
Генерация документации	Нет возможности выгружать отчеты, только просмотр	Нет	Нет	Есть	Нет

В результате проделанного анализа можно сделать следующие выводы. Все приложения обладают функционалом работы с заданиями: создание, назначение и установка дедлайнов. Отправлять выполненные задания на проверку напрямую нельзя, но почти во всех средствах есть возможность добавлять комментарии с выполненными заданиями. Остальной приведенный функционал частично присутствует в большинстве приложений, возможностью генерации документов обладает только Megaplan. Представленные программные средства являются платными, некоторые предоставляют бесплатный доступ к ограниченному функционалу.

Проведенный анализ показал, что разрабатываемое в рамках выпускной квалификационной работы приложение будет являться конкурентноспособным за счет реализации функций, которые лишь частично представлены в приведенных трекерах задач: отображение прогресса, просмотр статистики и генерация документации, причем последнее представлено только в одном решении. Также были определены главные функции, которые обязательно должны быть реализованы в разрабатываемой системе: создание заданий, назначение исполнителей и установка дедлайнов. Результаты проведенного анализа будут применены при разработке приложения.

1.4 Планирование и оценка затрат процесса разработки

Прежде чем приступить к реализации программного продукта, проведем планирование разработки и оценку бюджета. Планирование производится с целью организации процесса разработки, при котором работа будет успешно завершена в указанный срок. Для этого разделим работу на этапы, для каждого из которых укажем продолжительность в днях. Планирование и оценка бюджета были выполнены в программе Microsoft Office Project – программное обеспечение для управления проектами. Оно помогает планировать, контролировать и управлять ресурсами и задачами для успешного выполнения проектов [14].

Разработка приложения – трекера практических заданий была разделена на несколько этапов, у каждого из которых были выделены подзадачи, на основе выделенных пунктов был составлен план в project. В качестве начала работы была выбрана дата 8 апреля 2025 года. В результате выставления длительности каждого этапа работы дата окончания проекта назначена на когда-то там. Проект с планом работ представлен на рисунке 1.

Название задачи	Длительность	Начало	Окончание	Предшественники	Названия ресурсов
Разработка приложения	42 дней	Вт 08.04.25	Ср 04.06.25		
Начало проекта	0 дней	Вт 08.04.25	Вт 08.04.25		
Планирование и подготовка	9 дней	Вт 08.04.25	Пт 18.04.25		
Постановка цели на разработку	1 день	Вт 08.04.25	Вт 08.04.25		Руководитель проекта
Анализ предметной области	5 дней	Ср 09.04.25	Вт 15.04.25	4	Руководитель проекта
Анализ требований к разрабатываемым программным средствам	1 день	Ср 16.04.25	Ср 16.04.25	5	Руководитель проекта
Анализ аналогов	2 дней	Чт 17.04.25	Пт 18.04.25	6	Руководитель проекта
Планирование завершено	0 дней	Пт 18.04.25	Пт 18.04.25	7	Руководитель проекта
Проектирование программных средств	11 дней	Пн 21.04.25	Пн 05.05.25		
Анализ функциональных требований	1 день	Пн 21.04.25	Пн 21.04.25	8	Руководитель проекта
Разработка концептуальной модели базы данных	2 дней	Вт 22.04.25	Ср 23.04.25	10	Руководитель проекта
Разработка логической модели базы данных	2 дней	Чт 24.04.25	Пт 25.04.25	11	Руководитель проекта
Анализ требований к интерфейсу	1 день	Пн 28.04.25	Пн 28.04.25	12	Руководитель проекта
Анализ требований к архитектуре приложения	1 день	Вт 29.04.25	Вт 29.04.25	13	Руководитель проекта
Выбор технологий и средств разработки	1 день	Ср 30.04.25	Ср 30.04.25	14	Руководитель проекта
Планирование разработки и оценка затрат	1 день	Чт 01.05.25	Чт 01.05.25	15	Руководитель проекта
Составление отчета о проектировании программных средств	2 дней	Пт 02.05.25	Пн 05.05.25	16;10;11;12;13;14;15	Руководитель проекта
Проектирование программных средств завершено	0 дней	Пн 05.05.25	Пн 05.05.25	17	
Реализация приложения	18 дней	Вт 06.05.25	Чт 29.05.25		
Реализация базы данных	7 дней	Вт 06.05.25	Ср 14.05.25	18	Руководитель проекта
Разработка приложения	14 дней	Чт 08.05.25	Вт 27.05.25	18ОН+2 дней	Руководитель проекта
Реализация интерфейса	3 дней	Вт 13.05.25	Чт 15.05.25	18ОН+5 дней	Руководитель проекта
Составление отчета о разработке приложения	2 дней	Ср 28.05.25	Чт 29.05.25	22;20;21	Руководитель проекта
Реализация приложения завершена	0 дней	Чт 29.05.25	Чт 29.05.25	20;21;23	
Тестирование программных средств	3 дней	Пт 30.05.25	Вт 03.06.25		
Выбор методов тестирования и оценки качества кода	1 день	Пт 30.05.25	Пт 30.05.25	24	Руководитель проекта
Проведение функционального тестирования	1 день	Пн 02.06.25	Пн 02.06.25	26	Руководитель проекта
Проведение попарного тестирования	1 день	Пн 02.06.25	Пн 02.06.25	26	Руководитель проекта
Анализ метрик кода	1 день	Пн 02.06.25	Пн 02.06.25	26	Руководитель проекта
Составление отчета о результатах тестирования	1 день	Вт 03.06.25	Вт 03.06.25	29;27;28	Руководитель проекта
Тестирование программных средств завершено	0 дней	Вт 03.06.25	Вт 03.06.25	30	
Составление итогового отчета	1 день	Ср 04.06.25	Ср 04.06.25	31	
Проект завершен	0 дней	Ср 04.06.25	Ср 04.06.25	32	

Рисунок 1 – Этапы разработки приложения

Для более наглядной демонстрации этапов работы была выбрана диаграмма Ганта – визуальный инструмент управления проектами, который отображает задачи по времени в виде горизонтальных полос. Она помогает планировать и контролировать выполнение работ, показывая начало и конец задачи, их продолжительность и взаимосвязи [15]. По результатам планирования разработки приложения для выпускной квалификационной работы было получено время, необходимое на реализацию программного

продукта: 408 часов. По результатам оценки бюджета была получена стоимость проекта: 127500 рублей.

Полученная в результате планирования диаграмма Ганта представлена на рисунке 2.

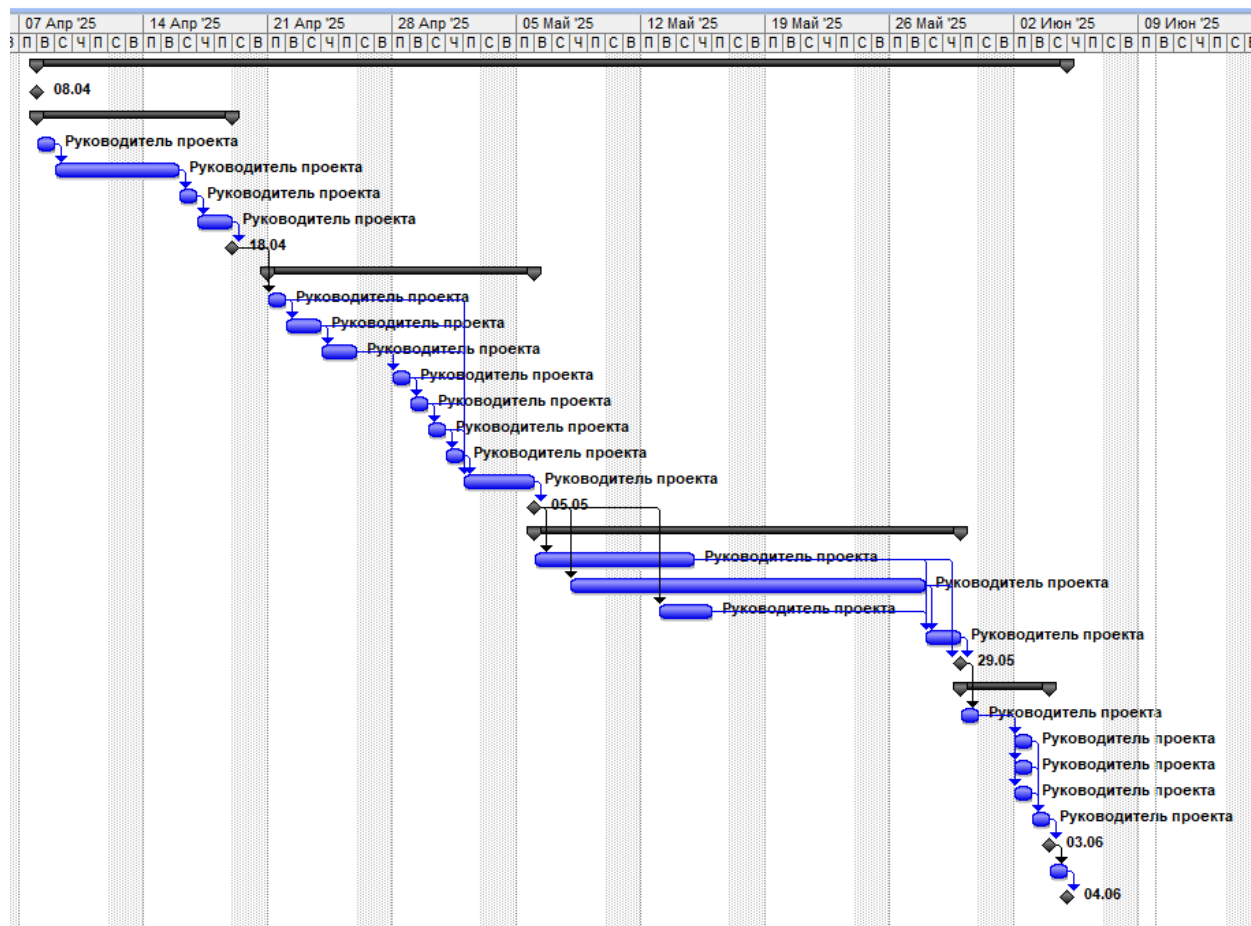


Рисунок 2 – Диаграмма Ганта для плана разработки приложения трекера заданий

В результате планирования разработки приложения был составлен план работы над проектом. Полученное время работы позволит завершить разработку приложения в выделенные сроки.

2 Анализ требований и постановка задач на разработку

2.1 Анализ функциональных требований

В результате проведенного анализа предметной области были определены основные пользователи разрабатываемого в рамках выпускной квалификационной работы приложения, а также бизнес-требования. На их основе были сформированы и поделены по пользовательским ролям следующие функциональные требования:

Преподаватели:

- создание учебных предметов;
- создание практических заданий;
- назначение выполнения практических заданий предмета;
- приём практических заданий;
- просмотр прогресса выполнения заданий созданного предмета;
- просмотр прогресса выполнения назначенных заданий

индивидуального студента;

- просмотр успеваемости студентов группы;
- генерация документов.

Студенты:

- просмотр назначенных практических заданий;
- отправка выполненной работы преподавателю;
- просмотр прогресса выполнения заданий предмета;
- просмотр текущей успеваемости.

Администраторы:

- регистрация пользователей;
- создание учебных групп;
- просмотр всей информации пользователя.

Диаграмма вариантов использования для пользователя с ролью «Преподаватель» представлена на рисунке 3.



Рисунок 3 – Диаграмма вариантов использования приложения для преподавателя

В таблице 2 приведена спецификация варианта использования «Создать учебный предмет» для пользователя с ролью «Преподаватель».

Таблица 2 – Сценарий выполнения варианта использования «Создать учебный предмет»

Наименование	Создать учебный предмет
ID	1
Краткое описание	Преподаватель создает новый учебный предмет
Действующие лица	Преподаватель
Предусловие	Преподаватель вошел в систему
Основной поток	1. Преподаватель указывает название создаваемого предмета 2. Преподаватель выбирает тип оценивания создаваемого предмета (Зачёт/Зачёт с оценкой/Экзамен) 3. Преподаватель указывает семестр, на котором пользователи будут изучать создаваемый предмет 4. Преподаватель указывает длительность изучения предмета в семестрах
Постусловие	Преподаватель успешно создал новый предмет с указанными параметрами

Диаграмма вариантов использования для пользователя с ролью «Студент» представлена на рисунке 4.



Рисунок 4 – Диаграмма вариантов использования приложения для студента

В таблице 3 приведена спецификация варианта использования «Отправить выполненную работу на проверку» для пользователя с ролью «Обучающийся».

Таблица 3 – Сценарий выполнения варианта использования «Отправить выполненную работу на проверку»

Наименование	Отправить выполненную работу на проверку
ID	2
Краткое описание	Пользователь отправляет выполненное задание на проверку преподавателю
Действующие лица	Обучающиеся
Предусловие	Пользователю было назначено задание
Основной поток	<ol style="list-style-type: none"> 1. Пользователь выбирает нужный предмет и нужное задание предмета 2. Пользователь открывает форму отправки ответа 3. Пользователь прикрепляет файл с выполненной работой 4. Пользователь оставляет комментарии при необходимости 4. Пользователь подтверждает отправку задания
Постусловие	Выполненная работа была отправлена преподавателю

Диаграмма вариантов использования для пользователя с ролью «Администратор» представлена на рисунке 5.



Рисунок 5 – Диаграмма вариантов использования приложения для администратора

В таблице 4 приведена спецификация варианта использования «Создать учебную группу» для пользователя с ролью «Администратор».

Таблица 4 – Сценарий выполнения варианта использования «Создать учебную группу»

Наименование	Создать учебную группу
ID	3
Краткое описание	Администратор создает новую учебную группу
Действующие лица	Администратор
Предусловие	Администратор вошел в систему
Основной поток	1. Администратор открывает форму создания новой учебной группы 2. Администратор указывает необходимую информацию 3. Администратор выбирает куратора 4. Администратор подтверждает создание учебной группы
Постусловие	Новая учебная группа была добавлена в систему

В результате проведения анализа функциональных требований была разработана диаграмма вариантов использования. Выявленные варианты использования будут реализованы при разработке приложения таск-трекера для планирования и контроля выполнения практических заданий студентами.

2.2 Анализ требований к данным

Концептуальная модель – это модель базы данных, которая отражает связи между сущностями и хранящиеся в них данные для дальнейшей разработки полноценной базы данных [16].

Разработанная концептуальная модель базы данных представлена на рисунке 6.

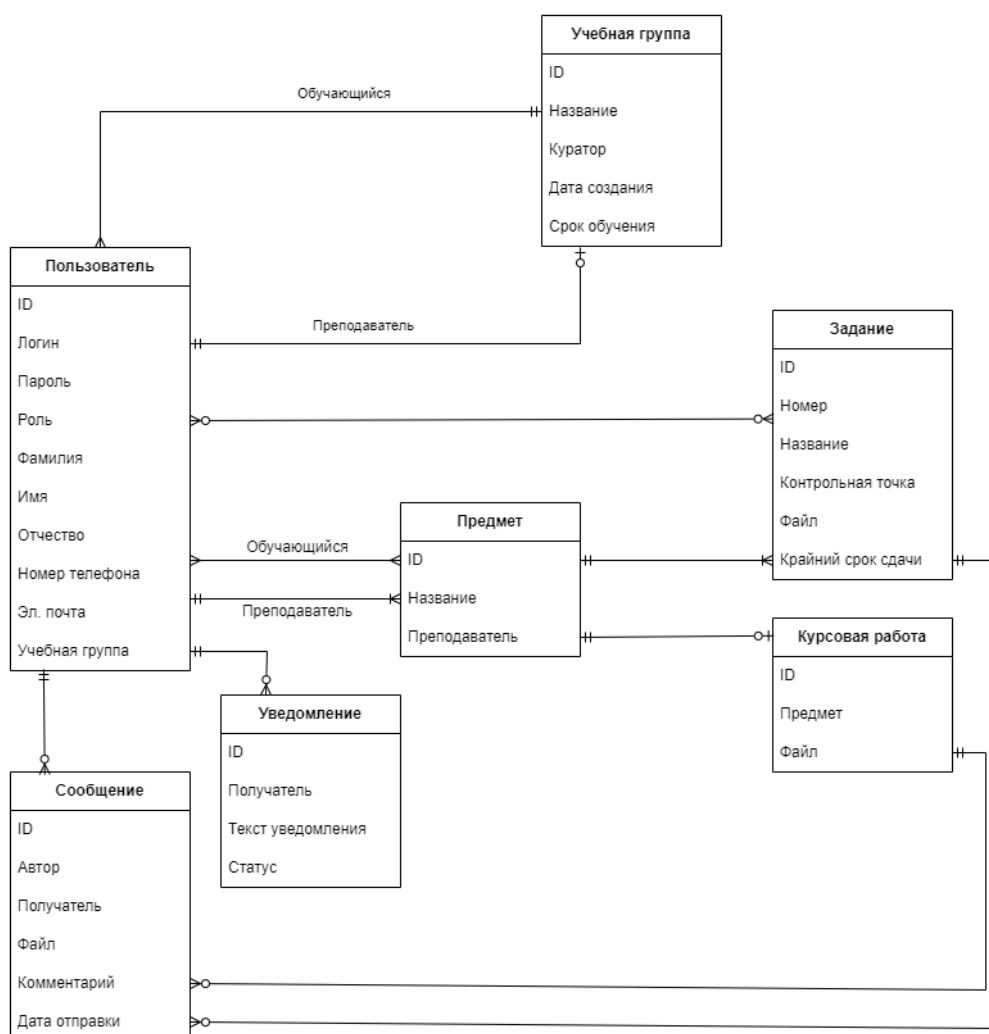


Рисунок 6 – Концептуальная модель базы данных для разрабатываемого приложения трекера заданий

Для разработки концептуальной модели базы данных были выделены основные сущности, полученные в результате проведения анализа предметной области. Атрибуты отображенных в модели сущностей представлены в таблицах Ж1 – Ж7 в приложении Ж. На основе выявленных сущностей была разработана концептуальная модель базы данных, содержащая выявленные сущности, а также связи между ними.

Описание сущностей разрабатываемой базы данных представлено в таблице 5.

Таблица 5 – Описание сущностей

Название сущности	Назначение
Пользователь	Содержит данные о пользователях
Учебная группа	Содержит данные об учебных группах
Предмет	Содержит данные об учебном предмете
Задание	Содержит данные о практическом задании
Курсовая работа	Содержит данные о курсовой работе
Уведомление	Содержит уведомления пользователя
Сообщение	Содержит сообщения пользователя

Спецификация связей сущностей разработанной концептуальной модели базы данных представлена в таблице 6.

Таблица 6 – Спецификация связей сущностей

№	Тип сущности 1	Тип сущности 2	Семантика связи	Мощность	Обязательность
1	Пользователь	Задание	Нескольким обучающимся может быть назначено несколько заданий	N:M	не обязательная
2	Преподаватель	Группа	У преподавателя может быть только одна группа	1:0,1	не обязательная
3	Группа	Обучающийся	Несколько обучающихся должны быть только в одной группе	1:M	обязательная
4	Пользователь	Курсовая работа	Сколько угодно обучающихся могут иметь сколько угодно курсовых работ	0,N:0,M	не обязательная
5	Преподаватель	Предмет	Хотя бы один предмет должен быть только у одного преподавателя	1:1,M	обязательная
6	Обучающийся	Предмет	Нескольким обучающимся должно быть назначено выполнение заданий по нескольким предметам	N:M	обязательная

Продолжение таблицы 6

№	Тип сущности 1	Тип сущности 2	Семантика связи	Мощность	Обязательность
7	Предмет	Задание	У одного предмета должно быть одно или несколько заданий	1:1,М	обязательная
8	Предмет	Курсовая работа	У одного предмета может быть только одна курсовая работа	1:0,1	не обязательная
9	Пользователь	Сообщение	Одному пользователю может быть отправлено несколько сообщений	1:0,М	не обязательная
10	Пользователь	Уведомление	Одному пользователю может быть отправлено несколько уведомлений	1:0,М	не обязательная
11	Задание	Сообщение	Несколько сообщений могут содержать решение только одного задания	1:0,М	не обязательная
12	Курсовая работа	Сообщение	Несколько сообщений могут содержать решение только одной курсовой работы	1:0,М	не обязательная

На основе разработанной концептуальной модели базы данных будет разработана логическая и физическая модели.

