

МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ 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»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программная система обучения грамматике с
возможностью проверки знаний**

Студент	<u>28.06.2021</u> (подпись, дата)	<u>Кузнецов Никита Сергеевич</u> (ФИО полностью)
Руководитель	<u>28.06.2021</u> (подпись, дата)	<u>Кольчугина Е.А.</u> (фамилия, инициалы)
Нормоконтролёр	<u>28.06.21</u> (подпись, дата)	<u>Такташкин Д.В.</u> (фамилия, инициалы)

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

Заведующий кафедрой	<u>(подпись)</u>	<u>Макарычев П.П.</u> (фамилия, инициалы)
---------------------	------------------	--

Работа защищена с отметкой 4 (протокол заседания ГЭК от 25.06.21 № 18)

Секретарь ГЭК	<u>(подпись)</u>	<u>Попова Н.А.</u> (фамилия, инициалы)
---------------	------------------	---

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
Вычислительной техники

Кафедра
«МОиПЭВМ»

«Утверждаю»
Заведующий кафедрой
П.П. Макарычев
« 12 » _____ 2021 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Кузнецов Никита Сергеевич гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Программная система обучения грамматике с возможностью проверки знаний

Тема утверждена приказом ПГУ № 0027 а/о от " 21 " января 2020

2. Руководитель работы д.т.н., профессор кафедры «МОиПЭВМ» Кольчугина Е.А.

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

Назначение: программный продукт реализует систему обучения грамматике с
возможностью проверки знаний

Основные функции:

1) Аутентификация и авторизация пользователей

2) Обучение теоретическому материалу по заданной теме

3) Проверка знаний по заданной теме

Технология разработки: ООП

Язык программирования: C#

Инструментальные средства разработки: Visual Studio

Среда исполнения: Windows

4. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) План разработки и оценка бюджета

3) Проектирование программных средств

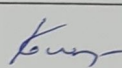
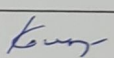
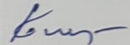
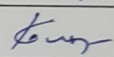
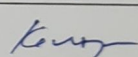
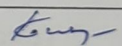
4) Реализация программных средств

5) Тестирование и анализ качества программного кода

6) Рефакторинг программных средств

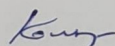
7) Оформление пояснительный записки

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10	14.04.2021	
2	План разработки и оценка бюджета	5	21.04.2021	
3	Проектирование программных средств	30	25.04.2021	
4	Реализация программных средств	25	16.05.2021	
5	Тестирование и анализ качества программного кода	20	30.05.2021	
6	Оформление пояснительный записки	10	14.06.2021	

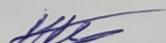
Дата выдачи задания 28.12.2020

Руководитель бакалаврской работы



(подпись, дата)

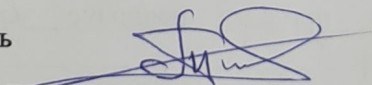
Задание к исполнению принял студент



(подпись, дата)

Работу к защите допустить

Декан факультета



(подпись, дата)

Реферат

Пояснительная записка содержит 101 листов, 15 рисунка, 7 таблиц, 7 использованных источников, 5 приложения.

WINDOWS, ВЕБ ПРИЛОЖЕНИЕ, БАЗЫ ДАННЫХ, MySQL, ПРЕДМЕТНАЯ ОБЛАСТЬ.

Объект исследования – системы образовательного программного обеспечения.

Цель выпускной квалификационной работы – разработка программной системы обучения грамматике с возможностью проверки знаний.

Технология разработки – ООП.

Среда исполнения – Windows.

В качестве базы данных программного сервиса выбрана MySQL. На сегодняшний день это одна из самых популярных и востребованных баз данных и СУБД на рынке. К ее преимуществам можно отнести надежные механизмы транзакций, механизм репликации и легкую масштабируемость.

Клиентское приложение создано с использованием языка C# на основе в среде Visual Studio.

					ПГУ 09.03.04 – 08БР171.11 ПЗ				
Изм.	Лист	№ докум.	Подп.	Дата	Программная система обучения грамматике с возможностью проверки знаний.	Лит.	Лист	Листов	
Разраб.		Кузнецов Н.С.		28.06.21					
Пров.		Кольчугина Е.А.		28.06.21			4	101	
Н. контр.		Такташкин Д.В.		29.06.21		Группа 17ВП1			
Утверд.		Макарычев П.П.		29.06.21					

Содержание

Введение.....	7
1 Анализ предметной области	8
1.1 Образовательное программное обеспечение	8
1.2 Классификация образовательного программного обеспечения.....	9
1.3 Исследование программных продуктов-аналогов.....	12
1.3.1 Moodle.....	13
1.3.2 Google Class	13
1.3.3 iSpring Online	14
1.3.4 Сравнительный анализ аналогичных программных средств	14
1.4 Постановка задачи на разработку программной системы обучения грамматики с возможностью проверки знаний.....	16
1.5 Анализ требований к программным средствам	16
1.5.1 Анализ функциональных требований.....	17
1.5.2 Моделирование бизнес-процессов предметной области	20
1.5.3 Разработка концептуальной модели базы данных	20
1.5.4 Анализ нефункциональных требований.....	23
1.6 Технологии и средства разработки	24
2 Проектирование и развертывание базы данных	25
2.1 Проектирование логической модели базы данных	25
2.2 Проектирование физической модели базы данных.....	26
2.3 Развертывание базы данных	27
3 Реализация приложения	28
3.1 Разработка базы данных.....	28
3.1.1 Разработка запросов к базе данных и их представлений для реализации аналитических бизнес-процессов.....	28

	6
3.1.2 Разработка функций и хранимых процедур	29
3.1.3 Разработка триггеров	29
3.2 Разработка серверного приложения	30
3.3 Разработка клиентской части приложения	30
3.4 Тестирование программного сервиса уделенного управления	32
3.4.1 Выбор режима тестирования	32
3.5 Анализ качества кода	35
3.6 Планирование разработки и оценка бюджета	37
Заключение	41
Список использованных источников	43
Приложение А Глоссарий	44
Приложение Б Концептуальная модель базы данных	46
Приложение В Скрипты для создания объектов базы данных	51
Приложение Г Результаты тестирования	64
Приложение Д Листинг приложения	68

Введение

Целью работы является изучение основных видов и представителей образовательного программного обеспечения. Это позволит оценить плюсы и минусы каждой системы и классифицировать системы образовательного программного обеспечения по их назначению.

Выбранная тема является актуальной, т.к. системы образовательного программного обеспечения получили большую популярность в последнее время. Это произошло из-за того, что всё больше людей испытывают потребность в получении дополнительного или расширению уже имеющегося образования. В этом случае автоматизированные программные системы обучения являются более привлекательной альтернативой очной системе образования, так как позволяют значительно экономить время, самостоятельно корректировать направление обучения и расширять знания по мере необходимости.

Для данной работы поставлены следующие задачи:

- ознакомиться с основными принципами работы программных систем обучения;
- провести классификацию программных систем обучения;
- провести сравнительный анализ наиболее распространенных систем образовательного программного обеспечения;
- выделить основные особенности будущей программы.

В данной работе было решено использовать следующие теоретические методы исследования: классификация и сравнительный анализ.

Предметом исследования являются характеристики систем образовательного программного обеспечения.

Объектом исследования являются системы образовательного программного обеспечения.

1 Анализ предметной области

1.1 Образовательное программное обеспечение

Образовательное программное обеспечение - вид программного обеспечения, выполнения образовательных задач. Оно включает в себя различные виды программ, например, интернет-сервисы для изучения языка, программное обеспечение для управления классом, справочное программное обеспечение, образовательные игры и среды и т. д. Подобное программное обеспечение нацелено на повышение эффективности и действенности какой-либо части образования человека.

История создания образовательного программного обеспечения берёт своё начало с 1940-х, когда американские разработчики симуляторов полёта использовали аналоговый компьютер для создания эмуляции показаний бортовых систем самолёта. До середины 70-х годов образовательное ПО напрямую было связано с мейнфреймами, на которых оно и выполнялось. В эти годы основоположниками образовательных компьютерных систем были: система PLATO, разработанная в Университете Иллинойса, и система TICSIT. Стоимость этих ранних терминалов превышала 10.000 долларов, и институты не могли себе позволить приобрести их из-за высокой цены. Языки программирования того времени, такие как BASIC и Logo, тоже можно рассматривать как образовательные системы, рассчитанные на студентов и начинающих пользователей компьютеров. Система PLATO IV, созданная в 1972 году, имела ряд функций, которые позднее стали неким стандартом в образовательном ПО для персональных компьютеров: растровые изображения, звуковое сопровождение, неклавиатурные устройства ввода, включая сенсорный экран. Создание первых персональных компьютеров оказало огромное влияние на программное обеспечение, и образовательное, в частности. Если до 1975 года пользователи могли получить доступ к компьютеру только в вычислительных центрах или университетах, то теперь компьютеры начали появляться в домах и школах.

1.2 Классификация образовательного программного обеспечения

Системы образовательного программного обеспечения довольно разнообразны, и для успешной разработки подобной системы необходимо ознакомиться с существующими продуктами. В зависимости от назначения системы образовательного программного обеспечения можно классифицировать следующим образом:

- учебные программы;
- учебные пособия;
- программное обеспечение для оценки;
- справочное программное обеспечение;
- образовательное программное обеспечение для специализированных платформ;
- программное обеспечение для корпоративного обучения;
- видеоигровые образовательные системы.

Термин «Учебные программы» первоначально использовался для описания дополнительных учебных материалов, предназначенных как комплекты для учителей или инструкторов или как учебные пособия для студентов, как правило, предназначенное для использования с компьютером. В наше время значение этого термина и его использование расширились и могут относиться ко всему курсу и любым дополнительным материалам. Многие компании используют этот термин для описания всего «пакета», состоящего из курса, объединенного с различными уроками, тестами и другим необходимым материалом. Само учебное ПО может быть представлено в разных форматах: некоторые из них доступны только в виде интернет-ресурсов, а другие могут быть загружены в виде файлов PDF или других типов документов. Многие формы образовательных технологий теперь охватываются термином «учебные программы». Большинство ведущих образовательных компаний предлагают учебное ПО или включают учебные программы в свои учебные комплексы.

Некоторое образовательное программное обеспечение предназначено для использования в условиях школьной аудитории. Обычно такое

программное обеспечение может проецироваться на доску или запускаться одновременно на нескольких настольных компьютерах в классе. Наиболее заметными из них являются доски SMART, которые используют ПО SMART Notebook для взаимодействия с доской, что позволяет использовать ручку-стилус для рукописного ввода на доске. Этот тип программного обеспечения часто называют программным обеспечением для управления классом. Системы образовательного программного обеспечения часто очень специализированы и производятся различными компаниями, в том числе многими известными издателями учебных книг. Программное обеспечение для оценки.

В связи с необходимостью перехода с бумажных носителей информации на цифровые, все больше учебных заведений ищут альтернативные способы оценки и тестирования. Программное обеспечение для оценки позволяет студентам проходить тесты и экзамены с помощью компьютера, как правило подключенного к сети интернет. Затем программное обеспечение оценивает каждую транскрипцию теста и выводит результаты для каждого учащегося. Программное обеспечение для оценки доступно с различными способами доставки, наиболее популярными из которых являются автономное программное обеспечение, онлайн-программное обеспечение и портативные системы голосования. Также существуют проприетарное программное обеспечение и программные системы с открытым исходным кодом. Технически относясь к категории учебных программ, лаборатория оценки навыков является примером компьютерного программного обеспечения для оценки с методологией PPA-2 (Plan, Prove, Assess) для создания и проведения компьютерных онлайн-экзаменов. Moodle - это пример программного обеспечения с открытым исходным кодом с набирающей популярность компонентом оценки. Другие популярные международные системы оценивания - Assessment Master, Google Classroom, Blackboard Learn, EvaluNet XT и Eduroan.

Многие издатели печатных словарей и энциклопедий были вовлечены в производство учебного справочного программного обеспечения с середины

1990-х годов. На рынке эталонного программного обеспечения к ним присоединились как начинающие компании, так и известные издатели программного обеспечения, в первую очередь Microsoft. Первыми коммерческими справочными программными продуктами были оцифрованные печатные издания, часто дополняемые новым мультимедийным контентом, включая сжатое видео и звук. В более поздних продуктах использовались интернет-технологии для дополнения продуктов, выпускаемых на CD-ROM, а в последнее время - для их полной замены. Википедия и ее ответвления (например, Викисловарь) ознаменовали новый поворот в образовательном справочном программном обеспечении. Ранее энциклопедии и словари составляли свое содержание на основе приглашенных и закрытых групп специалистов. Концепция Wiki позволила разработать совместные справочные материалы путем открытого сотрудничества с участием экспертов и рядовых пользователей системы.

Некоторые производители считали обычные персональные компьютеры неподходящей платформой для обучения программному обеспечению для детей младшего возраста и вместо этого производили специализированные, удобные для детей устройства. Аппаратное и программное обеспечение обычно объединяется в единый продукт, например, похожий на детский ноутбук. Самый известный пример - продукция Leapfrog. К ним относятся оригинальные портативные консоли с различными съемными картриджами для образовательных игр и электронные устройства, похожие на книги, в которые можно загружать различные электронные пособия. Эти продукты более портативны, чем портативные компьютеры, но имеют гораздо более ограниченный круг применений, сосредоточенных на образовании.

Ранее образовательное программное обеспечение для важных рынков корпоративного и высшего образования было разработано для работы на одном настольном компьютере (или эквивалентном пользовательском устройстве). В первые годы 21 века многие компании перешли на серверные приложения с высокой степенью стандартизации. Это означает, что образовательное программное обеспечение работает в основном на серверах,

которые могут находиться в сотнях или тысячах миль от реального пользователя. Пользователь получает только небольшие части учебного модуля или теста, поочерёдно передаваемые через Интернет. Серверное программное обеспечение решает, какие учебные материалы распространять, собирает результаты и отображает прогресс для преподавателей. Другой способ выразить это изменение - сказать, что образовательное программное обеспечение превратилось в образовательную онлайн-услугу. McDonald's также экспериментировал с этим в рамках программы разработки программного обеспечения eCrew для игровой приставки Nintendo DS.

Видеоигры можно использовать для обучения пользователей технологической грамотности или облегчения изучения предмета. Эти функции есть в некоторых операционных системах и мобильных телефонах. Ярким примером является Microsoft Solitaire, который был разработан для ознакомления пользователей с использованием графических пользовательских интерфейсов, особенно с мышью и техникой перетаскивания. Mavis Beacon Teaches Typing - широко известная программа со встроенными мини-играми, развивающими навыки набора текста. При создании подобных продуктов используется приём геймификации - использование элементов игрового дизайна в неигровом контексте, которое позволяет воспринимать процесс обучения как развлечение. Элементы игры обеспечивают «мотивационные вознаграждения», позволяющие значительно разнообразить и упростить процесс освоения материала. Примером этому может служить Classcraft, программный инструмент, используемый учителями, в котором наряду с образовательной целью есть игровые элементы.

1.3 Исследование программных продуктов-аналогов

В настоящее время существует множество приложений, предоставляющих возможность удалённого обучения и проверки знаний. Рассмотрим некоторые из них.

1.3.1 Moodle

Moodle - система управления, также известная как система управления обучением или виртуальная обучающая среда. Является аббревиатурой от англ. Modular Object-Oriented Dynamic Learning Environment (модульная объектно-ориентированная динамическая обучающая среда). Представляет собой свободное (распространяющееся по лицензии GNU GPL) веб-приложение, предоставляющее возможность создавать сайты для онлайн-обучения. Первая версия написана 20 августа 2002 года.

Преимущества:

- бесплатный и открытый проект, распространяемый по лицензии GPL;
- поддерживает установку плагинов (дополнений), расширяющих функционал системы;
- отслеживания прогресса учащихся посредством визуализации;
- содержит мощный аппарат тестирования;
- возможность публикации учебного контента различного формата аудио, видео, текст, флэш и т.д.;
- система адаптирована для использования на мобильных устройствах.

Недостатки:

- требует от организации сервера для установки;
- система требовательна к ресурсам сервера;
- требует значительного времени для освоения.

1.3.2 Google Class

Google Class - бесплатный веб-сервис, разработанный Google для школ, который призван упростить создание, распространение и оценку заданий безбумажным способом. Основная цель Google Class – упростить процесс обмена файлами между учителями и учениками.

Преимущества:

- бесплатный проект;

- качественная поддержка;
- использует экосистему google, а значит не потребует регистрации для большого числа пользователей.

Недостатки:

- отсутствие полноценного журнала успеваемости;
- невозможность редактирования заданий после их публикации;
- отсутствие автоматических тестов;
- объём хранилища бесплатного аккаунта ограничен.

1.3.3 iSpring Online

iSpring Online – система для организации курсов и дистанционного обучения. Состоит из нескольких модулей, предназначенных создания курсов и их прохождения.

