ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ

КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

ИНФОРМАЦИОННАЯ СИСТЕМА ПОДДЕРЖКИ ИЗУЧЕНИЯ РУССКОГО ЯЗЫКА КАК ИНОСТРАННОГО ДЛЯ СТУДЕНТОВ НИУ БЕЛГУ

Магистерская диссертация
обучающегося по направлению подготовки
09.04.02 Информационные системы и технологии
очной формы обучения,
группы 12001735
Або-Рашед Кнаан

Научный руководитель
к.т.н., доцент
Зимовец О.А.

Рецензент
директор ООО
«Управление ОПТИМА»
Ермолаев А.О.

БЕЛГОРОД 2019

РЕФЕРАТ

Информационная система поддержки изучения русского языка как иностранного для студентов НИУ БелГУ. – Або-Рашед Кнаан, магистерская диссертация Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 53, включая приложения 97, количество рисунков 40, количество, количество использованных источников 42.

КЛЮЧЕВЫЕ СЛОВА: информационная система, русский язык, обучение.

ОБЪЕКТ ИССЛЕДОВАНИЯ: информационная система поддержки изучения русского языка, как иностранного для студентов НИУ БелГУ.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: процесс изучения русского языка и его компьютерная поддержка.

ЦЕЛЬ РАБОТЫ: совершенствование процесса обучения иностранных студентов НИУ БелГУ посредством разработки информационной системы поддержки изучения русского языка как иностранного.

ЗАДАЧИ ИССЛЕДОВАНИЯ: ознакомиться с общими сведениями об информационных системах; провести обзор программных средств для разработки информационных систем; обосновать выбор программных средств; провести характеристику и анализ объекта исследования; разработать требования к информационной системе; разработать структуру информационной системы; разработать базу данных; разработать программные модули.

МЕТОДЫ ИССЛЕДОВАНИЯ: информационное моделирование и объектно-ориентированное проектирование.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы была спроектирована и разработана информационная система по изучения русского языка как иностранного с удобным пользовательским интерфейсом и широким спектром возможностей в области науки и образования.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Жизнедеятельность современного общества неразрывно связана с применением новых информационных технологий. Существенно возрастает роль информатики и коммуникаций. При всем разнообразии сфер использования современной информационно-вычислительной техники, главным является ее производственное применение.

Актуальность работы заключается в том, что в последнее время тема автоматизации учебного процесса учебных заведений является обсуждаемой и востребованной. В современном мире наблюдается скачок интереса к изучению русского языка, и это связано с экономическим ростом, который переживает Россия.

Русский язык пользуется большим спросом на рынке в сфере бизнеса и особенно в сфере науки и связи. Ежегодно в Россию приезжают более 200 000 иностранных студентов со всего мира.

В этой работе создается информационная система для поддержки изучения русского языка, как иностранного. Информационная система предназначена для ввода, хранения и обработки информации уроков русского языка для студентов, поступающих в систему, а также для сотрудников, работающих в институте.

Система создана в виде сайта на основе интегрированной среды разработки Microsoft Visual Studio. В ходе создания информационной системы, изучены материалы учебников, сайтов, журналов, статей по исследуемой теме и методики применения Asp.net и C# SQL Server и интегрированной среды разработки Microsoft Visual Studio.

Объект исследования: информационная система поддержки изучения русского языка, как иностранного для студентов НИУ БелГУ.

Предмет исследования: Процесс изучения русского языка и его компьютерная поддержка.

Целью работы является совершенствование процесса обучения

иностранных студентов НИУ БелГУ посредством разработки информационной системы поддержки изучения русского языка как иностранного.

Исходя из поставленной цели, определены следующие задачи:

- ознакомиться с общими сведениями об информационных системах;
- провести обзор программных средств для разработки информационных систем;
- обосновать выбор программных средств;
- провести характеристику и анализ объекта исследования;
- разработать требования к информационной системе;
- разработать структуру информационной системы;
- разработать базу данных;
- разработать программные модули.

Магистерская диссертация состоит из 3 глав, в которых описан процесс анализа, проектирования и разработки информационной системы.

В первой главе представлена общая информация о системах и моделях электронного обучения, исследования интереса к русскому языку в интернете, а также описаны проблемы иностранных студентов при изучении нового языка.

Во второй главе описаны функциональные модели существующего процесса и модели классов, а также модель UML.

Третья глава описаны требований системы и диаграмма сущность-связь, а также реализация системы.

Основная часть пояснительной записки написана на 53 стр., содержит 40 рисунков, 42 библиографических источника и приложение общим объемом 43 стр.

1 Анализ предметной области и существующих подходов в сфере поддержки изучения русского языка

1.1 Исследования интереса к изучению русского языка

Характерной чертой современной эпохи является глобализация, интеграция культур, языков, ментальности, формирование в конечном итоге поликультурной личности. Стало престижным получать образование, работать в других странах. В связи с этим особую актуальность приобретает процесс обучения иностранных студентов русскому языку как иностранному.

Для исследования интереса по изучению русского языка была собрана статистика запросов в поисковой системе Google и Яндекс.

На рисунке 1 показан анализ запросов по словосочетанию «Russian language» в поисковой системе Google за период между 2014 и 2019 гг. [2].

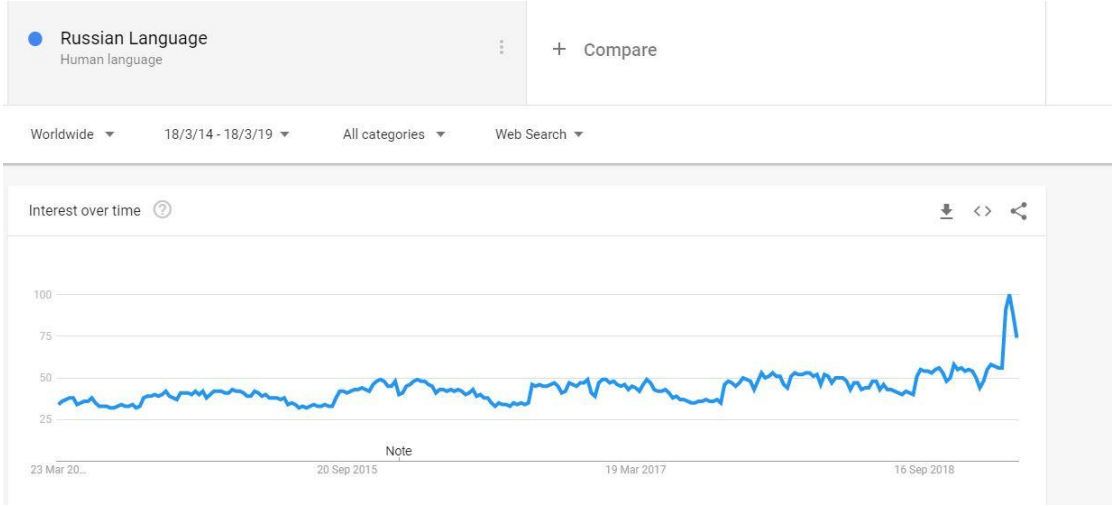


Рисунок 1 – Гугл Тренд Система

На рисунке 2 по запросам видно, что к данной теме проявляется значимый интерес. 11,129 запросов в месяц только по словосочетанию «Russian language» [3].

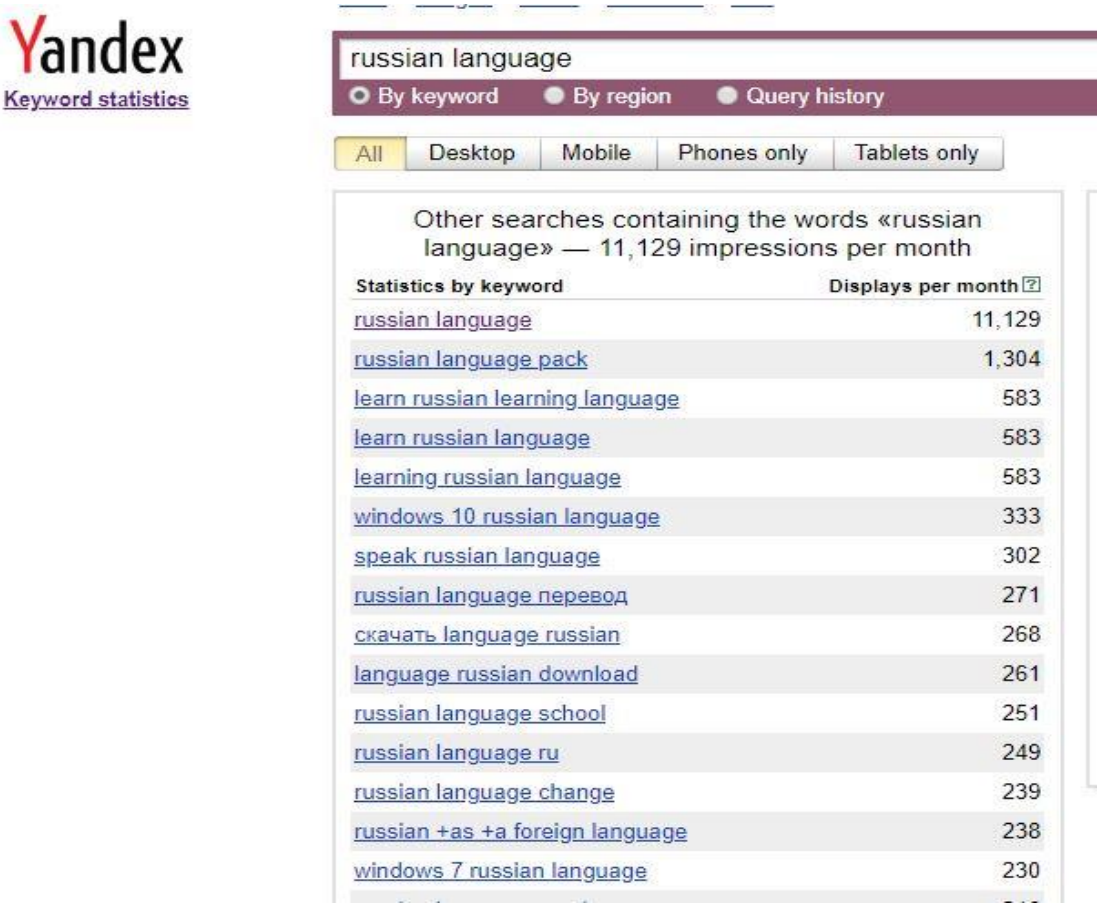


Рисунок 2 – Яндекс Система

1.2 Обзор и анализ основных проблем, возникающих при изучении русского языка

Изучение нового языка имеет важное значение для тех, кто ищет работу, представляет более широкие возможности, расширяет восприятие человека и дает ему возможность взглянуть на культуру и мысли говорящих на этом языке.

Тем не менее, большинство учащихся сталкиваются с некоторыми препятствиями и трудностями, которые затрудняют изучение иностранного язык. Это связано с несколькими причинами:

- транспорт до и от языка;
- культурные различия;
- эффекты обучения в классе.

Транспорт от и до языка. Изучающие язык, обычно набирают и форматируют грамматические формы своего собственного языка на втором языке, который они изучают, то есть они смотрят на второй язык с точки зрения и угла зрения своего родного языка.

Учащиеся делают неправильное произношение слов, или испытывают затруднения при произнесении некоторых символов, которые отсутствуют в аудиосистеме их родного языка, и обычно произносят одинаковые буквы между двумя языками способом, аналогичным тому, как символы произносятся на родном языке.

Окружающая среда играет важную роль, так как учащийся может выучить слова и переводы, которые могут быть неправильными или совершенно неправильными, в дополнение к отсутствию общения учащегося с носителями языка.

Культурные различия. Огромные культурные различия могут быть реальным препятствием в изучении второго языка, потому что это может означать разницу в понимании другого и значения слов и пословиц, популярных изречений и разговорного диалекта.

Учащийся должен достичь высокого уровня понимания и компетентности, изучая второй язык и используя его в соответствующих и различных контекстах.

Эффекты обучения в классе. Учащиеся, как правило, делают предположения о втором языке своего родного языка, чтобы заполнить пробелы в понимании и изучении второго языка. Педагог должен распознать эти предположения и пробелы и прояснить общие черты между двумя языками и отличительные особенности иностранного языка, которые помогают учащемуся быстрее дифференцироваться и учиться [4].

Отзывы учителя не должны быть потеряны в отношении информации и ответов учащегося, которые должны быть конструктивными и целенаправленными, и он должен понимать механизм ошибок, их причину и происхождение, а также соответствующие исправления.