2.3 Анализ требований к пользовательскому интерфейсу

Интерфейс разрабатываемого приложения должен предоставлять пользователю удобное отображение необходимых данных с возможностью фильтрации, а также должен обеспечивать доступ к управлению данными, а именно:

- изменению данных;
- удалению данных;
- добавлению данных;
- выгрузке данных.

Отображаемая страница должна менять свое содержимое в зависимости от того, какую страницу пользователь выбрал в навигационной панели. Отображаемая страница может содержать следующие элементы:

- таблица, которая предоставляет пользователю запрашиваемые данные в удобном формате, кроме необходимых пользователю данных таблица также может содержать кнопки для взаимодействия с существующими данными (изменение/удаление/отправка ответа и т.д.);

- выпадающие списки, которые позволят пользователю фильтровать данные по необходимым критериями (пользователь, группа, предмет и т. д.);
- кнопки для добавления новых данных (заданий, предметов, курсовых работ).

Разработанная схема пользовательского окна представлена на рисунке 7.

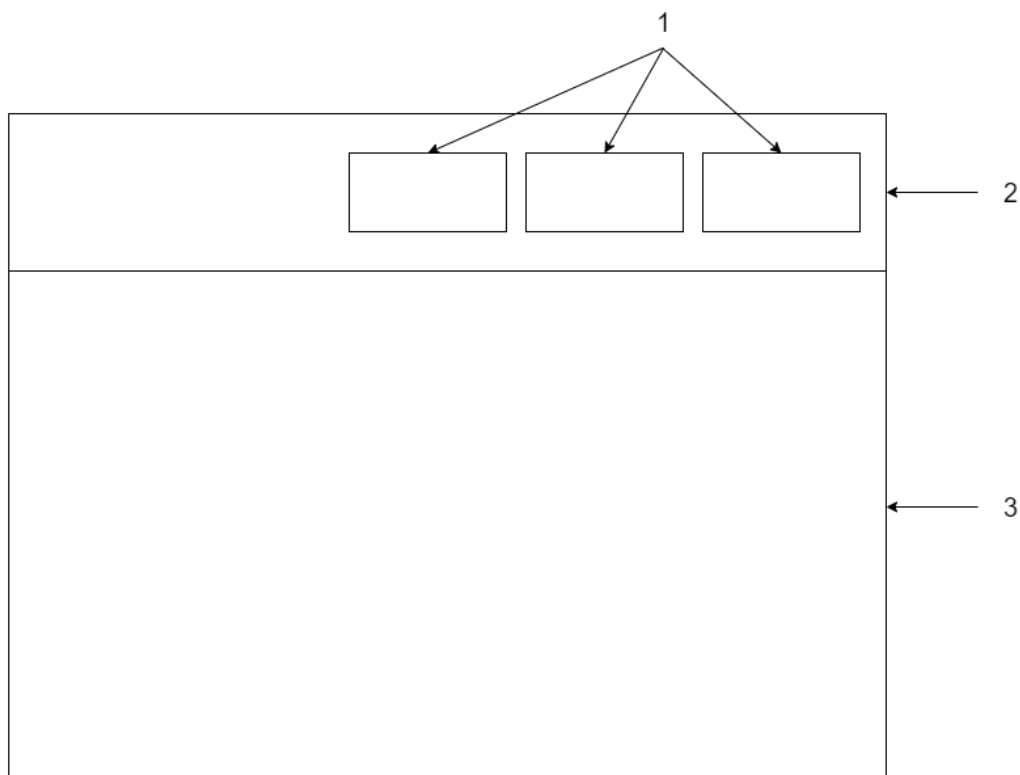


Рисунок 7 – Схема пользовательского окна

На схеме кнопками обозначены следующие элементы:

- 1 – навигационные кнопки, с помощью которых осуществляется переключение отображаемых страниц;
- 2 – навигационная панель, содержащая в себе навигационные кнопки;
- 3 – отображаемая страница, содержащая информацию о странице, которую открыл пользователь.

Разработанная схема пользовательского окна и определенные элементы отображаемых страниц в дальнейшем будут применены при разработке пользовательского интерфейса.

2.4 Анализ требований к архитектуре приложения

Для разработки приложения-трекера практических заданий была выбрана многоуровневая архитектура.

Чёткое разделение ответственности между слоями позволяет изолировать изменения в одном модуле и минимизировать побочные эффекты при развитии проекта. Плюсом можно отметить масштабируемость: за счёт распределения кеша и логики обработки на отдельные слои система может эффективно использовать многопоточность и расширяться без переработки всего приложения. Наличие обособленных слоёв снижает связанность компонентов, что ускоряет отладку и тестирование, а также облегчает перенос частей системы в новые проекты. За счёт чёткого структурирования кода по слоям разработчик легче ориентируется в проекте, быстрее внедряет новые функции и может поэтапно наращивать функционал без риска запутаться в едином монолитном коде [17].

Схема многоуровневой архитектуры представлена на рисунке 8.

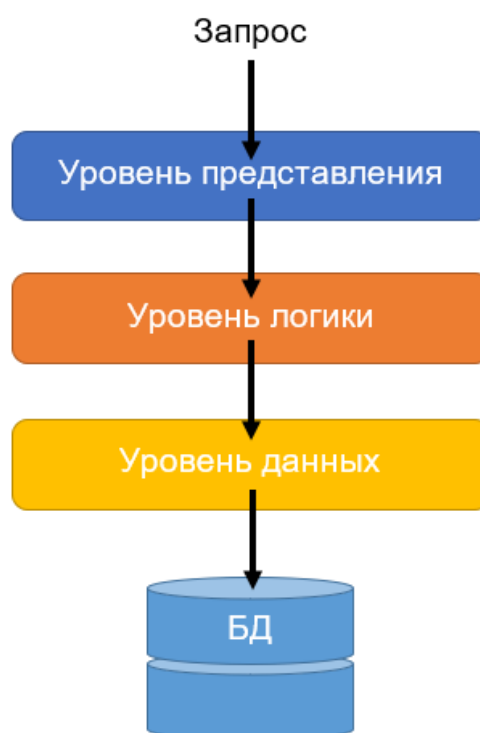


Рисунок 8 – Схема многоуровневой архитектуры

Основные элементы схемы:

- 1 – уровень представления: отображение данных пользователю;
- 2 – уровень логики: обработка запросов пользователя;
- 3 – уровень данных: получение данных из базы данных;
- 4 – бд: хранит в себе необходимые для обработки данные.

Требования к архитектуре, выявленные в результате проведения анализа будут применены при разработке приложения для выпускной квалификационной работы.

2.5 Обоснование выбора технологий и средств разработки

Для разработки настольного приложения для управления практическими заданиями и отслеживания прогресса выполнения был выбран язык программирования C# с использованием технологии Windows Forms для создания пользовательского интерфейса. Windows Forms предоставляет разработчику быстрый и доступный способ создания интерфейса с помощью визуального конструктора, позволяющего строить интерфейс путем простого перетаскивания элементов на форму, что упрощает разработку для тех, кто мало знаком с дизайном и разработкой пользовательского интерфейса. Windows Forms обеспечивает стабильную структуру приложения: разделение логики обработки данных и их отображения значительно упрощает поддерживаемость и расширение функционала. Использование платформы .NET и языка C# предоставляет преимущества в виде строгой типизации, автоматического сборщика мусора и т.д. [18].

Для хранения обрабатываемых данных была выбрана SQL Server Management Studio. SSMS предоставляет интегрированную среду для создания и управления объектами баз данных (таблицами, представлениями, процедурами, триггерами), что позволяет разработчику оперативно изменять схему данных и выполнять операции вставки, удаления или обновления без прибегания к сторонним утилитам. Во-вторых, мощный редактор запросов с подсветкой синтаксиса, автодополнением и возможностью просмотра плана

выполнения упрощает формирование сложных SQL-запросов и их оптимизацию, а встроенные механизмы отладки (точки останова, пошаговое выполнение) повышают надёжность взаимодействия клиентской части приложения с базой.

Третьим важным аспектом является наличие инструментов мониторинга и профилирования (SQL Server Profiler, Performance Monitor, Database Engine Tuning Advisor), которые помогают выявлять «узкие места» в производительности и своевременно настраивать индексы и параметры сервера, что особенно актуально для многопоточных десктопных систем с высокими требованиями к скорости обработки данных [19].

Помимо описанных средств при разработке также были использованы инструменты и библиотеки, перечисленные ниже.

Supabase – платформа Backend-as-a-Service (BaaS) с открытым исходным кодом, которая предоставляет полный набор инструментов для разработки приложений, включая аутентификацию, базу данных PostgreSQL, хранилище файлов, Realtime API и Функции [20]. Работа с удаленными данными в приложении осуществляется с помощью библиотеки supabase – единый NuGet-пакет, предоставляющий средства для взаимодействия с Supabase Storage (загрузка, скачивание и генерация подписанных URL) [21].

EPPlus – библиотека .NET, предназначенная для работы с файлами Excel в формате Office Open XML (.xlsx). Она позволяет разработчикам создавать, читать, редактировать и управлять электронными таблицами в .NET [22].

Draw.io – бесплатный онлайн-сервис, который помогает создавать блок-схемы, прототипы, инфографику и диаграммы любого вида. Сервисом пользуются разные IT-специалисты – от бизнес-аналитиков до разработчиков и дизайнеров. Это связано с тем, что у Draw.io много функций: с его помощью можно продумывать даже планировку помещений. Хотя основное назначение – то всё-таки визуализация данных, моделирование процессов и системное проектирование [23].

Перечисленные программные средства и инструменты будут использованы при разработке приложения – трекера студенческих заданий.

3 Разработка приложения

3.1 Разработка базы данных

На основе разработанной концептуальной модели базы данных была разработана логическая модель. При реализации концептуальной модели базы данных для разрабатываемого в рамках выпускной квалификационной работы приложения были реализованы определенные ранее сущности, а также были добавлены вспомогательные, осуществляющие связь между некоторыми сущностями. Диаграмма разработанной логической базы данных представлена на рисунке 9.

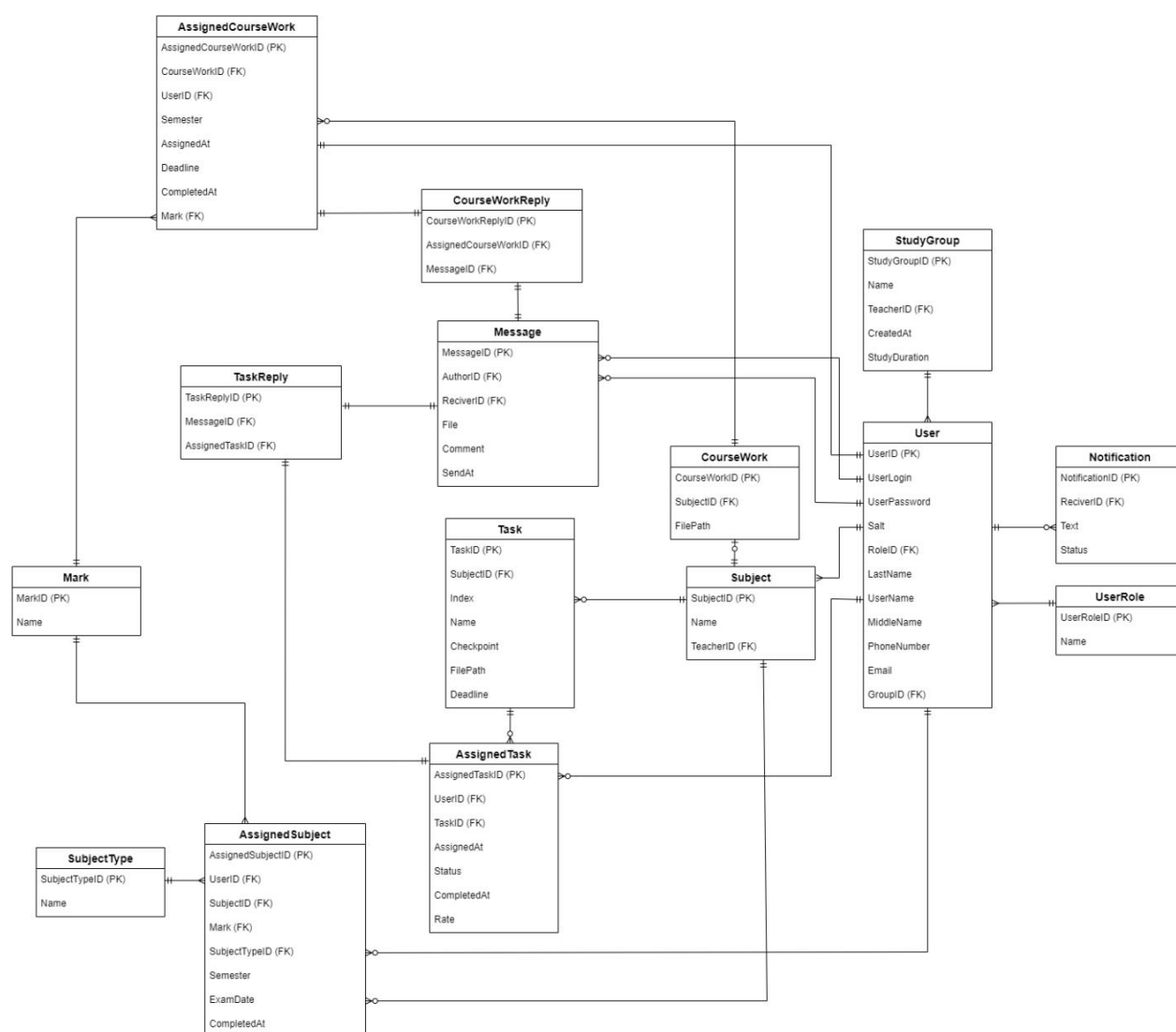


Рисунок 9 – Логическая модель базы данных разрабатываемого приложения трекера заданий

На основе разработанной логической базы данных была разработана физическая модель базы данных с использованием SQL Server Management Studio. Диаграмма разработанной базы данных представлена на рисунке 10.

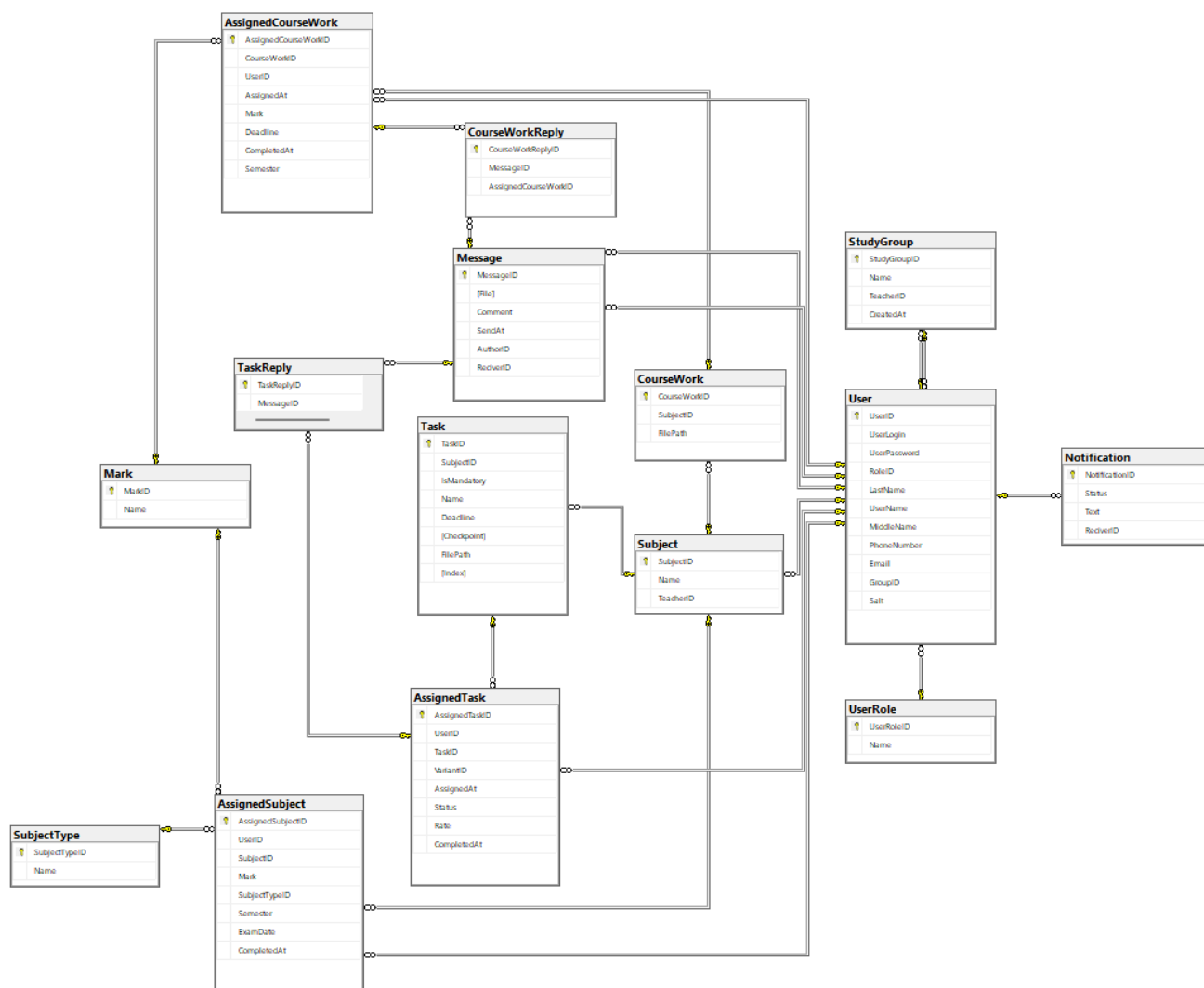


Рисунок 10 – Физическая модель базы данных разрабатываемого приложения трекера заданий

Спецификация таблиц разработанной базы данных приведена в таблице 7.

Таблица 7 – Спецификация таблиц базы данных

Название таблицы	Назначение таблицы
AssignedCourseWork	Хранит информацию о курсовой работе, назначенной студенту
AssignedSubject	Хранит информацию об учебном предмете, который изучает студент
AssignedTask	Хранит информацию о практических заданиях, выполнение которые назначены студенту
CourseWork	Хранит информацию о курсовой работе учебного предмета
CourseWorkReply	Хранит сообщение с отправленной на проверку курсовой работой

Продолжение таблицы 7

Название таблицы	Назначение таблицы
Mark	Хранит оценки, выставяемые пользователям
Message	Хранит отправленное пользователю сообщение
Notification	Хранит отправленное пользователю уведомление
StudyGroup	Хранит информацию об учебных группах
Subject	Хранит информацию о учебном предмете
SubjectType	Хранит способы аттестации по учебным предметам
Task	Хранит информацию о практическом задании
TaskReply	Хранит сообщение с отправленной на проверку практической работой
User	Хранит информацию о пользователе
UserRole	Хранит роли пользователей

Для разработанной физической модели базы данных были описаны атрибуты таблиц, представленные в таблицах И1 – И15 в приложении И.

3.2 Разработка пользовательского интерфейса

Для приложения был создан пользовательский интерфейс с применением технологии Windows Forms. Навигационная схема страниц для пользователя с ролью «Преподаватель» представлена на рисунке 11.