Преимущества:

- безлимитное хранилище данных;
- широкие возможности персонализации;
- присутствует мобильное приложение;
- присутствует возможность проведения вебинаров, с опциями демонстрации рабочего стола и сохранения записи конференции.

Недостатки:

- система требует платной подписки для каждого пользователя;
- минимальное число пользователей – 50.

1.3.4 Сравнительный анализ аналогичных программных средств

Для того чтобы выполнить сравнительный анализ описанных выше средств удаленного управления компьютерами с собственной разработкой, необходимо определиться с критериями сравнения.

Как уже было отмечено, для системы обучения грамматике необходимы следующие функциональные возможности:

- наличие бесплатной подписки;
- функция вебинаров;

- наличие функции создания тестов;
- наличие мобильной версии.

Сравнение описанных программных средств по выбранным функциональным критериям приведено в таблице 1.

Таблица 1 – Сравнительный анализ программных средств

Системы Критерии	Moodle	Google Class	iSpring Online	Ё-стади
Наличие бесплатной подписки	Присутствует	Присутствует (ограничен объём хранилища данных)	Отсутствует	Присутствует (ограничено количество пользователей и объём хранилища данных)
Функция вебинаров	Присутствует (при установке плагина)	Присутствует	Присутствует	Отсутствует
Наличие функции создания тестов	Присутствует	Отсутствует	Присутствует	Присутствует
Наличие мобильной версии	Присутствует	Присутствует	Присутствует	Отсутствует

Проведенный сравнительный анализ показал, что ни одна из систем не имеет необходимого для сервиса набора функций. Также, большинство из перечисленных систем имеют и прочие недостатки. Можно сделать вывод, что необходима разработка собственного программного средства.

1.4 Постановка задачи на разработку программной системы обучения грамматики с возможностью проверки знаний

Задача обучения грамматике представляет собой процесс предоставления знаний и контроля их усвоения.

Необходимо предоставить ученикам материал для обучения, а также вопросы по нему, по которым будет проверяться их успеваемость. Помимо этого учителя должны иметь возможность просматривать данные об успеваемости учеников.

Сам процесс обучения проходит в 3 этапа:

- ученику предоставляется текст темы учебного материала;
- ученику предлагается ответить на вопрос по теме;
- учитель просматривает и контролирует успеваемость ученика.

Получаем, что для работы с учебными материалами и тестами необходимо организовать:

- web-доступ к системе для просмотра и работы;
- хранение информации об успеваемости учеников;
- возможность отслеживания прогресса учеников

Опираясь на полученные данные, планируется разработать программный продукт, который позволит автоматизировать процесс обучения грамматике.

Созданное программное обеспечение должно быть отлажено и протестировано на наборах данных предметной области.

1.5 Анализ требований к программным средствам

Согласно поставленной задаче, в рамках данной работы необходимо разработать программную систему, способную выполнять следующие информационные задачи:

- 1) обеспечение авторизации пользователей;
- 2) ознакомление с теорией;
- 3) выполнения задания (ответ на вопрос);
- 4) ознакомление с прогрессом учеников;

Категории пользователей разрабатываемой программной системы и соответствующие им информационные задачи представлены в таблице 2.

Таблица 2 – Выделенные категории пользователей

Категория пользователей	Фрагмент структуры предметной области	Информационные задачи
Гость	Обеспечение авторизации в системе.	1
Ученик	Обучение	2, 3
Учитель	Контроль за прогрессом обучения	4

Исходя из всего описанного выше, целью разработки программной системы обучения грамматике с возможностью проверки знаний является создание работоспособного программного сервиса, обладающего функционалом для решения всех описанных выше информационных задач в рамках предметной области и поддерживающей все выделенные категории пользователей.

1.5.1 Анализ функциональных требований

Согласно поставленным информационным задачам, были определены следующие функциональные требования для системы обучения грамматике с возможностью проверки знаний:

- авторизироваться в системе;
- выбрать тему;
- ознакомиться с теорией;
- ответить на вопрос;
- просмотреть список прогресса по темам;

На рисунке 1 представлена диаграмма вариантов использования.



Рисунок 1 – Диаграмма вариантов использования

На диаграмме вариантов использования представлены функциональные требования к разрабатываемой программной системе. Была составлена спецификация для основных прецедентов диаграммы вариантов использования (рисунок 2, 3).

Прецедент: авторизоваться
ID: 1
Краткое описание: Пользователь авторизуется в системе с помощью логина и пароля.
Главные актёры: Пользователь.
Предусловие: Пользователь не авторизован в системе.
Основной поток: 1. Пользователь открывает страницу приложения 2. Пользователь вводит логин и пароль 3. Пользователь отправляет запрос на авторизацию 4. Система возвращает пользователю его роль в системе и токен Альтернативный поток: 4.a. Система возвращает сообщение об ошибке
Постусловие: Пользователь авторизован в системе.

Рисунок 2 – Спецификация прецедента «Авторизоваться в системе»

Прецедент: пройти тему
ID: 2
Краткое описание: Ученик получает теорию, а затем проверяет свои знания по выбранной теме.
Главные актёры: Ученик.
Предусловия: пользователь авторизировался в системе как ученик.
Основной поток: <ol style="list-style-type: none"> 1. Ученик выбирает тему и знакомится с предложенной теорией 2. Ученик отвечает на вопрос по изученной теории 3. Ответ верный, система отмечает тему как пройденную Альтернативный поток: 3.а. Ответ неверный, система предлагает заново ознакомиться с теорией
Постусловие: Тема отмечена как пройденная.

Рисунок 3 – Спецификация прецедента «Пройти тему»

Прецедент: просмотреть список прогресса по темам
ID: 3
Краткое описание: Ученик получает теорию, а затем проверяет свои знания по выбранной теме.
Главные актёры: Учитель.
Предусловия: пользователь авторизировался в системе как учитель.
Основной поток: <ol style="list-style-type: none"> 1. Учитель переходит на страницу списка прогресса прохождения тем 2. Учитель выбирает ученика для просмотра прогресса обучения
Постусловие: Тема отмечена как пройденная.

Рисунок 4 – Спецификация прецедента «Пройти тему»

В результате анализа функциональных требований были выделены варианты использования программы. Для основных вариантов использования была приведена спецификация, которая позволяет понять последовательность действий, которые выполняет система при использовании функций пользователем [8].

1.5.2 Моделирование бизнес-процессов предметной области

Для разрабатываемой программной системы обучения грамматике были выделены бизнес-процессы, представленные в таблице 3.

Таблица 3 – Выделенные бизнес-процессы

№	Бизнес-процесс	Исполнитель	Входные данные		Выходные данные		Ограничения
			Поставщик	Содержание	Потребитель	Содержание	
1	Авторизация	Ученик, учитель	Ученик, учитель	Логин, пароль	Клиентское приложение	Результат авторизации	- Данные о пользователе должны быть в системе
2	Просмотр списка тем и результатов учеников	Ученик, учитель	Ученик, учитель	Запрос	Ученик, учитель	Данные о списке тем и результатах	- Быть авторизованным
3	Просмотр содержания темы	Ученик	Ученик	Запрос	Ученик	Данные о содержании темы	- Быть авторизованным
4	Прохождение тестового задания	Ученик	Ученик	Запрос, ответ на задание	Ученик	Текст вопроса, варианты ответов, правильность выбранного варианта	- Быть авторизованным, - Ученик еще не проходил тестирование по теме

Среди выделенных бизнес-процессов присутствуют аналитические бизнес-процессы, требующие использования представлений и триггеров.

1.5.3 Разработка концептуальной модели базы данных

На основе анализа предметной области и выделенных функциональных требований построена концептуальная модель базы данных.

В концептуальной модели выделены следующие сущности:

- пользователь;

- роль;
- прогресс;
- статус темы;
- тема;
- вопрос;
- ответ.

Подробное описание сущностей разрабатываемой системы представлено в таблице 4.

Таблица 4 – Описание сущностей

№	Наименование	Описание	Количество экземпляров	Режим обновления
1	Вопрос	Список вопросов	100	Обновления производятся очень редко
2	Ответ	Список ответов	400	Обновления производятся очень редко
3	Тема	Список тем и их содержимого	100	Обновления производятся очень редко
4	Прогресс	Прогресс прохождения тем	1000	Обновления производятся часто
5	Статус темы	Список статусов тем	3	Обновления производятся очень редко
6	Пользователь	Список пользователей	200	В среднем раз в месяц
7	Роль	Список ролей пользователей	3	Обновления производятся очень редко

Связи между сущностями описаны в таблице 5.

Таблица 5 – Описание типов связей между сущностями

№	Тип сущности 1	Тип сущности 2	Описание	Мощность	Обязательность	Ограничения
1	Вопрос	Ответ	На один и тот же вопрос может быть дано множество ответов	1:M	Обязательная	

Продолжение Таблицы 5

2	Роль	Пользователь	У множества пользователей могут быть одинаковые роли	1:M	Обязательная	
3	Пользователь	Прогресс	Одному пользователю соответствует	1:M	Обязательная	
4	Ответ	Вопрос	У вопроса может быть только один правильный ответ	1:M	Обязательная	
5	Статус темы	Прогресс	Один статус темы может соответствовать состоянию множества тем	1:M	Обязательная	
6	Тема	Прогресс	Одна тема может проходиться множеством пользователей	1:M	Обязательная	
7	Вопрос	Тема	Один вопрос может подходить к нескольким темам	1:M	Обязательная	

Как предполагает стандарт нотации «сущность-связь», в концептуальной модели отображены мощности связей между сущностями предметной области. Во всех случаях, использован тип связи «один-ко-многим». При использовании данной связи несколько экземпляров зависимой сущности могут принадлежать одному экземпляру родительской сущности. Например, один пользователь может проходить несколько тем.

После проведения структурирования информации в предметной области без учета деталей возможной реализации базы данных была составлена концептуальная модель связей между сущностями в нотации «сущность-

связь». Данная диаграмма представлена на рисунке 4.

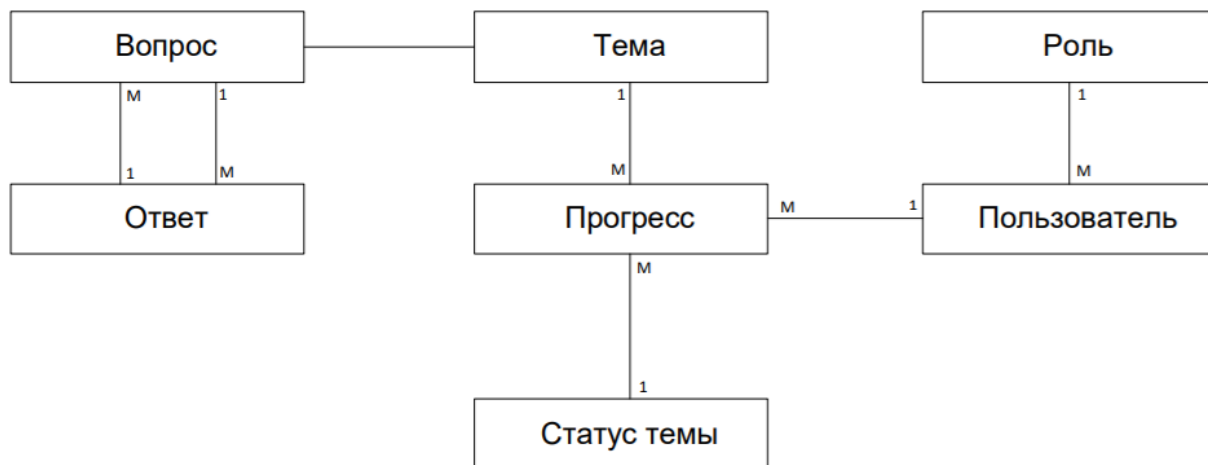


Рисунок 5 – Концептуальная модель базы данных в нотации «сущность-связь»

Последним этапом разработки концептуальной модели было проведено проектирование атрибутов сущностей. На данном уровне моделирования были спроектированы лишь атрибуты, несущие информационную нагрузку для каждой сущности, то есть, не рассматривая фиктивные атрибуты и внешние ключи. Атрибуты сущностей представлены в таблицах 6 – 12.

Таким, образом, созданная концептуальная модель предоставляет исходную формализованную информацию о сущностях и семантики предметной области системы обучения грамматике.

1.5.4 Анализ нефункциональных требований

Качество программного обеспечения является постоянным объектом заботы программной инженерии. Одним из основных критериев качества является надежность. Надежность – набор атрибутов, относящихся к способности программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленный период времени. Требования к надежности разрабатываемого программного средства планирования и анализа расписания можно сформулировать следующим образом:

- ошибки и неправильные действия учителя не должны приводить к критическим нарушениям работы системы, искажению или потере данных;
- ввод любых данных в систему не должен нарушать непротиворечивость и целостность данных в базе;
- система должна предоставлять возможность резервного копирования базы данных;
- система должна предоставлять возможность авторизации пользователя при входе в систему.

Аппаратные средства представляют комплекс технических средств, которые используются для функционирования системы обработки данных, а также устройства, реализующие типовые операции обработки данных.

Для работы сайта с программной системой обучения грамматике с возможностью проверки знаний на сервере должна быть установлена и развернута платформа ASP.NET.

1.6 Технологии и средства разработки

Разработка серверного приложения системы обучения грамматике проводилась с использованием языка программирования C#.

В качестве среды разработки было решено использовать Visual Studio. Данная среда разработки включает в себя возможности работы с данными технологиями, а также она предоставляет возможности подсветки синтаксиса, интеллектуального анализа кода, средства для генерации технической документации и автоматического рефакторинга для C#.

2 Проектирование и развертывание базы данных

При построении программной системы обучения грамматике для хранения учетных данных пользователей, их ролей в системе сервиса, а также данных о прогрессе их обучения необходимо будет хранить в базе данных. База данных может размещаться на одном сервере с системой обучения грамматике, либо же на удаленном сервере. В случае, если сервер базы данных будет развернут на удаленном сервере, потребуется дополнительная конфигурация СУБД для обеспечения доступа к базе данным со сторонних IP адресов.

Для доступа к сервису удобнее всего будет использовать веб-интерфейс в качестве клиентского приложения. Это позволит обеспечить доступ к сервису из любой точки планеты, где есть доступ к интернет-соединению. Также использование веб-приложения позволяет исключить отдельные реализации для каждой новой платформы, что позволяет без дополнительных затрат значительно увеличить охват аудитории.

2.1 Проектирование логической модели базы данных

Используя концептуальную модель, разработанную ранее, была создана логическая модель базы данных с помощью программы проектирования и документирования баз данных Erwin Data Modeler. Полученная логическая модель в виде IDEF1X-диаграммы продемонстрирована на рисунке 6.

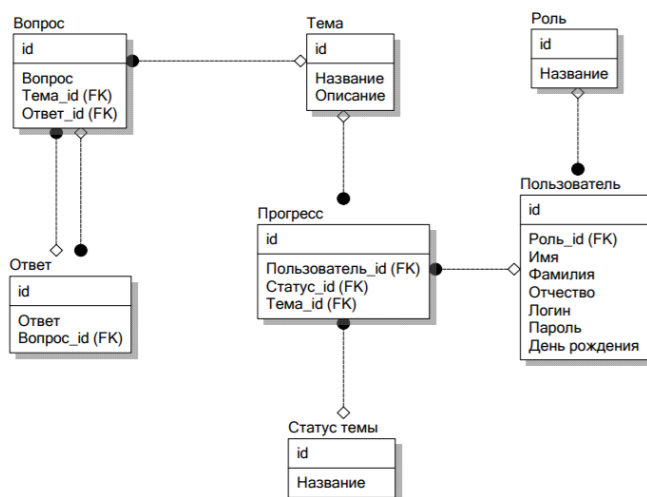


Рисунок 6 – Логическая модель базы данных в виде IDEF1X- диаграмм

Созданная логическая модель является начальным прототипом проектируемой базы данных и описывает отношения, связи между отношениями и атрибуты.

2.2 Проектирование физической модели базы данных

Используя логическую модель базы данных, была создана физическая модель базы данных с помощью программы проектирования и документирования баз данных Erwin Data Modeler. Полученная физическая модель в виде IDEF1X-диаграммы продемонстрирована на рисунке 7.