И.Я. Лернер в книге «Дидактические основы методов обучения» рекомендует следующие методы:

– объяснительно-иллюстративный, состоящий в предъявлении готовой информации учителем, и ее усвоении в готовом виде учащимися;

– репродуктивный, предполагающий выполнение учащимися заданий по образцам, предложенным учителем или данным в учебнике;

– проблемного изложения, состоящий в том, что проблему ставит учитель, он же ее решает, показывая противоречивые пути самого процесса познания, иллюстрируя культуру мышления в ходе решения проблемы;

– частично-поисковый, или эвристический – учитель предъявляет проблемную задачу и делит ее на подзадачи (отдельные вопросы), а учащиеся сначала решают каждую подзадачу, а затем обращаются к изначальной задаче;

– исследовательский, требующий от учащихся осуществления поисковой деятельности без участия учителя [5].

Основным методом обучения русскому языку можно считать сознательно – практический, выдвинутый известным психологом Б.В. Беляевым в книге «Очерки по психологии обучения иностранным языкам» [6].

Другой метод, ранее широко используемый на уроках русского языка – это метод, в основу которого лег источник получения знаний:

– слово (рассказ) учителя;

– беседа;

– анализ языка (наблюдения над языком, грамматический разбор);

– упражнения;

– использование наглядных пособий;

– работа с учебником.

Студенты, приезжающие в Россию на учебу, сталкиваются в первую очередь с языковым барьером, с трудностями в изучении грамматической системы русского языка, которые объясняются большим расхождением в структуре языков [7].

Преподавание русский как иностранный очень трудоемкая и ответственная работа, связанная с разницей в грамматической структуре языков, что создает определенные трудности для иностранных студентов при изучении грамматики. Они сталкиваются с различными проблемами, невольно сравнивают изучаемый язык с родным, ищут аналогии. Имеет место языковая интерференция, когда происходит неправомерный перенос некоторых элементов родного языка в грамматическую систему изучаемого, когда «в сознании говорящего образуется некоторая третья система, в которой смешиваются признаки языков, то есть учащиеся устанавливают ложные соответствия между единицами родного и изучаемого языка».

1.3 Особенности и возможности систем электронного обучения

За последние несколько десятилетий интернет совершил настоящую революцию во многих профессиональных сферах. СМИ, медиа, коммуникации, службы доставки – список можно продолжать долго. Все эти области сильно трансформировались за последние 50 лет. Развитие технологий добавило инструменты, которые значительно ускорили темпы работ и улучшили их качество. Но есть одна отрасль, активно сопротивляющаяся технологическому прогрессу – это образование [8]. Долгое время в данной сфере не происходило никаких изменений. Однако, многие понимали, что и в ней будет совершён прорыв. Так и случилось – в 2001 году Массачусетский технический университет решил изменить текущую ситуацию и начал выкладывать свои курсы в сети в открытом доступе [9]. Эта инициатива вскоре была подхвачена ведущими вузами мира, людям приоткрыли доступ к элитному образованию. По использованию интернет-технологий и по предоставленному интерактиву, онлайн-образование тех дней сильно отличалось от нынешнего. Но, тем не менее, именно тогда были посеяны семена революции в сфере онлайн-образования.

Электронное обучение – это образование, которое близко к концепции

обучения через Интернет, но отличается тем, что в нем используются интернет-технологии, а также добавлены инструменты для контроля проектирования, реализации, управления и оценки процесса преподавания и обучения с использованием программного обеспечения для управления контентом и обучением. Электронное обучение некоторые определили, как любое использование веб-технологий и Интернета для создания обучения, а другие рассматривают электронное обучение как современный глобальный термин для образования и обучения, предоставляемый компьютером на основе сетей [10].

Электронное обучение определяется Федеральным законом «Об образовании в Российской Федерации» как: организация образовательного процесса с применением содержащейся в базах данных и используемой при реализации образовательных программ информации и обеспечивающих её обработку информационных технологий, технических средств, а также информационно-телекоммуникационных сетей, обеспечивающих передачу по линиям связи указанной информации, взаимодействие участников образовательного процесса. Далее приведены модели электронного обучения.

Модель 1. Обучение с веб-поддержкой:

– электронная среда используется в дополнение к основному традиционному учебному процессу для организации СРС (электронные материалы для самоподготовки, подготовка к лабораторным работам с использованием виртуальных лабораторных комплексов, самотестирование и др.);

– проведение консультаций с использованием форумов и вебинаров;

– организация текущего и промежуточного контроля;

– организация проектной работы студентов в электронной среде.

Модель 2. Смешанное обучение: эта модель объединяет традиционное и электронное обучение в учебной комнате или местах, оборудованных специальным оснащением, и сочетает в себе преимущества традиционного и электронного обучения.

При смешанном обучении в электронную среду частично переносятся отдельные виды учебной деятельности (лекции, практические занятия, лабораторные работы). Именно смешанная форма обучения является приоритетной для развития ЭО в ТПУ, так как позволяет оптимизировать распределение временных затрат преподавателя, освободить его от части аудиторной нагрузки.

Модель 3. Полного электронного обучения: эта модель использует электронное обучение в качестве альтернативы традиционного обучения, так чтобы обучение могло быть доступным в любом месте и в любое время, сеть выступает в качестве основного посредника для обеспечения всего процесса обучения [11].

Вывод по первому разделу

В первом разделе изучена литература по системам анкетирования данных. Проведен обзор и анализ основных проблем, возникающих при изучении русского языка, возможности систем электронного обучения. Можно сделать вывод о том, что внедрение новых систем образования на основе информационных технологий является жизненно необходимым в современном мире.

2 Моделирование процесса изучения русского языка

2.1 Разработка функциональной модели существующего процесса изучения русского языка

Для отображения процесса работы информационной системы поддержки изучения русского языка были разработаны модели в нотации IDEF 0.

На рисунке 3 представлена контекстная диаграмма функциональной модели.

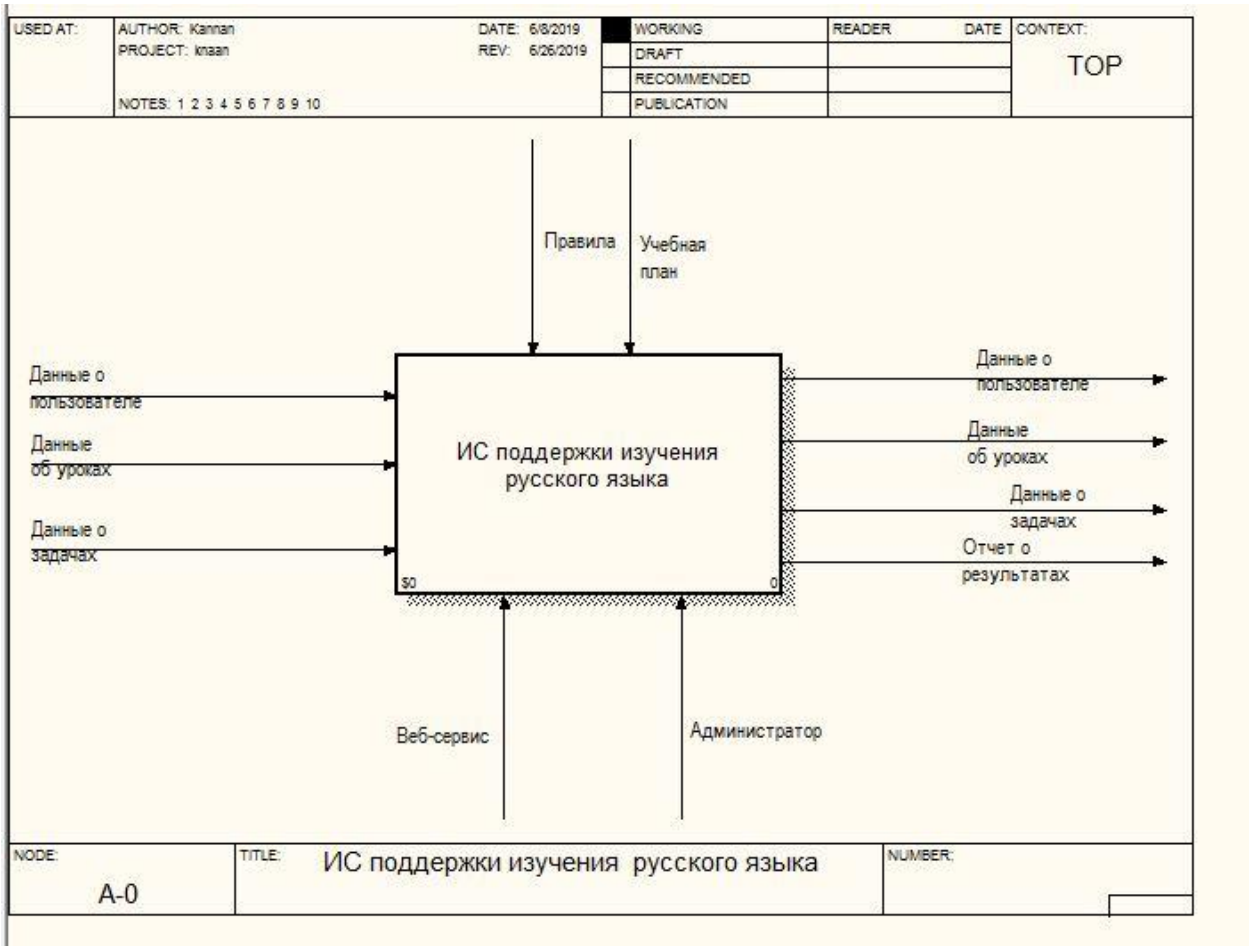


Рисунок 3 – Контекстная диаграмма функциональной модели

Модель описывает основную задачу системы – функцию изучения русского языка. Первоначальной входной информацией, являются:

– данные о пользователе;

– данные об уроках;

– данные о задачах.

Ресурсами, которые выполняют данную работу, являются: администратор и пользователь и веб-сервис.

Результатом работы являются:

– данные о пользователе;

– данные об уроках;

– данные о задачах;

– отчет о результатах.

На рисунке 4 показана декомпозиция контекстной диаграммы.

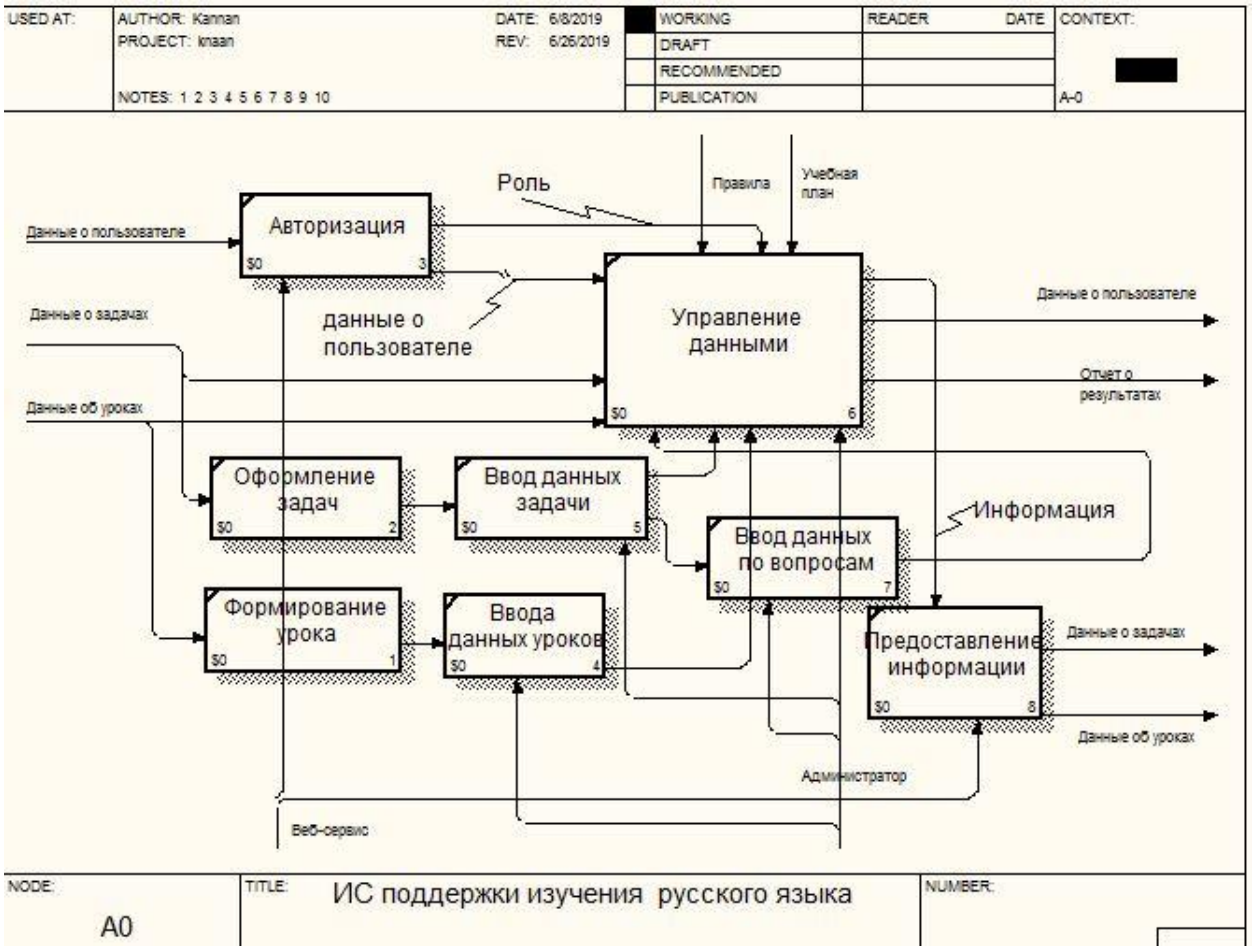


Рисунок 4 – Функциональная модель на втором уровне декомпозиции

На втором уровне декомпозиции процесс делится на 8 подпроцессов:

– авторизация;

– управление данными;
– оформление задач;
– ввод данных задачи;
– ввод данных по вопросам;
– формирование урока;
– ввода данных уроков;
– предоставление информации.