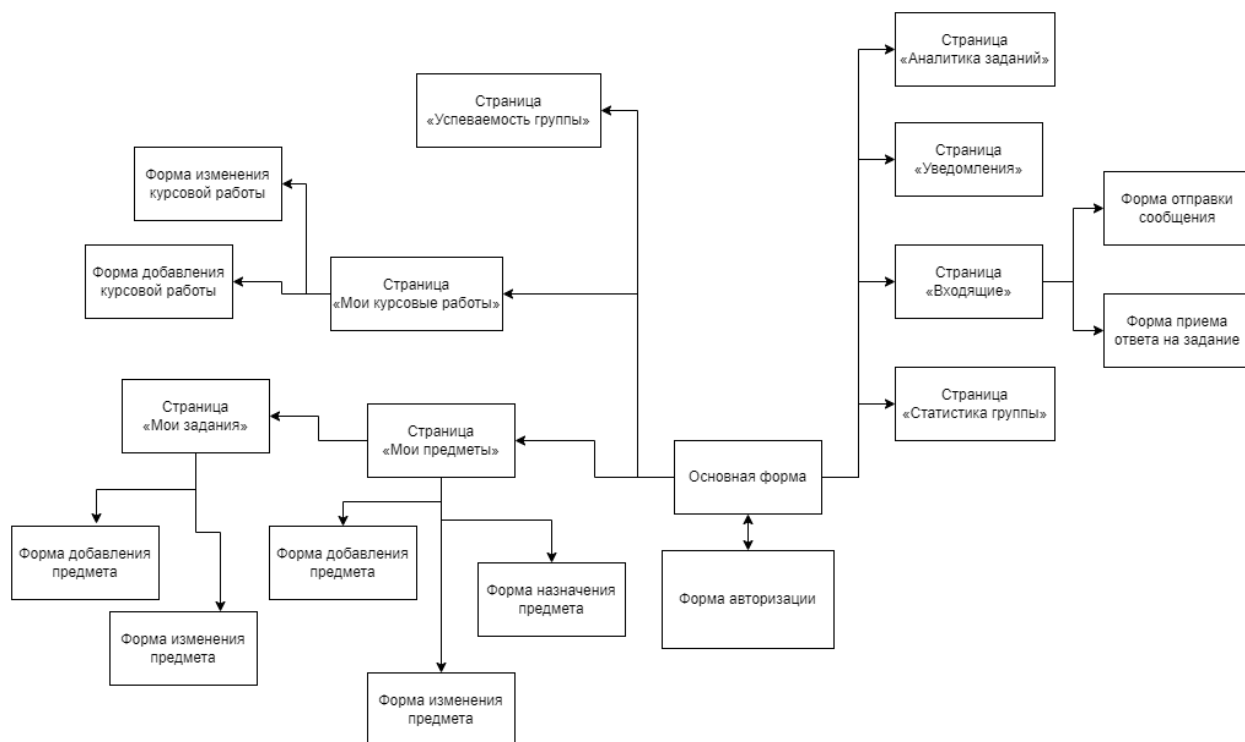


Рисунок 11 – Навигационная схема страниц для преподавателя

Навигационная схема страниц для пользователя с ролью «Студент» представлена на рисунке 12.



Рисунок 12 – Навигационная схема страниц для студента

Навигационная схема страниц для пользователя с ролью «Администратор» представлена на рисунке 13.

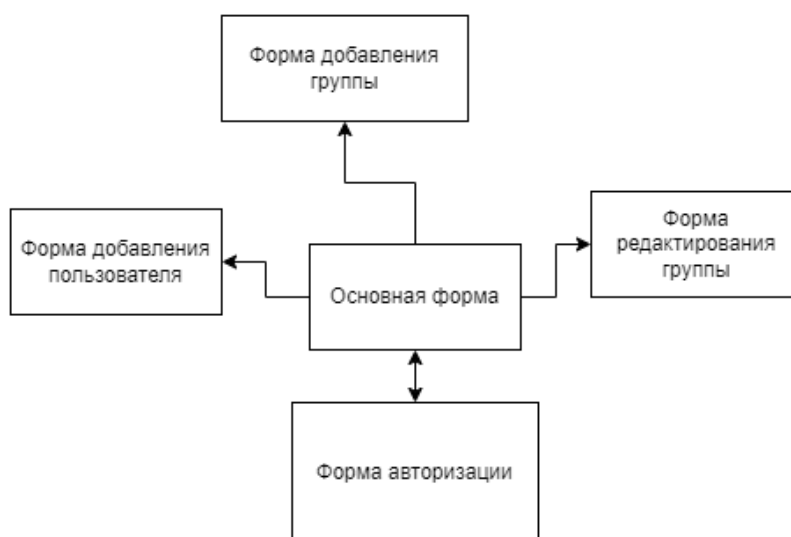


Рисунок 13 – Навигационная схема страниц для администратора

На рисунке 14 представлена демонстрация разработанного пользовательского интерфейса, представляющее собой окно с полями, в которые необходимо ввести данные учетной записи для входа в аккаунт зарегистрированного пользователя.

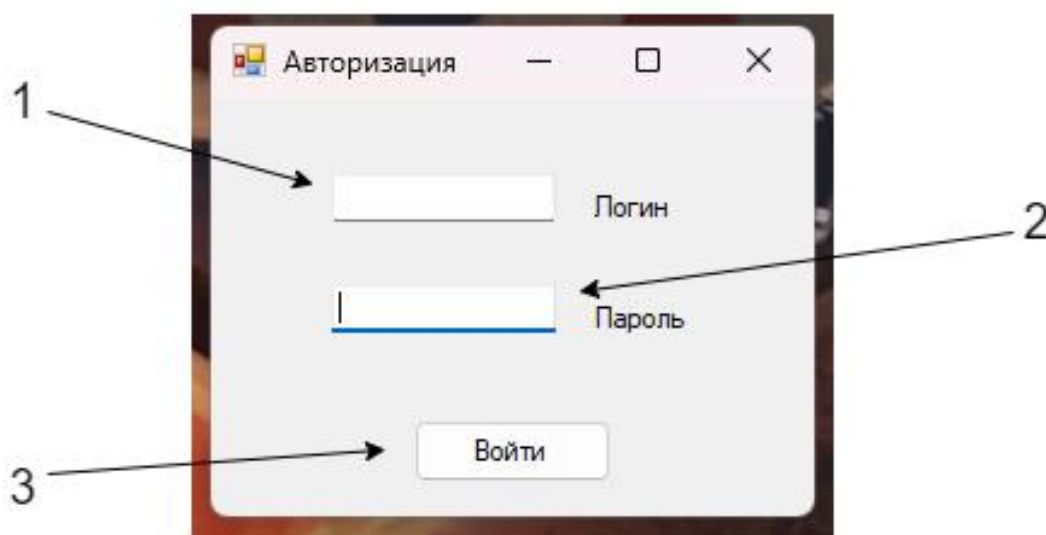


Рисунок 14 – Интерфейс авторизации

Цифрами на рисунке обозначены:

- 1 – поле для ввода логина;
- 2 – поле для ввода пароля;
- 3 – кнопка для авторизации пользователя.

При нажатии на кнопку «Войти» система проверит введенные данные на корректность. Если введенные данные корректны, пользователь войдет в свой аккаунт, иначе будет показано сообщение об ошибке. При успешной авторизации пользователь увидит приветственное сообщение и перейдет на начальную страницу в пользовательском окне.

Пользовательское окно имеет навигационное меню, которое позволяет пользователям переключаться между страницами, а также осуществлять выход из аккаунта.

На странице «Успеваемость группы» преподаватель может выбрать интересующего студента и семестр для просмотра успеваемости за соответствующий период, студент может просмотреть только свою успеваемость. В таблице представлена информация о предметах, а именно способ аттестации и выставленная пользователю оценка. Интерфейс начальной страницы представлен на рисунке 15.

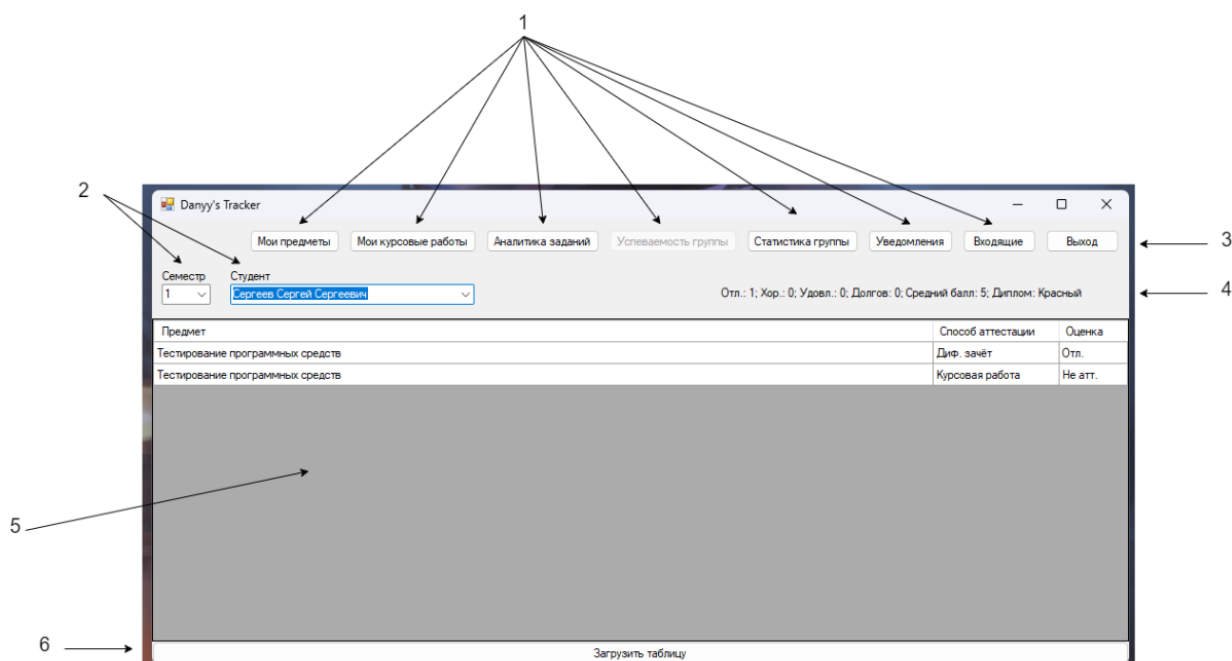


Рисунок 15 – Интерфейс страницы «Успеваемость группы»

Цифрами на рисунке обозначены:

- 1 – навигационные кнопки;
- 2 – фильтры отображаемых данных;
- 3 – кнопка выхода из аккаунта;
- 4 – статистика по успеваемости студента;
- 5 – отображаемые данные;
- 6 – кнопка для выгрузки отображаемых данных.

При нажатии на кнопку «Загрузить таблицу» приложение сформирует и загрузит на компьютер преподавателя таблицу формата .xlsx, содержащую отображаемые в приложении данные. Кнопка доступна только для преподавателя. Также страница выводит статистику по успеваемости пользователя: количество оценок, количество долгов, средний балл и диплом, на получение которого студент может претендовать. При выборе фильтра программа сначала фильтрует данные, а затем на основе отфильтрованных данных генерирует статистику. Статистику может просматривать как сам студент, так и преподаватель, причем всех студентов группы.

На рисунке 16 представлен интерфейс страницы «Мои предметы».

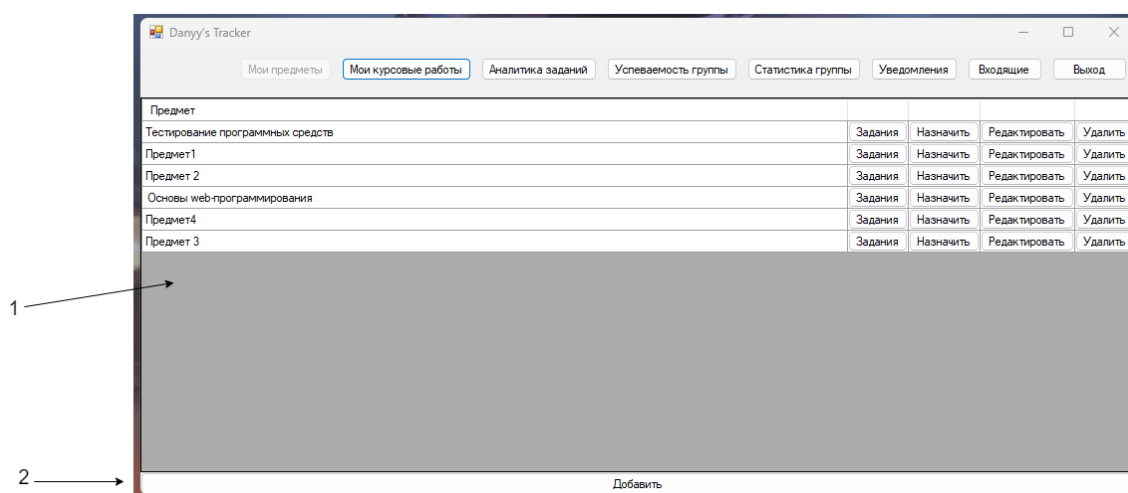


Рисунок 16 – Интерфейс страницы «Мои предметы»

Цифрам на рисунке обозначены:

- 1 – отображаемые созданные пользователем данные;
- 2 – кнопка для добавления новых данных.

Страница «Мои предметы» выводит преподавателю все созданные им предметы с возможностью их изменения.

Страницы управления заданиями представлена на рисунке 17

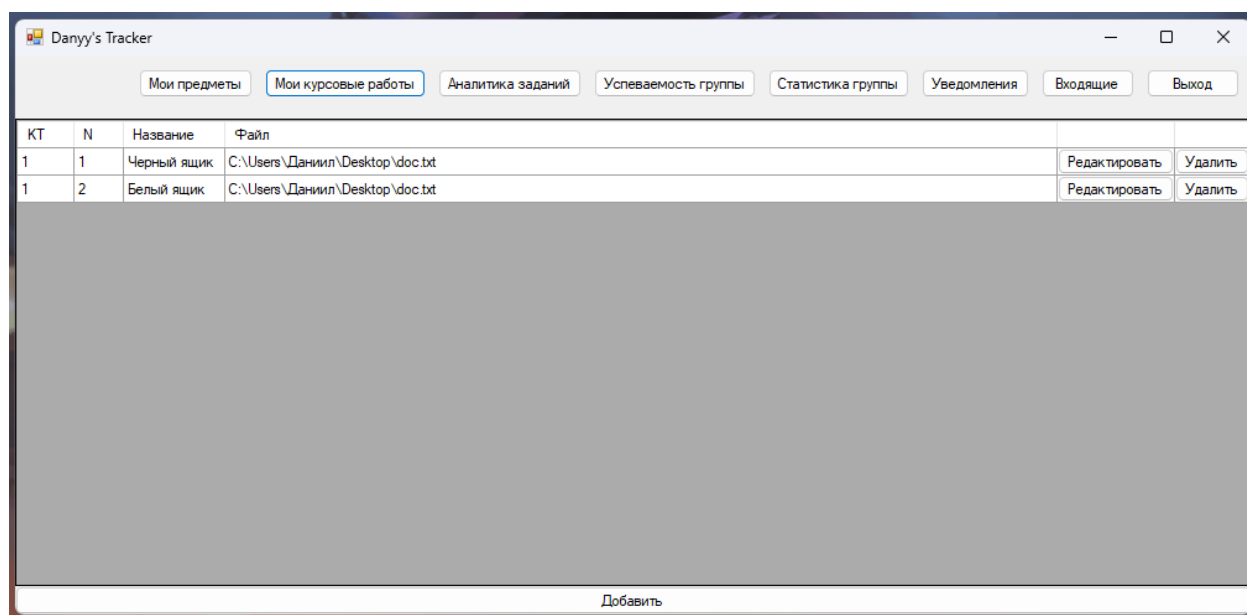


Рисунок 17 – Интерфейс страницы «Задания предмета»

Страница имеет идентичный интерфейс и позволяет преподавателям просматривать и изменять созданные практические задания учебного предмета.

Аналогичный интерфейс имеет страница просмотра курсовых работ, представленная на рисунке 18.

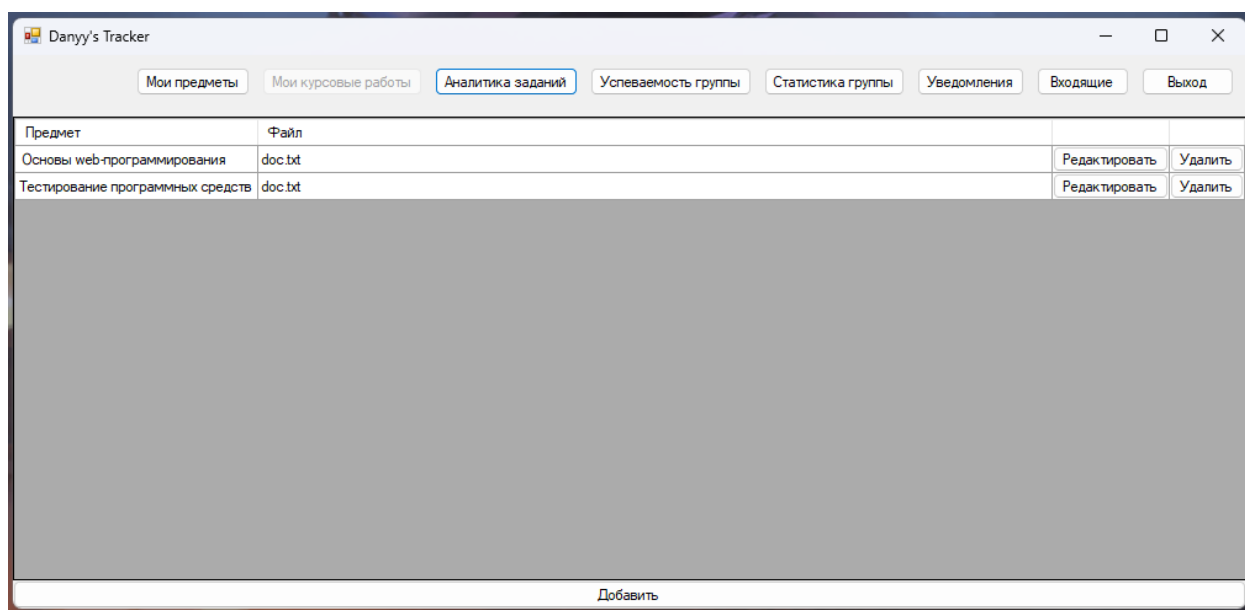


Рисунок 18 – Интерфейс страницы «Мои курсовые работы»

Страница позволяет преподавателям просматривать и изменять созданные курсовые работы учебного предмета.

На рисунке 19 представлен интерфейс страницы «Аналитика заданий».

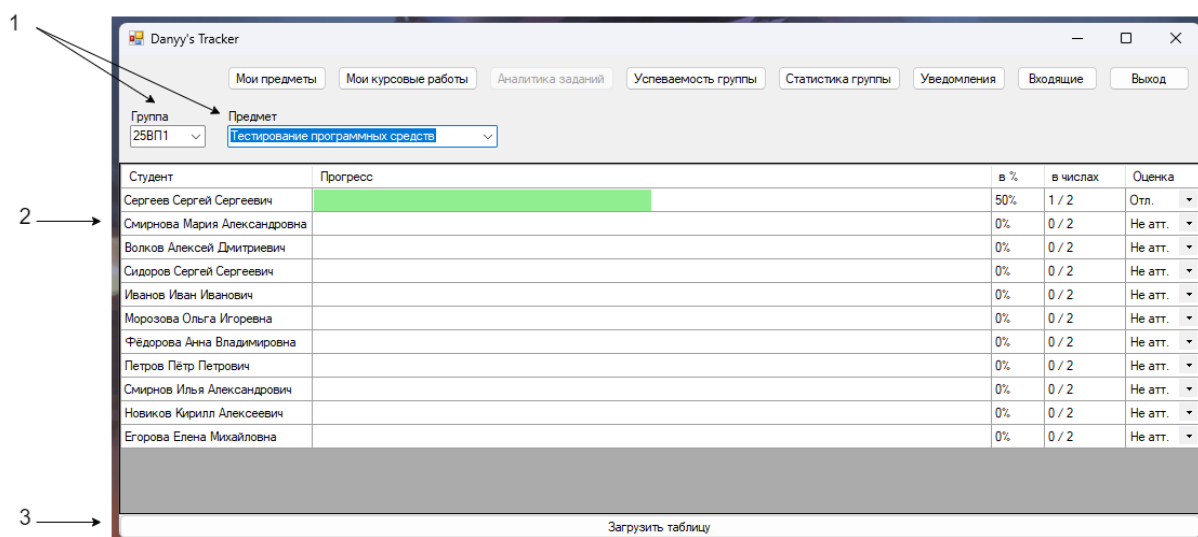


Рисунок 19 – Интерфейс страницы «Аналитика заданий»

Цифрами на рисунке обозначены:

1 – фильтры отображаемых данных;

- 2 – отображаемые данные пользователей;
- 3 – кнопка загрузки данных из приложения.

На странице «Аналитика заданий» преподаватель может просматривать статистику пользователей, которым назначено выполнение заданий созданного им учебного предмета. С помощью фильтров преподаватель может выбирать созданные им предметы, а также группы, которым выполнение заданий выбранного предмета было назначено. Преподаватель может выставить студенту оценку по выбранному предмету. Также преподаватель может выгрузить отображаемые данные в таблицу.