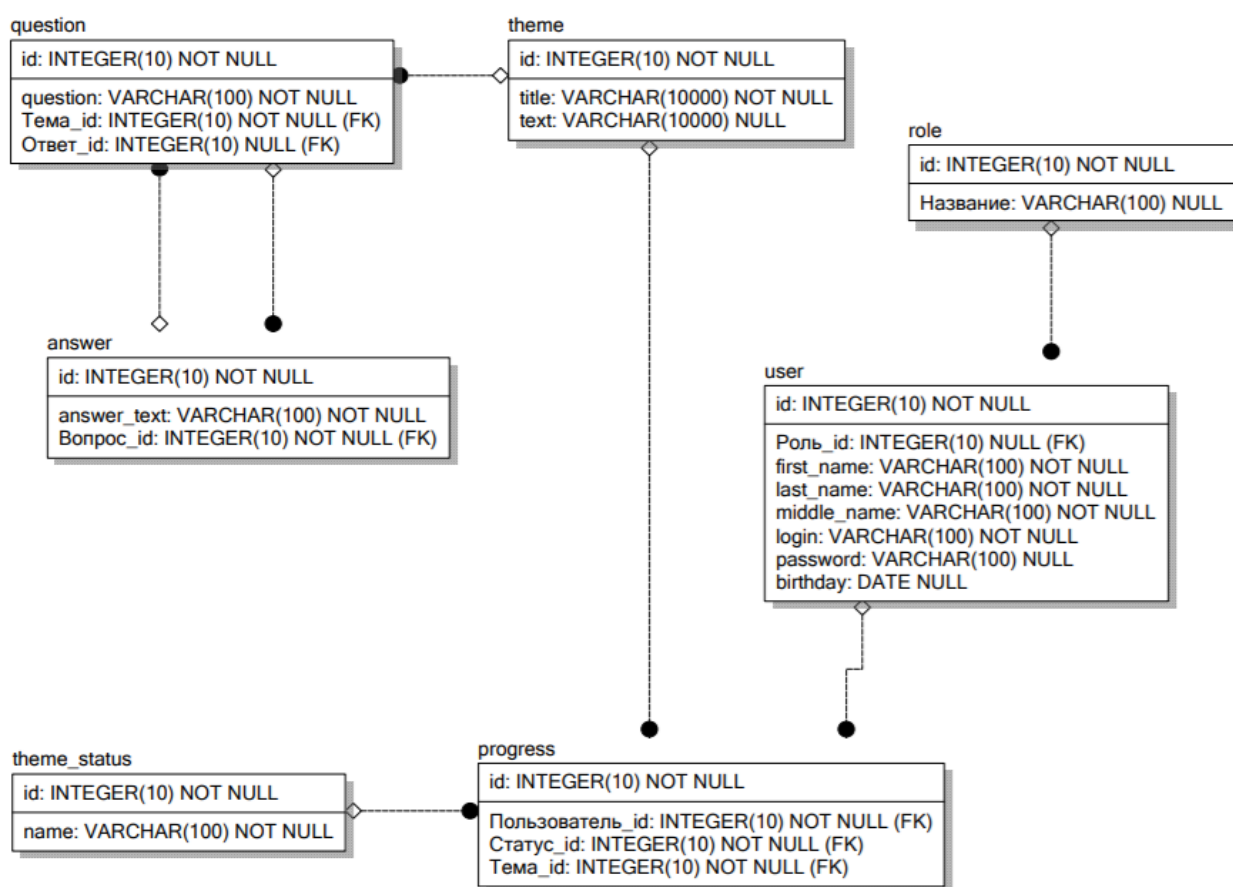


Рисунок 7 – Физическая модель базы данных в виде IDEF1X-диаграммы

Созданная физическая модель базы данных отображает детали физического уровня организации проектируемой базы данных, включая определение типов данных и режимов доступа к данным в специфике выбранной базы данных.

2.3 Развертывание базы данных

В качестве СУБД программной системы выбрана MySQL. На сегодняшний день это одна из самых популярных и востребованных баз данных и СУБД на рынке. К ее преимуществам можно отнести надежные механизмы транзакций, механизм репликации и легкую масштабируемость. Кроме того, установочный пакет СУБД MySQL содержит набор востребованных утилит. Например, утилиты для резервного копирования базы данных или воспроизведения базы данных из сохраненных копий.

Для работы с базой данных через СУБД MySQL использована программа DBeaver, содержащая удобный редактор SQL скриптов, редактор таблиц баз данных с возможностью генерации диаграммы данных.

Используя созданную в среде CA Erwin Data Modeler физическую модель, был создан скрипт создания базы данных на языке SQL, который представлен в приложении А.

Результат выполнения данного скрипта в среде DBeaver представлен на рисунке 8.

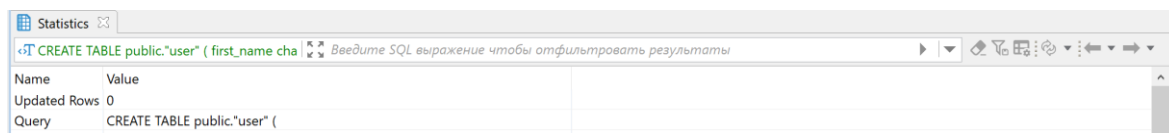


Рисунок 8 – Результат выполнения скрипта создания базы данных

На рисунке 9 представлена диаграмма данных развернутой базы данных, построенная в среде DBeaver.

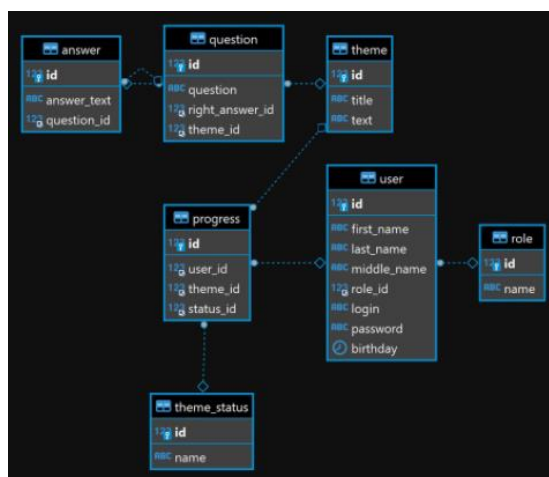


Рисунок 9 – Диаграмма данных развернутой базы данных

Построенная в среде DBeaver диаграмма данных отражает состояние базы данных после выполнения скрипта генерации данной базы данных и также отражает результат создания базы данных.

Диаграмма отражает таблицы и атрибуты с указанием использованных типов данных.

3 Реализация приложения

3.1 Разработка базы данных

3.1.1 Разработка запросов к базе данных и их представлений для реализации аналитических бизнес-процессов

Для удобства вывода информации было разработано представление, отображающее в удобной форме данные о прогрессе ученика. Код представления отражен в листинге 1.

Листинг 1 - представление

```
create or replace
algorithm = UNDEFINED view `grammar`.`student_progress` as
select
    `p`.`id` as `id`,
    `u`.`first_name` as `first_name`,
    `u`.`last_name` as `last_name`,
    `u`.`middle_name` as `middle_name`,
    `t`.`title` as `title`,
    `s`.`name` as `status`
from
    (((`grammar`.`user` `u`
join `grammar`.`progress` `p`)
join `grammar`.`theme` `t`)
join `grammar`.`theme_status` `s`)
where
    ((`p`.`user_id` = `u`.`id`)
and (`p`.`theme_id` = `t`.`id`)
and (`p`.`status_id` = `s`.`id`))
```

Результат обращения к данному представлению показан на рисунке 10.

	123 id	ABC first_name	ABC last_name	ABC middle_name	ABC title	ABC status	
1	1	Margaret	Smith	Ann	Verb to be: ai	complete	
2	2	Margaret	Smith	Ann	Verb to be: q	complete	
3	3	Margaret	Smith	Ann	Present Simp	not complete	

Рисунок 10 – Результат обращения к представлению student_progress

Созданное представление используются для реализации бизнес-процесса просмотра результатов.

3.1.2 Разработка функций и хранимых процедур

Для просмотра данных прогресса конкретного ученика была разработана хранимая процедура progress_for_student. Ее код представлен в приложении в Листинге 2

Листинг 2 - процедура progress_for_student

```
CREATE DEFINER=`root`@`localhost` PROCEDURE
`grammar`.`progress_for_student`(in student_id int)
begin
    select
        `p`.`id` as `id`,
        `t`.`title` as `title`,
        `s`.`name` as `status`
    from
        ((`grammar`.`progress` `p`
    join `grammar`.`theme` `t`)
    join `grammar`.`theme_status` `s`)
    where
        ((`p`.`user_id` = student_id)
        and (`p`.`theme_id` = `t`.`id`)
        and (`p`.`status_id` = `s`.`id`));
end
```

3.1.3 Разработка триггеров

Для бизнес-процесса «Просмотр прогресса» создан триггер, который при добавлении данных в таблицу прогресса учеников проверяет, была ли уже запись по этой теме в таблице. Его код представлен в Листинге 2.

Листинг 2 - триггер insert_in_progress

```
CREATE DEFINER=`root`@`localhost` TRIGGER insert_in_progress
Before INSERT
ON schedule FOR EACH ROW
begin
```

```

DECLARE student_theme_count INT;
set student_theme_count = (select COUNT(*) from progress
where user_id = new.user_id and theme_id = new.theme_id;
if student_theme_count > 0 then
signal sqlstate '45000';
end if;
end

```

Таким образом, при попытке добавить несколько записей для одного ученика и темы, запросившая это действие транзакция будет отменена.

3.2 Разработка серверного приложения

Систему планирования и анализа расписания разместим на сервере IIS от Microsoft. Для проверки приложения во время разработки среда Visual Studio автоматически разворачивает приложение на сервере и запускает его с помощью одного из установленных на компьютере браузеров.

3.3 Разработка клиентской части приложения

Для реализации пользовательского веб-интерфейса и функционала программного средства было создано 19 файлов, часть из которых представляет из себя веб-страницы для взаимодействия с приложением, другая – описание модели данных и подключения к базе данных.

Краткое описание созданных файлов и страниц представлено в таблице 12. Необходимо учесть, что доступ без авторизации возможен только к странице авторизации. Для доступа к другим файлам, имеющим страницы с отображением, необходимо зайти на сайт под одной из учетных записей. Так же некоторые страницы доступны только пользователям с определенными ролями.

Программный код разработанных страниц представлен в приложении Г.

Таблица 5 – Созданные файлы сайта

Файл		Назначение файла
1	2	3
1	Authorization.aspx	Форма авторизации
2	Default.aspx	Домашняя страница с данными пользователя
3	QuestionMaterial.aspx	Форма вопроса

Продолжение Таблицы 5

4	Student.aspx	Форма со списком прогресса для ученика, где он может просмотреть свои результаты и перейти на страницу с материалом темы
5	Study.aspx	Страница, определяющая доступ к страницам прогресса по ролям
6	Teacher.aspx	Форма со списком результатов всех учеников, выводимых для учителя
7	ThemeMaterial.aspx	Материал темы
8	Answer.cs	Модель ответов
9	Progress.cs	Модель прогресса
10	Question.cs	Модель вопроса
11	Role.cs	Модель роли пользователя
12	Theme.cs	Модель темы
13	ThemeStatus.cs	Модель статуса темы
14	User.cs	Модель пользователя
15	DbController.cs	Подключение к БД
16	IController.cs	Интерфейсы для подключения к БД
17	MySQLDbContext.cs	Описание подключения к БД
18	DataBaseManager.cs	Менеджер загрузки данных из БД
19	SessionManager.cs	Менеджер данных сессии

Коротко работу приложения можно описать следующим образом. В форме `Authorization.aspx` пользователь вводит свои логин и пароль и авторизуется в системе. После входа в аккаунт система перенаправляет его на страницу `Default.aspx`, где он может просмотреть записанные в системе данные о себе. С этой страницы он может перейти на страницу просмотра прогресса `Study.aspx`, которая, в зависимости от роли, покажет страницы `Teacher.aspx` или `Student.aspx`. На первой показываются данные о прогрессе всех учеников. На второй – только одного текущего пользователя. Со страницы `Student.aspx` может перейти на страницу с текстом учебного материала `ThemeMaterial.aspx`, откуда, в свою очередь, он может перейти на страницу прохождения вопроса `QuestionMaterial.aspx`, если не проходил тестирование ранее. Листинг приложения представлен в Приложении Д.

3.4 Тестирование программного сервиса уделенного управления

3.4.1 Выбор режима тестирования

Для выбора режима тестирования созданной информационной системы, рассмотрим основные подходы к тестированию программного обеспечения и проанализируем их применимость для данного программного продукта.

Модульное тестирование (Unit tests) – метод низкоуровневого тестирования отдельных модулей или классов программы на правильность выполнения тех или иных функций. Выполняется автоматически, однако существенным минусом является необходимость ручного создания тестов по средствам программного кода. Кроме того, модульное тестирование может не выявить некоторые типы ошибок, например ошибки в содержании данных.

Тестируемая система состоит из двух программных компонентов, что может потребовать написание отдельных модульных тестов для каждого из них, так как модульные тесты не принимают во внимание взаимодействие на уровне системы.

Системное тестирование – это высокоуровневая оценка работоспособности программы или системы в целом. При тестировании система рассматривается по принципу «черного ящика». Данный подход требует создание сложных систем автоматизации, а для небольших программ и информационных архитектур, системное тестирование без автоматизации равносильно функциональному тестированию. Следовательно, данный вид тестирования не подходит для созданной системы.

Интеграционное тестирование – высокоуровневое тестирование, направленное на проверку интеграции модулей системы и их взаимодействия между собой, а также на интеграцию подсистем в одну общую систему. Модули тестируемой системы связаны по средствам HTTP запросов, что является универсальным методом взаимодействия компонентов через сеть. Данная модель связи обладает низким уровнем интеграции и не требует отдельного тестирования связей между компонентами. Ошибки связей между компонентами могут быть покрыты функциональными тестами.

Тестирование производительности, включая нагрузочное и стресс-тестирование. Позволяет оценить работу системы под нагрузкой больших массивов данных и большого количества единовременных запросов. Из недостатков стоит отметить, что данный вид тестирования требует использования дополнительных программных средств для тестирования на отказоустойчивость высоконагруженных систем. На текущей итерации разработки тестируемой системы данными параметрами можно пренебречь. Кроме того, данный вид тестирования не позволяет оценить правильность работы программы с точки зрения функциональных требований.

Юзабилити тестирование позволяет получить данные о качестве пользовательского интерфейса, о его уровне удобства и эргономики с точки зрения пользователя. Данный вид тестирования не подходит для текущего этапа разработки, так как подразумевает, что функции программы должны быть уже протестированы и отлажены.

Функциональное тестирование – вид тестирования, направленный на проверку функциональных требований программы или системы. Может проводиться как в режиме «черного ящика», так и в режиме «белого ящика». В данном случае тестирование полностью имитирует фактическое использование системы и может быть выполнено вручную. Из преимуществ стоит отметить, то, что тестирование может проводиться на ранних этапах разработки программного продукта, над только частично законченными продуктами. Следовательно, выявленные ошибки могут быть исправлены на одном этапе с тестированием.

Перечисленные преимущества функционального тестирования позволяют сделать вывод о том, что применение данного типа тестирования может максимально оперативно дать информацию о стабильности работы системы и выполнении всех функциональных требований.

3.4.3 Тестирование клиентской части приложения

Основываясь на функциональных требованиях, определенных на этапе анализа и на основе бизнес-процессов предметной области был проведен набор функциональных тестов, представленных в таблице 13.

Таблица 13 – Результаты функционального тестирования приложения

Бизнес-процесс (Функц. треб.)	Тест и параметры	Результат
Авторизоваться	Ввести валидные данные Логин test Пароль test	Тест пройден. Открыта страница с данными пользователя.
	Ввести не валидные данные Логин test2 Пароль test1234	Тест пройден. Выведено уведомление.
Просмотреть список прогресса	Просмотреть список прогресса всех учеников	Тест пройден. Список отображен. Рисунок 9.
	Просмотреть список текущего ученика	Тест пройден. Список отображен. Рисунок 10.
Просмотреть материал темы	Выбрать тему	Тест пройден. Тема отображена. Рисунок 11.
Запустить тестирование по теме и ввести вариант ответа	Выбран неправильный вариант ответа	Тест пройден. Вывод сообщение о том, что ответ неправильный. Рисунок 12.
	Выбран правильный вариант ответа	Тест пройден. Вывод сообщения о том, что ответ правильный. Рисунок 13.

Скриншоты результатов тестирования представлены на рисунках Г.1-Г.6.

Результаты функционального тестирования показывают стабильную работу приложения на данном наборе тестов. Приложение выполняет все функциональные требования.

3.5 Анализ качества кода

Для анализа качества кода был вычислен комплексный показатель качества кода. Этот показатель разработан специалистами из Carnegie Mellon Software Engineering Institute. Комплексный показатель качества кода может принимать значения от 0 до 100 и показывает относительную сложность поддержки кода. Чем больше значение этой метрики, тем легче поддерживать код.

Для вычисления комплексного показателя качества кода необходимо подсчитать количество операторов и операндов.

Вычисление показателя качества для всего проекта нецелесообразно из-за сложности вычислений. Также нет необходимости вычислять данный показатель для модулей, обновление которых планируется крайне редко. Поэтому показатель качества кода стоит вычислять для критически важных в проекте модулей или же методов.

В таблице 18 представлены список операторов и операндов для функции получения данных пользователя. Функция представлена в листинге X.