2.2 Разработка диаграммы классов информационной системы.

Диаграмма классов (Class diagram) служит для описания состава атрибутов классов, а также используется для отображения взаимосвязей между классами [12].

Диаграмма классов показывает коллекцию классов, интерфейсов, ассоциаций, сотрудничества и ограничений. Она также известен как структурная схема [13].

На рисунке 5 изображена диаграмма классов информационной системы поддержки изучения русского языка.

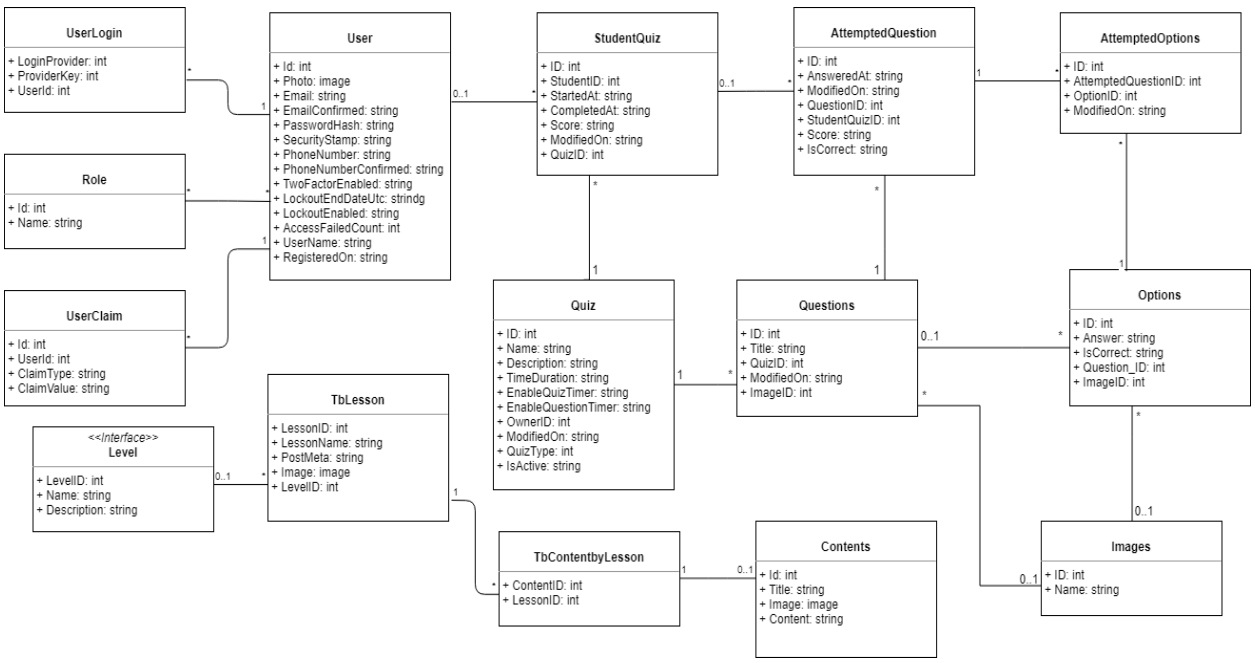


Рисунок 5 – Диаграмма классов

По результатам анализа предметной области было принято решение о создании следующего набора сущностей:

– таблица «Lesson» – это таблица «Урок», которая используется для хранения данных о печатных изданиях, поступающих в систему;

– таблица «Level» – это таблица «Уровень», которая используется для хранения данных об уровнях уроков, поступающих в систему;

– таблица «Content» – это таблица, которая используется для хранения данных контента урока;

– таблица «ContentbyLesoon» – это таблица, которая используется для связывания таблицы урок и таблицы контент;

– таблица «Users»– это таблица, которая используется для хранения данных о пользователях;

– таблица «Role» – это таблица, в которой приведены права администратора или пользователя и привязаны к таблице пользователя, и это одно отношение ко всем, где каждый пользователь имеет определенные полномочия;

– таблица «UserLogins» – это таблица, предназначенная для хранения пользовательских логинов;

– таблица «UserClaims» – это таблица, предназначенная для авторизации пользователей;

– таблица «Images» – это таблица, предназначенная для хранения изображений, связанных с параметрами;

– таблица «Quiz» – это таблица, предназначенная для хранения информации о задачах;

– таблица «StudentQuizzes» – это таблица, предназначенная для хранения информации о пройденных заданиях студентами;

– таблица «Questions» – это таблица, предназначенная для хранения вопросов, связанных с заданиями;

– таблица «Options» – это таблица, предназначенная для хранения опций,

связанных с вопросами;

– таблица «AttemptedOptions» – это таблица, предназначенная для хранения опций учащихся, связанных с каждым заданным вопросом;

– таблица «AttemptedQuestions» – это таблица, предназначенная для хранения заданных вопросов студентом для каждого задания.

2.3 Разработка UML-модели проектируемой информационной системы

UML – это графический язык моделирования общего назначения, предназначенный для спецификации, визуализации, проектирования и документирования всех данных, создаваемых при разработке программных систем [14].

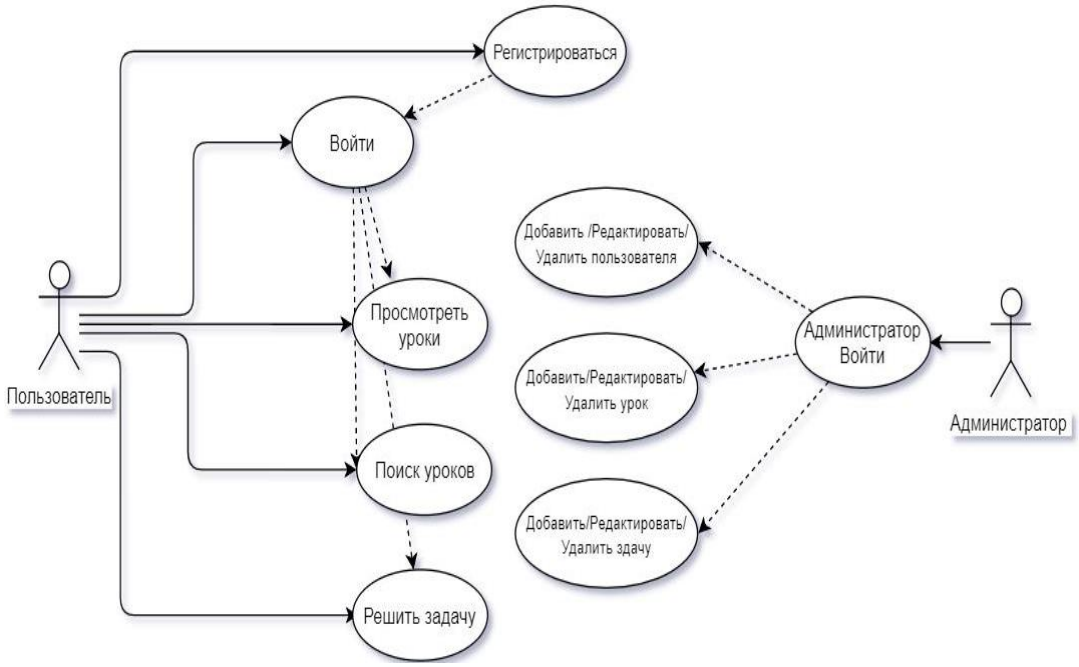


Рисунок 6 – UML Диаграмма использования

1. Описание случая использования: добавить нового пользователя.

Номер состояния: UN01.

Описание: здесь администратор создаёт нового пользователя, система проверяет правильность данных для возможности добавления пользователя,

если он имеет полномочия (только для администратора), то он может создать нового пользователя.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– ввести информацию о пользователе;

– система проверяет, что обязательные поля не оставлены пустыми (например, имя, адрес электронной почты);

– нажатие на кнопку задач;

– установка информации.

Альтернативные потоки:

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что у вас нет доступа к этой зоне;

– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

2. Описание случая использования: редактировать пользователя.

Номер состояния: UN02.

Описание: здесь администратор изменяет пользователя, и система проверяет достоверность возможности изменения, и, если он имеет полномочия администратора, то он может изменять информацию пользователя.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– изменение информации пользователя;

– система проверяет, что обязательные поля не оставлены пустыми

(например, имя, адрес электронной почты);

– нажатие на кнопку задач;

– установка информации.

Альтернативные потоки:

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что у вас нет доступа к этой зоне;

– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

3. Описание случая использования: удалить пользователя.

Номер состояния: UN03.

Описание: здесь администратор удаляет пользователя, и система проверяет правильность удаления, если он имеет полномочия только администратора, он может создать нового пользователя.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– удалить пользователя;

– давление на кнопку задач;

– удаление пользователя из базы данных.

Альтернативные потоки: при входе в систему у пользователя высвечивается на экране ограничение к доступу этой области, так как эти права, закрепленные за администратором.

Последующие строки: нет.

4. Описание случая использования: добавить урок.

Номер состояния: UN04.

Описание: здесь администратор создает урок, и система проверяет правильные возможности добавления, если он имеет полномочия

администратора, то он может создать новый урок.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– ввести информацию об уроке;
– система проверяет, что обязательные поля не оставлены пустыми;
– нажатие на кнопку задач;
– установка информации.

Альтернативные потоки

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что нет доступа к этой зоне;
– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

5. Описание случая использования: редактировать урок.

Номер состояния: UN05.

Описание: здесь администратор изменяет информацию об уроке, и система проверяет достоверность возможности изменения, и, если он имеет полномочия администратора, он может изменять информационный урок.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– введите информацию об уроке;
– система проверяет, что обязательные поля не оставлены пустыми;
– давление на кнопку задач;
– установка информации.

Альтернативные потоки:

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что нет доступа к этой зоне;

– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

6. Описание случая использования: удалить урок.

Номер состояния: UN06.

Описание: здесь администратор удаляет урок, и система проверяет возможность удаления, так как возможность удаления есть только у администратора.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– удалить урок;

– нажатие на кнопку задач;

– удаление урока из базы данных.

Альтернативные потоки:

– при входе в систему у пользователя высвечивается на экране ограничение к доступу этой области, так как эти права, закреплены за администратором.

Последующие строки: нет.

7. Описание случая использования: добавить задачу.

Номер состояния: UN07.

Описание: здесь администратор создает задачу, система проверяет правильность возможности добавления.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– ввести информацию о задаче;
– система проверяет, что обязательные поля не оставлены пустыми;
– нажатие на кнопку задач;
– установка информации.

Альтернативные потоки:

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что у вас нет доступа к этой зоне;
– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

8. Описание случая использования: редактировать задачу.

Номер состояния: UN08.

Описание: здесь администратор изменяет информацию о задаче, и система проверяет достоверность возможности изменения, и, если он имеет полномочия администратора, он может изменять информационную задачу.

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– ввести информацию об уроке;
– система проверяет, что обязательные поля не оставлены пустыми;
– нажатие на кнопку задач;
– установка информации.

Альтернативные потоки:

– при вводе пользователя, который не имеет полномочий админа на экране появляется сообщение о том, что у вас нет доступа к этой зоне;
– когда система обнаруживает, что введённая информация неполная, процесс не возникает, и сообщение отображается не в том месте.

Последующие строки: нет.