Кроме просмотра статистики своих заданий преподаватель может просмотреть статистику студентов в своей учебной группе, интерфейс страницы с этими данным идентичен представленной, кроме возможности выставлять пользователям оценку.

Интерфейс страницы «Задания» представлен на рисунке 20.

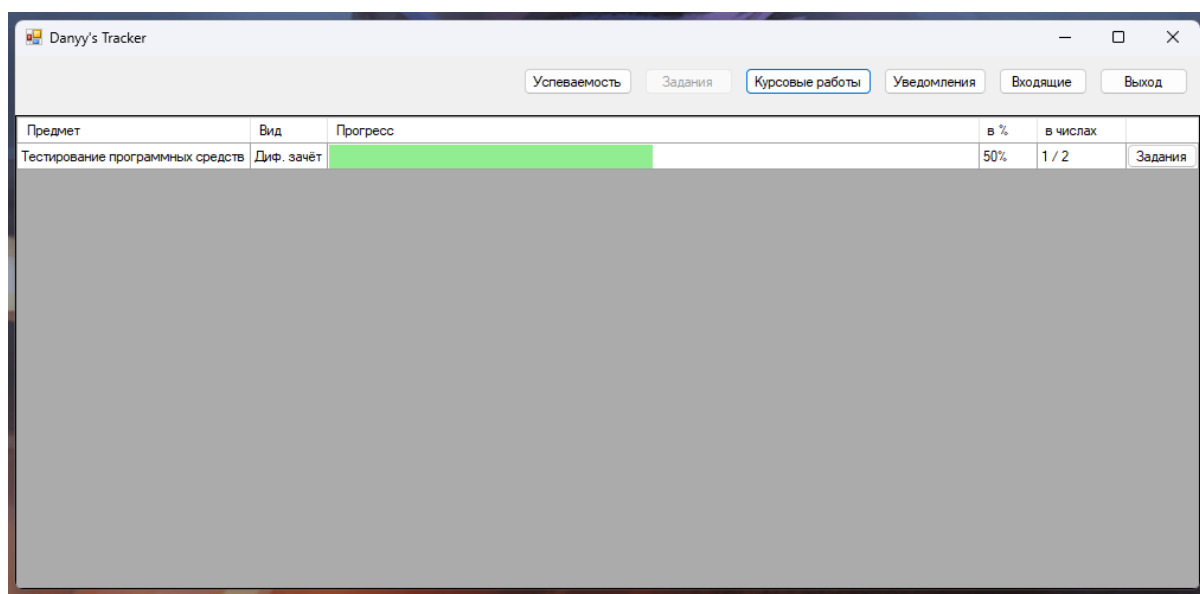


Рисунок 20 – Интерфейс страницы «Задания»

На этой странице студент может просматривать назначенные ему практические задания, сгруппированные по учебным предметам. Также студент может просмотреть свою персональную статистику. Сами задания студент

может просмотреть на соответствующей странице, интерфейс которой представлен на рисунке 21.

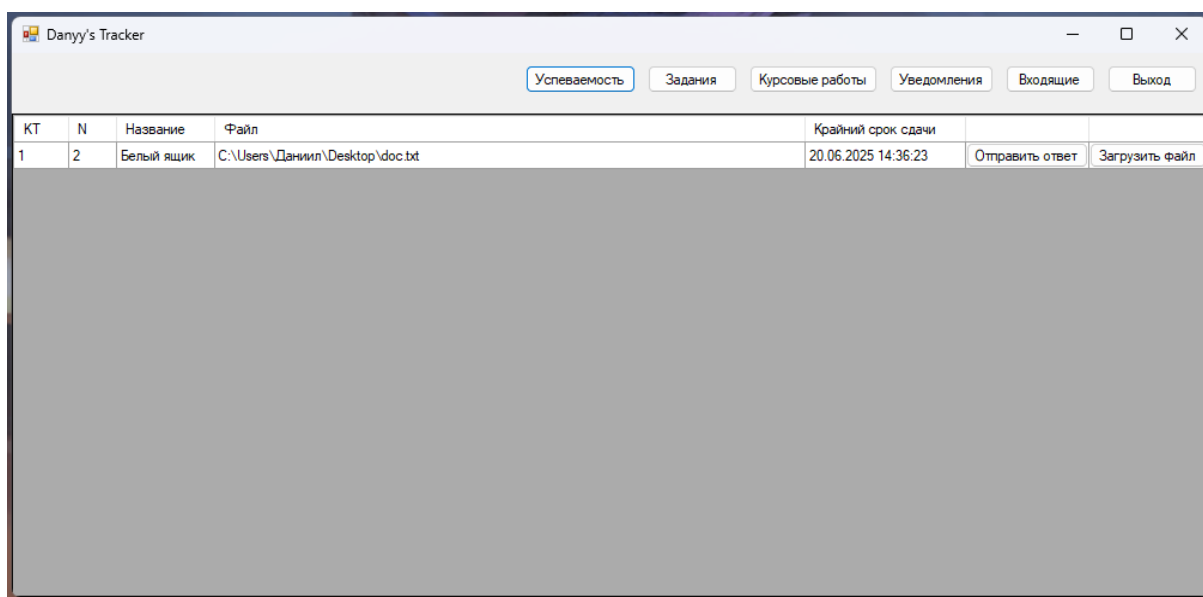


Рисунок 21 – Интерфейс страницы с назначенными заданиями

На эту страницу выводятся задания учебного предмета, выполнение которых назначено студенту. На этой странице пользователь может отправить преподавателю выполненную работу на проверку. Страница просмотра назначенных курсовых работ имеет идентичный интерфейс, представленный на рисунке 22.

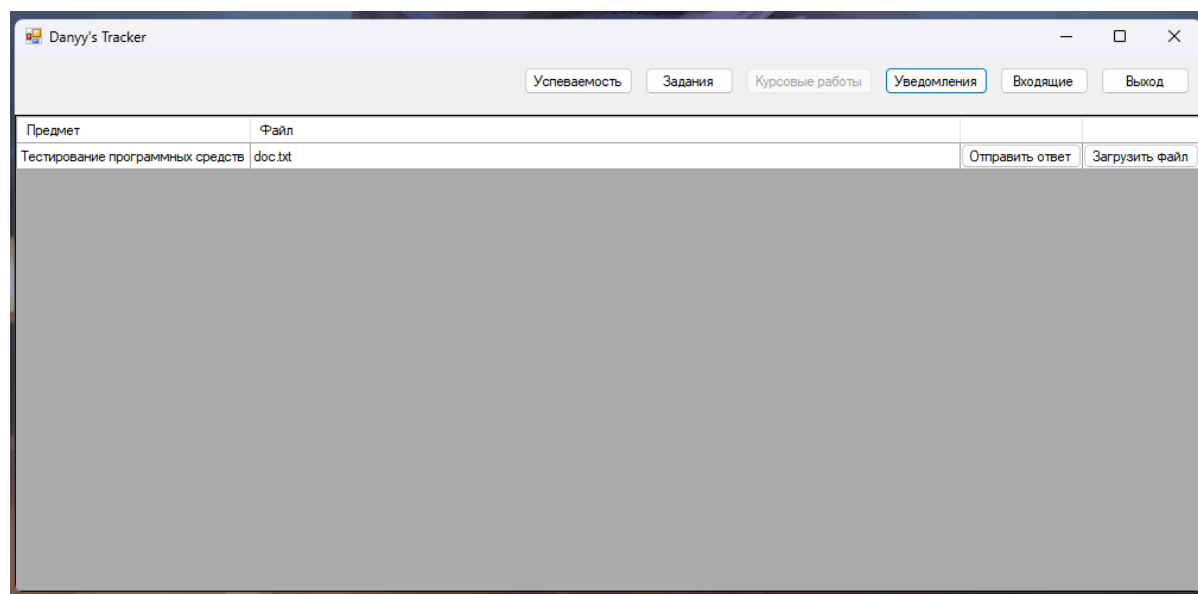


Рисунок 22 – Интерфейс страницы «Курсовые работы»

Интерфейс страницы «Входящие» представлен на рисунке 23.

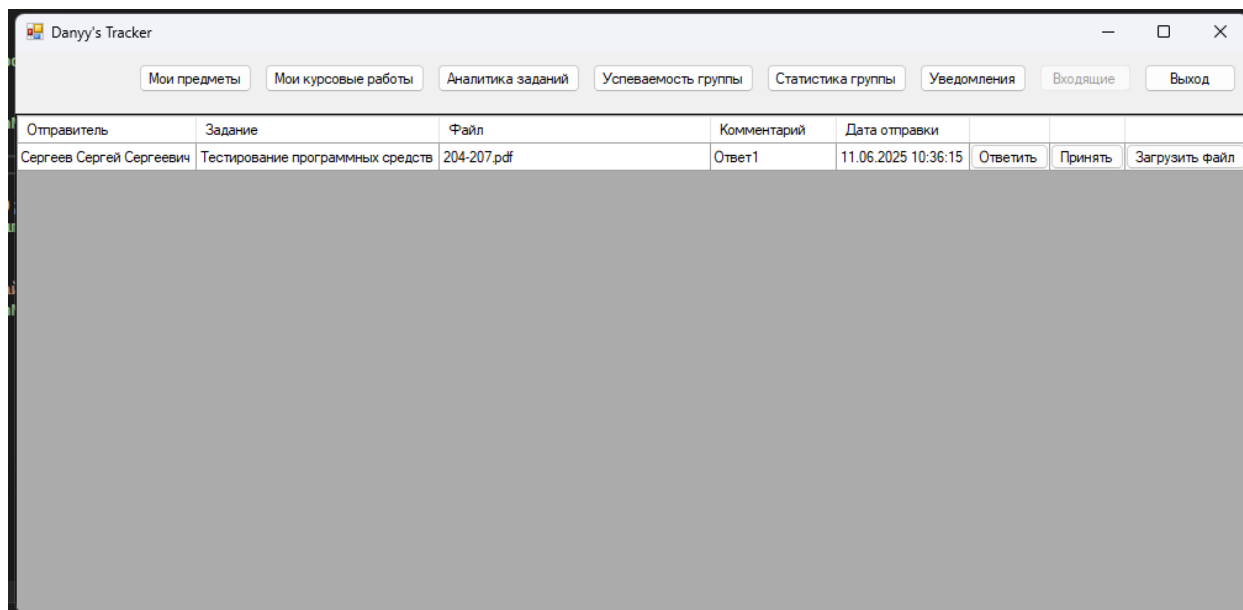


Рисунок 23 – Интерфейс страницы «Входящие»

На этой странице пользователь может просматривать и отвечать на входящие сообщения. Сообщения могут содержать в себе прикрепленные файлы, которые пользователь может загрузить и просмотреть. У преподавателя также есть возможность принять отправленную студентом работу с выставлением оценки.

Интерфейс страницы «Уведомления» представлен на рисунке 24.

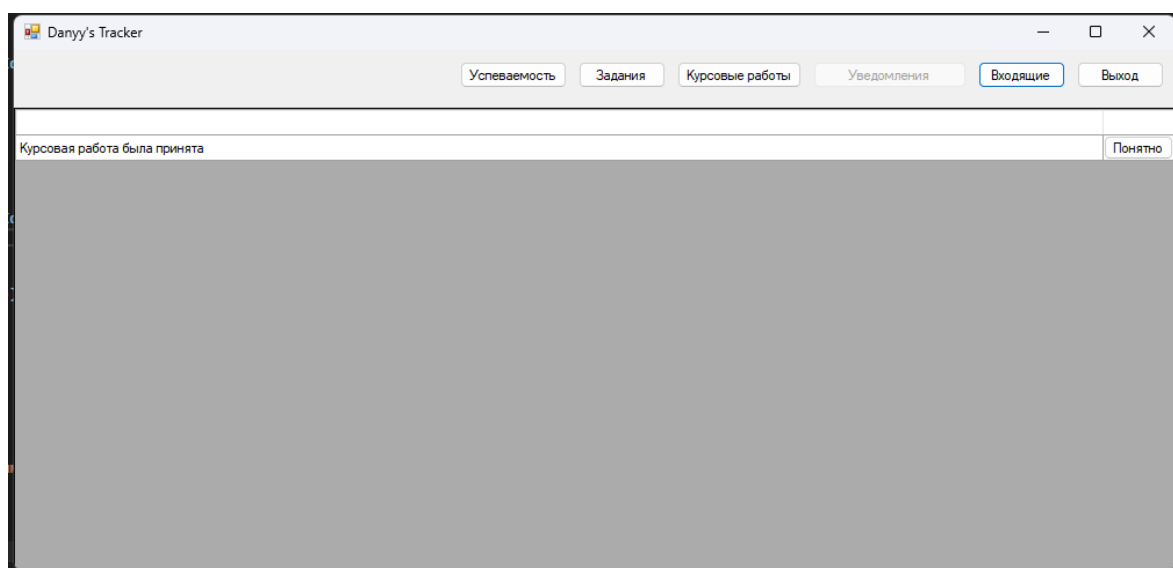


Рисунок 24 – Интерфейс страницы «Уведомления»

При приеме задания студенту будет отправлено уведомление о приеме отправленного ответа.

3.3 Разработка программного обеспечения

3.3.1 Разработка архитектуры программных средств

На основе выявленных функциональных требований и требований к архитектуре было разработано приложение таск-трекер для планирования и контроля выполнения практических заданий студентами. Взаимодействие между классами соответствует выбранной многоуровневой архитектуре: слои взаимодействуют только с соседними слоями и отвечают за соответствующий функционал. Для демонстрации связей между слоями была реализована схема, представленная на рисунке 25.

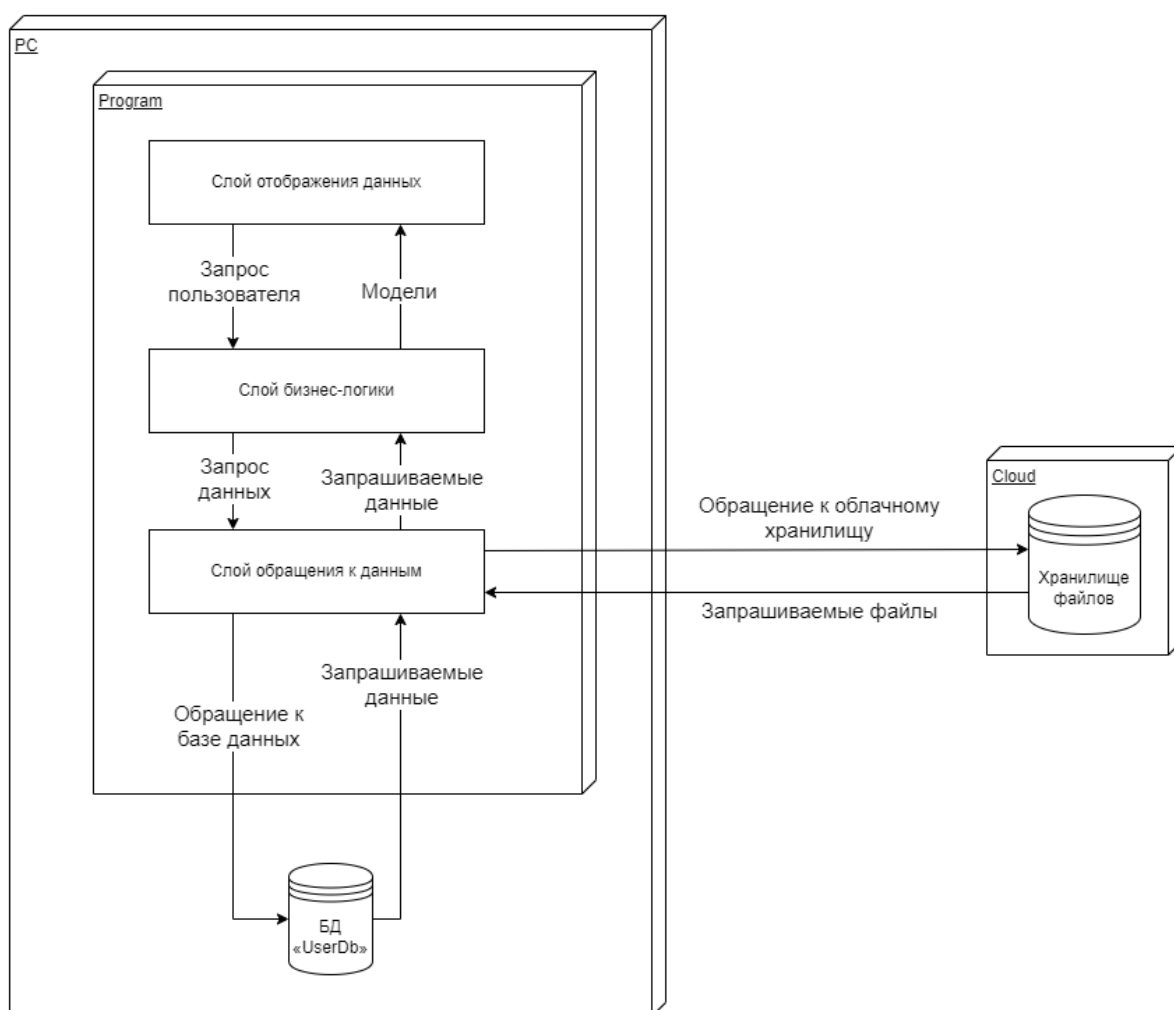


Рисунок 25 – Схема архитектуры разрабатываемого приложения

Каждый слой разрабатываемого в рамках выпускной квалификационной работы приложения таск-трекера для планирования и контроля выполнения практических заданий студентами включает в себя набор классов, которые отвечают за реализацию определенного функционала, характерного для многоуровневой архитектуры. Разработанные классы выполняют определенные слоем задачи, обеспечивая логическую изоляцию и разделение обязанностей между уровнями.

Благодаря многоуровневой архитектуре достигается лучшая структурированность кода, упрощается его сопровождение и повышается масштабируемость приложения.

Ниже представлено подробное описание классов каждого слоя, приведено описание и диаграммы классов каждого слоя.

3.3.2 Программное обеспечение слоя представления данных

Краткое описание основных классов слоя представления данных приведены в таблице 8.

Таблица 8 – Описание основных классов слоя представления данных

Компонент	Назначение
LoginForm.cs	Форма, позволяющая пользователю ввести данные для авторизации
MainForm.cs	Форма, отображающая страницы приложения
GradesView.cs	Страница, отображающая успеваемость пользователя
TaskStatisticView.cs	Страница, отображаемая статистику выполнения студентам заданий учебного предмета преподавателя
GroupStatisticView.cs	Страница, отображающая статистику выполнения заданий студентами учебной группы преподавателя
ProgressView.cs	Страница, отображающая прогресс выполнения заданий студентами

Диаграмма классов, отвечающих за отображение данных представлена на рисунке 26.