Листинг X – Функция получения данных о пользователе

```
public User GetUserData(string login, string password)
{
    try {
        User user = new User();
        using (MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConectionData)) {
            conn.Open();
            MySqlCommand comm = new MySqlCommand($"select * from
            user_with_role where login='{login}' and password='{password}'", conn);
            var reader = comm.ExecuteReader();

            if (reader.HasRows) {
                while (reader.Read()){
                    user.id = reader.GetUInt32("id");
                    user.first_name =
                    reader.GetString("first_name");
                    user.last_name = reader.GetString("last_name");
                    user.middle_name =
                    reader.GetString("middle_name");
                    user.role_name = reader.GetString("role_name");
                    user.login = reader.GetString("login");
                    user.password = reader.GetString("password");
                    user.birthday = reader.GetString("birthday");
                }
            } else {
                user = null;
            }
        }
        return user;
    }
```

```

    } catch (Exception e) {
        return null;
    }
}

```

Таблица 6 – Данные для расчета комплексного показателя качества кода

Операторы		Операнды	
Оператор	Количество	Операнд	Количество
=	14	password	2
try	1	user	11
;	15	conn	3
""	8	reader	11
while	1	else	1
' '	2	e	1
new	3	null	2
using	1	comm	2
{ }	9	login	2
var	1		
catch	1		
if	1		
return	2		
else	1		

Для вычисления комплексного показателя качества кода необходимо подсчитать следующие переменные:

- η_1 – число уникальных операторов;
- η_2 – число уникальных операндов;
- N_1 – общее число операторов;
- N_2 – общее число операндов;
- CC – структурная сложность программы;
- LoC – количество строк кода в программе.

Основываясь на таблице 17, были подсчитаны необходимые параметры:

- $\eta_1 = 14$;
- $\eta_2 = 9$;
- $N_1 = 60$;
- $N_2 = 35$.

Для вычисления словаря программы необходимо использовать следующую формулу:

$$\eta = \eta_1 + \eta_2.$$

Длина программы вычисляется при помощи данной формулы:

$$N = N_1 + N_2.$$

Таким образом, получаем что словарь программы $\eta = 23$, а длина программы $N = 95$. Используя данные значения, можно подсчитать вычислительную сложность программы при помощи следующей формулы:

$$V = N \cdot \log_2 \eta.$$

После подстановки словаря и длины программы получим, что вычислительная сложность программы $V = 429,4$.

Основываясь на листинге 1, были вычислены показатели:

- структурной сложности программы $CC = 8$;
- количества строк кода $LoC = 29$.

Для вычисления комплексного показателя сложности кода необходимо использовать следующую формулу [N]:

$$MI = \max(0, (171 - 5,2 \cdot \ln(V) - 0,23 \cdot CC - 16,2 \cdot \ln(LoC)) \cdot 100/171).$$

При помощи данной формулы был вычислен комплексный показатель сложности кода $MI = 48$. Показатель является достаточно высоким, что говорит о хорошем качестве кода. Понимание и поддержка разработанного алгоритма не должны вызывать проблем.

3.6 Планирование разработки и оценка бюджета

Планирование разработки и оценка бюджета была выполнена при помощи программы для управления проектами Microsoft Project. Проект был

начат 19 апреля 2021 года. При планировании разработки было выделено 25 задач. Список задач и их описание представлено на рисунке 11.

	i	Название задачи	Длительность	Начало	Окончание	Пре	Названия ресурсов
1		Начало проекта	0 дней	Пн 19.04.21	Пн 19.04.21		
2		«Анализ требований и постановка задачи»	4 дней	Пн 19.04.21	Чт 22.04.21	1	
3		Анализ предметной области	1 день	Пн 19.04.21	Пн 19.04.21		Аналитик[30%];Программист
4		Сравнение программ-аналогов	2 дней	Пн 19.04.21	Вт 20.04.21		Аналитик[70%]
5		Постановка задачи	1 день	Ср 21.04.21	Ср 21.04.21	3;4	Аналитик
6		Анализ функциональных требований	1 день	Чт 22.04.21	Чт 22.04.21	5	Аналитик;Программист
7		Анализ требования и постановка задачи завершена	0 дней	Чт 22.04.21	Чт 22.04.21	6	
8		«Проектирование программных средств»	8 дней	Пт 23.04.21	Вт 04.05.21	2	
9		Проектирование структур данных сервиса	4 дней	Пт 23.04.21	Ср 28.04.21		Аналитик
10		Проектирование REST API приложения	4 дней	Пт 23.04.21	Ср 28.04.21		Программист
11		Проектирование интерфейса	2 дней	Чт 29.04.21	Пт 30.04.21	9;10	Аналитик
12		Проектирование клиентского приложения	2 дней	Пн 03.05.21	Вт 04.05.21	11	Аналитик
13		Проектирование программных средств завершено	0 дней	Вт 04.05.21	Вт 04.05.21	11;12	
14		«Реализация»	12 дней	Ср 05.05.21	Чт 20.05.21	8	
15		Реализация структур данных сервиса	5 дней	Ср 05.05.21	Вт 11.05.21		Программист
16		Реализация REST API приложения	5 дней	Ср 12.05.21	Вт 18.05.21	15	Программист[80%]
17		Реализация интерфейса	2 дней	Ср 12.05.21	Чт 13.05.21	15	Программист[20%]
18		Реализация клиентского приложения	2 дней	Ср 19.05.21	Чт 20.05.21	16;17	Программист
19		Реализация завершена	0 дней	Чт 20.05.21	Чт 20.05.21	18	
20		«Тестирование»	4 дней	Пт 21.05.21	Ср 26.05.21	14	
21		Функциональное тестирование	2 дней	Пт 21.05.21	Пн 24.05.21		Тестировщик
22		Модульное тестирование	2 дней	Вт 25.05.21	Ср 26.05.21	21	Тестировщик
23		Тестирование завершено	0 дней	Ср 26.05.21	Ср 26.05.21	22	
24		Составление и оформление документации	3 дней	Чт 27.05.21	Пн 31.05.21	20	Аналитик;Программист
25		Конец проекта	0 дней	Пн 31.05.21	Пн 31.05.21	24	

Рисунок 11 – Список задач проекта

Для реализации проекта было выделено три трудовых ресурса: аналитик, программист и тестировщик. Описание ресурсов представлено на рисунке 12.

	i	Название ресурса	Тип	Единицы измерения	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на исполн.	Начисление	Базовый календарь
1		Аналитик	Трудовой		А		100%	170,00 Р/ч	340,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
2		Программист	Трудовой		П		100%	160,00 Р/ч	320,00 Р/ч	0,00 Р	Пропорциональное	Стандартный
3		Тестировщик	Трудовой		Т		100%	140,00 Р/ч	280,00 Р/ч	0,00 Р	Пропорциональное	Стандартный

Рисунок 12 – Список ресурсов проекта

В результате выделения задач и необходимых ресурсов проекта была построена диаграмма Ганта (рисунок 13). На диаграмме Ганта представлены задачи продолжительностью и взаимосвязи с между задачами в проекте. Также на диаграмме отображены даты начала и конца каждой задачи и назначенные ресурсы для выполнения задачи.

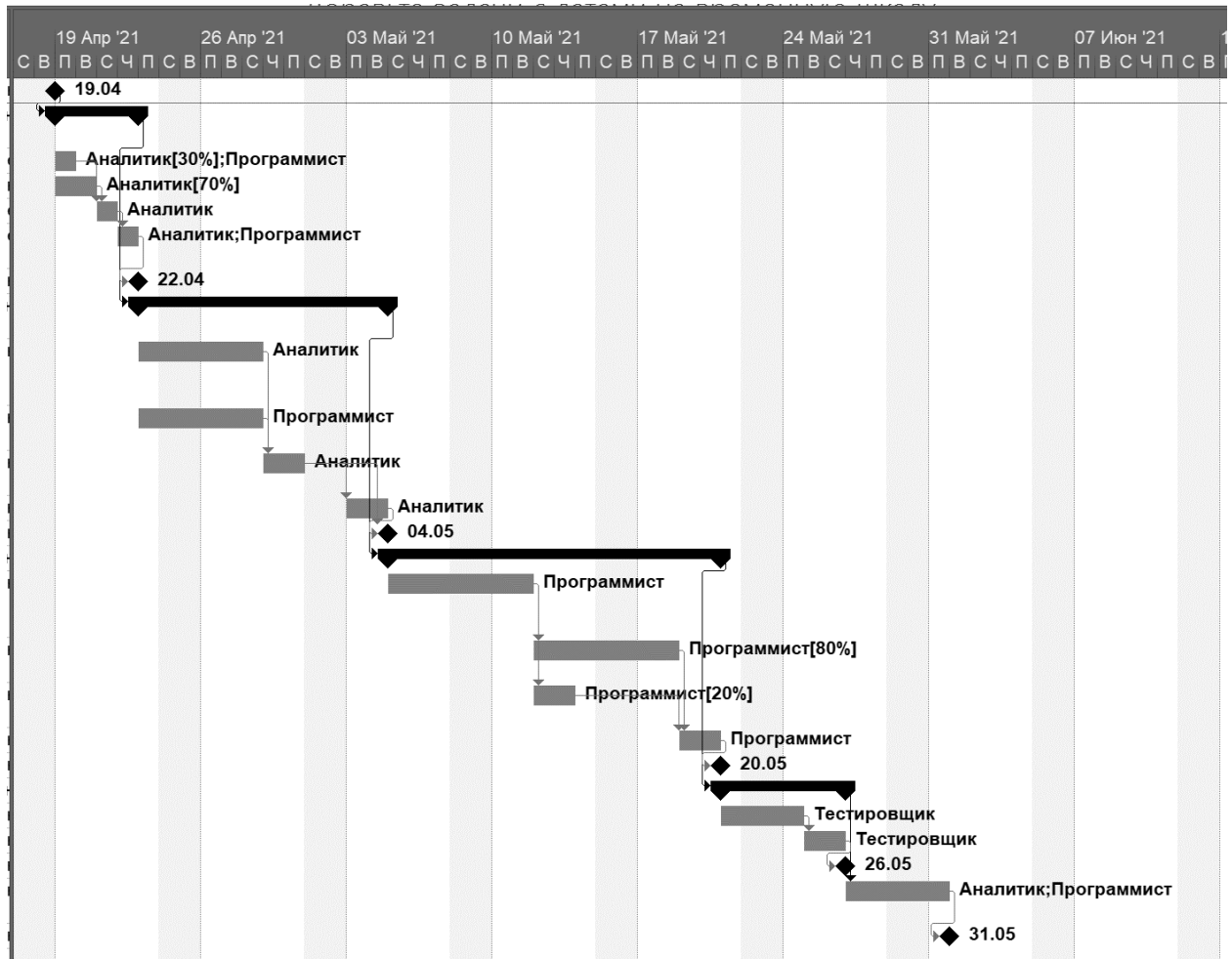


Рисунок 13 – Диаграмма Ганта

На рисунке 14 – 15 представлены гистограмма загрузки ресурсов и гистограмма затрат на задачи соответственно.

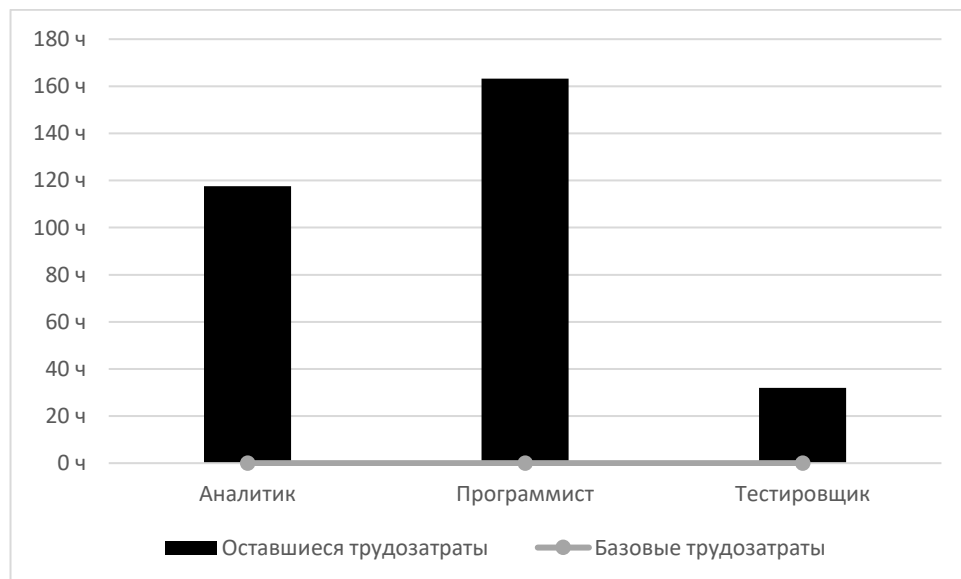


Рисунок 14 – Гистограмма загрузки ресурсов

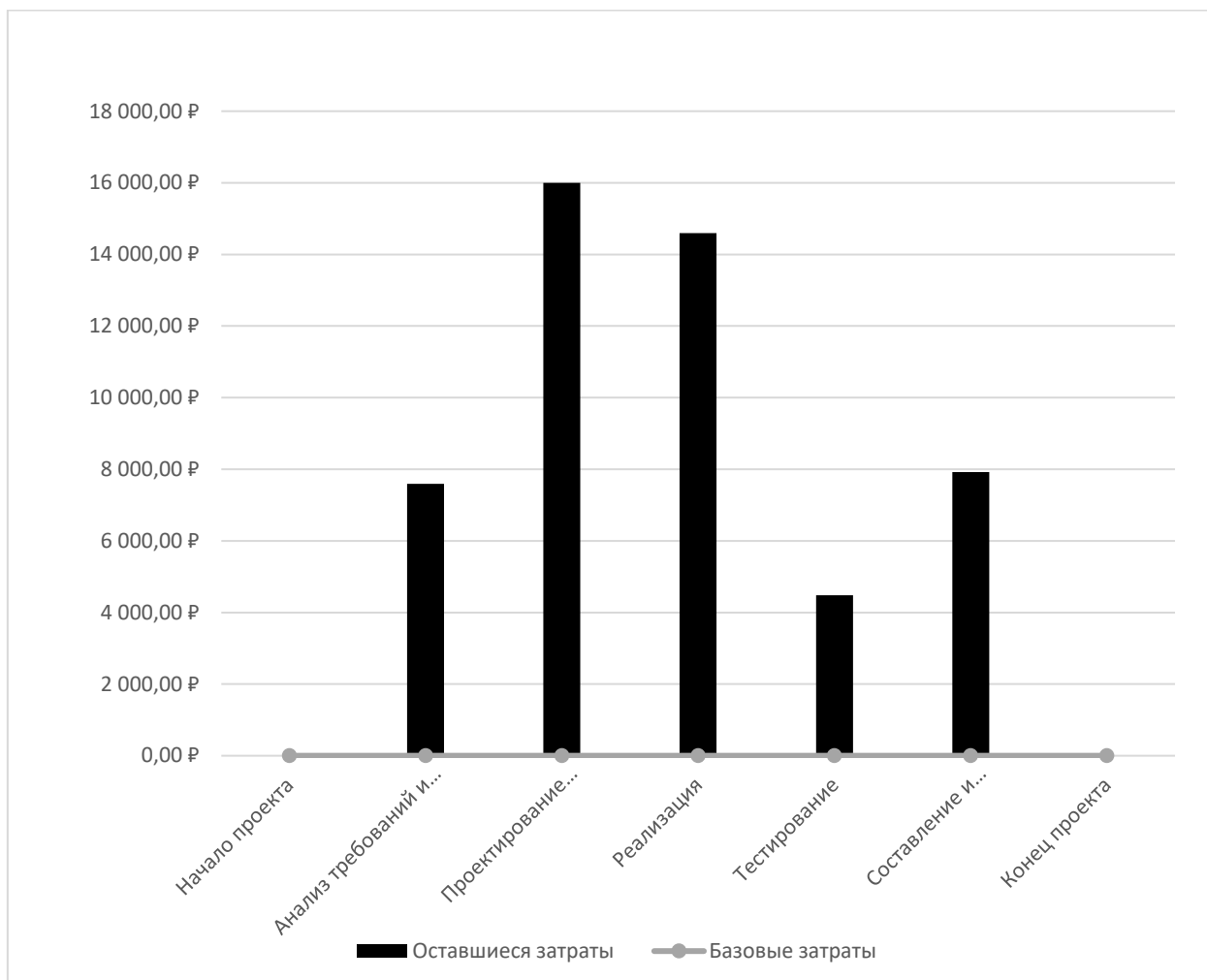


Рисунок 15 – Гистограмма затрат на задачи

Таким образом разработка проекта составит 29 полных рабочих дней с 19 апреля 2021 года по 31 мая 2021 года. Трудозатраты составят 312 человеко-часов. Оценка бюджета проекта составляет 50 584 рубля.