9. Описание случая использования: удалить задачу.

Номер состояния: UN09.

Описание: здесь администратор удаляет задачу, и система проверяет достоверность удаления, если она имеет полномочия администратора, он может удалить задачу

Основной пользователь: Администратор.

Вторичный пользователь: нет.

Предыдущие условия: убедиться, что портал является администратором.

Основной поток:

– удалить урок;
– нажатие на кнопку задач;
– удаление урока из базы данных.

Альтернативные потоки:

– при входе в систему у пользователя высвечивается на экране ограничение к доступу этой области, так как эти права закреплены за администратором.

Последующие строки: нет.

10. Описание случай использования: просмотреть уроки.

Номер состояния: UN010.

Описание: здесь админ или пользователь отображает все существующие уроки.

Основной пользователь: Администратор, пользователь.

Вторичный пользователь: нет.

Предыдущие условия: нет.

Основной поток:

– нажатие на кнопку задач;
– просмотр списка уроков.

Альтернативные потоки: нет.

Последующие строки: нет.

11. Описание случая использования: поиск уроков.

Номер состояния: UN011.

Описание: Администратор или пользователь ищет уроки.

Основной пользователь: Администратор, пользователь.

Вторичный пользователь: нет.

Предыдущие условия: нет.

Основной поток:

- ввести ключевые слова;
- запросить базу данных для слов;
- показать список результатов.

Альтернативные потоки: показывать сообщение, когда требуемой информации нет.

Последующие строки: нет.

12. Описание случая использования: решить задачу.

Номер состояния: UN12

Описание: Администратор или пользователь может решать задачи.

Основной пользователь: Администратор, пользователь.

Вторичный пользователь: нет.

Предыдущие условия: нет.

Основной поток:

- нажатие на кнопку задач;
- показать информацию.

Альтернативные потоки: показывать сообщение, когда требуемой информации нет.

Последующие строки: нет.

Вывод по второму разделу

Во второй главе был проведен анализ информационной системы для поддержки изучения русского языка как иностранного. произведено

моделирование основных бизнес-процессов. Разработаны диаграммы классов и UML-модели.

3 Разработка информационной системы поддержки изучения русского языка как иностранного

3.1 Разработка требований к информационной системе по видам обеспечения

В системе поддержки изучения необходимо своевременно обеспечить обучающихся информационной учебной литературой необходимую для осуществления учебного процесса. Важно формировать библиотеки в соответствии с профилем учебного заведения и информационными потребностями студентов. Системой также можно управлять, добавляя и удаляя уроки или добавляя соответствующее редактирование и добавляя пользователей.

Основание для разработки: программа разрабатывается на основе учебного плана в рамках курса «Системная инженерия».

Назначение разработки: автоматизированная информационная система поддержки изучения предназначена для решения следующих задач:

- ввода, хранения и обработки информации об изданиях;
- ввода, хранения и обработки информации о пользователе;
- контроль доступа в систему;
- ввода, хранения и обработки информации о сотрудниках.

Требования к функциональным характеристикам. Функциональные требования делятся на два уровня полномочий (админ, пользователь).

Админ: создать, редактировать, удалить уроки. Создать, редактировать, удалить пользователей, просмотр списка уроков.

Пользователь: просмотреть уроки.

Требования к надёжности. Необходимо, чтобы система обладала

устойчивостью к возникновению проблем, связанных с оборудованием и программными системами, а также электропитанием. Для надёжной работы электронного ресурса необходимы высоконадёжные аппаратные и программные системы [16].

Требования надёжности должны быть регламентированы для следующих аварийных ситуаций:

– отсутствие электроэнергии;
– выход из строя программных средств системы;
– неверные действия пользователей;
– пожар, взрыв;
– попадание вирусов в систему и так далее.

Система должна:

– проводить контроль вводимой информации о печатных изданиях и пользователях;
– блокировать некорректные действия пользователя при работе с системой;
– обеспечивать целостность данных;
– разграничивать права доступа;
– хранить данные в отдельной базе.

Требования к лингвистическому обеспечению:

– использование русского языка в интерфейсе информационной системы;
– эффективные интерфейсы должны быть простыми и доступными для пользователя. Необходимо, чтобы пользователь мог сразу понять свои задачи и мог достичь своих целей;
– интерфейс не должен беспокоить пользователя внутренним взаимодействием с системой. Необходимо бережное и непрерывное сохранение работы, с предоставлением пользователю возможности отменять любые действия в любое время.

Требования к составу и параметрам технических средств. Техническое

обеспечение должно удовлетворять следующим требованиям: технические средства должны обладать уровнем надёжности, отвечающие современным требованиям; необходимо предусмотреть возможность эффективного увеличения размера системы без осуществления значительных затрат [18].

Сервер должен удовлетворять следующим требованиям:

- процессор Intel Core i3 или i5;
- 1Gb и более оперативной памяти;
- 100 Gb - жёсткий диск;
- монитор – 17;
- сетевая карта PCIGeniusGF100TXRRL-8139 1Гб или выше клавиатура;
- манипулятор типа «мышь»;
- процессор 1.3 GHz или выше;
- 128 Мб и более оперативной памяти;
- монитор - 19. Сетевая карта PCI GeniusGF100TXRRL-8139 10/100Мб;
- клавиатура;
- манипулятор типа «мышь».

Требования к информационной и программной совместимости:

Система должна работать под управлением ОС Microsoft Windows. Система управления базы данных SQL Server. Другое ПО выбирается по решению разработчика. Основным критерием является низкая стоимость.

Требования к маркировке и упаковке. Готовое программное изделие и документация поставляются на компакт дисках в стандартной упаковке. Один комплект программной документации должен быть распечатан с помощью лазерного принтера на листах формата А4 и иметь типографский переплёт.

Требования к транспортированию и хранению:

Требования программного изделия совпадают с аналогичными требованиями, предъявляемыми к компакт дискам.

Требования к математическому обеспечению. Математические методы и алгоритмы, используемые для шифрования дешифрования данных, а также

программное обеспечение, реализующее их, должны быть сертифицированы уполномоченными организациями для использования в государственных органах Российской Федерации [17].

Требования к информационному обеспечению. В состав информационного обеспечения программы входит база данных, входная, внутренняя и выходная информация. Структура базы данных поддерживает кодирование хранимой и обрабатываемой информации в соответствии с общероссийскими классификаторами.

В качестве входной информации выступает:

- информация о принятии на учёт в фонд библиотеки литературы;
- информация об абонентах;
- требования на получение литературы из фонда;
- требования на сдачу литературы в фонд.

В качестве выходной информации служат:

- экранные формы с отображением на них результатов поиска литературы;
- экранные формы с отображением списков читателей библиотеки;
- экранные формы с отображением карточек читателей.

Требования к организационному обеспечению. Организационное обеспечение системы должно быть достаточным для эффективного выполнения персоналом возложенных обязанностей при осуществлении автоматизированных и связанных с ними неавтоматизированных функций системы [19]. К работе с системой допускаются сотрудники, имеющие навыки работы на персональном компьютере, ознакомленные с правилами эксплуатации и прошедшие обучение при работе с системой.

3.2 Выбор программных средств для реализации информационной системы

Выбор программных средств для разработки информационной системы определяются архитектурой информационной системы. Архитектура – это совокупность существенных решений об организации информационной системы. Обычно в понятие архитектуры входят решения об основных аппаратных и программных составляющих системы, их функциональном назначении и организации связей между ними [20].

Известны несколько типов архитектур: файл-сервер, клиент-сервер, многоуровневая архитектура, архитектура на основе Интернет/интранет-технологии. Компромиссным решением для создания удобных и простых в использовании и сопровождении информационных систем, эффективно работающих с базами данных, стало объединение интранет-технологии с архитектурой клиент-сервер [21].

При этом структура информационного приложения приобретает следующий вид: приложение – сервер приложений – сервер баз данных.

3.2.1 Система управления базами данных SQL Server

SQL Server – это система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft [22].

Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия, конкурирует с другими СУБД в этом сегменте рынка [23]. SQL Server поддерживает большую часть стандарта SQL и предлагает множество современных функций:

- сложные запросы;
- внешние ключи;
- триггеры;
- изменяемые представления;

– транзакционная целостность;
– много версионность.

Кроме того, пользователи могут всячески расширять возможности SQL Server, например, создавая свои

– типы данных;
– функции;
– операторов;
– агрегатные функции;
– методы индексирования;
– процедурные языки.

А благодаря свободной лицензии, SQL Server разрешает бесплатно использовать, изменять и распространять всем и для любых целей личных, коммерческих или учебных.

3.2.2 Язык программирования C# и Asp.net

C# – язык программирования, сочетающий объектно-ориентированные и контекстно-ориентированные концепции. Разработан в 1998-2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств вроде Visual Studio.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет строгую статическую типизацию, поддерживает полиморфизм, перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения, комментарии в формате XML. Переняв многое от своих предшественников – языков C++, Delphi, Modula и Smalltalk – C#, опираясь на практику их

использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++) или вывода типов (в отличие от Haskell) [24].

Asp.net – технология создания веб-приложений и веб-сервисов от компании Майкрософт. Она является составной частью платформы Microsoft .NET и более старой технологии Microsoft ASP.

Asp.net внешне во многом сохраняет схожесть с более старой технологией ASP, что позволяет разработчикам относительно легко перейти на Asp.net. В то же время внутреннее устройство ASP.NET существенно отличается от ASP, поскольку оно основано на платформе .NET и, следовательно, использует все новые возможности, предоставляемые этой платформой [25].

После выпуска сервера Internet Information Services 4.0 в 1997 году, компания Microsoft начала исследовать возможность новой модели веб-приложения, которая удовлетворила бы жалобы на ASP, особенно связанные с отделением оформления содержания, и которая позволит писать «чистый» код. Работа по разработке такой модели была поручена Марку Андерсу, менеджеру команды IIS, и Скотту Гатри, поступившему на работу в Microsoft в 1997. Андерс и Гатри разработали первоначальный проект в течение двух месяцев, и Гатри написал код первоначального прототипа во время рождественских каникул 1997 года.

Первоначальный проект назывался «XSP»; Гатри объяснил в интервью 2007 года, что «всегда спрашивают, что означает буква X. В то время она ничего не значила. XML начинается с неё; XSLT начинается с неё. Все клевое начинается с X, поэтому мы его так и назвали.» Прототип XSP был написан на Java, так как на тот момент у Microsoft не было Java-подобной технологии. В то время уже предполагалось, что лицензирование Java для Microsoft не будет продлено в 2003 году (в 2003 истекал срок выданной Sun Microsystems лицензии). В 1999 было решено построить новую платформу на основе

Common Language Runtime (CLR), так как в нём, как и в Java, наличествовало программирование по принципам ООП, сборка мусора и другие возможности. Гатри описал это решение как «огромный риск», так как успех новой разработки был связан с успехом CLR, которая, как и XSP, находилась на ранней стадии разработки [26].

3.2.3 Интегрированная среда разработки Microsoft Visual Studio

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемых кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight [27].

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса

разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Таким образом, все эти программы взаимосвязаны, можно сказать, что они дополняют друг друга. При их использовании создаются приложения, которые необходимы пользователям. Таким образом, выбор средств разработки сделан на основе выбранной архитектуры системы и удобства разработки.

3.3 Разработка информационной модели

Построение информационной модели предполагает выделение сущностей, их атрибутов и первичных ключей, идентификацию связей между сущностями. Общепринятым видом графического изображения реляционной модели данных является ER диаграмма, на которой сущности изображаются прямоугольниками, соединённые между собой связями. Такое графическое представление облегчит восприятие структуры базы данных по сравнению с текстовым описанием [28].

На рисунке 10 представлена логическая и физическая модель для автоматизированной системы.