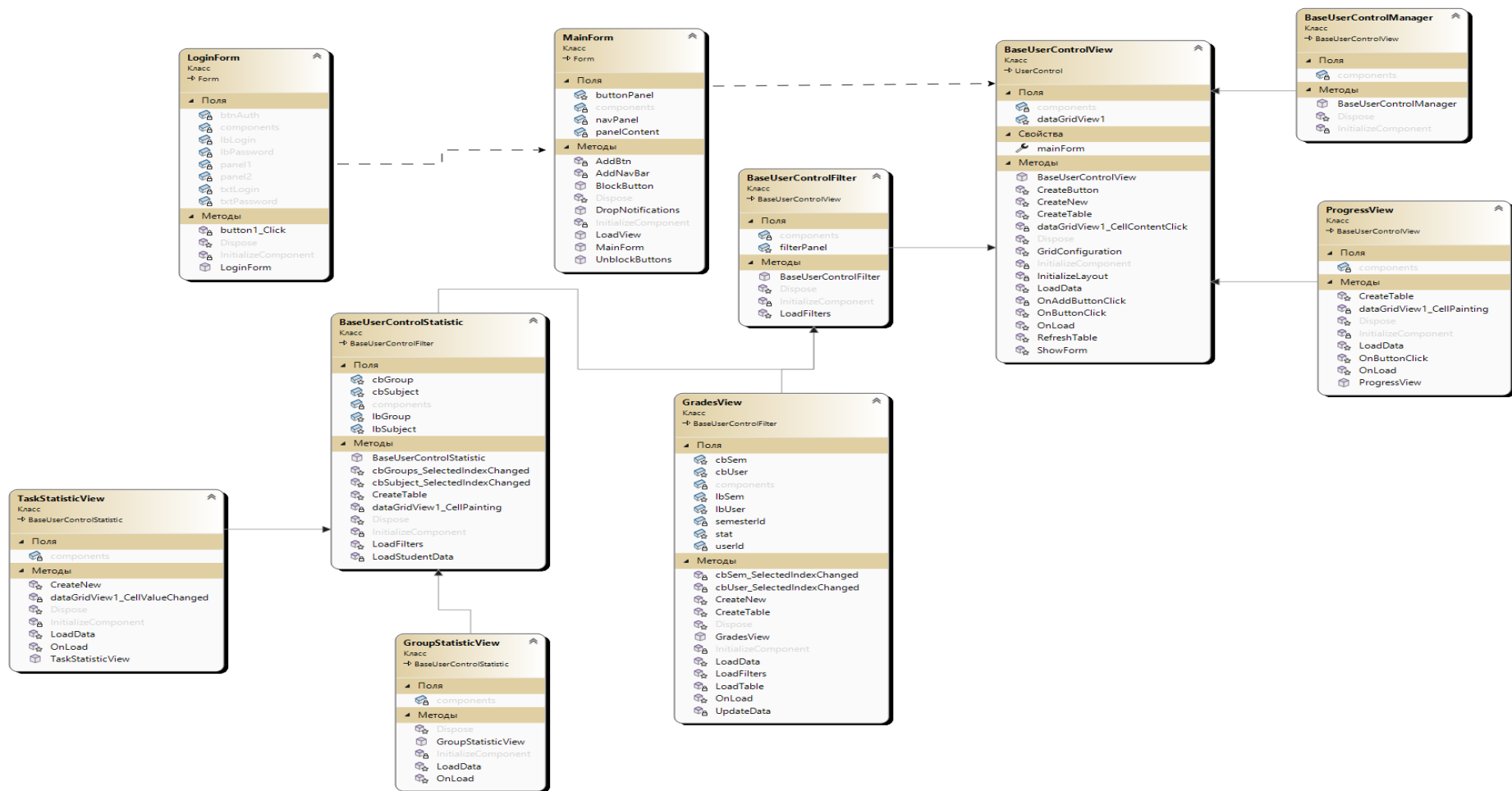


Рисунок 26 – Диаграмма основных классов слоя отображения данных разрабатываемого приложения

Листинг рассмотренных классов слоя представления данных представлен в приложении Б.

3.3.3 Программное обеспечение слоя бизнес-логики

Краткое описание основных классов слоя бизнес – логики представлено в таблице 9.

Таблица 9 – Описание основных классов слоя бизнес – логики

Компонент	Назначение
AssignedTaskController.cs	Контроллер, обрабатывающий получение назначенных пользователю практических заданий
NewSubjectController.cs	Контроллер, обрабатывающий добавление нового учебного предмета
ProgressViewController.cs	Контроллер, обрабатывающий данные пользователя и вычисляющий текущий прогресс выполнения заданий
StatisticController.cs	Контроллер, вычисляющий статистику студента
NotificationsController.cs	Контроллер, отвечающий за получение и удаление уведомлений пользователя

Диаграмма рассмотренных классов слоя бизнес-логики представлена на рисунке 27.

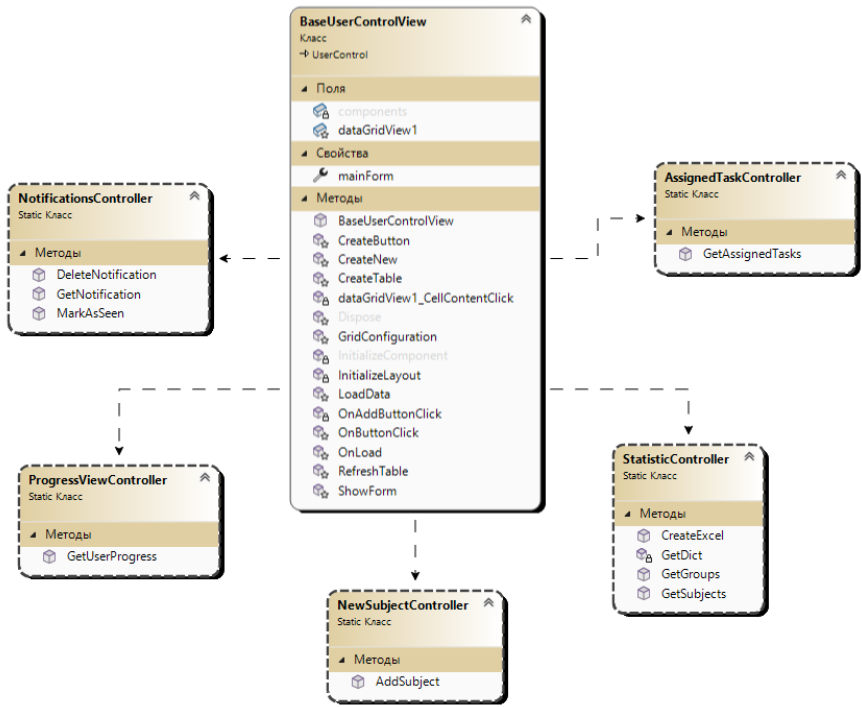


Рисунок 27 – Диаграмма основных классов слоя бизнес-логики разрабатываемого приложения

Листинг рассмотренных классов слоя бизнес-логики представлен в приложении В.

3.3.4 Программное обеспечение слоя обращения к данным

Краткое описание основных классов слоя обращения к данным представлено в таблице 10.

Таблица 10 – Описание основных классов слоя обращения к данным

Компонент	Назначение
DbConnection.cs	Класс, отвечающий за связь с базой данных
FileController.cs	Класс, отвечающий за связь с облачным хранилищем

Диаграмма рассмотренных классов слоя обращения к данным представлена на рисунке 28.

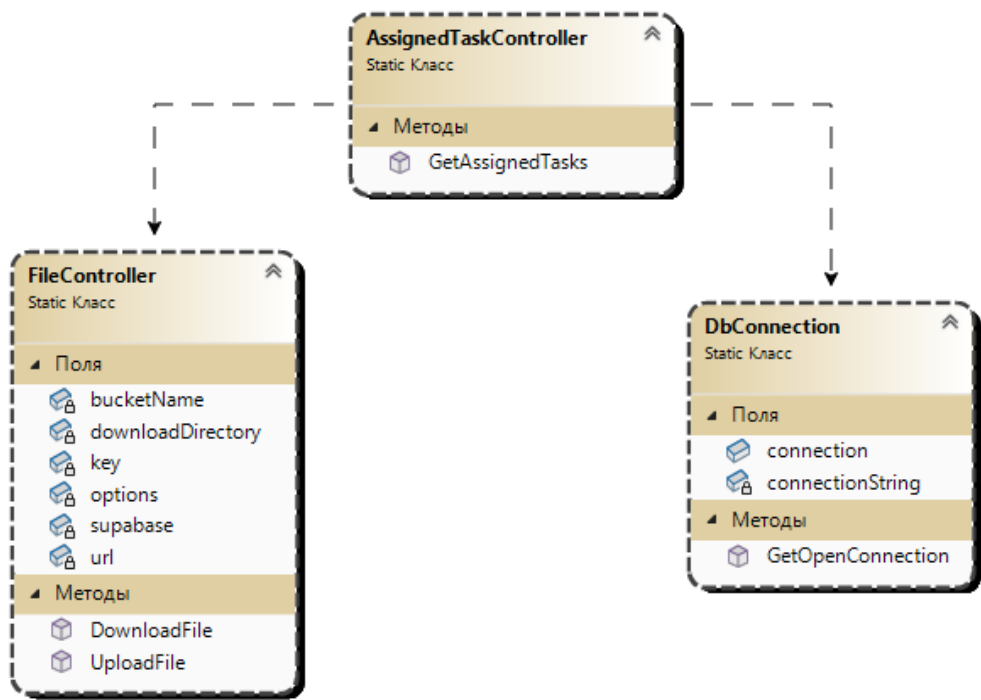


Рисунок 28 – Диаграмма основных классов слоя обращения к данным

Листинг рассмотренных классов слоя обращения к данным представлен в приложении Г.

3.3.5 Сценарии выполнения обращений

Рассмотрим работу разработанного приложения на примере сценария выгрузки данных в таблицу. Диаграмма деятельности для выбранного для примера сценария приведена на рисунке 29.

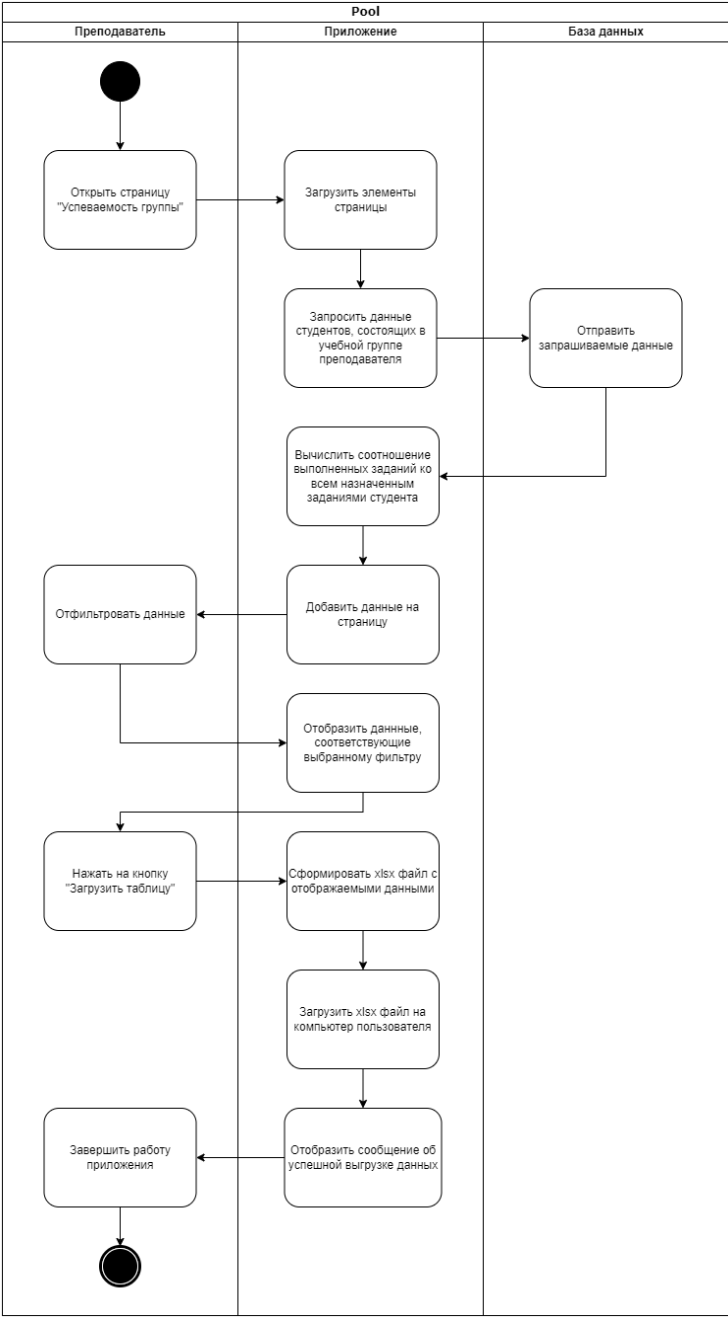


Рисунок 29 – Диаграмма деятельности для сценария выгрузки данных в таблицу

Листинг класса, отвечающего за исполнение этого сценария представлен в приложении Д. В результате выполнения данного сценария пользователь с

ролью «Преподаватель» должен получить файл с расширением .xlsx, содержащий отображаемые приложением данные. Результат работы рассмотренного сценария приведен на рисунке 30.

	A	B	C	D	E
1			Тестирование программных средств		
2	Группа	Обучающийся	Черный ящик	Белый ящик	
3	25ВП1	Волков Алексей Дмитриевич	-	-	
4	25ВП1	Егорова Елена Михайловна	-	-	
5	25ВП1	Иванов Иван Иванович	-	-	
6	25ВП2	Лебедев Дмитрий Николаевич	-	-	
7	25ВП1	Морозова Ольга Игоревна	-	-	
8	25ВП1	Новиков Кирилл Алексеевич	-	-	
9	25ВП1	Петров Пётр Петрович	-	-	
10	25ВП1	Сергеев Сергей Сергеевич	+	-	
11	25ВП1	Сидоров Сергей Сергеевич	-	-	
12	25ВП1	Смирнов Илья Александрович	-	-	
13	25ВП1	Смирнова Мария Александровна	-	-	
14	25ВП1	Фёдорова Анна Владимировна	-	-	
15					
16					
17					

Рисунок 30 – Результат выгрузки данных в таблицу

Таким образом, функционал разработанного приложения соответствует выявленным требованиям.

4 Оценка качества программного обеспечения

4.1 Обзор мероприятий оценки качества программного обеспечения

Оценка качества программного обеспечения играет ключевую роль в обеспечении его стабильности, безопасности и удобства для пользователя. Без системного подхода к проверке качества сложно гарантировать корректную работу даже при выполнении базовых функций.

Один из базовых методов – функциональное тестирование. Оно проверяет, насколько поведение приложения соответствует заданным требованиям: правильно ли обрабатываются действия пользователя, корректно ли работают элементы интерфейса. Такой тип тестирования может быть как ручным, так и автоматизированным с помощью специализированных инструментов, например TestComplete или Ranorex.

Модульное тестирование направлено на изолированную проверку логики отдельных функций или классов. Это позволяет выявлять ошибки на раннем этапе и снижает риск возникновения проблем при дальнейшей интеграции. В свою очередь, интеграционное тестирование проверяет взаимодействие между компонентами, выявляя сбои на стыке модулей.

Для десктопных приложений важную роль играет и тестирование пользовательского интерфейса. Оно позволяет убедиться, что элементы отображаются корректно и взаимодействие с ними удобно. Автоматизация UI-тестов особенно полезна при частых обновлениях, когда требуется быстро проверять уже реализованный функционал [24].

Дополнительно к тестированию применяется анализ кода с использованием метрик качества. Метрика цикломатической сложности позволяет оценить логическую нагруженность кода: чем выше показатель, тем сложнее сопровождать и тестировать такой участок. Метрика покрытия кода тестами показывает, насколько полно протестированы модули. Также важны

показатели связности и заимствованности, которые отражают архитектурную устойчивость программы. [25].

4.2 Тестирование методом черного ящика с применением функциональной диаграммы и таблицы решений

Для проведения мероприятия по оценке качества программного обеспечения было выбрано тестирование методом черного ящика с применением функциональной диаграммы и таблицы решений. Выбранный подход позволяет проверить систему без необходимости анализа самого кода, что актуально при работе с многоуровневой архитектурой.

Для проведения функционального тестирования был выбран участок кода, отвечающий за добавление новых пользователей. Листинг классов, отвечающих за добавление новых пользователей представлен в приложении Е. Процесс добавления пользователя включает в себя указание адреса электронной почты и номера телефона, которые ранее не были внесены в систему. Для вводимых при регистрации пользователя данных были определены следующие требования:

- адрес электронной почты должен быть корректной длины (15 – 255 символов);
- номер телефона должен быть корректной длины (10 цифр);
- адрес электронной почты должен содержать корректные символы;
- номер телефона должен состоять из цифр;
- адрес электронной почты должен быть соответствующего формата;
- номер телефона не должен быть зарегистрирован;
- адрес почты не должен быть зарегистрирован.

Приведенные требования были преобразованы в причины и следствия, представленные в таблице 11.

Таблица 11 – Причины и следствия

Причины	
П1	Корректная длина номера телефона

На основе информации из функциональной диаграммы была составлена таблица решений, содержащая все возможные варианты ввода и реакции программы. Таблица решений представлена в таблице 12.

Таблица 12 – Таблица решений

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35		
П1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		
П2	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
П3	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	
П4	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	
П5	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	
П6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	
П7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	
С1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
С2	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
С3	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	
С4	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	0
С5	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0	0
С6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	
С7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	
С8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	

На основе полученной таблицы с возможными решениями были описаны тест-кейсы, направленные на выявление ошибок в работе разработанного приложения. Описанные тест-кейсы для тестирования программного средства представлены в таблице 13.

Таблица 13 – Тест-кейсы

Входные данные	Выходные данные				
123_ long user...name.example@com	Некорректная длина номера телефона	Некорректные символы номера телефона	Некорректная длина адреса почты	Некорректн ые символы адреса почты	Некорректный формат адреса почты
123_ long user...name@example.com	Некорректная длина номера телефона	Некорректные символы номера телефона	Некорректная длина адреса почты	Некорректн ые символы адреса почты	
123_ longuser...name.example@com	Некорректная длина номера телефона	Некорректные символы номера телефона	Некорректная длина адреса почты		Некорректный формат адреса почты
123_ longuser...name@example.com	Некорректная длина номера телефона	Некорректные символы номера телефона	Некорректная длина адреса почты		
123 long user...name.example@com	Некорректная длина номера телефона		Некорректная длина адреса почты	Некорректн ые символы адреса почты	Некорректный формат адреса почты
123 long user...name@example.com	Некорректная длина номера телефона		Некорректная длина адреса почты	Некорректн ые символы адреса почты	
123 longuser...name.example@com	Некорректная длина номера телефона		Некорректная длина адреса почты		Некорректный формат адреса почты
123 longuser...name@example.com	Некорректная длина номера телефона		Некорректная длина адреса почты		
123_ user name.example@com	Некорректная длина номера телефона	Некорректные символы номера телефона		Некорректн ые символы адреса почты	Некорректный формат адреса почты
123_ user name@example.com	Некорректная длина номера телефона	Некорректные символы номера телефона		Некорректн ые символы адреса почты	
123_ username@example.com	Некорректная длина номера телефона	Некорректные символы номера телефона			Некорректный формат адреса почты
123_ username@example.com	Некорректная длина номера телефона	Некорректные символы номера телефона			
123 user name.example@com	Некорректная длина номера телефона			Некорректн ые символы адреса почты	Некорректный формат адреса почты
123 user name@example.com	Некорректная длина номера телефона			Некорректн ые символы адреса почты	

Продолжение таблицы 13

Входные данные	Выходные данные				
123 username@example.com	Некорректная длина номера телефона				Некорректный формат адреса почты
123 username@example.com	Некорректная длина номера телефона				
123456789_ long user...name.example@com		Некорректные символы номера телефона	Некорректная длина адреса почты	Некорректн ые символы адреса почты	Некорректный формат адреса почты
123456789_ long user...name@example.com		Некорректные символы номера телефона	Некорректная длина адреса почты	Некорректн ые символы адреса почты	
123456789_ longuser...name.example@com		Некорректные символы номера телефона	Некорректная длина адреса почты		Некорректный формат адреса почты
123456789_ longuser...name@example.com		Некорректные символы номера телефона	Некорректная длина адреса почты		
0123456789 long user...name.example@com			Некорректная длина адреса почты	Некорректн ые символы адреса почты	Некорректный формат адреса почты
0123456789 long user...name@example.com			Некорректная длина адреса почты	Некорректн ые символы адреса почты	
0123456789 longuser...name.example@com			Некорректная длина адреса почты		Некорректный формат адреса почты
0123456789 longuser...name@example.com			Некорректная длина адреса почты		
123456789_ user name.example@com		Некорректные символы номера телефона		Некорректн ые символы адреса почты	Некорректный формат адреса почты
123456789_ user name@example.com		Некорректные символы номера телефона		Некорректн ые символы адреса почты	
123456789_ username@example.com		Некорректные символы номера телефона			Некорректный формат адреса почты
123456789_ username@example.com		Некорректные символы номера телефона			

Окончание таблицы 13

Входные данные	Выходные данные				
0123456789 user name.example@com				Некорректные символы адреса почты	Некорректный формат адреса почты
0123456789 user name@example.com				Некорректные символы адреса почты	
0123456789 username.example@com					Некорректный формат адреса почты
0123456789 username@example.com	Пользователь успешно добавлен				

Программа корректно реагирует на вводимые данные и соответствует функциональным требованиям.