Заключение

В ходе выполнения выпускной квалификационной работы разработан программная система обучения грамматике с возможностью проверки знаний..

В результате научной исследовательской работы был проведен анализ систем удалённого обучения и проверки знаний. Были рассмотрены и охарактеризованы технологии удаленного управления компьютерами.

Протоколы и технологии удаленного управления компьютерными системами были классифицированы. Для каждой категории классификации были приведены описание и примеры.

На этапе анализа предметной области и постановки требований были выделены бизнес-процессы, для решения которых разрабатывалось приложение. Была создана модель данных предметной области. На основе модели данных было проведено логическое и физическое моделирование базы данных. Для базы данных были разработаны запросы, процедуры и триггеры.

Клиентское приложение было разработано с использованием языка программирования C#. Разработка проводилась в среде программирования Visual Studio 2019.

Серверная часть представлена СУБД MySQL и серверным приложением, разработанным на языке программирования C#. СУБД содержит в себе сущности, их атрибуты и связи, задействованные в предметной области и основную бизнес-логику приложения, в виде представлений, хранимых процедур и триггеров.

По всем компонентам полученного приложения было проведено функциональное тестирование. Тестирование показало стабильность работы системы на использованном наборе тестов.

В ходе выполнения выпускной квалификационной работы проведено планирование и оценка бюджета разработки при помощи программы для управления проектами Microsoft Project. Затраты на проект составили 50 584 рубля. Время разработки составило 29 рабочих дней.

Таким образом в результате выполнения выпускной квалификационной работы разработан программная система обучения грамматике с возможностью проверки знаний, который удовлетворяет всем требованиям технического задания.

Список использованных источников

1. Якубайтис Э. А. Информационные сети и системы: Справочная книга / Якубайтис Э. А. – М. : Финансы и статистика, 1996. – 336 с.
2. Хомоненко А. Д. Основы современных компьютерных технологий / Хомоненко А. Д. – СПб.: КОРОНА-Принт, 2009. – 672 с.
3. Что такое облачные вычисления? // Облачная платформа компании Microsoft [Электронный ресурс]. URL: [\(https://azure.microsoft.com/ru-ru/overview/what-is-cloud-computing/#:~:text=%D0%9F%D1%80%D0%BE%D1%89%D0%B5%20%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D1%8F%2C%20%D0%BE%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%20%E2%80%94%D1%8D%D1%82%D0%BE,%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%20\(%D0%22%D0%BE%D0%B1%D0%BB%D0%B0%D0%BA%D0%BE%22\)\)](https://azure.microsoft.com/ru-ru/overview/what-is-cloud-computing/#:~:text=%D0%9F%D1%80%D0%BE%D1%89%D0%B5%20%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D1%8F%2C%20%D0%BE%D0%B1%D0%BB%D0%B0%D1%87%D0%BD%D1%8B%D0%B5%20%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F%20%E2%80%94%D1%8D%D1%82%D0%BE,%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%20(%D0%22%D0%BE%D0%B1%D0%BB%D0%B0%D0%BA%D0%BE%22)) (дата обращения: 21.04.2021).
4. Брайтфельд Т. Системы удаленного управления // Журнал сетевых решений/LAN. 2003. № 04 [Электронный ресурс]. URL: <https://www.osp.ru/lan/2003/04/137451> (дата обращения: 21.04.2021).
5. Титтел Э. TCP/IP / Титтел Э., Хадсон К., Стюарт Дж. М. – СПб. : ПИТЕР, 1999. – 416 с.
6. Точилкина, Т.Е. Принципы создания информационных систем и моделирования бизнес-процессов с использованием пакета программ Allfusion Modeling Suite / Т.Е. Точилкина. – Академия бюджета и казначейства, 2009. – 169 с.
7. Джим Арлоу, Айла Нейштадт / UML2 и Унифицированный процесс. Практический объектно-ориентированный анализ и проектирование – Санкт-Петербург, Изд-во Символ-Плюс, 2007.
8. Вирт Н. Алгоритмы и структуры данных. Новая версия для Оберона + CD / Никлаус Вирт – Москва: ДМК Пресс, 2010 – 272 с

Приложение А
Глоссарий
(справочное)

БД – база данных – совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

Веб-интерфейс – совокупность средств, при помощи которых пользователь взаимодействует с веб-сайтом или веб-приложением через браузер.

Веб-сервер – сервер, принимающий HTTP-запросы от клиентов, обычно веб-браузеров, и выдающий им HTTP-ответы, как правило, вместе с HTML-страницей, изображением, файлом, медиа-поток или другими данными.

Хостинг (англ. hosting) — услуга по предоставлению ресурсов для размещения информации на сервере, постоянно находящемся в сети (обычно Интернет).

СУБД – система управления базами данных – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

API – программный интерфейс приложения - набор готовых классов, процедур, функций, структур и констант, предоставляемых для использования во внешних программных продуктах.

Алгоритм – конечная совокупность точно заданных правил решения произвольного класса задач или набор инструкций, описывающих порядок действий исполнителя для решения некоторой задачи.

Приложение Б
Концептуальная модель базы данных
(Рекомендуемое)

Таблица Б.1 – Атрибуты сущности «Вопрос»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор вопроса	Код вопроса	Целочисленный	10	0-9999999999	Нет	PK	1234567890	
Вопрос	Текст вопроса	Символьный	100	A–я, A-z, “-“, “ ”, “ “, “ _	Нет		Hello, what _____ your name?	
Идентификатор темы	Код темы	Целочисленный	10	0-9999999999	Нет	FK	1234567890	
Идентификатор ответа	Код ответа	Целочисленный	10	0-9999999999	Нет	FK	1234567890	

Таблица Б.2 – Атрибуты сущности «Ответ»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор ответа	Код ответа	Целочисленный	10	0-9999999999	Нет	PK	1234567890	
Текст ответа	Текст ответа	Символьный	100	A–я, A-z, “-“, “ ”, “	Нет		is	
Идентификатор вопроса	Код вопроса	Целочисленный	10	0-9999999999	Нет	FK	1234567890	

Таблица Б.3 – Атрибуты сущности «Тема»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор темы	Код темы	Целочисленный	10	0-9999999999	Нет	PK	1234567890	
Заголовок	Название темы	Символьный	10000	A-я, A-z, “-“, “,” “	Нет		Verb to be: am, is, are	
Текст	Теория по теме	Целый	10000	A-я, A-z, “-“, “,” “	Нет			

Таблица Б.4 – Атрибуты сущности «Прогресс»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор прогресса	Код прогресса	Целочисленный	10	0-9999999999	Нет	PK	1234567890	
Идентификатор пользователя	Код пользователя	Целочисленный	10	0-9999999999	Нет	FK	1234567890	
Идентификатор статуса	Код статуса темы	Целочисленный	10	0-9999999999	Нет	FK	1234567890	
Идентификатор темы	Код темы	Целочисленный	10	0-9999999999	Нет	FK	1234567890	

Таблица Б.5 – Атрибуты сущности «Статус темы»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор статуса темы	Код статуса темы	Целочисленный	10	0-9999999999	Нет	РК	1234567890	
Имя статуса	Название статуса темы	Символьный	100	A-z, A-я	Нет		Complete	

Таблица Б.6 – Атрибуты сущности «Роль»

Наименование	Описание	Тип данных	Размерность	Область допустимых значение	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор роли	Код роли	Целочисленный	10	0-9999999999	Нет	РК	1234567890	
Название	Название роли	Символьный	100	A-z	Нет		admin	

Таблица Б.7 – Атрибуты сущности «Пользователь»

Наименование	Описание	Тип данных	Размерность	Область допустимых значения	Возможность значения NULL	Роль	Пример	Примечание
Идентификатор пользователя	Код пользователя	Целочисленный	10	0-9999999999	Нет	PK	1234567890	
Идентификатор роли	Код роли	Целочисленный	10	0-9999999999	Нет	FK	1234567890	
Имя	Имя пользователя	Символьный	100	A-я, A-z, “-“,	Нет		Петр	
Фамилия	Фамилия пользователя	Символьный	100	A-я, A-z, “-“,	Нет		Иванов	
Отчество	Отчество пользователя	Символьный	100	A-я, A-z, “-“,	Нет		Семёнович	
Логин	Системное имя пользователя	Символьный	100	A-z, 0-9	Нет		petrivanov001	
Пароль	Пароль пользователя	Символьный	100	A-z, 0-9	Да		abc123	
День рождения	Дата рождения пользователя	Дата		01.01.1950 – 01.01.2040	Нет		16.03.1998	

Приложение В
Скрипты для создания объектов базы данных
на языке PostgreSQL
(рекомендуемое)

```

--

-- Table structure for table `answer`

--

DROP TABLE IF EXISTS `answer`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `answer` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `answer_text` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `question_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `answer_FK` (`question_id`),
  CONSTRAINT `answer_FK` FOREIGN KEY (`question_id`) REFERENCES
`question` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=10 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Dumping data for table `answer`

--

LOCK TABLES `answer` WRITE;

/*!40000 ALTER TABLE `answer` DISABLE KEYS */;

INSERT INTO `answer` VALUES
(1, '1', 1), (2, '2', 1), (3, '3', 1), (4, '1', 2), (5, '2', 2), (6, '3', 2), (7, '4', 3),
(8, '5', 3), (9, '6', 3);

/*!40000 ALTER TABLE `answer` ENABLE KEYS */;

UNLOCK TABLES;

--

```

```
-- Table structure for table `progress`
```

```
--
```

```
DROP TABLE IF EXISTS `progress`;
```

```
/*!40101 SET @saved_cs_client      = @@character_set_client */;
```

```
/*!40101 SET character_set_client = utf8 */;
```

```
CREATE TABLE `progress` (
```

```
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
```

```
  `user_id` int(10) unsigned NOT NULL,
```

```
  `theme_id` int(10) unsigned NOT NULL,
```

```
  `status_id` int(10) unsigned NOT NULL,
```

```
  PRIMARY KEY (`id`),
```

```
  KEY `progress_FK` (`user_id`),
```

```
  KEY `progress_FK_1` (`theme_id`),
```

```
  KEY `progress_FK_2` (`status_id`),
```

```
  CONSTRAINT `progress_FK_2` FOREIGN KEY (`status_id`) REFERENCES  
  `theme_status` (`id`),
```

```
  CONSTRAINT `progress_FK` FOREIGN KEY (`user_id`) REFERENCES `user`  
  (`id`),
```

```
  CONSTRAINT `progress_FK_1` FOREIGN KEY (`theme_id`) REFERENCES  
  `theme` (`id`)
```

```
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
```

```
COLLATE=utf8_unicode_ci;
```

```
/*!40101 SET character_set_client = @saved_cs_client */;
```

```
--
```

```
-- Dumping data for table `progress`
```

```
--
```

```
LOCK TABLES `progress` WRITE;
```

```
/*!40000 ALTER TABLE `progress` DISABLE KEYS */;
```

```
INSERT INTO `progress` VALUES (1,2,1,1),(2,2,2,1),(3,2,3,2);
```

```

/*!40000 ALTER TABLE `progress` ENABLE KEYS */;

UNLOCK TABLES;

--
-- Table structure for table `question`
--

DROP TABLE IF EXISTS `question`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `question` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `question` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
  `right_answer_id` int(10) unsigned DEFAULT NULL,
  `theme_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`),
  KEY `question_FK_1` (`theme_id`),
  KEY `question_FK` (`right_answer_id`),
  CONSTRAINT `question_FK` FOREIGN KEY (`right_answer_id`) REFERENCES
`answer` (`id`),
  CONSTRAINT `question_FK_1` FOREIGN KEY (`theme_id`) REFERENCES
`theme` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `question`
--

LOCK TABLES `question` WRITE;

/*!40000 ALTER TABLE `question` DISABLE KEYS */;

```

```

INSERT INTO `question` VALUES (1,'1',1,1),(2,'question
2?',4,2),(3,'question 3?',7,3);

/*!40000 ALTER TABLE `question` ENABLE KEYS */;

UNLOCK TABLES;

--

-- Table structure for table `role`
--

DROP TABLE IF EXISTS `role`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `role` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Dumping data for table `role`
--

LOCK TABLES `role` WRITE;

/*!40000 ALTER TABLE `role` DISABLE KEYS */;

INSERT INTO `role` VALUES (1,'teacher'),(2,'student');

/*!40000 ALTER TABLE `role` ENABLE KEYS */;

UNLOCK TABLES;

--

-- Temporary table structure for view `student_progress`

```

```
--

DROP TABLE IF EXISTS `student_progress`;

/*!50001 DROP VIEW IF EXISTS `student_progress`*/;

SET @saved_cs_client      = @@character_set_client;
SET character_set_client = utf8;

/*!50001 CREATE TABLE `student_progress` (
  `id` int(10) unsigned,
  `first_name` varchar(100),
  `last_name` varchar(100),
  `middle_name` varchar(100),
  `title` varchar(10000),
  `status` varchar(100)
) ENGINE=MyISAM */;

SET character_set_client = @saved_cs_client;

--
-- Table structure for table `theme`
--

DROP TABLE IF EXISTS `theme`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `theme` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `title` varchar(10000) COLLATE utf8_unicode_ci NOT NULL,
  `text` varchar(10000) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;
```



```

    `name` varchar(100) COLLATE utf8_unicode_ci NOT NULL,

    PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `theme_status`
--

LOCK TABLES `theme_status` WRITE;

/*!40000 ALTER TABLE `theme_status` DISABLE KEYS */;

INSERT INTO `theme_status` VALUES (1,'complete'),(2,'not complete');

/*!40000 ALTER TABLE `theme_status` ENABLE KEYS */;

UNLOCK TABLES;

--
-- Table structure for table `user`
--

DROP TABLE IF EXISTS `user`;

/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;

CREATE TABLE `user` (
    `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
    `first_name` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
    `last_name` varchar(100) COLLATE utf8_unicode_ci NOT NULL,
    `middle_name` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
    `role_id` int(10) unsigned DEFAULT NULL,
    `login` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
    `password` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,

```

```

    `birthday` date DEFAULT NULL,

    PRIMARY KEY (`id`),

    KEY `user_FK` (`role_id`),

    CONSTRAINT `user_FK` FOREIGN KEY (`role_id`) REFERENCES `role`
    (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8
COLLATE=utf8_unicode_ci;

/*!40101 SET character_set_client = @saved_cs_client */;

--

-- Dumping data for table `user`

--

LOCK TABLES `user` WRITE;

/*!40000 ALTER TABLE `user` DISABLE KEYS */;

INSERT INTO `user` VALUES
(1,'Gorden','Freeman','11',1,'11','111','1998-01-
01'),(2,'Margaret','Smith','Ann',2,'22','222','2004-01-01');

/*!40000 ALTER TABLE `user` ENABLE KEYS */;

UNLOCK TABLES;

--

-- Temporary table structure for view `user_with_role`

--

DROP TABLE IF EXISTS `user_with_role`;

/*!50001 DROP VIEW IF EXISTS `user_with_role`*/;

SET @saved_cs_client      = @@character_set_client;
SET character_set_client = utf8;

/*!50001 CREATE TABLE `user_with_role` (
    `id` int(10) unsigned,
    `first_name` varchar(100),

```

```

`last_name` varchar(100),
`middle_name` varchar(100),
`birthday` date,
`login` varchar(100),
`password` varchar(100),
`role_name` varchar(100)
) ENGINE=MyISAM */;

SET character_set_client = @saved_cs_client;

--
-- Dumping routines for database 'grammar'
--

/*!50003 DROP PROCEDURE IF EXISTS `progress_for_student` */;
/*!50003 SET @saved_cs_client      = @@character_set_client */ ;
/*!50003 SET @saved_cs_results    = @@character_set_results */ ;
/*!50003 SET @saved_col_connection = @@collation_connection */ ;
/*!50003 SET character_set_client = utf8mb4 */ ;
/*!50003 SET character_set_results = utf8mb4 */ ;
/*!50003 SET collation_connection = utf8mb4_general_ci */ ;
/*!50003 SET @saved_sql_mode      = @@sql_mode */ ;
/*!50003 SET sql_mode              =
'STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION' */ ;

DELIMITER ;;

/*!50003 CREATE*/ /*!50020 DEFINER=`root`@`localhost`*/ /*!50003
PROCEDURE `progress_for_student`(in student_id int)

begin

    select

        `p`.`id` as `id`,

        `t`.`title` as `title`,

        `s`.`name` as `status`

from

    ((`grammar`.`progress` `p`