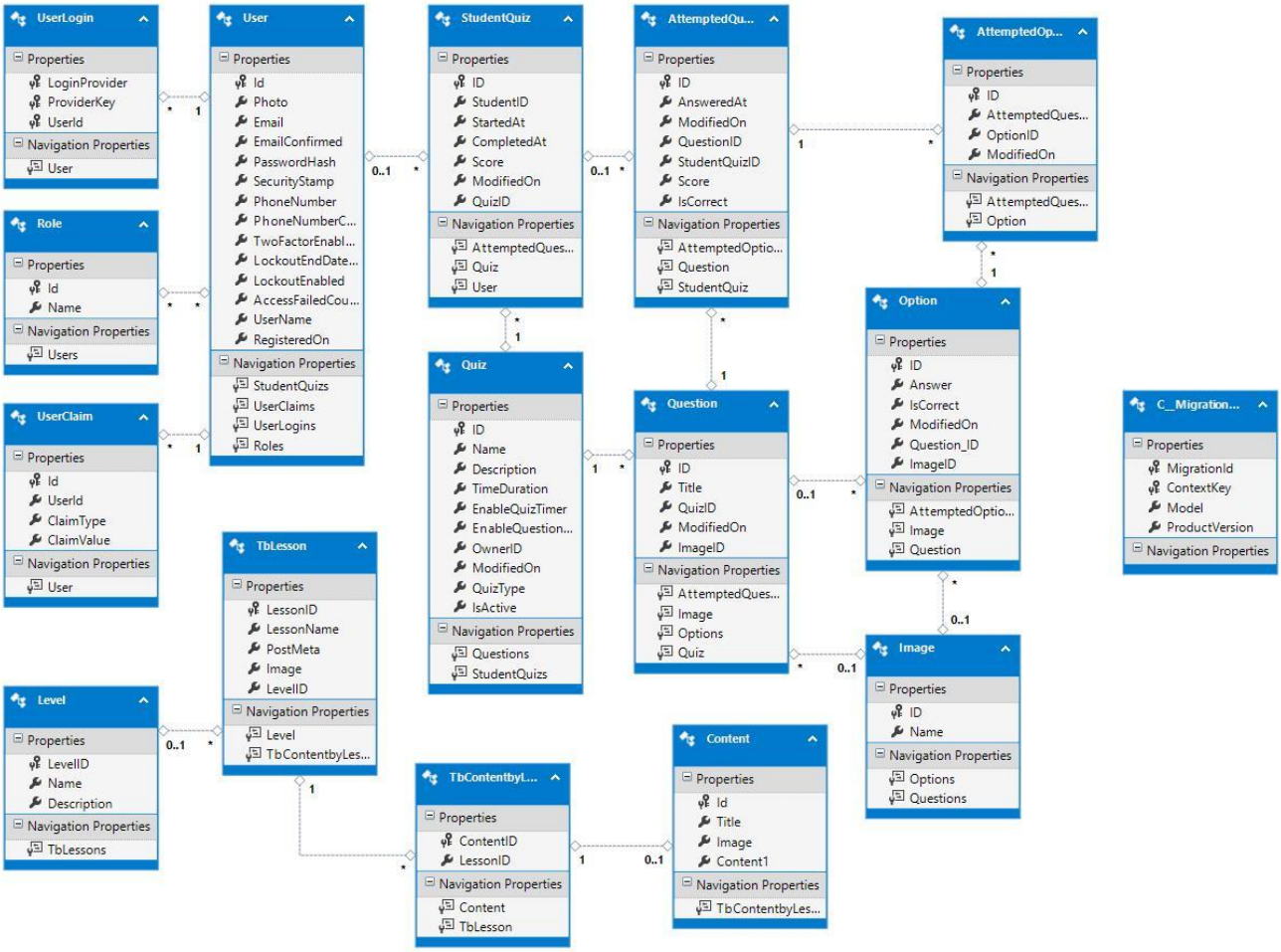


Рисунок 7 – Реляционная схема ERD

Разработанная модель базы данных содержит шесть таблиц, каждая из которых хранит определенную информацию. Каждая из таблиц непосредственно связана друг с другом.

Далее приведено описание каждой таблицы. На рисунке 8 представлена таблица «Lesson».

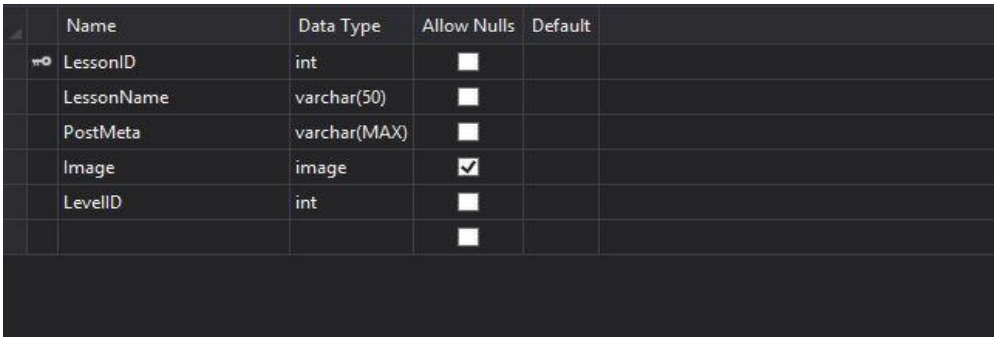


Рисунок 8 – Таблица «Lesson»

Таблица «Lesson» – это таблица «Урок», которая используется для хранения данных о печатных изданиях, поступающих в систему.

На рисунке 9 представлена таблица «Level».

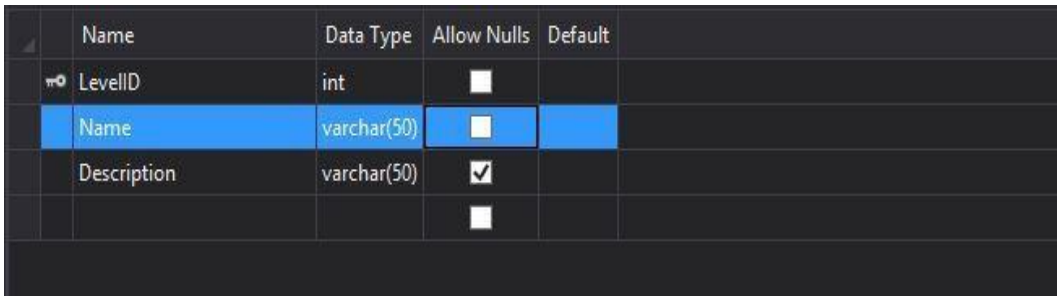


Рисунок 9 – Таблица «Level»

Таблица «Level» – это таблица «Уровень», которая используется для хранения данных об уровнях уроков, поступающих в систему.

На рисунке 10 представлена таблица «Content».

Рисунок 10 – Таблица «Content»

Таблица «Content» – это таблица, которая используется для хранения данных контента урока.

На рисунке 11представлена таблица «ContentbyLesoon».

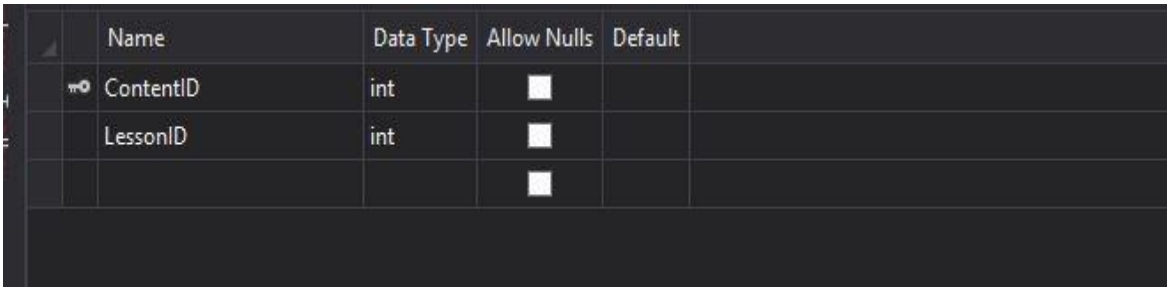


Рисунок 11 – Таблица «ContentbyLesoon»

Таблица «ContentbyLesoon» – это таблица, которая используется для связывания таблицы урок и таблицы контент.

На рисунке 12 представлена таблица «Users».

Name	Data Type	Allow Nulls	Default
Photo	nvarchar(MAX)	☑	
Email	nvarchar(256)	☑	
EmailConfirmed	bit	☐	
PasswordHash	nvarchar(MAX)	☑	
SecurityStamp	nvarchar(MAX)	☑	
PhoneNumber	nvarchar(MAX)	☑	
PhoneNumberConfirmed	bit	☐	
TwoFactorEnabled	bit	☐	
LockoutEndDateUtc	datetime	☑	
LockoutEnabled	bit	☐	
AccessFailedCount	int	☐	
UserName	nvarchar(256)	☐	
RegisteredOn	datetime	☑	
		☐	

Рисунок 12 – Таблица «Users»

Таблица «Users»– это таблица, которая используется для хранения данных о пользователях.

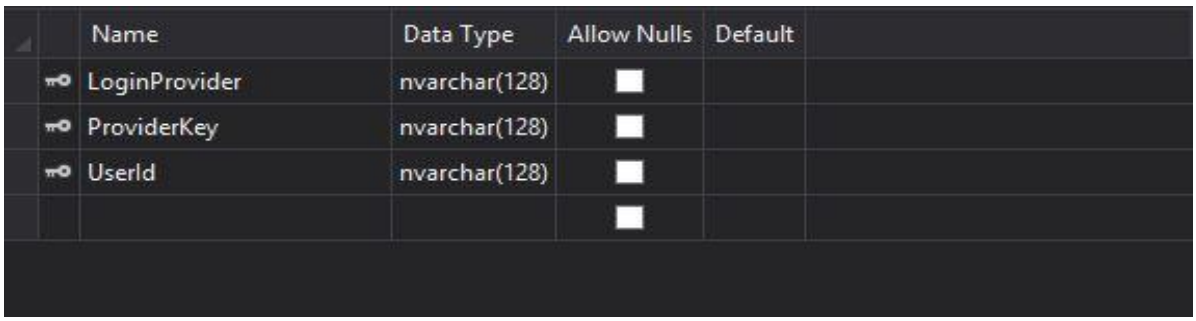
На рисунке 13 представлена таблица «Role».

Name	Data Type	Allow Nulls	Default
Id	nvarchar(128)	☐	
Name	nvarchar(256)	☐	
		☐	

Рисунок 13 – Таблица «Role»

Таблица «Role» – это таблица, в которой приведены права администратора или пользователя и привязаны к таблице пользователя, и это одно отношение ко всем, где каждый пользователь имеет определенные полномочия.

На рисунке 14 представлена таблица «UserLogins».

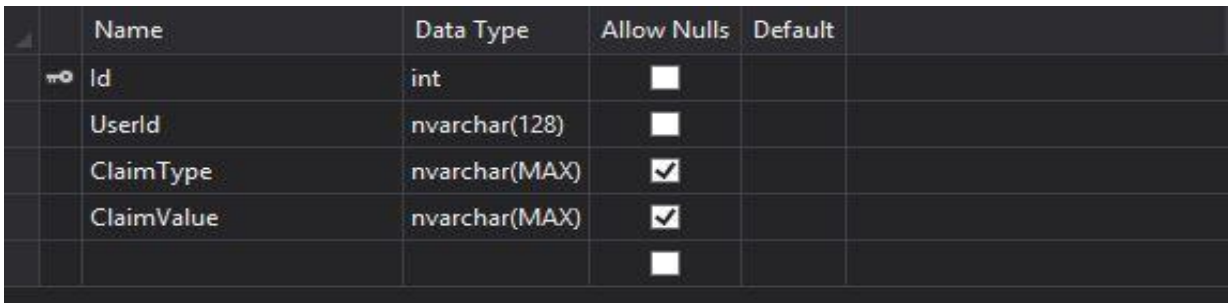


	Name	Data Type	Allow Nulls	Default
🔑	LoginProvider	nvarchar(128)	☐	
🔑	ProviderKey	nvarchar(128)	☐	
🔑	UserId	nvarchar(128)	☐	
			☐	

Рисунок 14 – Таблица «UserLogins»

Таблица «UserLogins» – это таблица, предназначенная для хранения пользовательских логинов.

На рисунке 15 представлена таблица «UserClaims».



	Name	Data Type	Allow Nulls	Default
🔑	Id	int	☐	
	UserId	nvarchar(128)	☐	
	ClaimType	nvarchar(MAX)	☑	
	ClaimValue	nvarchar(MAX)	☑	
			☐	

Рисунок 15 – Таблица «UserClaims»

Таблица «UserClaims» – это таблица, предназначенная для авторизации пользователей.