4.3 Попарное тестирование

Попарное тестирование является эффективной техникой тестирования, направленной на сокращение количества тест-кейсов при сохранении высокой степени покрытия входных данных. Данный метод позволяет выявить дефекты, возникающие при взаимодействии различных параметров системы, за счет комбинирования пар значений.

Для проведения попарного тестирования был выбран рассмотренный ранее процесс регистрации пользователя, включающий ввод адреса электронной почты и номера телефона. Основными параметрами, влияющими на корректность ввода, являются:

- длина номера телефона;
- символы номера телефона;
- длина адреса почты;
- символы адреса почты;
- формат адреса почты.

В ходе анализа перечисленных выше параметров были выведены исходные данные для формирования пар тестов, приведенные в таблице 14.

Таблица 14 – Данные для формирования пар

Корректная длина номера телефона	Корректные символы номера телефона	Корректная длина адреса почты	Корректные символы адреса почты	Корректный формат адреса почты
10	0 – 9	5 – 64	Печатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
0 – 9	Не 0 – 9	0 – 4	Непечатаемые символы	Остальное
11+		65+		

На основе этих данных были определены возможные комбинации значений, использованные для тестирования. Созданные тест-кейсы перечислены в таблице 15.

Таблица 15 – Тест-кейсы

№	Длина номера телефона	Символы номера телефона	Длина адреса почты	Символы адреса почты	Формат адреса почты
1	10	Не 0 – 9	0 – 4	Непечатаемые символы	Остальное
2	0 – 9	Не 0 – 9	65+	Непечатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
3	0 – 9	Не 0 – 9	5 – 64	Непечатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
4	0 – 9	0 – 9	0 – 4	Печатаемые символы	Остальное
5	11+	0 – 9	5 – 64	Печатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
6	11+	Не 0 – 9	0 – 4	Непечатаемые символы	Остальное
7	11+	0 – 9	65+	Печатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
8	11+	Не 0 – 9	5 – 64	Непечатаемые символы	Остальное
9	10	Не 0 – 9	65+	Непечатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
10	10	0 – 9	5 – 64	Печатаемые символы	Остальное
11	10	Не 0 – 9	5 – 64	Непечатаемые символы	$\wedge[\wedge s@.] + @ [a-zA-Z] + \backslash [a-zA-Z] + \$$
12	10	0 – 9	0 – 4	Печатаемые символы	Остальное

Попарное тестирование позволило минимизировать затраты на тестирование и выявить потенциальные ошибки при меньшем количестве проверок, обеспечивая надежность и качество регистрации пользователей.

4.4 Оценка качества кода с помощью метрик

Количество строк кода (англ. Source Lines of Code – SLOC) – это метрика программного обеспечения, используемая для измерения его объёма с помощью подсчёта количества строк в тексте исходного кода. Обычно код измеряется в двух вариантах:

physical SLOC (PSLOC) – количество всех непустых строк кода, включая объявления, операторы, скобки и т. д., но исключая комментарии и пустые строки;

logical SLOC (LSLOC) – количество логических операторов или выражений (например, if, for, присваивания и вызовы функций), каждый из которых считается за одну логическую строку.

Для анализа был выбран участок кода, отвечающий за обработку успеваемости студентов, код представлен в приложении В.

Подсчёт Physical SLOC:

- объем класса: 179 строк;
- пустых строк: 29.

Подсчёт Logical SLOC:

- метод SelectGroup(): 10 LSLOC;
- метод LoadGroup(...): 20 LSLOC;
- метод GetAssignedSubject(...): 10 LSLOC;
- метод GetAssignedCW(...): 10 LSLOC;
- метод CreateExcel(...): 15 LSLOC.

Результаты проведенного анализа:

- физические строки: 150;
- логические строки: 55.

В результате проведенного анализа был сделан следующий вывод: на 1 физическую строку приходится 0.5 логической – это нормальное соотношение, свидетельствующее о хорошей читаемости и умеренной плотности логики.

Заключение

В результате выполнения выпускной квалификационной работы разработано приложение таск-трекер для планирования и контроля выполнения практических заданий студентами, соответствующее всем обозначенным на этапе анализа требованиям:

- обеспечивает доступ данным при отсутствии выхода в глобальную сеть;
- предоставляет возможность общения студента и преподавателя;
- отображает прогресс выполнения назначенных студенту заданий в удобном для восприятия виде;
- выводит статистику по успеваемости студента.

В работе приведен анализ предметной области, обоснование необходимости разрабатываемого программного средства, а также сравнительный анализ аналогов. По результатам анализа составлен план с этапами работ по созданию, в соответствии с которым проходила реализация приложения-трекера прогресса. В результате планирования затраты времени составили 408 часов, а денежные затраты – 127500 рублей.

На этапе проектирования определены требования к функционалу, хранению данных и пользовательскому интерфейсу. Спроектированы концептуальная и логическая базы данных. Выбран архитектурный стиль, технологии и средства разработки.

В ходе реализации разработана база данных в среде SQL Server Management Studio, приложение на языке программирования C# и пользовательский интерфейс с применением технологий Windows Forms. Написанный код соответствует выбранному архитектурному стилю, разделен на слои, каждый со своей логикой и зоной ответственности.

После реализации программного средства проведен контроль качества с помощью различных методов тестирования и оценок качества кода. В результате проведенного тестирования было определено, что разработанное

приложение-трекер прогресса работает корректно и имеет хороший читаемый код.

Таким образом, разработанное в ходе выполнения выпускной квалификационной работы приложение таск-трекер для планирования и контроля выполнения практических заданий студентами полностью удовлетворяет требованиям потенциальных пользователей – студентов и преподавателей ВУЗов, работает корректно и предоставляет богатый функционал, обеспечивающий площадку для организации электронного обучения.

Список использованных источников

1. Болкунов И.А. Электронное обучение: проблемы, перспективы, задачи // Таврический научный обозреватель. 2012. №3. С. 129 – 132. [Электронный ресурс] URL: <https://cyberleninka.ru/article/n/elektronnoe-obuchenie-problemy-perspektivy-zadachi> (дата обращения: 18.05.2025).
2. Сатунина А.Е. ЭЛЕКТРОННОЕ ОБУЧЕНИЕ: ПЛЮСЫ И МИНУСЫ // Современные проблемы науки и образования. 2006. № 1 С. 89 – 90 [Электронный ресурс] URL: <https://science-education.ru/ru/article/view?id=103> (дата обращения: 18.05.2025).
3. Казаков М.М. Информационные технологии и новая парадигма образования // VIII межрегиональная специализированная выставка-семинар по компьютерным и телекоммуникационным технологиям. 18 – 21 января 2005 г. Сборник трудов. – Смоленск, 2005. – С.86 – 91. (дата обращения: 18.05.2025)
4. Digital classrooms evolved: How real-time feedback fuels student progress // timesofindia.indiatimes.com [Электронный ресурс]. URL: <https://timesofindia.indiatimes.com/education/india-schools-online/digital-classrooms/digital-classrooms-evolved-how-real-time-feedback-fuels-student-progress/articleshow/118560015.cms> (дата обращения: 18.05.2025).
5. What are the psychological benefits of automating student progress tracking in LMS platforms, and how can educators leverage these insights to enhance learning outcomes? Include references to studies on learning psychology and the impact of tracking on student motivation. // psicosmart.net [Электронный ресурс]. URL: <https://psicosmart.net/blogs/blog-what-are-the-psychological-benefits-of-automating-student-progress-tra-243193> (дата обращения: 18.05.2025).
6. Revolutionize Learning: The Impact of Visual Course Completion Indicators // learnexus.com [Электронный ресурс]. URL: <https://learnexus.com/blog/visual-course-completion-indicators/> (дата обращения: 18.05.2025).
7. What are the psychological effects of automated student progress tracking in LMS on learner motivation and engagement? Include insights from educational

psychology studies and psychology journals. // learnexus.com [Электронный ресурс]. URL: <https://psicosmart.net/blogs/blog-what-are-the-psychological-effects-of-automated-student-progress-track-241482> (дата обращения: 18.05.2025).

8. Task Tracker: что это такое и как использовать // sky.pro. [Электронный ресурс]. URL: <https://sky.pro/wiki/profession/task-tracker-что-это-такое-и-как-использовать/> (дата обращения: 18.05.2025).

9. Asana: система управления проектами [Электронный ресурс]. Режим доступа: asana.com (дата обращения: 18.05.2025).

10. Moodle: система управления обучением [Электронный ресурс]. Режим доступа: <https://moodle.com> (дата обращения: 18.05.2025).

11. Trello: онлайн-сервис для управления проектами [Электронный ресурс]. Режим доступа: <https://trello.com/ru> (дата обращения: 18.05.2025).

12. ClickUp уходит. Как METEOR поможет пользователям уходящего трекера? // <https://habr.com> [Электронный ресурс]. URL: <https://habr.com/ru/articles/843178/> (дата обращения: 18.05.2025).

13. CRM Мегаплан – обзор функционала системы // <https://crm.ru/> [Электронный ресурс]. URL: <https://crm.ru/blog/crm-megaplan-obzor/> (дата обращения: 18.05.2025).

14. Microsoft Office Project: инструмент для планирования [Электронный ресурс]. Режим доступа: <https://www.microsoft.com/ru-ru/microsoft-365/planner/microsoft-planner?market=ru> (дата обращения: 18.05.2025).

15. Все о Диаграмме Ганта: Как Успешно и Вовремя Завершать Проекты // blog.ganttpro.com [Электронный ресурс]. URL: <https://blog.ganttpro.com/ru/diagramma-gantta-gantt-chart/> (дата обращения: 18.05.2025).

16. Рудь В. Концептуальное логическое и физическое моделирование данных // Информационный портал компании «Марк Аврелий» [Электронный ресурс]. URL: https://marcus-aurelius.ru/articles/Data_modeling.html (дата обращения 03.06.2024).

17. Многоуровневая архитектура организации распределенных информационных систем / Н. В. Пацей [и др.] // Информационные технологии : материалы 86-й научно-технической конференции профессорско-преподавательского состава, научных сотрудников и аспирантов, Минск, 31 января – 12 февраля 2022 г. – Минск : БГТУ, 2022. – С. 72 – 75. (дата обращения: 18.05.2025).

18. Наварко, А. С. Windows forms как современная модель программирования и ее возможности / А. С. Наварко, Н. И. Белодед // Информационные технологии в образовании, науке и производстве [Электронный ресурс] : материалы научно-технической интернет-конференции, Минск, 21 – 22 ноября 2022 г. / сост. М. Г. Карасёва. – Минск : БНТУ, 2023. – С. 204 – 207. (дата обращения: 18.05.2025).

19. Функциональные возможности и применение Microsoft SQL Server Management Studio – полный обзор инструмента для работы с базами данных // perepolokh.ru [Электронный ресурс]. URL: <https://perepolokh.ru/blog/funkcionalnye-vozmozhnosti-i-primenenie-microsoft-sql-server-management-studio-polnyi-obzor-instrumenta-dlya-raboty-s-bazami-dannyx/> (дата обращения: 18.05.2025).

20. Supabase Documentation [Электронный ресурс]. Режим доступа: supabase.com/docs (дата обращения: 18.05.2025).

21. C# Client Library [Электронный ресурс]. Режим доступа: <https://supabase.com/docs/reference/csharp/introduction> (дата обращения: 18.05.2025).

22. EPPlus – Таблицы Excel для .NET [Электронный ресурс]. Режим доступа: <https://epplussoftware.com> (дата обращения: 18.05.2025).

23. Сервис Draw.io: за что его любят и как им пользоваться // <https://practicum.yandex.ru> [Электронный ресурс]. URL: <https://practicum.yandex.ru/blog/vozmozhnosti-servisa-drawio/> (дата обращения: 18.05.2025).

24. Тестирование десктопных приложений // <https://ios-apps.ru/> [Электронный ресурс]. URL: ios-apps.ru/blog/testirovanie-desktopnyx-prilozenii/ (дата обращения: 18.05.2025).

25. Метрики кода, или Как определить внутреннее качество продукта // <https://dou.ua/> [Электронный ресурс]. URL: <https://dou.ua/lenta/articles/code-metrics/> (дата обращения: 18.05.2025).

Приложение А
(справочное)
Глоссарий

БД (База данных) – организованная совокупность взаимосвязанных данных, хранящихся и обрабатываемых таким образом, чтобы обеспечить быстрый поиск, добавление, изменение и удаление информации.

Система управления базами данных (СУБД) – это программный комплекс, который обеспечивает централизованное хранение и организацию данных, предоставляет механизмы для их поиска и изменения, а также гарантирует целостность, безопасность и согласованность информации при одновременной работе нескольких пользователей.

E-learning – это организация учебного процесса с помощью цифровых технологий и интернет-ресурсов, при которой обучение проходит дистанционно, независимо от места и времени с использованием онлайн-платформ, мультимедийного контента и интерактивных инструментов для передачи знаний и поддержки взаимодействия между преподавателем и учащимися.

Десктопное приложение – это устанавливаемая на персональный компьютер или ноутбук программа, работающая в операционной системе пользователя, взаимодействующая с локальными ресурсами (файловой системой, периферией и оборудованием) для решения конкретных задач, обеспечивающая быструю отзывчивость и стабильную работу без постоянного выхода в интернет.

.NET – это кроссплатформенная экосистема и фреймворк от Microsoft, предоставляющая общую инфраструктуру библиотек, среду выполнения и инструменты для разработки приложений под Windows, Linux, macOS, а также веб- и мобильных решений.

Таск-трекер – программный инструмент для постановки и управления задачами, позволяющий отслеживать их статусы, сроки и распределение ответственности, а в учебной деятельности помогает преподавателям и студентам планировать учебные задания, контролировать сроки их выполнения, отслеживать прогресс каждого участника и обеспечивать оперативную коммуникацию в образовательном процессе.

Приложение Б

(обязательное)

Листинг основных классов слоя отображения данных

Листинг Б.1 – Файл LoginForm.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;

namespace StudentTaskTracker
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            string login = txtLogin.Text.Trim();
            string password = txtPassword.Text;

            if (string.IsNullOrEmpty(login) ||
string.IsNullOrEmpty(password))
            {
                MessageBox.Show("Пожалуйста, введите логин и
пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            try
            {
                if (LoginController.UserLogin(login, password))
                {
                    MessageBox.Show($"Добро пожаловать,
{SessionManager.UserName} {SessionManager.LastName}!");

                    var mainForm = new MainForm();
                    mainForm.Show();
                    this.Hide();
                }

                else
                {

```

```

        MessageBox.Show("Неверный логин или пароль.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при подключении к базе
данных: " + ex.Message, "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}
}
}

```

Листинг Б.2 – Файл GradesView.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using StudentTaskTracker.Controllers;
using StudentTaskTracker.Models;
using static System.Windows.Forms.AxHost;

namespace StudentTaskTracker
{
    public partial class GradesView : BaseUserControlFilter
    {
        Guid userId;
        int semesterId;

        protected ComboBox cbSem;
        protected Label lbSem;

        protected ComboBox cbUser;
        protected Label lbUser;

        protected Label stat;

        public GradesView()
        {
            InitializeComponent();

            stat = new Label
            {

```

```

        AutoSize = true,
        Location = new Point(580, 17)
    };
    filterPanel.Controls.Add(stat);
    cbSem.SelectedIndexChanged +=
cbSem_SelectedIndexChanged;
    cbUser.SelectedIndexChanged +=
cbUser_SelectedIndexChanged;
    }

    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);

        if (SessionManager.RoleID == 1)
        {
            mainForm.BlockButton("Успеваемость группы");
            CreateButton("Загрузить таблицу");
        }

        else if (SessionManager.RoleID == 0)
        {
            mainForm.BlockButton("Успеваемость");
        }
    }

    protected override void CreateNew()
    {
        GradesController.CreateExcel(dataGridView1,
cbUser.Text);
        MessageBox.Show("Данные были успешно выгружены в
таблицу!", "Сообщение", MessageBoxButtons.OK,
MessageBoxIcon.Information);
    }

    protected override void LoadFilters()
    {
        cbSem = new ComboBox
        {
            Width = 50,
            Location = new Point(10, 15),
        };
        cbSem.Items.AddRange(new object[] {
            "1",
            "2",
            "3",
            "4",
            "5",
            "6",
            "7",
            "8",
            "9",
            "10",

```

```

        "11",
        "12",
        "13",
        "14"});
filterPanel.Controls.Add(cbSem);

lbSem = new Label
{
    Text = "Семестр",
    AutoSize = true,
    Location = new Point(8, 0),
};
filterPanel.Controls.Add(lbSem);

cbUser = new ComboBox
{
    Width = 250,
    Location = new Point(10 + cbSem.Width + 20, 15)
};
filterPanel.Controls.Add(cbUser);

lbUser = new Label
{
    Text = "Студент",
    AutoSize = true,
    Location = new Point(8 + cbSem.Width + 20, 0),
};
filterPanel.Controls.Add(lbUser);
}

protected override void CreateTable()
{
    dataGridView1.Columns.Add("UserID", "");
    dataGridView1.Columns["UserID"].Visible = false;

    dataGridView1.Columns.Add("Group", "");
    dataGridView1.Columns["Group"].Visible = false;

    dataGridView1.Columns.Add("Semester", "");
    dataGridView1.Columns["Semester"].Visible = false;

    dataGridView1.Columns.Add("Name", "Предмет");

    dataGridView1.Columns.Add("Type", "Способ
аттестации");
    dataGridView1.Columns["Type"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

    dataGridView1.Columns.Add("Mark", "Оценка");
    dataGridView1.Columns["Mark"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
}

```

```

private void LoadTable(List<GradeModel> list)
{
    foreach (var item in list)
    {
        dataGridView1.Rows.Add(item.UserID,
item.GroupName, item.Semester, item.SubjectName, item.Type,
item.Mark);
    }
}

protected override void LoadData()
{
    if (SessionManager.RoleID == 1)
    {
        var group =
GradesController.SelectGroup(SessionManager.UserID);

        cbUser.DataSource = new BindingSource(group,
null);

        cbUser.DisplayMember = "Value";
        cbUser.ValueMember = "Key";

        cbUser.SelectedIndex = 0;
    }

    else if (SessionManager.RoleID == 0)
    {
        cbUser.Hide();
        lbUser.Hide();
        userId = SessionManager.UserID;
    }

    var asbList =
GradesController.GetAssignedSubject(SessionManager.UserID);
    LoadTable(asbList);

    var acwList =
GradesController.GetAssignedCW(SessionManager.UserID);
    LoadTable(acwList);

    cbSem.SelectedIndex = 0;
}

private void UpdateData(Guid userId, int semId)
{
    foreach (DataGridViewRow row in dataGridView1.Rows)
    {
        if (((Guid)row.Cells["UserID"].Value == userId) &&
(Convert.ToInt32(row.Cells["Semester"].Value) == semId))
        {
            row.Visible = true;
        }
        else

```

```

        {
            row.Visible = false;
        }
    }

    private void cbUser_SelectedIndexChanged(object sender,
EventArgs e)
    {
        var dict = (KeyValuePair<Guid,
string>)cbUser.Items[cbUser.SelectedIndex];
        userId = dict.Key;

        UpdateData(userId, semesterId);

        stat.Text =
GradesController.GetStatistic(dataGridView1, userId);
        stat.Refresh();

        //LoadSubjectData(userId, semesterId);
    }

    private void cbSem_SelectedIndexChanged(object sender,
EventArgs e)
    {
        semesterId = Convert.ToInt32(cbSem.SelectedItem);

        UpdateData(userId, semesterId);

        //LoadSubjectData(userId, semesterId);
    }
}

```

Листинг Б.3 – Файл MainForm.cs

```

using StudentTaskTracker.Controllers.Controllers;
using StudentTaskTracker.MainApp.Teachers;
using StudentTaskTracker.UI.UI.Forms;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