```

```

join `grammar`.`theme` `t`)
join `grammar`.`theme_status` `s`)
where
    ((`p`.`user_id` = student_id)
    and (`p`.`theme_id` = `t`.`id`)
    and (`p`.`status_id` = `s`.`id`));
end *;;

DELIMITER ;

/*!50003 SET sql_mode            = @saved_sql_mode */ ;
/*!50003 SET character_set_client = @saved_cs_client */ ;
/*!50003 SET character_set_results = @saved_cs_results */ ;
/*!50003 SET collation_connection = @saved_col_connection */ ;

--
-- Final view structure for view `student_progress`
--

/*!50001 DROP TABLE IF EXISTS `student_progress`*/;
/*!50001 DROP VIEW IF EXISTS `student_progress`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results        = @@character_set_results */;
/*!50001 SET @saved_col_connection    = @@collation_connection */;
/*!50001 SET character_set_client      = utf8mb4 */;
/*!50001 SET character_set_results     = utf8mb4 */;
/*!50001 SET collation_connection      = utf8mb4_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `student_progress` AS select `p`.`id` AS
`id`,`u`.`first_name` AS `first_name`,`u`.`last_name` AS
`last_name`,`u`.`middle_name` AS `middle_name`,`t`.`title` AS
`title`,`s`.`name` AS `status` from (((`user` `u` join `progress` `p`)
join `theme` `t`) join `theme_status` `s`) where ((`p`.`user_id` =

```

```

`u`.`id`) and (`p`.`theme_id` = `t`.`id`) and (`p`.`status_id` =
`s`.`id`)) */;

/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

--
-- Final view structure for view `user_with_role`
--

/*!50001 DROP TABLE IF EXISTS `user_with_role`*/;
/*!50001 DROP VIEW IF EXISTS `user_with_role`*/;
/*!50001 SET @saved_cs_client          = @@character_set_client */;
/*!50001 SET @saved_cs_results         = @@character_set_results */;
/*!50001 SET @saved_col_connection     = @@collation_connection */;
/*!50001 SET character_set_client      = utf8mb4 */;
/*!50001 SET character_set_results     = utf8mb4 */;
/*!50001 SET collation_connection      = utf8mb4_general_ci */;
/*!50001 CREATE ALGORITHM=UNDEFINED */
/*!50013 DEFINER=`root`@`localhost` SQL SECURITY DEFINER */
/*!50001 VIEW `user_with_role` AS select `u`.`id` AS
`id`,`u`.`first_name` AS `first_name`,`u`.`last_name` AS
`last_name`,`u`.`middle_name` AS `middle_name`,`u`.`birthday` AS
`birthday`,`u`.`login` AS `login`,`u`.`password` AS
`password`,`r`.`name` AS `role_name` from (`user` `u` join `role` `r`)
where (`u`.`role_id` = `r`.`id`) */;

/*!50001 SET character_set_client      = @saved_cs_client */;
/*!50001 SET character_set_results     = @saved_cs_results */;
/*!50001 SET collation_connection      = @saved_col_connection */;

/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;

```

```
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;  
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;  
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;  
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;  
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;  
  
-- Dump completed on 2021-06-21 22:17:49
```

Приложение Г
Результаты тестирования
(рекомендуемое)