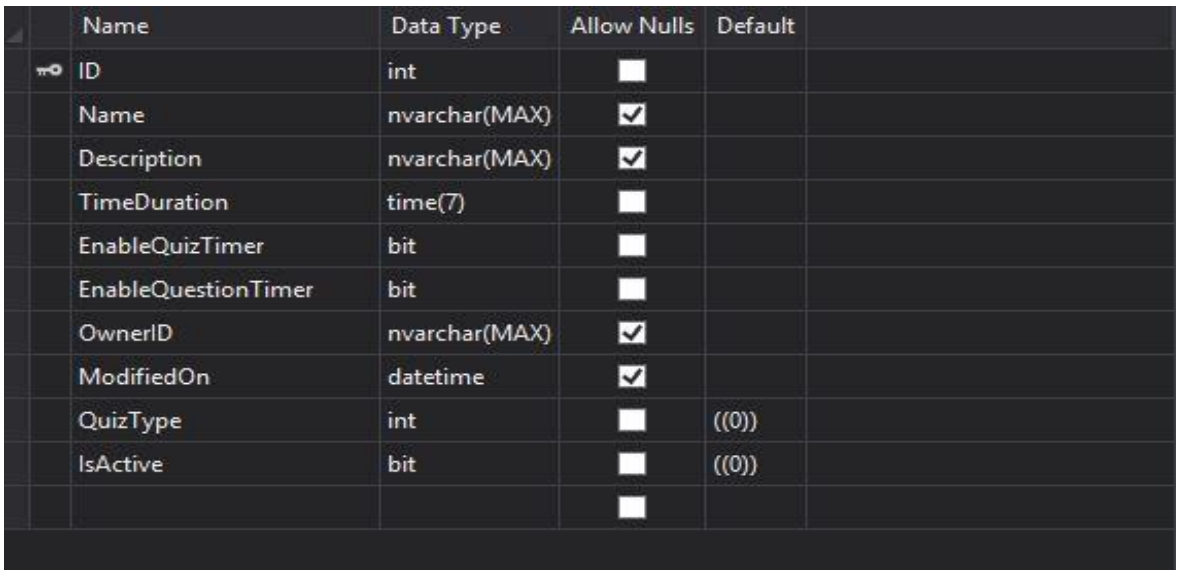
На рисунке 16 представлена таблица «Images».

	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	Name	nvarchar(MAX)	☑	
			☐	

Рисунок 16 – Таблица «Images»

Таблица «Images» – это таблица, предназначенная для хранения изображений, связанных с параметрами.

На рисунке 17 представлена таблица «Quiz».

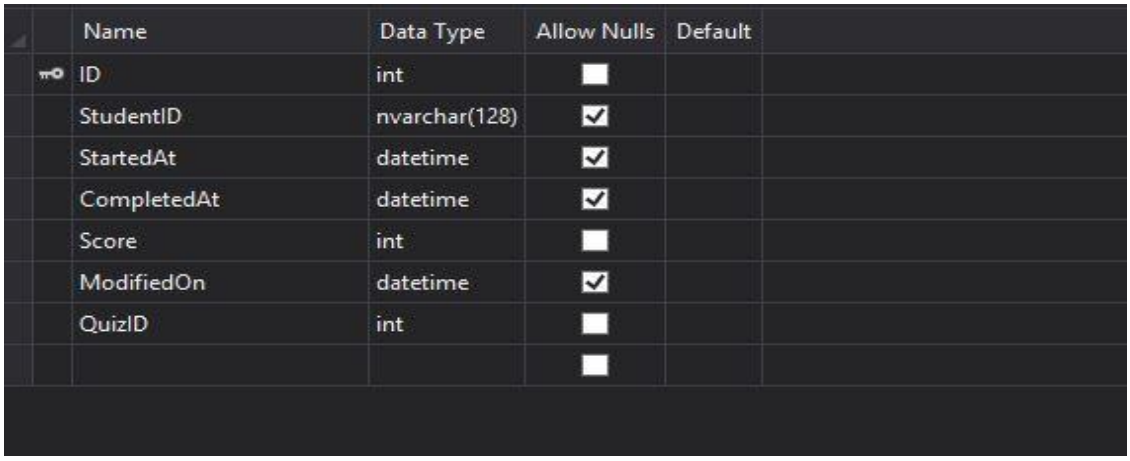


Name	Data Type	Allow Nulls	Default
ID	int	☐	
Name	nvarchar(MAX)	☑	
Description	nvarchar(MAX)	☑	
TimeDuration	time(7)	☐	
EnableQuizTimer	bit	☐	
EnableQuestionTimer	bit	☐	
OwnerID	nvarchar(MAX)	☑	
ModifiedOn	datetime	☑	
QuizType	int	☐	((0))
IsActive	bit	☐	((0))
		☐	

Рисунок 17 – Таблица «Quiz»

Таблица «Quiz» – это таблица, предназначенная для хранения информации о задачах.

На рисунке 18 представлена таблица «StudentQuizzes».

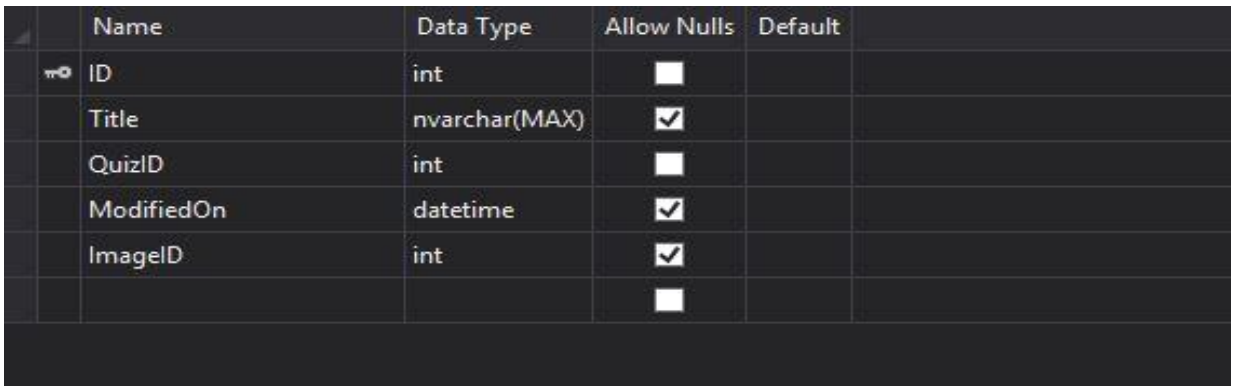


Name	Data Type	Allow Nulls	Default
ID	int	☐	
StudentID	nvarchar(128)	☑	
StartedAt	datetime	☑	
CompletedAt	datetime	☑	
Score	int	☐	
ModifiedOn	datetime	☑	
QuizID	int	☐	
		☐	

Рисунок 18 – Таблица «StudentQuizzes»

Таблица «StudentQuizzes» – это таблица, предназначенная для хранения информации о пройденных заданиях студентами.

На рисунке 19 представлена таблица «Questions».

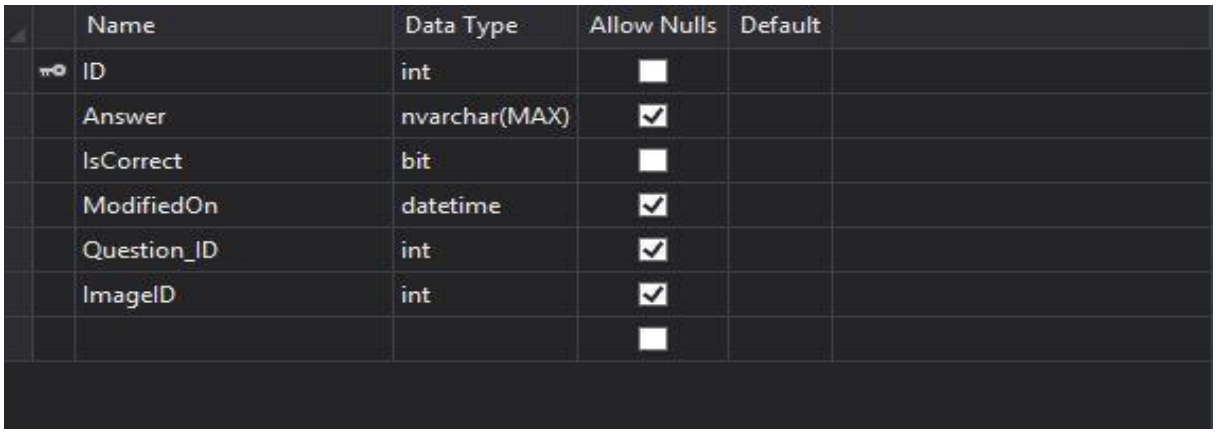


	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	Title	nvarchar(MAX)	☑	
	QuizID	int	☐	
	ModifiedOn	datetime	☑	
	ImageID	int	☑	
			☐	

Рисунок 19 – Таблица «Questions»

Таблица «Questions» – это таблица, предназначенная для хранения вопросов, связанных с заданиями.

На рисунке 20 представлена таблица «Options».

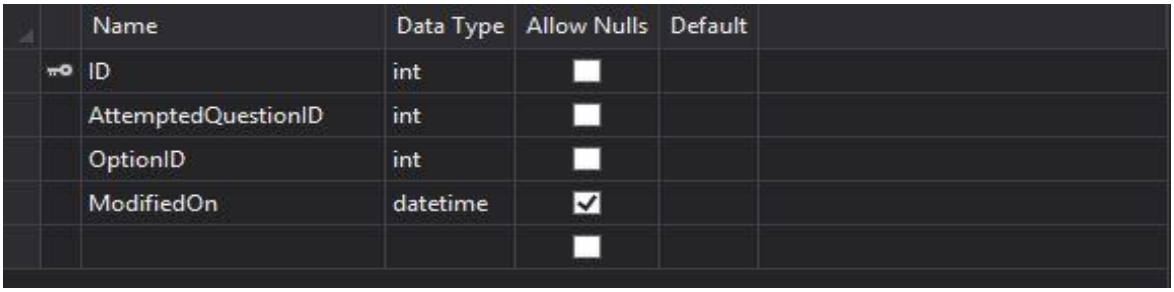


	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	Answer	nvarchar(MAX)	☑	
	IsCorrect	bit	☐	
	ModifiedOn	datetime	☑	
	Question_ID	int	☑	
	ImageID	int	☑	
			☐	

Рисунок 20 – Таблица «Options»

Таблица «Options» – это таблица, предназначенная для хранения опций, связанных с вопросами.

На рисунке 21 представлена таблица «AttemptedOptions».

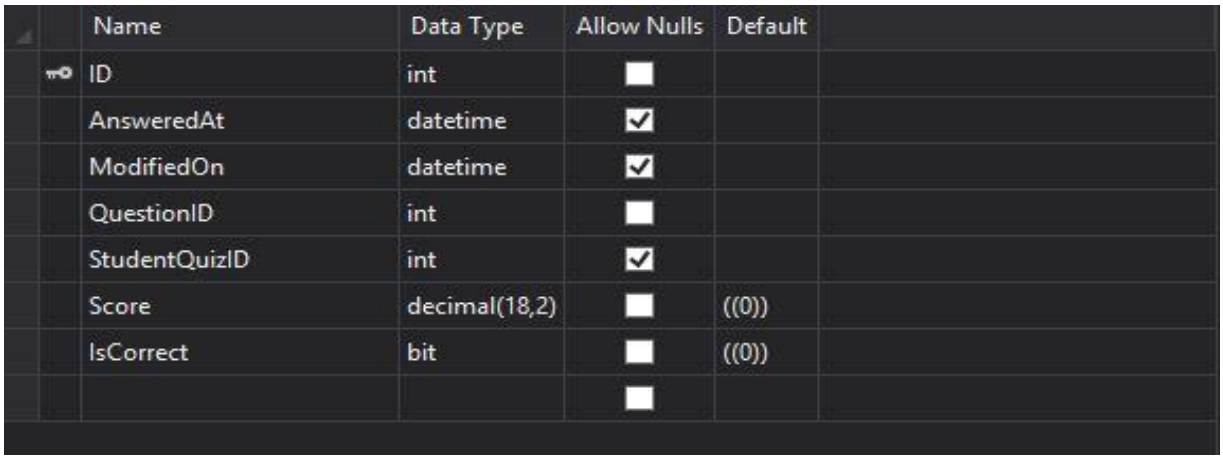


	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	AttemptedQuestionID	int	☐	
	OptionID	int	☐	
	ModifiedOn	datetime	☑	
			☐	

Рисунок 21 – Таблица «AttemptedOptions»

Таблица «AttemptedOptions» – это таблица, предназначенная для хранения опций учащихся, связанных с каждым заданным вопросом.

На рисунке 22 представлена таблица «AttemptedQuestions».