```

```

namespace StudentTaskTracker
{
    public partial class MainForm : Form
    {
        private Panel navPanel;
        private Panel panelContent;
        protected FlowLayoutPanel buttonPanel;

        public MainForm()
        {
            InitializeComponent();

            navPanel = new Panel
            {
                Dock = DockStyle.Top,
                Height = 50,
                BackColor = SystemColors.Control
            };

            panelContent = new Panel
            {
                Dock = DockStyle.Fill
            };

            Controls.Add(panelContent);
            Controls.Add(navPanel);

            AddNavBar();

            if (SessionManager.RoleID != 2)
            {
                LoadView(new GradesView());
            }
        }

        public void LoadView(BaseUserControlView view)
        {
            view.mainForm = this;

            view.Dock = DockStyle.Fill;
            panelContent.Controls.Clear();
            panelContent.Controls.Add(view);
        }

        public void UnblockButtons()
        {
            foreach (Control ctrl in buttonPanel.Controls)
            {
                if (ctrl is Button btn)
                {
                    btn.Enabled = true;
                }
            }
        }
    }
}

```

```

    }
}

public void BlockButton(string name)
{
    foreach (Control ctrl in buttonPanel.Controls)
    {
        if (ctrl is Button btn && btn.Text == name)
        {
            btn.Enabled = false;
            break;
        }
    }
}

public void DropNotifications(string name)
{
    foreach (Control ctrl in buttonPanel.Controls)
    {
        if (ctrl is Button btn && btn.Text == name)
        {
            string newName = "Уведомления";
            btn.Text = newName;
            btn.Name = newName;
            break;
        }
    }
}

private void AddBtn(string text, Action action, DockStyle
dock = DockStyle.None, Control parent = null)
{
    var btn = LoadHelper.CreateButton(text, action);
    btn.Dock = dock;
    (parent ?? buttonPanel).Controls.Add(btn);
}

private void AddNavBar()
{
    // Панель с кнопками справа внутри верхней панели
    buttonPanel = new FlowLayoutPanel
    {
        Dock = DockStyle.Right,
        FlowDirection = FlowDirection.LeftToRight,
        AutoSize = true,
        WrapContents = false,
        Padding = new Padding(5),
        BackColor = SystemColors.Control
    };

    if (SessionManager.RoleID == 2)
    {
        var regForm = new Registration();
    }
}

```

```

        AddBtn("Добавить пользователя", () =>
regForm.ShowDialog());

        var newGroupForm = new CreateStudyGroup();
        AddBtn("Добавить учебную группу", () =>
newGroupForm.ShowDialog());

        var alterGroupForm = new UserGroupForm();
        AddBtn("Добавить учебную группу", () =>
alterGroupForm.ShowDialog());
    }
    else
    {
        if (SessionManager.RoleID == 1)
        {
            AddBtn("Мои предметы", () => LoadView(new
SubjectsManager()));
            AddBtn("Мои курсовые работы", () =>
LoadView(new CWSManager()));
            AddBtn("Аналитика заданий", () => LoadView(new
TaskStatisticView()));
            AddBtn("Успеваемость группы", () =>
LoadView(new GradesView()));
            AddBtn("Статистика группы", () => LoadView(new
GroupStatisticView()));
        }

        if (SessionManager.RoleID == 0)
        {
            AddBtn("Успеваемость", () => LoadView(new
GradesView()));
            AddBtn("Задания", () => LoadView(new
ProgressView()));
            AddBtn("Курсовые работы", () => LoadView(new
AssignedCWView()));
        }

        var notifName =
MainFormController.UnseenNotifications();

        AddBtn(notifName, () => LoadView(new
NotificationsView(notifName)));
        AddBtn("Входящие", () => LoadView(new
MessageView()));
    }

    AddBtn("Выход", () => FormManager.GoToLogin(this));

    navPanel.Controls.Add(buttonPanel);
}
}
}

```

Листинг Б.4 – Файл TaskStatisticView.cs

```

using StudentTaskTracker.Controllers;
using StudentTaskTracker.MainApp.Teachers;
using StudentTaskTracker.MainApp.Teachers.BaseUC;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace StudentTaskTracker
{
    public partial class TaskStatisticView :
BaseUserControlStatistic
    {
        public TaskStatisticView()
        {
            InitializeComponent();

            protected override void OnLoad(EventArgs e)
            {
                base.OnLoad(e);
                mainForm.BlockButton("Аналитика заданий");
                dataGridView1.CellValueChanged +=
dataGridView1_CellValueChanged;
                CreateButton("Загрузить таблицу");
            }

            protected override void CreateNew()
            {
                StatisticController.CreateExcel((Guid)cbSubject.SelectedValue);
            }

            protected override void LoadData()
            {
                dataGridView1.Columns.Add(LoadHelper.CreateCbMarks());
                dataGridView1.Columns["Mark"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dataGridView1.ReadOnly = false;
                foreach (DataGridViewColumn col in
dataGridView1.Columns)
                {
                    if (col.Name != "Mark")

```

```

        col.ReadOnly = true;
    }

    var subjects =
StatisticController.GetSubjects(SessionManager.UserID);

    cbSubject.SelectedIndexChanged -=
cbSubject_SelectedIndexChanged;

    cbSubject.DataSource = new BindingSource(subjects,
null);

    cbSubject.DisplayMember = "Value";
    cbSubject.ValueMember = "Key";

    cbSubject.SelectedIndexChanged +=
cbSubject_SelectedIndexChanged;

    var groups =
StatisticController.GetGroups(SessionManager.UserID);

    cbGroup.SelectedIndexChanged -=
cbGroups_SelectedIndexChanged;

    cbGroup.DataSource = new BindingSource(groups, null);
    cbGroup.DisplayMember = "Value";
    cbGroup.ValueMember = "Key";

    cbGroup.SelectedIndexChanged +=
cbGroups_SelectedIndexChanged;
}

private void dataGridView1_CellValueChanged(object sender,
DataGridViewCellEventArgs e)
{
    if (dataGridView1.Columns[e.ColumnIndex] is
DataGridViewComboBoxColumn)
    {
        var comboBoxCell =
dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex];
        string newMark = comboBoxCell.Value.ToString();
        Guid assignedSubjectId =
(Guid)dataGridView1.Rows[e.RowIndex].Cells[0].Value;

        LoadHelper.SetNewMark(assignedSubjectId, newMark);
    }
}
}
}

```

Листинг Б.5 – Файл GroupStatisticView.cs

```

using StudentTaskTracker.Controllers;
using StudentTaskTracker.MainApp.Teachers.BaseUC;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace StudentTaskTracker.MainApp.Teachers
{
    public partial class GroupStatisticView :
BaseUserControlStatistic
    {
        public GroupStatisticView()
        {
            InitializeComponent();
        }
        protected override void OnLoad(EventArgs e)
        {
            base.OnLoad(e);
            mainForm.BlockButton("Статистика группы");
        }

        protected override void LoadData()
        {
            dataGridView1.Columns.Add("Mark", "Оценка");
            dataGridView1.Columns["Mark"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

            var groups =
StatisticController.GetGroups(SessionManager.UserID);

            cbGroup.SelectedIndexChanged -=
cbGroups_SelectedIndexChanged;

            cbGroup.DataSource = new BindingSource(groups, null);
            cbGroup.DisplayMember = "Value";
            cbGroup.ValueMember = "Key";

            cbGroup.SelectedIndexChanged +=
cbGroups_SelectedIndexChanged;

            var subjects =
StatisticController.GetSubjects(SessionManager.UserID);

```

```

        cbSubject.SelectedIndexChanged -=
cbSubject_SelectedIndexChanged;

        cbSubject.DataSource = new BindingSource(subjects,
null);

        cbSubject.DisplayMember = "Value";
        cbSubject.ValueMember = "Key";

        cbSubject.SelectedIndexChanged +=
cbSubject_SelectedIndexChanged;
    }
}
}

```

Листинг Б.6 – Файл ProgressView.cs

```

using StudentTaskTracker.Controllers;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace StudentTaskTracker
{
    public partial class ProgressView : BaseUserControlView
    {
        public ProgressView()
        {
            dataGridView1.CellPainting +=
dataGridView1_CellPainting;
        }

        protected override void OnLoad(EventArgs e)
        {
            base.OnLoad(e);

            mainForm.BlockButton("Задания");
        }

        protected override void CreateTable()
        {
            dataGridView1.Columns.Add("ID", "");
            dataGridView1.Columns["ID"].Visible = false;

            dataGridView1.Columns.Add("Subject", "Предмет");
        }
    }
}

```

```

        dataGridView1.Columns["Subject"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

        dataGridView1.Columns.Add("Type", "Вид");
        dataGridView1.Columns["Type"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

        dataGridView1.Columns.Add("Progress", "Прогресс");

        dataGridView1.Columns.Add("Percent", "В %");
        dataGridView1.Columns["Percent"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

        dataGridView1.Columns.Add("Ratio", "В числах");
        dataGridView1.Columns["Ratio"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

dataGridView1.Columns.Add(LoadHelper.ColumnButton("Tasks",
"Задания"));
        dataGridView1.Columns["Tasks"].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
    }

    protected override void LoadData()
    {
        var progressList =
ProgressViewController.GetUserProgress(SessionManager.UserID);

        foreach (var item in progressList)
        {
            int rowIndex = dataGridView1.Rows.Add();
            var row = dataGridView1.Rows[rowIndex];

            row.Cells["ID"].Value = item.AssignedSubjectID;
            row.Cells["Subject"].Value = item.SubjectName;
            row.Cells["Type"].Value = item.Type;
            row.Cells["Progress"].Value = item.Progress;
            row.Cells["Percent"].Value = item.Percent;
            row.Cells["Ratio"].Value = item.Ratio;
        }
    }

    protected override void OnButtonClick(string buttonName,
int rowIndex)
    {
        Guid subjectId = (Guid)dataGridView1["ID",
rowIndex].Value;

        if (buttonName == "Tasks")
        {
            mainForm.LoadView(new
AssignedTasksView(subjectId));

```

```

        }
    }

    private void dataGridView1_CellPainting(object sender,
DataGridViewCellPaintingEventArgs e)
    {
        if (e.ColumnIndex < 0 ||
dataGridView1.Columns[e.ColumnIndex].Name != "Progress" ||
e.RowIndex < 0)
            return;

        e.Handled = true;
        e.PaintBackground(e.CellBounds, true);

        var value =
dataGridView1.Rows[e.RowIndex].Cells[e.ColumnIndex].Value;
        if (value == null || !float.TryParse(value.ToString(),
out float percent))
            return;

        percent = Math.Max(0, Math.Min(1, percent));
        int fillWidth = (int)(e.CellBounds.Width * percent);

        Rectangle fillRect = new Rectangle(e.CellBounds.X + 1,
e.CellBounds.Y + 1, fillWidth - 2, e.CellBounds.Height - 2);
        using (Brush b = new SolidBrush(Color.LightGreen))
        {
            e.Graphics.FillRectangle(b, fillRect);
        }

        e.Paint(e.ClipBounds, DataGridViewPaintParts.Border);
    }
}

```

Приложение В

(обязательное)

Листинг основных классов слоя бизнес-логики

Листинг В.1 – Файл AssignedTaskController.cs

```

using StudentTaskTracker.Models;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentTaskTracker.Controllers
{
    public static class AssignedTaskController
    {
        public static List<AssignedTaskModel>
        GetAssignedTasks(Guid assignedSubjectId, Guid userId)
        {
            using (SqlConnection conn =
                DbConnection.GetOpenConnection())
            {
                var query = @"
                    select
                        ast.AssignedTaskID,
                        t.[Checkpoint],
                        t.[Index],
                        t.[Name],
                        t.FilePath,
                        t.Deadline
                    from
                        AssignedTask ast
                        join Task t on ast.TaskID = t.TaskID
                        join [Subject] s on t.SubjectID =
s.SubjectID
                        join AssignedSubject asb on
s.SubjectID = asb.SubjectID
                    where
                        asb.AssignedSubjectID =
@assignedSubjectId
                        and ast.UserID = @userId
                        and ast.[Status] != 2
                    order by
                        t.[Index]";

                using (SqlCommand cmd = new SqlCommand(query,
                    conn))
                {
                    cmd.Parameters.AddWithValue("@assignedSubjectId",
                        assignedSubjectId);
                    cmd.Parameters.AddWithValue("@userId",
                        userId);
                }
            }
        }
    }
}

```



```

    {
        static public void MarkAsSeen(Guid userId)
        {
            using (SqlConnection conn =
DbConnection.GetOpenConnection())
            {
                var query = @"update
                                [Notification]
                                set
                                [Status] = 1
                                where
                                [Status] != 1
                                and ReciverID = @userId";

                using (SqlCommand cmd = new SqlCommand(query,
conn))
                {
                    cmd.Parameters.AddWithValue("@userId",
userId);

                    cmd.ExecuteNonQuery();
                }
            }

            static public void DeleteNotification(Guid notifId)
            {
                using (SqlConnection conn =
DbConnection.GetOpenConnection())
                {
                    var query = @"delete from
                                    [Notification]
                                    where
                                    NotificationID =
@notifId";

                    using (SqlCommand cmd = new SqlCommand(query,
conn))
                    {
                        cmd.Parameters.AddWithValue("@notifId",
notifId);

                        cmd.ExecuteNonQuery();
                    }
                }
            }

            static public List<NotificationModel> GetNotification(Guid
userId)
            {
                List<NotificationModel> notifList = new
List<NotificationModel>();

                using (SqlConnection conn =
DbConnection.GetOpenConnection())

```

```

        {
            var query = @"
                select
                    n.NotificationID,
                    n.[Text]
                from
                    [Notification] n
                where
                    n.ReciverID = @userId";

            using (SqlCommand cmd = new SqlCommand(query,
conn))
            {
                cmd.Parameters.AddWithValue("@userId",
userId);
                using (SqlDataReader reader =
cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        NotificationModel model = new
NotificationModel
                        {
                            NotificationID =
(Guid)reader["NotificationID"],
                            Status = false,
                            Text = reader["Text"].ToString(),
                            ReciverID = userId
                        };

                        notifsList.Add(model);
                    }

                    return notifsList;
                }
            }
        }
    }
}

```

Листинг В.3 – Файл StatisticController.cs

```

using OfficeOpenXml;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

```

```

namespace StudentTaskTracker.Controllers
{
    public static class StatisticController
    {
        private static Dictionary<Guid, string> GetDict(string
query, Guid teacherID)
        {
            using (SqlConnection conn =
DbConnection.GetOpenConnection())
            {
                using (SqlCommand cmd = new SqlCommand(query,
conn))
                {
                    cmd.Parameters.AddWithValue("@teacherId",
teacherID);

                    using (SqlDataReader reader =
cmd.ExecuteReader())
                    {
                        Dictionary<Guid, string> dict = new
Dictionary<Guid, string>();
                        while (reader.Read())
                        {
                            Guid subjectId = reader.GetGuid(0);
                            string name = reader.GetString(1);
                            dict.Add(subjectId, name);
                        }

                        return dict;
                    }
                }
            }

            public static void CreateExcel(Guid subjectId)
            {
                using (SqlConnection conn =
DbConnection.GetOpenConnection())
                {
                    var query = @"select
                                sg.[Name] as GroupName,
                                u.LastName,
                                u.UserName,
                                u.MiddleName,
                                s.[Name] as SubjectName,
                                t.[Index],
                                t.[Name] as TaskName,
                                ast.[Status]
                                from
                                AssignedTask ast
                                join [User] u on ast.UserID =
u.UserID

```

```

                                join Task t on ast.TaskID =
t.TaskID                                join [Subject] s on t.SubjectID =
s.SubjectID                                join StudyGroup sg on u.GroupID =
sg.StudyGroupID
                                where
                                    s.SubjectID = @subjectId
                                order by
                                    u.LastName,
                                    t.[Index]";

                                using (SqlCommand cmd = new SqlCommand(query,
conn))
                                {
                                    cmd.Parameters.AddWithValue("@subjectId",
subjectId);

                                    using (SqlDataReader reader =
cmd.ExecuteReader())
                                    {
                                        if (reader.Read())
                                        {
                                            using (var package = new
ExcelPackage())
                                            {
                                                var worksheet =
package.Workbook.Worksheets.Add("МояТаблица");

                                                // Заголовки первой строки
                                                worksheet.Cells["A1"].Value =
"Группа";

                                                worksheet.Cells["A1"].Style.Font.Bold = true;
                                                worksheet.Cells["A1:A2"].Merge =
true;

                                                worksheet.Cells["B1"].Value =
"Обучающийся";

                                                worksheet.Cells["B1"].Style.Font.Bold = true;
                                                worksheet.Cells["B1:B2"].Merge =
true;

                                                worksheet.Cells["C1"].Value =
reader["SubjectName"].ToString();

                                                worksheet.Cells["C1"].Style.Font.Bold = true;

                                                string lastStudent =
$"{reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";

```

```

int row = 3;

while (true)
{
    string group =
reader["GroupName"].ToString();
    worksheet.Cells[row, 1].Value
= group;

    string currentStudent =
"${reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";
    worksheet.Cells[row, 2].Value
= currentStudent;

    for (int i = 3; currentStudent
== lastStudent; i++)
    {
        int statusValue =
Convert.ToInt32(reader["Status"]);
        string status =
statusValue == 2 ? "+" : "-";

        var cell =

        if (cell.Value == null)
        {
            cell.Value =

reader["TaskName"];

        }

        worksheet.Cells[row,

i].Value = status;

        if (reader.Read())
        {
            currentStudent =
"${reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";
        }
        else
        {
            worksheet.Cells[1, 3,
1, worksheet.Dimension.End.Column].Merge = true;

            var fileInfo = new
FileInfo(@"C:\Users\Даниил\Desktop\TaskStatistic.xlsx");

package.SaveAs(fileInfo);

MessageBox.Show("Данные были успешно выгружены в таблицу!",
"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);

```

```

return;
    }
}

row++;

lastStudent =
$"{reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";
    }
    }
    }
    }
    }
    }
    }

public static Dictionary<Guid, string> GetSubjects(Guid
teacherID)
{
    var query = "SELECT SubjectID, [Name] FROM Subject
WHERE TeacherID = @teacherId";
    return GetDict(query, teacherID);
}

public static Dictionary<Guid, string> GetGroups(Guid
teacherID)
{
    var query = @"
        select
            sg.StudyGroupID,
            sg.[Name]
        from
            AssignedSubject asb
            join [User] u on asb.UserID =
u.UserID
            join StudyGroup sg on u.GroupID =
sg.StudyGroupID
            join [Subject] s on asb.SubjectID
= s.SubjectID
        where
            s.TeacherID = @teacherId
        group by
            sg.StudyGroupID,
            sg.[Name]";

    return GetDict(query, teacherID);
}
}
}

```

Листинг В.4 – Файл ProgressViewController.cs

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentTaskTracker.Controllers
{
    public static class ProgressViewController
    {
        public static List<ProgressModel> GetUserProgress(Guid
userId)
        {
            using (SqlConnection conn =
DbConnection.GetOpenConnection())
            {
                var query = @"select
                                asb.AssignedSubjectID,
                                s.[Name] as subjectName,
                                st.[Name] as typeName,
                                COUNT(ast.AssignedTaskID) AS
TotalTasks,
                                SUM(CASE WHEN ast.Status = 2 THEN
1 ELSE 0 END) as CompletedTasks
                                from
                                    AssignedSubject asb
                                join [User] u on asb.UserID =
u.UserID
                                    join [Subject] s on asb.SubjectID
= s.SubjectID
                                    join SubjectType st on
asb.SubjectTypeID = st.SubjectTypeID
                                    join Mark m on asb.Mark = m.MarkID
                                    join AssignedTask ast on u.UserID
= ast.UserID
                                where
                                    u.UserID = @userId
                                group by
                                    asb.AssignedSubjectID,
                                    s.[Name],
                                    st.[Name]";

                using (SqlCommand cmd = new SqlCommand(query,
conn))
                {
                    cmd.Parameters.AddWithValue("@userId",
userId);

                    using (SqlDataReader reader =
cmd.ExecuteReader())
                    {

```

```
List<ProgressModel>();  
        while (reader.Read())  
        {  
            int completed =  
Convert.ToInt32(reader["CompletedTasks"]);  
            int total =  
Convert.ToInt32(reader["TotalTasks"]);  
            float progress = total > 0 ?  
(float)completed / total : 0;  
  