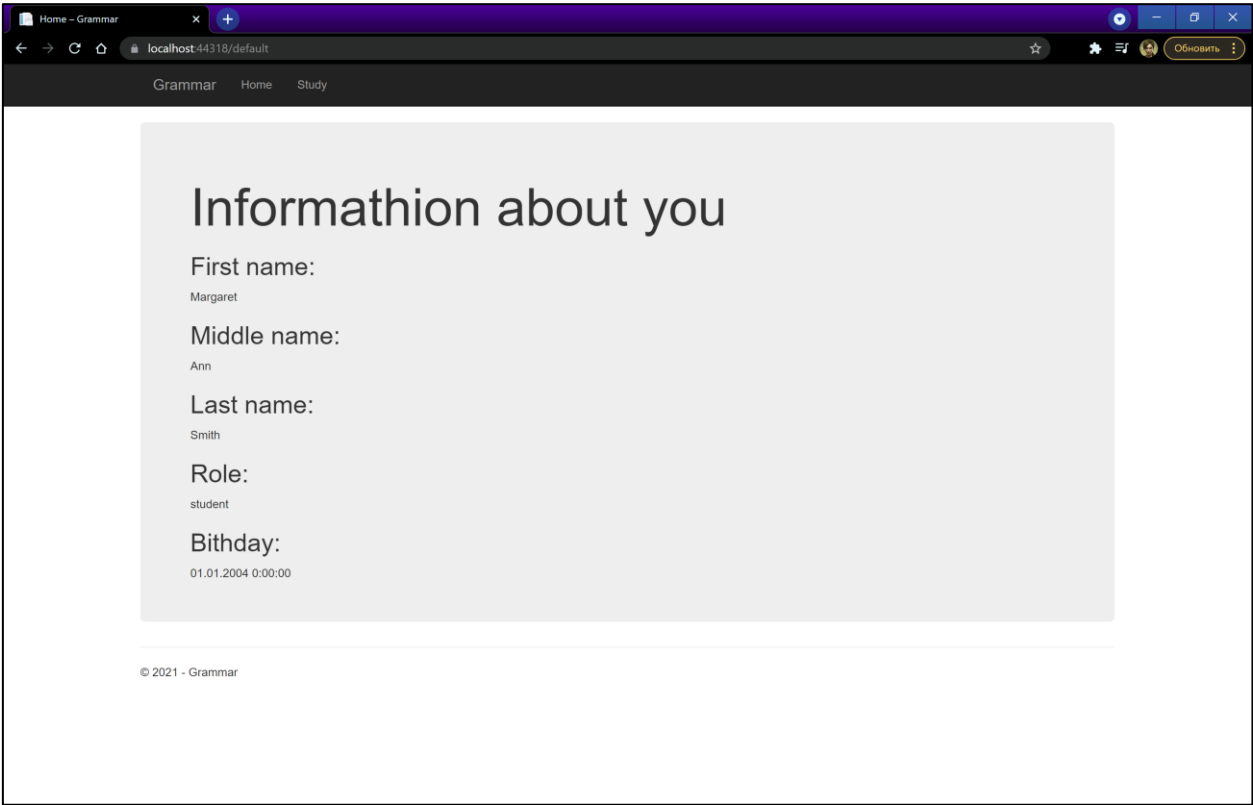


Рисунок Г.1 – Результат тестирования авторизации

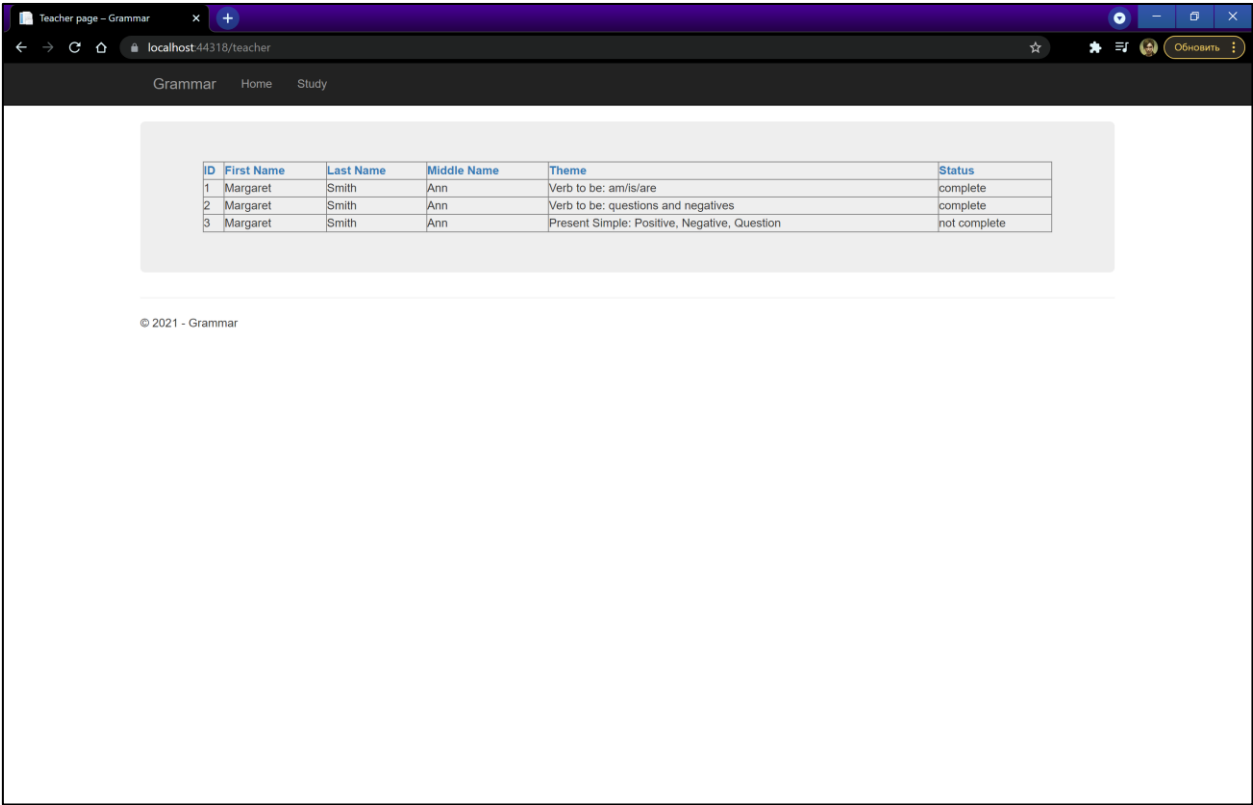


Рисунок Г.2 – Результат тестирования прогресса для учителей

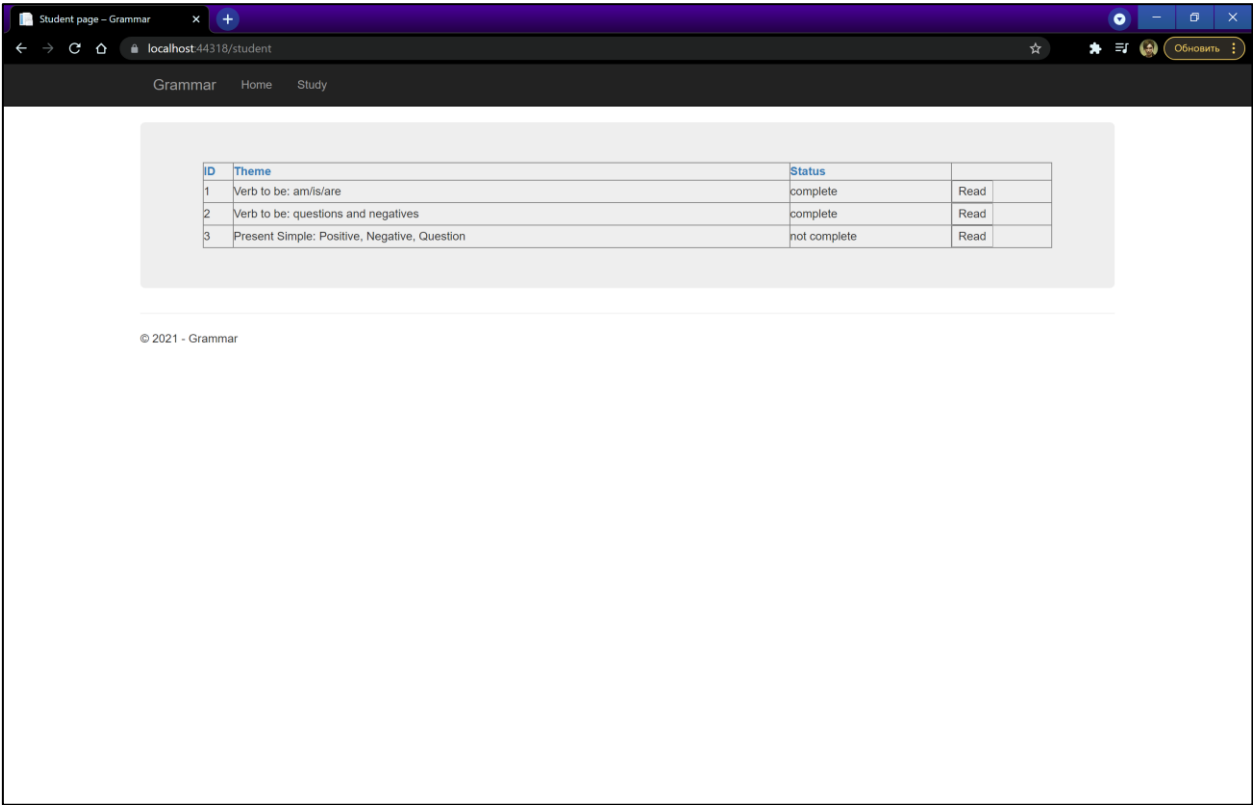


Рисунок Г.3 – Результат тестирования прогресса для учеников

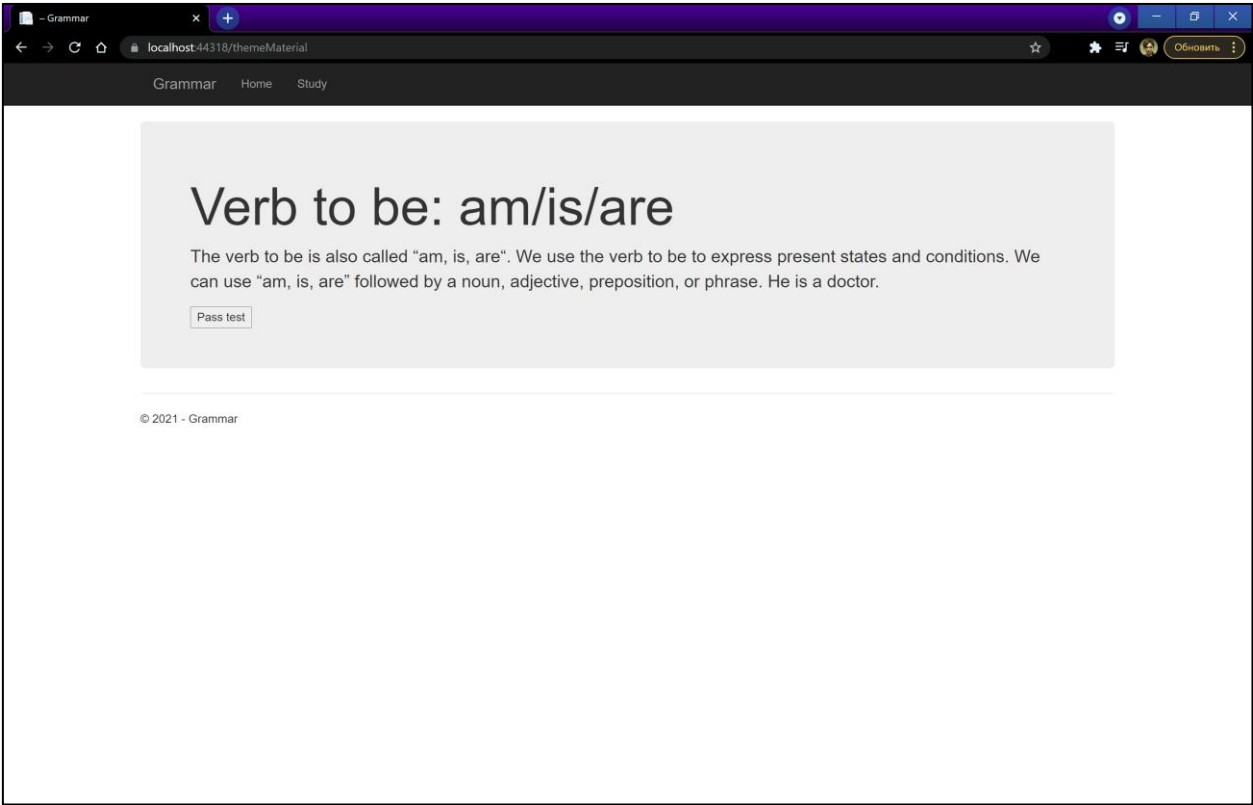


Рисунок Г.4 – Результат тестирования просмотра материала темы

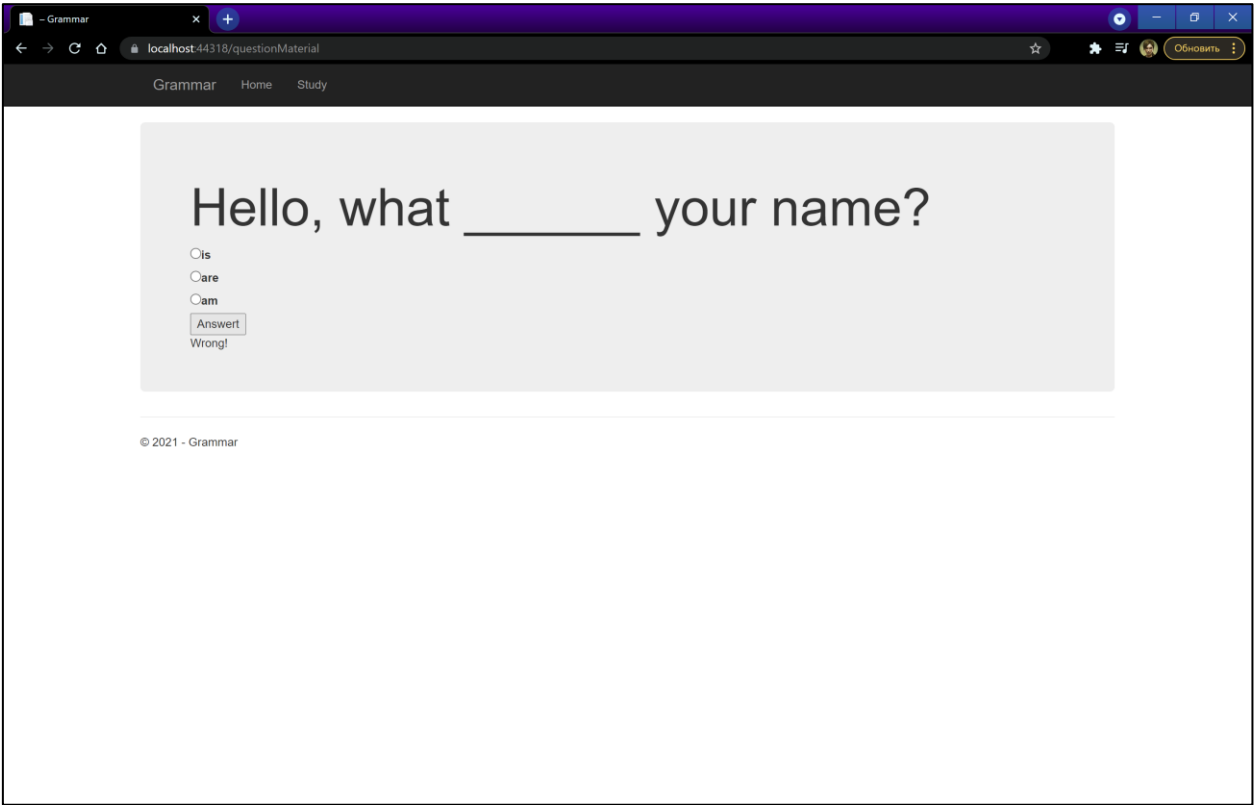


Рисунок Г.5 – Результат тестирования прохождения теста с неправильным ответом

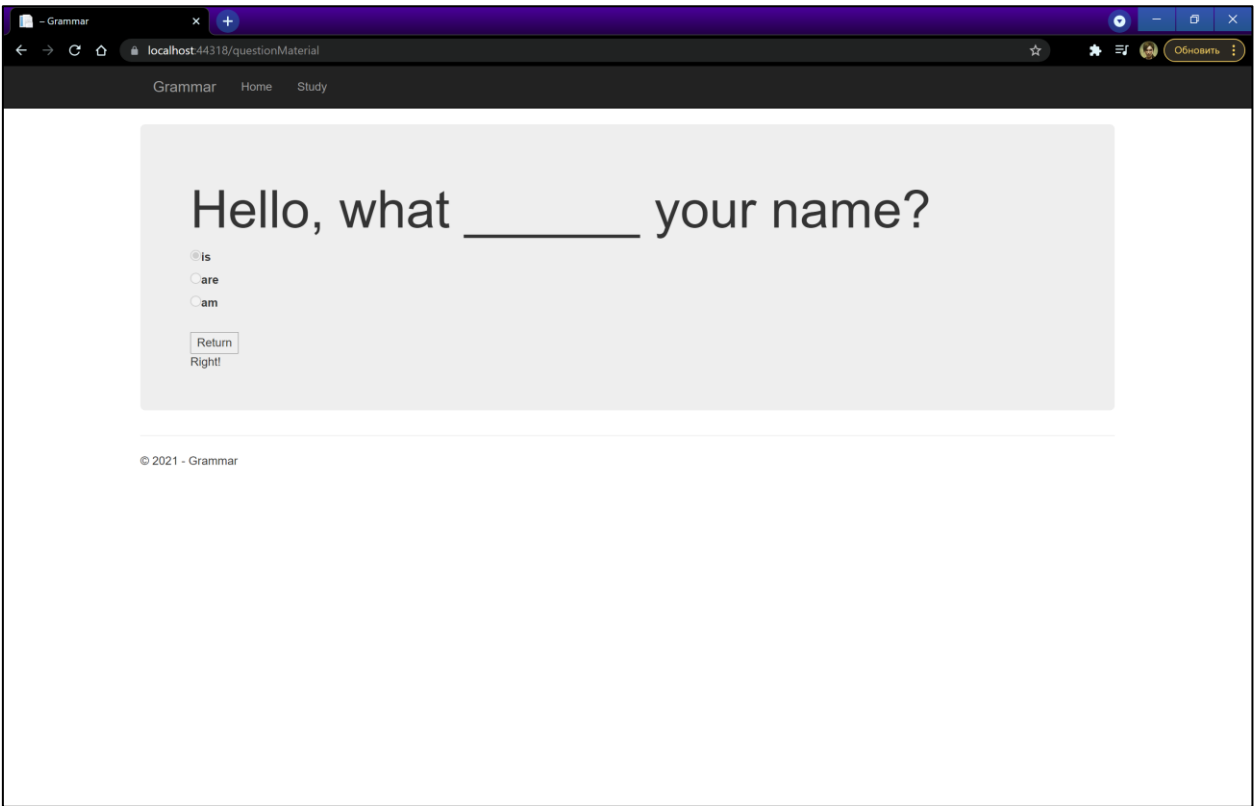


Рисунок Г.6 – Результат тестирования прохождения теста с правильным ответом

Приложение Д
Листинг приложения
(рекомендуемое)

Листинг Д.1 - Default.aspx

```

<%@ Page Title="Home" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.cs" Inherits="Grammar._Default" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

    <div class="jumbotron">
        <h1>Information about you </h1>
        <h2>First name:</h2> <%= GetUserFirstName()%>
        <h2>Middle name:</h2> <%= GetUserMiddleName()%>
        <h2>Last name:</h2> <%= GetUserLastName()%>
        <h2>Role:</h2> <%= GetUserRole()%>
        <h2>Birthday:</h2> <%= GetUserBirthday()%>
    </div>

</asp:Content>

using Grammar.Managers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Grammar
{
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (SessionData.user == null)
            {
                Response.Redirect("authorization.aspx");
            }
        }

        protected string GetUserFirstName()
        {
            return SessionData.user.first_name;
        }

        protected string GetUserMiddleName()
        {
            return SessionData.user.middle_name;
        }

        protected string GetUserLastName()
        {
            return SessionData.user.last_name;
        }

        protected string GetUserRole()
        {
            return SessionData.user.role_name;
        }

        protected string GetUserBirthday()
        {
            return SessionData.user.birthday;
        }
    }
}

```

Листинг Д.2 – Файл Authorization.aspx

```

<%@ Page Title="Authorization" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Authorization.aspx.cs" Inherits="Grammar.Contact" %>

    <asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
        <div class="jumbotron">
            <h1><%: Title %></h1>
            <h2>Login:</h2>
            <asp:TextBox
runat="server" ID="userLogin"
OnTextChanged="OnLoginChange"/>
            <h2>Password:</h2>
            <asp:TextBox
runat="server" ID="password"
OnTextChanged="OnPasswordChange"/>
            <br />
            <br />
            <asp:Button runat="server" id="AuthorizationButton"
Text="Authorization" onclick="OnAutorizationButtonClick"/>
        </div>
    </asp:Content>
using Grammar.Managers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Grammar
{
    public partial class Contact : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void OnAutorizationButtonClick(object sender, EventArgs e)
        {
            if (SessionData.login != "" && SessionData.password != "")
            {
                SessionData.user =
DataBaseControllersManager.UserController.GetUserData(SessionData.login,
SessionData.password);
                Response.Redirect("default.aspx");
            }
        }

        protected void OnLoginChange(object sender, EventArgs e)
        {
            var textBox = (TextBox)sender;
            SessionData.login = textBox.Text;
        }

        protected void OnPasswordChange(object sender, EventArgs e)
        {
            var textBox = (TextBox)sender;
            SessionData.password = textBox.Text;
        }
    }
}

```

Листинг Д.3 - QuestionMaterial.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/Site.Master"
CodeBehind="QuestionMaterial.aspx.cs" Inherits="Grammar.QuestionMaterial" %>

    <asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

```

```

<div class="jumbotron">
    <h1> <% = GetCurrentQuestion() %></h1>

    <asp:RadioButtonList ID="RadioButtonAnswersList" runat="server">
    </asp:RadioButtonList>

        <% if(!right.HasValue || !right.Value) { %>
            <asp:Button          runat="server"          id="Answer"          Text="Answer"
onclick="OnAnswerButtonClick"/>
        <% } %>
        <br />
        <% if(right.HasValue && !right.Value) { %>
            Wrong!
        <% } %>
        <% if(right.HasValue && right.Value) { %>
            <asp:Button          runat="server"          id="Return"          Text="Return"
onclick="Return_Click"/>
        <br />
            Right!
        <% } %>

    </div>

</asp:Content>
using Grammar.Managers;
using Grammar.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Grammar
{
    public partial class QuestionMaterial : System.Web.UI.Page
    {
        private Question _question;
        protected List<Answer> _answers;
        protected bool? right;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (SessionData.user == null)
            {
                Response.Redirect("authorization.aspx");
            }
            else
            {
                _question
DataBaseControllersManager.QuestionController.GetByThemeId(SessionData.CurrentTheme);
                _answers
DataBaseControllersManager.AnswerController.GetAnswersByQuestion(_question.id);

                if (RadioButtonAnswersList.Items.Count == 0)
                {
                    for (int i = 0; i < _answers.Count; i++)
                    {
                        RadioButtonAnswersList.Items.Insert((int)i, new ListItem($"
{_answers[i].answerText}", _answers[i].id.ToString(), true));
                    }
                }
            }
        }
    }
}

```

```

        protected string GetCurrentQuestion()
        {
            return _question.question;
        }

        protected void OnAnswerButtonClick(object sender, EventArgs e)
        {
            if (Convert.ToInt32(RadioButtonAnswersList.SelectedValue) ==
                _question.rightAnswerId)
            {
                right = true;

                foreach(ListItem radioButton in RadioButtonAnswersList.Items)
                {
                    radioButton.Enabled = false;
                }

                DataBaseControllersManager.ProgressController.ChangeStatus(SessionData.CurrentTheme, 1);
            }
            else
            {
                right = false;

                RadioButtonAnswersList.Items.Clear();

                for (int i = 0; i < _answers.Count; i++)
                {
                    RadioButtonAnswersList.Items.Insert((int)i, new ListItem($"
{_answers[i].answerText}", _answers[i].id.ToString(), true));
                }
            }
        }

        protected void Return_Click(object sender, EventArgs e)
        {
            Response.Redirect("student.aspx");
        }
    }
}

```

Листинг Д.4 – Файл Student.aspx

```

<%@ Page Title="Student page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Student.aspx.cs" Inherits="Grammar.Student" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <div class="jumbotron">
        <div class="container">
            <asp:GridView runat="server" ID="progressGrid"
                CssClass="gvData"
                ItemType="Grammar.Model.Progress" DataKeyNames="ID"
                SelectMethod="progressGrid_GetData"
                onrowcommand="CustomersGridView_RowCommand"
                AllowSorting="True"
                AutoGenerateColumns="false" Width="100%" >
                <Columns>
                    <asp:DynamicField DataField="id" />
                    <asp:DynamicField DataField="themeTitle" />
                    <asp:DynamicField DataField="status"/>
                    <asp:ButtonField ButtonType="Button" Text="Read"
commandname="Read"/>
                </Columns>
            </asp:GridView>
        </div>
    </div>

```

```

</asp:Content>

using Grammar.Managers;
using Grammar.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Grammar
{
    public partial class Student : System.Web.UI.Page
    {
        private List<Progress> _progress;

        protected void Page_Load(object sender, EventArgs e)
        {
            if (SessionData.user == null)
            {
                Response.Redirect("authorization.aspx");
            }
            else
            {
                _progress
DataBaseControllersManager.ProgressController.GetByStudent(SessionData.user.id);
            }
        }

        public IQueryable<Progress> progressGrid_GetData()
        {
            return _progress.AsQueryable();
        }

        public void CustomersGridView_RowCommand(Object sender,
GridViewCommandEventArgs e)
        {
            if (e.CommandName == "Read")
            {
                SessionData.CurrentTheme = Convert.ToInt32(e.CommandArgument) + 1;
                SessionData.IsCurrentThemeComplete
                _progress[(int)SessionData.CurrentTheme - 1].status == "complete";
                Response.Redirect("themeMaterial.aspx");
            }
        }
    }
}

```

Листинг Д.5 – Файл Study.aspx

```

<%@ Page Title="Study" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Study.aspx.cs" Inherits="Grammar.About" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

</asp:Content>
using Grammar.Managers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

```

```

namespace Grammar
{
    public partial class About : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            if (SessionData.user == null)
            {
                Response.Redirect("authorization.aspx");
            }

            else
            {
                if (SessionData.user.role_name == "teacher")
                {
                    Response.Redirect("teacher.aspx");
                }
                else
                {
                    Response.Redirect("student.aspx");
                }
            }
        }
    }
}

```

Листинг Д.6 – Файл Teacher.aspx

```

<%@ Page Title="Teacher page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Teacher.aspx.cs" Inherits="Grammar.Teacher" %>

```

```

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <div class="jumbotron">
        <div class="container">
            <asp:GridView runat="server" ID="progressGrid"
                CssClass="gvData"
                ItemType="Grammar.Model.Progress" DataKeyNames="ID"
                SelectMethod="progressGrid_GetData"
                AllowSorting="True"
                AutoGenerateColumns="false" Width="100%" >
                <Columns>
                    <asp:DynamicField DataField="id" />
                    <asp:DynamicField DataField="first_name" />
                    <asp:DynamicField DataField="last_name" />
                    <asp:DynamicField DataField="middle_name" />
                    <asp:DynamicField DataField="themeTitle" />
                    <asp:DynamicField DataField="status" />
                </Columns>
            </asp:GridView>
        </div>
    </div>
</asp:Content>

```

```

using Grammar.Managers;
using Grammar.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

```

```

namespace Grammar
{

```

```

public partial class Teacher : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (SessionData.user == null)
        {
            Response.Redirect("authorization.aspx");
        }
    }

    public IQueryable<Progress> progressGrid_GetData()
    {
        return
DataBaseControllersManager.ProgressController.GetAll().AsQueryable();
    }
}

```

Листинг Д.7 – Файл ThemeMaterial.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" MasterPageFile="~/Site.Master"
CodeBehind="ThemeMaterial.aspx.cs" Inherits="Grammar.ThemeMaterial" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

    <div class="jumbotron">
        <h1> <% = GetCurrentThemeTitle() %></h1>
        <p><% = GetCurrentThemeText() %></p>

        <% if(IsNotCompleted()) { %>
            <asp:Button runat="server" id="PassTest" Text="Pass test"
onclick="OnPassTestButtonClick"/>

            <% } %>
        <% else { %>
            <p>Complete!</p>
            <% } %>
    </div>

</asp:Content>
using Grammar.Managers;
using Grammar.Model;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

namespace Grammar
{
    public partial class ThemeMaterial : System.Web.UI.Page
    {
        private Theme _theme;
        protected void Page_Load(object sender, EventArgs e)
        {
            if (SessionData.user == null)
            {
                Response.Redirect("authorization.aspx");
            }
            else
            {
                _theme
DataBaseControllersManager.ThemeController.GetById(SessionData.CurrentTheme);
            }
        }
    }
}

```

```

        protected string GetCurrentThemeTitle()
        {
            return _theme.title;
        }

        protected string GetCurrentThemeText()
        {
            return _theme.text;
        }

        protected bool IsNotCompleted()
        {
            return !SessionData.IsCurrentThemeCompete;
        }

        protected void OnPassTestButtonClick(object sender, EventArgs e)
        {
            if (SessionData.login != "" && SessionData.password != "")
            {
                SessionData.user
                DataBaseControllersManager.UserController.GetUserData(SessionData.login,
                SessionData.password);
                Response.Redirect("questionMaterial.aspx");
            }
        }
    }
}

```

Answer.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Вариант ответа
    /// </summary>
    public class Answer
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Текст варианта ответа
        /// </summary>
        public string answerText;
        /// <summary>
        /// id вопроса, к которому вариант ответа принадлежит
        /// </summary>
        public uint questionID;
    }
}

```

Листинг Д.8 – Файл Progress.cs

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