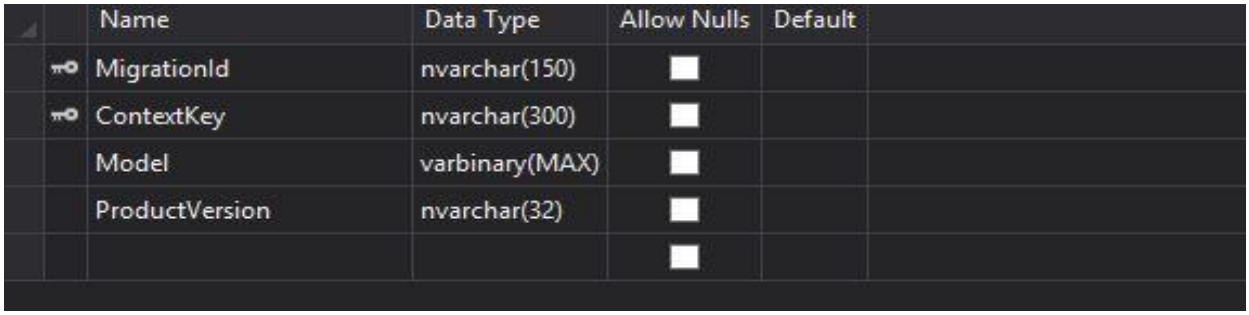


	Name	Data Type	Allow Nulls	Default
🔑	ID	int	☐	
	AnsweredAt	datetime	☑	
	ModifiedOn	datetime	☑	
	QuestionID	int	☐	
	StudentQuizID	int	☑	
	Score	decimal(18,2)	☐	((0))
	IsCorrect	bit	☐	((0))
			☐	

Рисунок 22 – Таблица «AttemptedQuestions»

Таблица «AttemptedQuestions» – это таблица, предназначенная для хранения заданных вопросов студентом для каждого задания.

На рисунке 23 представлена таблица «Migration».



	Name	Data Type	Allow Nulls	Default
🔑	MigrationId	nvarchar(150)	☐	
🔑	ContextKey	nvarchar(300)	☐	
	Model	varbinary(MAX)	☐	
	ProductVersion	nvarchar(32)	☐	
			☐	

Рисунок 23 – Таблица «Migration»

Таблица «Migration» – это таблица относится к данным первой миграции кода структуры объекта.

3.4 Программная реализация информационной системы

На рисунке 24 показана домашняя страница, на которой можно зарегистрироваться на сайте, что может позволить пользователю получить доступ к системе.

Рисунок 24 – Главная страница

На рисунке 25 представлен интерфейс окна ввода логина и пароля.

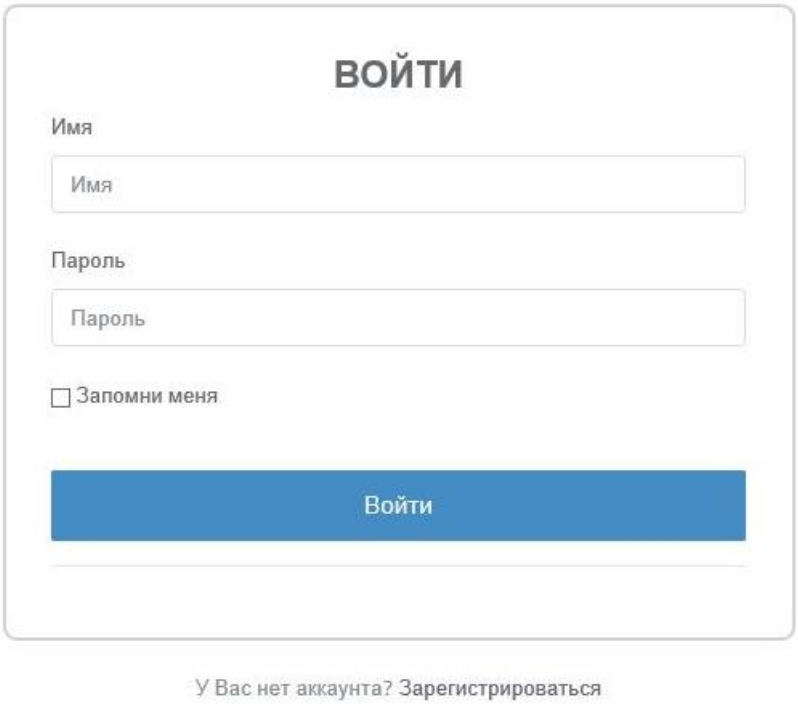


Рисунок 25 – Ввод данных в систему

При нажатии на кнопку «Войти», система проверяет входную информацию, связанную с информацией в базе данных, и позволяет пользователю получить доступ к системе.

Интерфейс ввода входной информации представлен на рисунке 26.

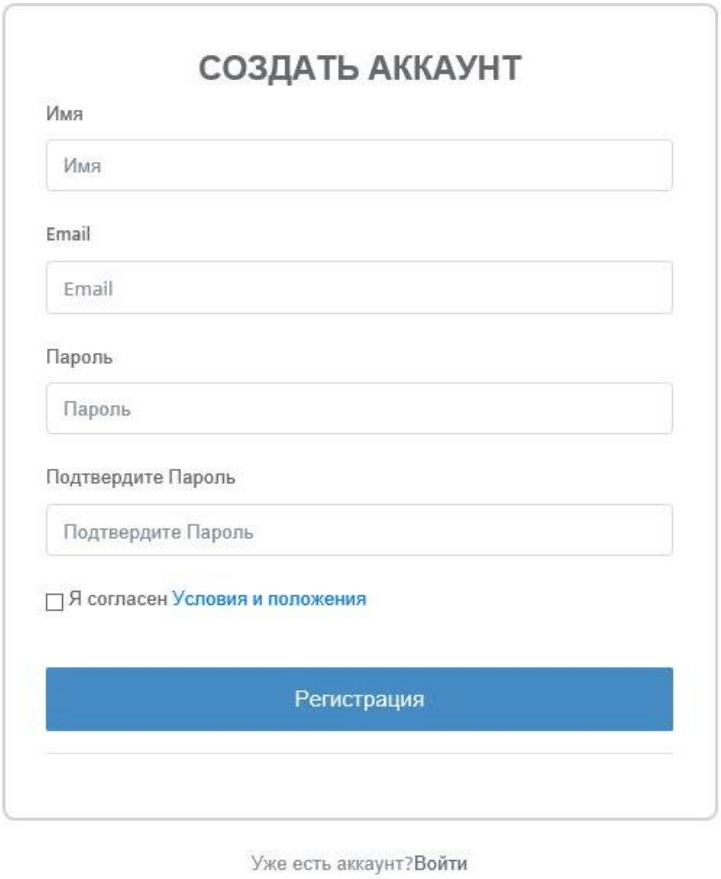


Рисунок 26 – Форма для регистрации

При нажатии на кнопку «Регистрация» система проверяет введённую информацию в отношении информации, содержащейся в базе данных, и вводятся введённые данные.

На рисунке 27 представлен интерфейс окна доступа к урокам и задачам.

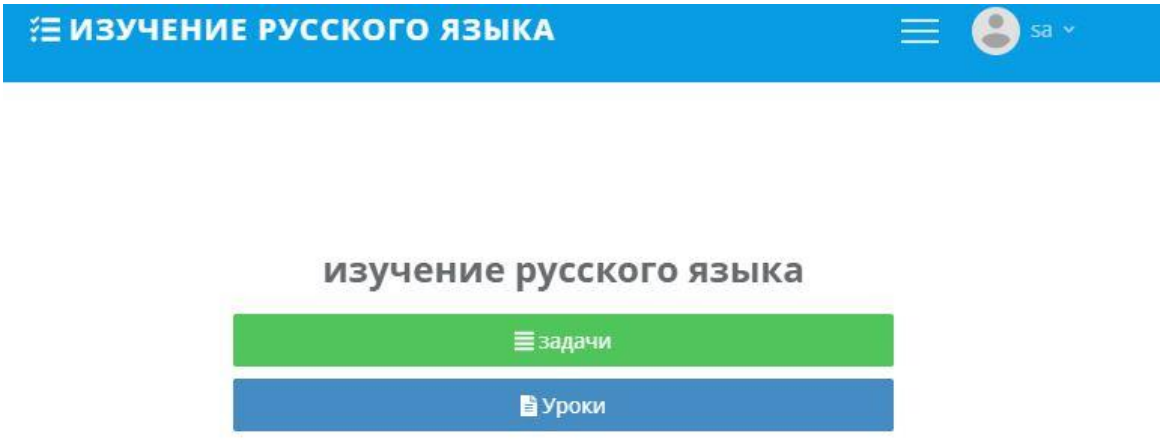


Рисунок 27 – Доступ к урокам и задачам

Данное окно отображается после входа в систему и предназначено для доступа к урокам и задачам.

На рисунке 28 представлен интерфейса окна управления задачами.

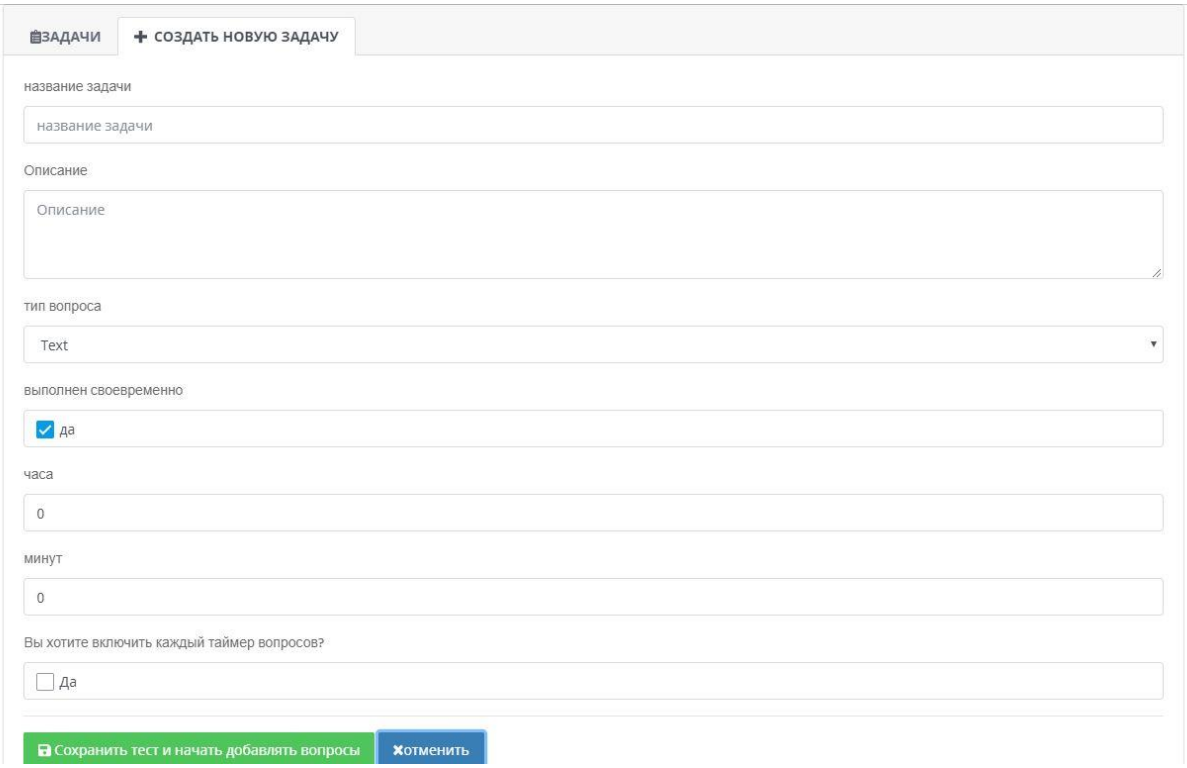


Рисунок 28 – Управление задачами

С помощью представленного окна администратор имеет возможность создания новой задачи.

На рисунке 29 представлен интерфейс окна создания задачи.

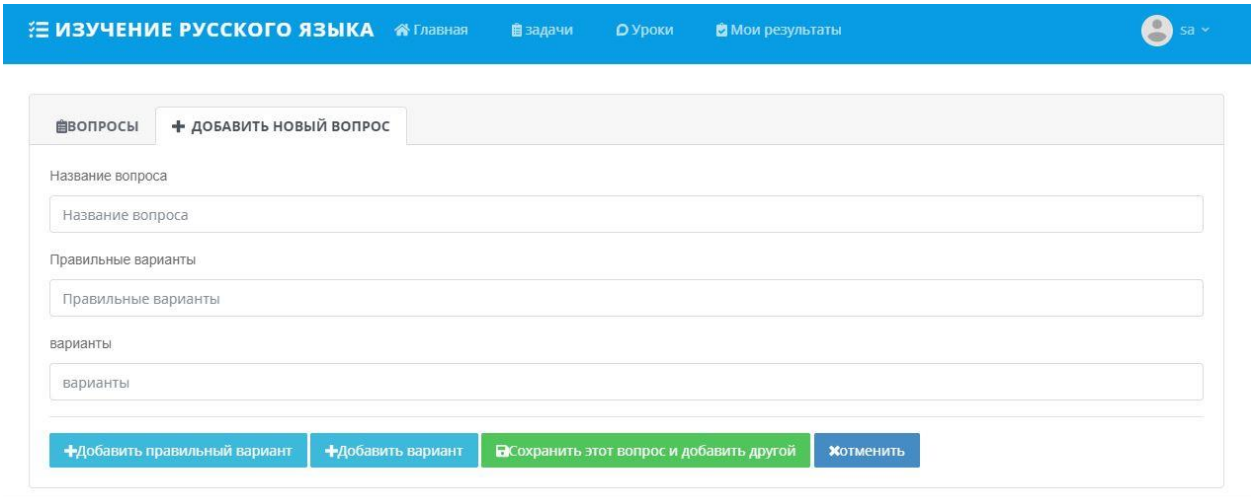


Рисунок 29 – Окно создания новой задачи

Через эту страницу администратор может создать новый вопрос.