            ProgressModel progressModel = new  
ProgressModel  
            {  
                AssignedSubjectID =  
(Guid)reader["AssignedSubjectID"],  
                SubjectName =  
reader["subjectName"].ToString(),  
                Type =  
reader["typeName"].ToString(),  
                Progress = progress,  
                Percent = $"{(int)(progress *  
100)}%",  
                Ratio = $"{completed} / {total}"  
            };  
  
            list.Add(progressModel);  
        }  
  
        return list;  
    }  
}  
}
```

Приложение Г

(обязательное)

Листинг основных классов слоя доступа к данным

Листинг Г.1 – Файл DbConnection.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;

namespace StudentTaskTracker
{
    public static class DbConnection
    {
        private static readonly string connectionString = "" +
            "Data Source=localhost;" +
            "Initial Catalog=StudentTaskTracker;" +
            "Integrated Security=True;";

        static public SqlConnection connection = new
        SqlConnection(connectionString);

        static public SqlConnection GetOpenConnection()
        {
            var connection = new SqlConnection(connectionString);
            connection.Open();
            return connection;
        }
    }
}
```

Листинг Г.2 – Файл FileController.cs

```
using Supabase.Storage;
using Supabase;
using System.IO;

namespace StudentTaskTracker.Controllers.Controllers
{
    static public class FileController
    {
        private static readonly string url =
        "https://twqyouductuamlxtxvywz.supabase.co";
        private static readonly string key =
        "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6InR3cXlvdWRjdHVhbWx4dHh2eXd6Iiwicm9sZSI6ImFub24iLCJpYXQiOiJE3NDkzODQ1NTcsImV4cCI6MjA2NDk2MDU1N30.fQ79Y1l5NSfuOo1ja_meonofePvjAGskh8bHR2w-Y5Y";

        private static SupabaseOptions options = new
        SupabaseOptions
        {

```

```

        AutoConnectRealtime = true
    };

    private static Supabase.Client supabase = new
Supabase.Client(url, key, options);
    private static string bucketName = "files";
    private static string downloadDirectory =
@"..\..\downloads";

    public static async void UploadFile(string fileLocation,
string fileName)
    {
        await supabase.InitializeAsync();

        var filePath = Path.Combine(fileLocation, fileName);

        await supabase.Storage
            .From(bucketName)
            .Upload(filePath, fileName);
    }

    public static async void DownloadFile(string fileName)
    {
        await supabase.InitializeAsync();

        var bytes = await
supabase.Storage.From(bucketName).Download(fileName,
(TransformOptions)null);
        var download = $"{downloadDirectory}\{fileName}";
        File.WriteAllBytes(download, bytes);
    }
}

```

Приложение Д

(обязательное)

Листинг класса выгрузки данных

```

using OfficeOpenXml;
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace StudentTaskTracker.Controllers
{
    public static class StatisticController
    {
        private static Dictionary<Guid, string> GetDict(string
query, Guid teacherID)
        {
            using (SqlConnection conn =
DbConnection.GetOpenConnection())
            {
                using (SqlCommand cmd = new SqlCommand(query,
conn))
                {
                    cmd.Parameters.AddWithValue("@teacherId",
teacherID);

                    using (SqlDataReader reader =
cmd.ExecuteReader())
                    {
                        Dictionary<Guid, string> dict = new
Dictionary<Guid, string>();
                        while (reader.Read())
                        {
                            Guid subjectId = reader.GetGuid(0);
                            string name = reader.GetString(1);
                            dict.Add(subjectId, name);
                        }

                        return dict;
                    }
                }
            }

            public static void CreateExcel(Guid subjectId)
            {
                using (SqlConnection conn =
DbConnection.GetOpenConnection())
                {
                    var query = @"select
                                sg.[Name] as GroupName,
                                u.LastName,

```

```

        u.UserName,
        u.MiddleName,
        s.[Name] as SubjectName,
        t.[Index],
        t.[Name] as TaskName,
        ast.[Status]
    from
        AssignedTask ast
    join [User] u on ast.UserID =
u.UserID
        join Task t on ast.TaskID =
t.TaskID
        join [Subject] s on t.SubjectID =
s.SubjectID
        join StudyGroup sg on u.GroupID =
sg.StudyGroupID
    where
        s.SubjectID = @subjectId
    order by
        u.LastName,
        t.[Index]";

    using (SqlCommand cmd = new SqlCommand(query,
conn))
    {
        cmd.Parameters.AddWithValue("@subjectId",
subjectId);

        using (SqlDataReader reader =
cmd.ExecuteReader())
        {
            if (reader.Read())
            {
                using (var package = new
ExcelPackage())
                {
                    var worksheet =
package.Workbook.Worksheets.Add("МояТаблица");

                    // Заголовки первой строки
                    worksheet.Cells["A1"].Value =
"Группа";

                    worksheet.Cells["A1"].Style.Font.Bold = true;
                    worksheet.Cells["A1:A2"].Merge =
true;

                    worksheet.Cells["B1"].Value =
"Обучающийся";

                    worksheet.Cells["B1"].Style.Font.Bold = true;
                    worksheet.Cells["B1:B2"].Merge =
true;

```

```

                                worksheet.Cells["C1"].Value =
reader["SubjectName"].ToString();

worksheet.Cells["C1"].Style.Font.Bold = true;

                                string lastStudent =
"${reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";

                                int row = 3;

                                while (true)
                                {
reader["GroupName"].ToString();
                                string group =
                                worksheet.Cells[row, 1].Value
= group;

                                string currentStudent =
"${reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";
                                worksheet.Cells[row, 2].Value
= currentStudent;

                                for (int i = 3; currentStudent
== lastStudent; i++)
                                {
                                int statusValue =
Convert.ToInt32(reader["Status"]);
                                string status =
statusValue == 2 ? "+" : "-";

                                var cell =

                                if (cell.Value == null)
                                {
                                    cell.Value =

                                }

                                worksheet.Cells[row,

                                if (reader.Read())
                                {
                                    currentStudent =
"${reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";

                                }
                                else
                                {

```

```

        worksheet.Cells[1, 3,
1, worksheet.Dimension.End.Column].Merge = true;

        var fileInfo = new
FileInfo(@"C:\Users\Даниил\Desktop\TaskStatistic.xlsx");

package.SaveAs(fileInfo);

MessageBox.Show("Данные были успешно выгружены в таблицу!",
"Сообщение", MessageBoxButtons.OK, MessageBoxIcon.Information);
        return;
    }
}

row++;

lastStudent =
$"{reader["LastName"]} {reader["UserName"]}
{reader["MiddleName"]}";
    }
}
}
}
}
}

public static Dictionary<Guid, string> GetSubjects(Guid
teacherID)
{
    var query = "SELECT SubjectID, [Name] FROM Subject
WHERE TeacherID = @teacherId";
    return GetDict(query, teacherID);
}

public static Dictionary<Guid, string> GetGroups(Guid
teacherID)
{
    var query = @"
        select
            sg.StudyGroupID,
            sg.[Name]
        from
            AssignedSubject asb
            join [User] u on asb.UserID =
u.UserID
            join StudyGroup sg on u.GroupID =
sg.StudyGroupID
            join [Subject] s on asb.SubjectID
= s.SubjectID
        where
            s.TeacherID = @teacherId
        group by

```

```
        sg.StudyGroupID,  
        sg.[Name]";  
  
    return GetDict(query, teacherID);  
    }  
    }  
}
```

Приложение Е

(обязательное)

Листинг классов добавления пользователя

Листинг Е.1 – Файл Registration.cs

```

using StudentTaskTracker.Controllers.Controllers;
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.ListView;
using static
System.Windows.Forms.VisualStyles.VisualStyleElement.StartPanel;
using StudentTaskTracker.Controllers;

namespace StudentTaskTracker
{
    public partial class Registration : Form
    {
        public Registration()
        {
            InitializeComponent();
        }

        private void Registration_Load(object sender, EventArgs e)
        {
            LoadHelper.Load("UserRole", cbRole);
            LoadHelper.Load("StudyGroup", cbGroup);
        }

        private void btnRegister_Click(object sender, EventArgs e)
        {
            var fields = new string[]
            {
                txtLogin.Text,
                txtPassword.Text,
                txtLastName.Text,
                txtUserName.Text,
                txtPhoneNumber.Text
            };

            if (cbRole.SelectedItem == null ||
fields.Any(string.IsNullOrEmpty))
            {
                MessageBox.Show("Заполните все обязательные
поля!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}

```

```

        return;
    }

    var email = string.IsNullOrEmpty(txtEmail.Text) ?
null : txtEmail.Text;
    var middleName =
string.IsNullOrEmpty(txtMiddleName.Text) ? null :
txtMiddleName.Text;
    var groupId = cbGroup.SelectedItem == null ? null :
cbGroup.SelectedItem.ToString();

RegistrationController.RegisterUser(cbRole.SelectedItem.ToString()
, fields[0], fields[1], fields[2], fields[3], middleName,
fields[4], email, groupId);

    MessageBox.Show("Пользователь был успешно добавлен!",
"Сообщения", MessageBoxButtons.OK, MessageBoxIcon.Information);
    this.Close();
}
}
}

```

Листинг Е.2 – Файл RegistrationController.cs

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace StudentTaskTracker.Controllers.Controllers
{
    public static class RegistrationController
    {
        public static void RegisterUser(string role, string login,
string password, string lastName, string userName, string
middleName, string phoneNumber, string email, string groupId)
        {
            Guid userId = Guid.NewGuid();
            byte[] salt = LoginController.GenerateSalt();
            byte[] passwordHash =
LoginController.HashPassword(password, salt);
            int roleId = LoadHelper.GetId("UserRole", role);

            using (SqlConnection conn =
DbConnection.GetOpenConnection())
            {
                string query = "INSERT INTO [User] " +
"(UserID, " +

```

```

        "UserLogin," +
        "UserPassword," +
        "Salt," +
        "RoleID," +
        "LastName," +
        "UserName," +
        "MiddleName," +
        "PhoneNumber," +
        "Email," +
        "GroupID)" +
        "VALUES (@userId, @userLogin, @userPassword,
@salt, @roleId, @lastName, @userName, @middleName, @phoneNumber,
@email, @groupId)";

        using (SqlCommand cmd = new SqlCommand(query,
conn))
        {
            cmd.Parameters.AddWithValue("@userId",
userId);
            cmd.Parameters.AddWithValue("@userLogin",
login);
            cmd.Parameters.AddWithValue("@userPassword",
passwordHash);
            cmd.Parameters.AddWithValue("@salt", salt);
            cmd.Parameters.AddWithValue("@roleId",
roleId);
            cmd.Parameters.AddWithValue("@lastName",
lastName);
            cmd.Parameters.AddWithValue("@userName",
userName);
            cmd.Parameters.AddWithValue("@middleName",
string.IsNullOrEmpty(middleName) ? (object)DBNull.Value :
middleName);
            cmd.Parameters.AddWithValue("@phoneNumber",
phoneNumber);
            cmd.Parameters.AddWithValue("@email",
string.IsNullOrEmpty(email) ? (object)DBNull.Value : email);
            cmd.Parameters.AddWithValue("@groupId",
string.IsNullOrEmpty(groupId) ? (object)DBNull.Value :
LoadHelper.GetGuid("StudyGroup", groupId));
            cmd.ExecuteNonQuery();
        }
    }
}
}

```

Приложение Ж
(рекомендуемое)

Атрибуты сущностей концептуальной модели базы данных

Таблица Ж.1 – Атрибуты сущности Пользователь

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID пользователя	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Логин	Логин пользователя	Строковый	100		нет	ivanivanov
3	Пароль	Пароль пользователя	Двоичный	256		нет	p@ssw0rd
4	Роль	Роль пользователя	Числовой		1 – 3	нет	0
5	Фамилия	Фамилия пользователя	Строковый	100	а-Я а-Z , - '	нет	Иванов
6	Имя	Имя пользователя	Строковый	100	а-Я а-Z , - '	нет	Иван
7	Отчество	Отчество пользователя	Строковый	100	а-Я а-Z , - '	да	Иванович
8	Номер телефона	Номер телефона пользователя	Строковый	15	^(+7 8)[0 – 9]{10}\$	нет	+71234567890
9	Эл. почта	Адрес эл. почты пользователя	Строковый	256	^[w]+@[a-z]+\.[a-z]+\$	да	example@mail.ru
10	Учебная группа	Учебная группа пользователя	GUID			да	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3

Таблица Ж.2 – Атрибуты сущности Предмет

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID предмета	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Название	Название предмета	Строковый	100	а-Я а-Z	нет	Программирование
3	Семестр	На каком семестре будет изучаться предмет	Числовой		1 – 14	нет	1

Таблица Ж.3 – Атрибуты сущности Курсовая работа

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID курсовой работы	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Предмет	Предмет, для которого создана курсовая работа	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
3	Адрес файла	Ссылка на файл в хранилище файлов	Строковый	500		нет	https://abc123xyz.supabase.co/storage/v1/object/public/user-files/sample-document.pdf

Таблица Ж.4 – Атрибуты сущности Учебная группа

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID учебной группы	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Название	Название учебной группы	Строковый	10	[1 – 9][0 – 9][А – Я]+[1 – 9]+	нет	21ВП1

Продолжение таблицы Ж.4

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
3	Куратор	Куратор учебной группы	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
4	Дата создания	Дата, с которой учебная группа начинает учиться	Дата-время			нет	01.09.2025
5	Срок обучения	Продолжительность обучения учебной группы	Числовой		4 – 5	нет	4

Таблица Ж.5 – Атрибуты сущности Уведомления

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID задания	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Получатель	Получатель уведомления	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
3	Текст	Текст уведомления	Строковый	100		нет	Ваш ответ принят
4	Статус	Флаг, который проверяет, было ли уведомление просмотрено пользователем	Двоичный		0 – 1	нет	0

Таблица Ж.6 – Атрибуты сущности Задание

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID задания	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3

Продолжение таблицы Ж.6

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
2	Номер	Номер задания	Числовой		1 – 100	нет	1
3	Название	Название задания	Строковый	255		нет	Тестирование методом черного ящика
4	Контрольная точка	Контрольная точка, до которой задание должно быть выполнено	Числовой		1 – 2	нет	1
5	Адрес файла	Ссылка на файл с заданием в хранилище	Строковый	500		нет	https://abc123xyz.supabase.co/storage/v1/object/public/user-files/sample-document.pdf
6	Крайний срок сдачи	Крайний срок сдачи задания	Дата-время			нет	31.09.2025

Таблица Ж.7 – Атрибуты сущности Сообщение

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
1	ID	Уникальный ID задания	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
2	Автор	Автор сообщения	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
3	Получатель	Получатель сообщения	GUID			нет	a4f5c9e0-7d3b-4f89-b6a8-1e2fc43d52a3
4	Адрес файла	Ссылка на файл с ответом в хранилище	Строковый	500		нет	https://abc123xyz.supabase.co/storage/v1/object/public/user-files/sample-document.pdf
5	Комментарий	Комментарий к сообщению	Строковый	100		да	Отправляю задание по первой лабораторной работе

Продолжение таблицы Ж.7

№п/п	Наименование	Описание	Тип данных	Размер	Область допустимых значений	Возможность значения Null	Пример
6	Дата отправки	Дата создания сообщения	Дата-время			нет	20.09.2025

Приложение И
(рекомендуемое)
Спецификации атрибутов таблиц логической модели базы данных

Таблица И.1 – Спецификация таблицы User

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
UserID	Уникальный ID пользователя	unid		PK	да	
UserLogin	Логин пользователя	varchar	100		да	
UserPassword	Пароль пользователя	varbinary	256		да	
Salt	Вспомогательные данные для шифровки пароля	varbinary	128		да	
RoleID	Роль пользователя	tinyint			да	0 – 2
LastName	Фамилия пользователя	varchar	100		да	
UserName	Имя пользователя	varchar	100		да	
MiddleName	Отчество пользователя	varchar	100		нет	
PhoneNumber	Номер телефона	varchar	15		да	^(+7 8)[0 – 9]{10}\$
Email	Адрес эл. почты	varchar	256		нет	^[\\w]+@[a-z]+\\. [a-z]+\$
GroupID	Группа пользователя	unid		FK	нет	

Таблица И.2 – Спецификация таблицы UserRole

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
UserRoleID	Порядковый номер роли	tinyint		PK	да	
Name	Название роли	varchar	50		да	

Таблица И.3 – Спецификация таблицы StudyGroup

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
StudyGroupID	Уникальный ID группы	unid		PK	да	
GroupName	Название группы	varchar	10		да	^[0 – 9]{2}[А-Я]+[0 – 9]+\$
TeacherID	Куратор группы	unid		FK	да	
CreatedAt	Дата создания группы	datetime			да	
StudyDuration	Срок обучения группы в годах	tinyint			да	1 – 6

Таблица И.4 – Спецификация таблицы SubjectType

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
SubjectTypeID	Порядковый номер типа предмета	tinyint		PK	да	
Name	Название типа предмета	varchar	50		да	

Таблица И.5 – Спецификация таблицы Notification

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
NotificationID	Уникальный ID уведомления	unid		PK	да	
Status	Статус уведомления	bit			да	0 – 1
Text	Текст уведомления	varchar	255		да	
ReciverID	Уникальный ID получателя уведомления	unid		FK	да	

Таблица И.6 – Спецификация таблицы Task

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
TaskID	Уникальный ID задания	unid		PK	да	
SubjectID	ID Предмета	unid		FK	да	
Name	Название задания	varchar	255		да	
Deadline	Срок сдачи	datetime			да	
Checkpoint	Контрольная точка	tinyint			да	1 – 2
FilePath	Путь файла	varchar	500		да	
Index	Порядковый номер практического задания	tinyint			да	

Таблица И.7 – Спецификация таблицы CourseWork

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
CourseWorkID	Уникальный ID курсовой работы	unid		PK	да	
SubjectID	ID Предмета	unid		FK	да	
FilePath	ID Файла	unid		FK	да	

Таблица И.8 – Спецификация таблицы AssignedSubject

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
AssignedSubjectID	Уникальный ID связи пользователя с предметом	unid		PK	да	

Продолжение таблицы И.8

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
SubjectID	ID Предмета	unid		FK	да	
UserID	Обучающийся	unid		FK	да	
Mark	Оценка	tinyint		FK	да	
SubjectTypeID	Тип отчетности	tinyint		FK	да	
Semester	Семестр	tinyint			да	1 – 14
ExamDate	Дата аттестации	datetime			да	
CompletedAt	Дата выполнения	datetime			да	

Таблица И.9 – Спецификация таблицы Message

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
MessageID	Уникальный ID сообщения	unid		PK	да	
File	Адрес файла	varchar	255		да	
Comment	Комментарий	varchar	500		нет	
SendAt	Дата отправки	datetime			да	
AuthorID	Уникальный ID отправителя сообщения	unid		FK	да	
ReciverID	Уникальный ID получателя сообщения	unid		FK	да	

Таблица И.10 – Спецификация таблицы AssignedTask

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
AssignedTaskID	Уникальный ID назначенного задания	unid		PK	да	
TaskID	Назначенное задание	unid		FK	да	
UserID	Пользователь, которому было назначено задание	unid		FK	да	
AssignedAt	Дата назначения	datetime			да	
Status	Статус задания	tinyint			да	0 – 2
CompletedAt	Дата выполнения	datetime			да	
Rate	Оценка качества выполнения задания	tinyint			да	0 – 5

Таблица И.11 – Спецификация таблицы TaskReply

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
TaskReplyID	Уникальный ID решения задания	unid		PK	да	
MessageID	Уникальный ID сообщения	unid		FK	да	
AssignedTaskID	Уникальный ID назначенного задания	unid		FK	да	

Таблица И.12 – Спецификация таблицы AssignedCourseWork

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
AssignedCourseWorkID	Уникальный ID назначенной курсовой работы	unid		PK	да	
CourseWorkID	ID Курсовой работы	unid		FK	да	
UserID	ID Пользователя	unid		FK	да	
Semester	Семестр	tinyint			да	1 – 14
AssignedAt	Дата назначения	datetime			да	
Deadline	Крайний срок сдачи	datetime			да	
CompletedAt	Дата выполнения	datetime			да	
Mark	Оценка	tinyint		FK	да	

Таблица И.13 – Спецификация таблицы Mark

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
MarkID	Порядковый номер оценки	tinyint		PK	да	
Name	Название оценки	varchar	20		да	

Таблица И.14 – Спецификация таблицы CourseWorkReply

Название	Описание	Тип	Размер	Роль	Обязательный	Ограничения
CourseWorkID	Уникальный ID решения курсовой работы	unid		PK	да	
MessageID	Уникальный ID сообщения	unid		FK	да	
AssignedCourseWorkID	Уникальный ID назначенной курсовой	unid		FK	да	