```

```

namespace Grammar.Model
{
    /// <summary>
    /// Прогресс ученика
    /// </summary>
    public class Progress
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        [Key, Display(Name = "ID")]
        [ScaffoldColumn(false)]
        public uint id { get; set; }
        /// <summary>
        /// Имя
        /// </summary>
        [Required, StringLength(40), Display(Name = "First Name")]
        public string first_name { get; set; }
        /// <summary>
        /// Фамилия
        /// </summary>
        [Required, StringLength(40), Display(Name = "Last Name")]
        public string last_name { get; set; }
        /// <summary>
        /// Отчество
        /// </summary>
        [Required, StringLength(40), Display(Name = "Middle Name")]
        public string middle_name { get; set; }
        /// <summary>
        /// Номер темы
        /// </summary>
        [Required, StringLength(40), Display(Name = "Theme")]
        public string themeTitle { get; set; }
        /// <summary>
        /// Статус
        /// </summary>
        [Required, StringLength(40), Display(Name = "Status")]
        public string status { get; set; }
    }
}

```

Листинг Д.9 – Файл Question.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Вопрос
    /// </summary>
    public class Question
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Текст вопроса
        /// </summary>
        public string question;
        /// <summary>
        /// Номер правильного ответа
        /// </summary>
    }
}

```

```

        public uint rightAnswerId;
        /// <summary>
        /// Номер темы, к которой привязан вопрос
        /// </summary>
        public uint themeId;
    }
}

```

Листинг Д.10 – Файл Role.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Роль
    /// </summary>
    public class Role
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Название
        /// </summary>
        public string name;
    }
}

```

Листинг Д.11 – Файл Theme.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Тема (теория)
    /// </summary>
    public class Theme
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Заголовок (название темы)
        /// </summary>
        public string title;
        /// <summary>
        /// Текст (теория)
        /// </summary>
        public string text;
    }
}

```

Листинг Д.12 - ThemeStatus.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Статус темы
    /// </summary>
    public class ThemeStatus
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Название
        /// </summary>
        public string name;
    }
}

```

Листинг Д.13 – Файл User.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Grammar.Model
{
    /// <summary>
    /// Пользователь
    /// </summary>
    public class User
    {
        /// <summary>
        /// Порядковый номер
        /// </summary>
        public uint id;
        /// <summary>
        /// Имя
        /// </summary>
        public string first_name;
        /// <summary>
        /// Фамилия
        /// </summary>
        public string last_name;
        /// <summary>
        /// Отчество
        /// </summary>
        public string middle_name;
        /// <summary>
        /// Номер роли
        /// </summary>
        public string role_name;
        /// <summary>
        /// Логин
        /// </summary>
        public string login;
    }
}

```

```

        /// <summary>
        /// Пароль
        /// </summary>
        public string password;
        /// <summary>
        /// Дата рождения
        /// </summary>
        public string birthday;

        public User()
        {
            first_name = "2";
        }
    }
}

```

Листинг Д.14 – Файл IDbController.cs

```

using Grammar.Model;

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Grammar.Controllers
{
    #region Answer

    public class AnswerController : IAnswerController
    {
        public bool Add(uint id, string answerText, uint questionID)
        {
            bool success = true;

            using (MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConnectionData))
            {
                try
                {
                    conn.Open();

                    MySqlCommand comm = new MySqlCommand(

```

```

        string.Format("insert into answer values ({0}, {1},
{2})", id, answerText, questionID), conn);

        var reader = comm.ExecuteReader();

    }

    catch (Exception e)

    {

        success = false;

    }

}

return success;

}

public bool Change(uint id, string answerText, uint questionID)

{

    bool success = true;

    using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConnectionData))

    {

        try

        {

            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("update
answer set answer_text = {1}, question_id = {2} where id = {0}", id, answerText,
questionID), conn);

            var reader = comm.ExecuteReader();

        }

        catch (Exception e)

        {

            success = false;

        }

    }

    return success;

}

public bool Delete(uint id)

{

```

```

        bool success = true;

        try
        {
            using ( MySqlConnection conn = new
            MySqlConnection(MySQLDbContext.ConectionData))
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(string.Format("delete
                from answer where id = {0}", id), conn);

                var reader = comm.ExecuteReader();

            }
        }
        catch (Exception e)
        {
            success = false;
        }

        return success;
    }

    public List<Answer> GetAll()
    {
        List<Answer> answers = new List<Answer>();

        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand("select * from answer",
            conn);

            var reader = comm.ExecuteReader();

            while (reader.Read())
            {
                Answer answer = new Answer
                {

                    id = reader.GetUInt32("id"),
                    answerText = reader.GetString("answer_text"),

```

```

        questionID = reader.GetUInt32("question_id")
    };

    answers.Add(answer);
}
}
return answers;
}

public List<Answer> GetAnswersByQuestion(uint id)
{
    try
    {
        List<Answer> answers = new List<Answer>();
        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand($"select * from answer
where question_id={id}", conn);

            var reader = comm.ExecuteReader();

            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    Answer answer = new Answer

                    {

                        id = reader.GetUInt32("id"),
                        answerText = reader.GetString("answer_text"),
                        questionID = reader.GetUInt32("question_id")
                    };

                    answers.Add(answer);
                }
            }
        }
    }
    catch { }
}

```

```

        }
    }
    else
    {
        answers = null;
    }
}
return answers;
}
catch
{
    return null;
}
}
}
#endregion

#region Progress
public class ProgressController : IProgressController
{
    public bool Add(uint id, uint userId, uint themeId, uint statusId)
    {
        bool success = true;
        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
        {
            try
            {
                conn.Open();
                MySqlCommand comm = new MySqlCommand(
                    string.Format("insert into progress values ({0}, {1},
{2}, {3})", id, userId, themeId, statusId),
                    conn);
                var reader = comm.ExecuteReader();
            }

```

```

        catch (Exception e)
        {
            success = false;
        }
    }
    return success;
}

public bool Change(uint id, uint userId, uint themeId, uint statusId)
{
    bool success = true;

    using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
    {
        try
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("update
progress set user_id = {1}, theme_id = {2}, status_id = {3} where id = {0}", id,
userId, themeId, statusId), conn);

            var reader = comm.ExecuteReader();
        }
        catch (Exception e)
        {
            success = false;
        }
    }
    return success;
}

public bool ChangeStatus(uint id, uint statusId)
{
    bool success = true;

    using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
    {

```

```

        try
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("update
progress set status_id = {1} where theme_id = {0}", id, statusId), conn);

            var reader = comm.ExecuteReader();

        }
        catch (Exception e)
        {
            success = false;
        }
    }
    return success;
}

public bool Delete(uint id)
{
    bool success = true;

    try
    {
        using (MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("delete
from schedule where id = {0}", id), conn);

            var reader = comm.ExecuteReader();

        }
    }
    catch (Exception e)
    {
        success = false;
    }

    return success;
}

```

```

public List<Progress> GetAll()
{
    List<Progress> progresses = new List<Progress>();

    using (MySqlConnection conn = new
    MySqlConnection(MySQLDbContext.ConnectionData))
    {
        conn.Open();

        MySqlCommand comm = new MySqlCommand("select * from
        student_progress", conn);

        var reader = comm.ExecuteReader();
        while (reader.Read())
        {
            Progress progress = new Progress
            {
                id = reader.GetUInt32("id"),
                first_name = reader.GetString("first_name"),
                last_name = reader.GetString("last_name"),
                middle_name = reader.GetString("middle_name"),
                themeTitle = reader.GetString("title"),
                status = reader.GetString("status"),

            };

            progresses.Add(progress);
        }
    }
    return progresses;
}

public List<Progress> GetByStudent(uint id)
{
    List<Progress> progresses = new List<Progress>();

    using (MySqlConnection conn = new
    MySqlConnection(MySQLDbContext.ConnectionData))
    {

```

```

        conn.Open();

        MySqlCommand comm = new MySqlCommand($"call
progress_for_student({id})", conn);

        var reader = comm.ExecuteReader();
        while (reader.Read())
        {
            Progress progress = new Progress
            {
                id = reader.GetUInt32("id"),
                themeTitle = reader.GetString("title"),
                status = reader.GetString("status"),
            };

            progresses.Add(progress);
        }
    }
    return progresses;
}

#endregion

#region Question
public class QuestionController : IQuestionController
{
    public bool Add(uint id, string question, uint rightAnswerId, uint
themeId)
    {
        bool success = true;

        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConnectionData))
        {
            try
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(

```

```

        string.Format("insert into question values ({0}, {1},
{2}, {3})", id, question, rightAnswerId, themeId),

        conn);

        var reader = comm.ExecuteReader();

    }

    catch (Exception e)

    {

        success = false;

    }

    }

    return success;

}

public bool Change(uint id, string question, uint rightAnswerId, uint
themeId)

{

    bool success = true;

    using ( MySqlConnection conn = new
 MySqlConnection(MySQLDbContext.ConnectionData))

    {

        try

        {

            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("update
question set question = {1}, rightAnswerId = {2}, themeId = {3} where id = {0}", id,
question, rightAnswerId, themeId), conn);

            var reader = comm.ExecuteReader();

        }

        catch (Exception e)

        {

            success = false;

        }

    }

    return success;

}

public bool Delete(uint id)

```

```

    {
        bool success = true;
        try
        {
            using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
            {
                conn.Open();
                MySqlCommand comm = new MySqlCommand(string.Format("delete
from question where id = {0}", id), conn);
                var reader = comm.ExecuteReader();
            }
        }
        catch (Exception e)
        {
            success = false;
        }
        return success;
    }

    public List<Question> GetAll()
    {
        List<Question> questions = new List<Question>();
        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
        {
            conn.Open();
            MySqlCommand comm = new MySqlCommand("select * from campus",
conn);
            var reader = comm.ExecuteReader();
            while (reader.Read())
            {
                Question question = new Question
                {
                    id = reader.GetUInt32("id"),
                    question = reader.GetString("adress"),

```

```

        rightAnswerId = reader.GetUInt32("right_answer_id"),
        themeId = reader.GetUInt32("theme_id"),
    };

    questions.Add(question);
}
}
return questions;
}

public Question GetByThemeId(uint id)
{
    try
    {
        Question question = new Question();
        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            conn.Open();
            MySqlCommand comm = new MySqlCommand($"select * from
            question where theme_id={id}", conn);
            var reader = comm.ExecuteReader();

            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    question.id = reader.GetUInt32("id");
                    question.question = reader.GetString("question");
                    question.rightAnswerId =
                    reader.GetUInt32("right_answer_id");
                }
            }
            else
            {
                question = null;
            }
        }
    }
}

```

```

        }
    }
    return question;
}
catch
{
    return null;
}
}
}
#endregion

#region Role
public class RoleController : IRoleController
{
    public List<Role> GetAll()
    {
        List<Role> roles = new List<Role>();
        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
        {
            conn.Open();
            MySqlCommand comm = new MySqlCommand("select * from role",
conn);
            var reader = comm.ExecuteReader();
            while (reader.Read())
            {
                Role role = new Role
                {
                    id = reader.GetUInt32("id"),
                    name = reader.GetString("name")
                };

                roles.Add(role);
            }
        }
    }
}

```

```

        }

        return roles;
    }
}

#endregion

#region Theme

public class ThemeController : IThemeController
{
    public bool Add(uint id, string title, string text)
    {
        bool success = true;

        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            try
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(
                    string.Format("insert into theme values ({0}, {1},
{2})", id, title, text),
                    conn);

                var reader = comm.ExecuteReader();

            }
            catch (Exception e)
            {
                success = false;
            }
        }

        return success;
    }

    public bool Change(uint id, string title, string text)
    {
        bool success = true;

```

```

        using (MySqlConnection conn = new
 MySqlConnection(MySQLDbContext.ConnectionData))
        {
            try
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(string.Format("update
theme set title = {1}, text = {2} where id = {0}", id, title, text), conn);

                var reader = comm.ExecuteReader();

            }
            catch (Exception e)
            {
                success = false;
            }
        }
        return success;
    }

    public bool Delete(uint id)
    {
        bool success = true;

        try
        {
            using (MySqlConnection conn = new
 MySqlConnection(MySQLDbContext.ConnectionData))
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(string.Format("delete
from schedule where id = {0}", id), conn);

                var reader = comm.ExecuteReader();

            }
        }
        catch (Exception e)
        {
            success = false;
        }
    }

```

```

        return success;
    }

    public List<Theme> GetAll()
    {
        List<Theme> campuses = new List<Theme>();

        using (MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand("select * from theme",
            conn);

            var reader = comm.ExecuteReader();
            while (reader.Read())
            {
                Theme campus = new Theme
                {
                    id = reader.GetUInt32("id"),
                    title = reader.GetString("title"),
                    text = reader.GetString("text")
                };

                campuses.Add(campus);
            }
        }

        return campuses;
    }

    public Theme GetById(uint id)
    {
        try
        {
            Theme theme = new Theme();

```

```

        using ( MySqlConnection conn = new
 MySqlConnection(MySQLDbContext.ConectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand($"select * from theme
where id={id}", conn);

            var reader = comm.ExecuteReader();

            if (reader.HasRows)
            {
                while (reader.Read())
                {
                    theme.id = reader.GetUInt32("id");
                    theme.title = reader.GetString("title");
                    theme.text = reader.GetString("text");
                }
            }
            else
            {
                theme = null;
            }
        }
        return theme;
    }
    catch
    {
        return null;
    }
}

#endregion

#region ThemeStatus
public class ThemeStatusController : IThemeStatusController
{

```

```

        public List<ThemeStatus> GetAll()
        {
            List<ThemeStatus> themeStatuses = new List<ThemeStatus>();

            using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
            {
                conn.Open();
                MySqlCommand comm = new MySqlCommand("select * from campus",
conn);

                var reader = comm.ExecuteReader();
                while (reader.Read())
                {
                    ThemeStatus themeStatus = new ThemeStatus
                    {
                        id = reader.GetUInt32("id"),
                        name = reader.GetString("name")
                    };

                    themeStatuses.Add(themeStatus);
                }
            }
            return themeStatuses;
        }
    }

#endregion

#region User

public class UserController : IUserController
{
    public bool Add(uint id, string first_name, string last_name, string
middle_name, uint role_id, string login, string password, string birthday)
    {
        bool success = true;

        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConectionData))
        {

```

```

        try
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand(
                string.Format("insert into user values ({0}, {1}, {2},
{3}, {4}, {5}, {6}, {7})", id, first_name, last_name, middle_name, role_id, login,
password, birthday),

                conn);

            var reader = comm.ExecuteReader();

        }
        catch (Exception e)
        {
            success = false;
        }
    }
    return success;
}

```

```

    public bool Change(uint id, string first_name, string last_name, string
middle_name, uint role_id, string login, string password, string birthday)
    {
        bool success = true;

        using ( MySqlConnection conn = new
MySqlConnection(MySQLDbContext.ConnectionData))
        {
            try
            {
                conn.Open();

                MySqlCommand comm = new MySqlCommand(string.Format("update
schedule set first_name = {1}, last_name = {2}, middle_name = {3}, role_id = {4}, login
= {5}, password = {6}, birthday = {7} where id = {0}", id, first_name, last_name,
middle_name, role_id, login, password, birthday), conn);

                var reader = comm.ExecuteReader();

            }
            catch (Exception e)
            {
                success = false;
            }
        }
    }
}

```

```

    }
    return success;
}

public bool Delete(uint id)
{
    bool success = true;
    try
    {
        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand(string.Format("delete
from user where id = {0}", id), conn);

            var reader = comm.ExecuteReader();
        }
    }
    catch (Exception e)
    {
        success = false;
    }

    return success;
}

public List<User> GetAll()
{
    List<User> users = new List<User>();

    using ( MySqlConnection conn = new
    MySqlConnection(MySQLDbContext.ConnectionData))
    {
        conn.Open();

        MySqlCommand comm = new MySqlCommand("select * from
user_with_role", conn);

        var reader = comm.ExecuteReader();
    }
}

```

```

while (reader.Read())
{
    User user = new User
    {
        id = reader.GetUInt32("id"),
        first_name = reader.GetString("first_name"),
        last_name = reader.GetString("last_name"),
        middle_name = reader.GetString("middle_name"),
        role_name = reader.GetString("role_name"),
        login = reader.GetString("login"),
        password = reader.GetString("password"),
        birthday = reader.GetString("birthday")
    };

    users.Add(user);
}
}
return users;
}

public User GetUserData(string login, string password)
{ try
    {
        User user = new User();

        using ( MySqlConnection conn = new
        MySqlConnection(MySQLDbContext.ConnectionData))
        {
            conn.Open();

            MySqlCommand comm = new MySqlCommand($"select * from
            user_with_role where login='{login}' and password='{password}'", conn);

            var reader = comm.ExecuteReader();

            if (reader.HasRows)
            {

```

```
        while (reader.Read())
        {

            user.id = reader.GetUInt32("id");
            user.first_name = reader.GetString("first_name");
            user.last_name = reader.GetString("last_name");
            user.middle_name = reader.GetString("middle_name");
            user.role_name = reader.GetString("role_name");
            user.login = reader.GetString("login");
            user.password = reader.GetString("password");
            user.birthday = reader.GetString("birthday");

        }
    }
    else
    {
        user = null;
    }
}
return user;
}
catch (Exception e)
{
    return null;
}
}
}
#endregion
}
```