На рисунке 30 представлено окно начала решения задачи.

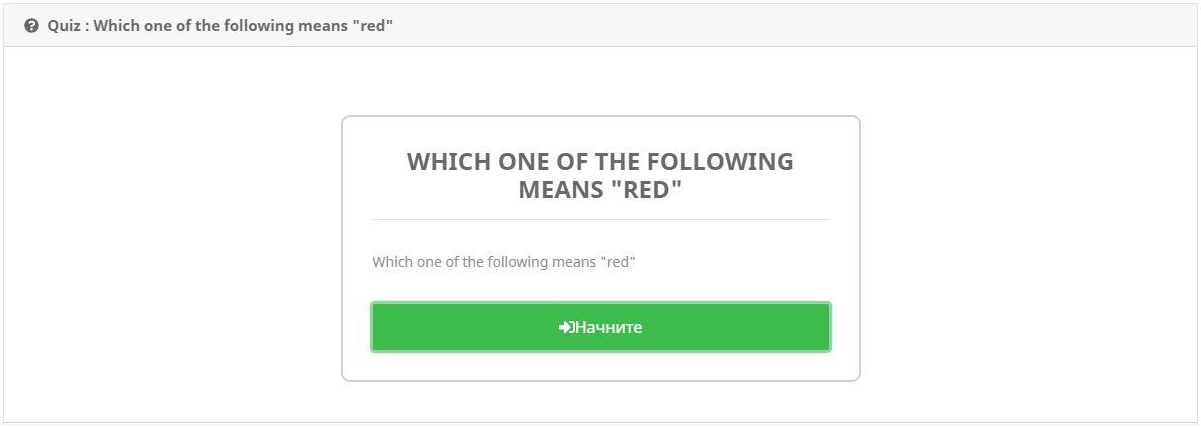


Рисунок 30 – Начать задачу

При нажатии кнопки «Начните» открывается окно с вопросами и заданиями для решения. Интерфейс данного окна представлен на рисунке 31.

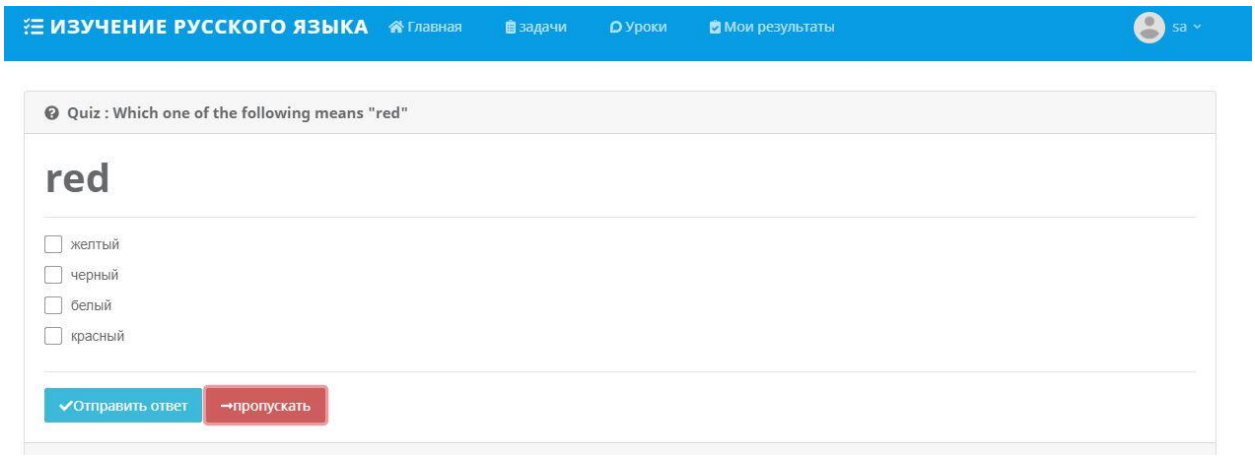


Рисунок 31 – Начало выполнения задачи

На приведенном рисунке отображается пример задания и варианты ответа для него.

На рисунке 32 показано окно просмотра результатов тестирования.

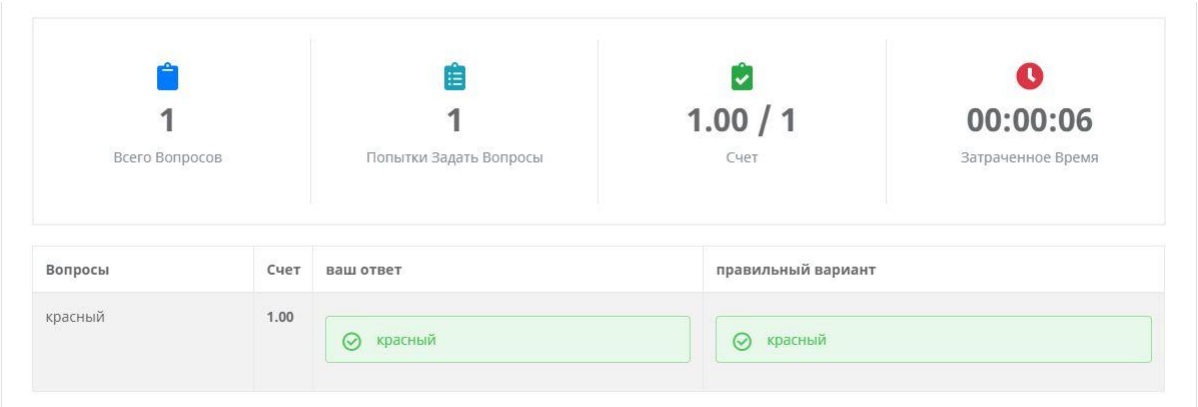


Рисунок 32 – Отчет о результатах

На приведенном рисунке отображается отчет о результатах задачи.

После того, как задача создана, имеется возможность создать новую, или редактировать, удалить существующую. Интерфейса окна, с помощью которого можно производить данные действия, представлен на рисунке 33.

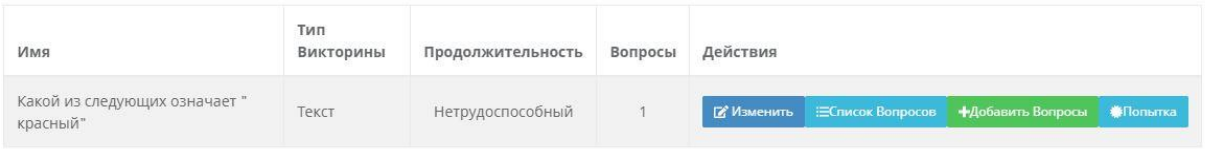


Рисунок 33 – Действия с задачей

При создании, редактировании, удалении задачи администратором, система проверяет возможность изменения.

На рисунке 34 показа профиль пользователя.

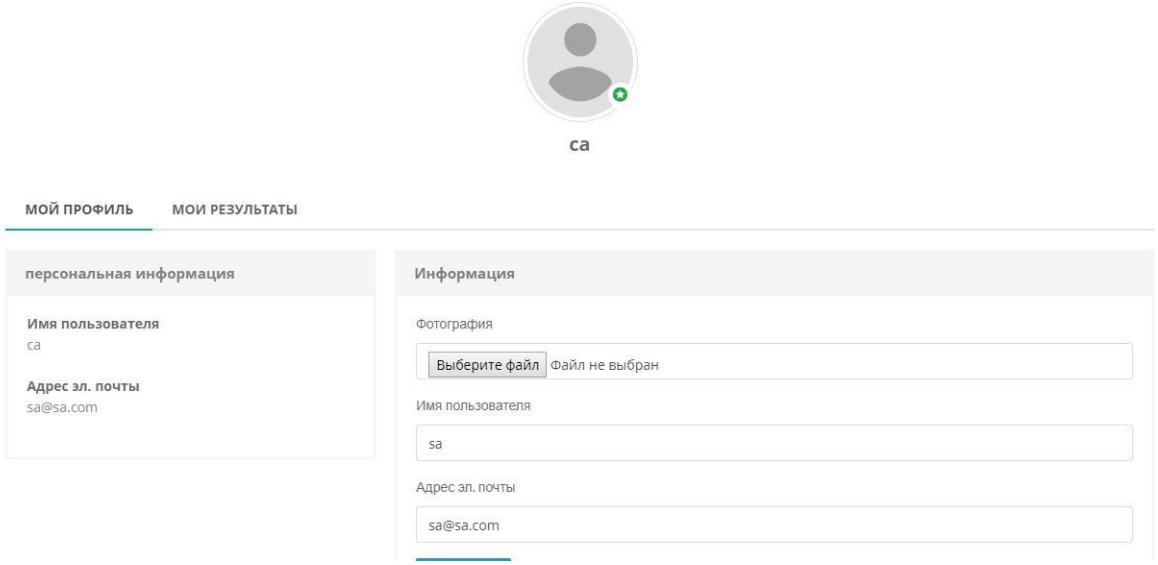


Рисунок 34 – Профиль пользователя

Эта страница показывает всю информацию о пользователе.

На рисунке 35 показан интерфейс окна для просмотра книг.

Рисунок 35 – Форма просмотра книг

В данной форме администратор, или пользователь имеет возможность просматривать все существующие книги.

На рисунке 36 показана возможность поиска книг из каталога.

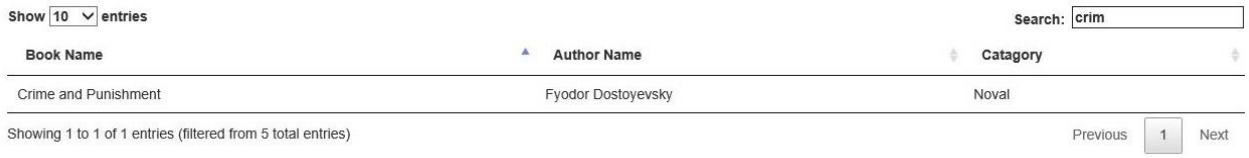


Рисунок 36 – Форма поиска

На рисунке 37 показано окно добавления новой книги.

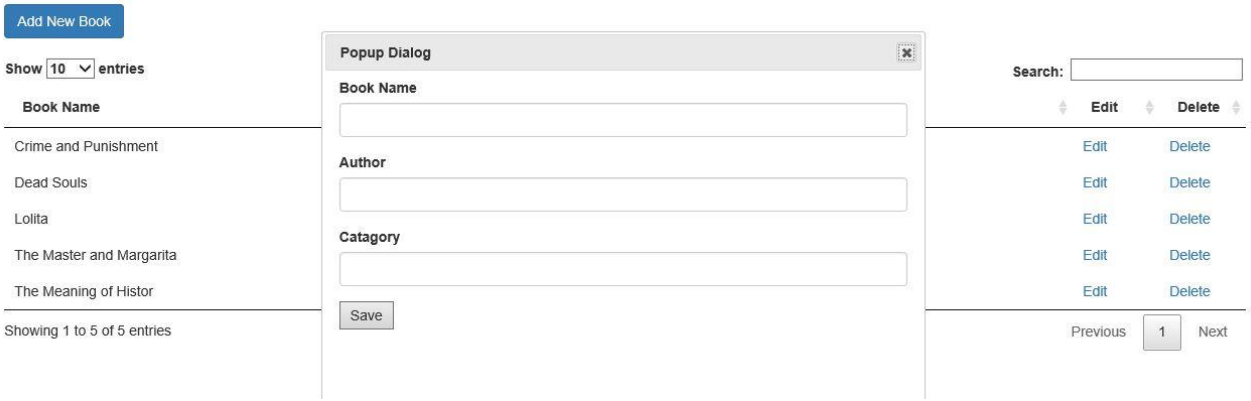


Рисунок 37 – Добавление книги

Здесь администратор создает книгу, и система проверяет возможность ее добавления.

С помощью этой же формы имеется возможность редактирования книги. Окно редактирования показано на рисунке 39.

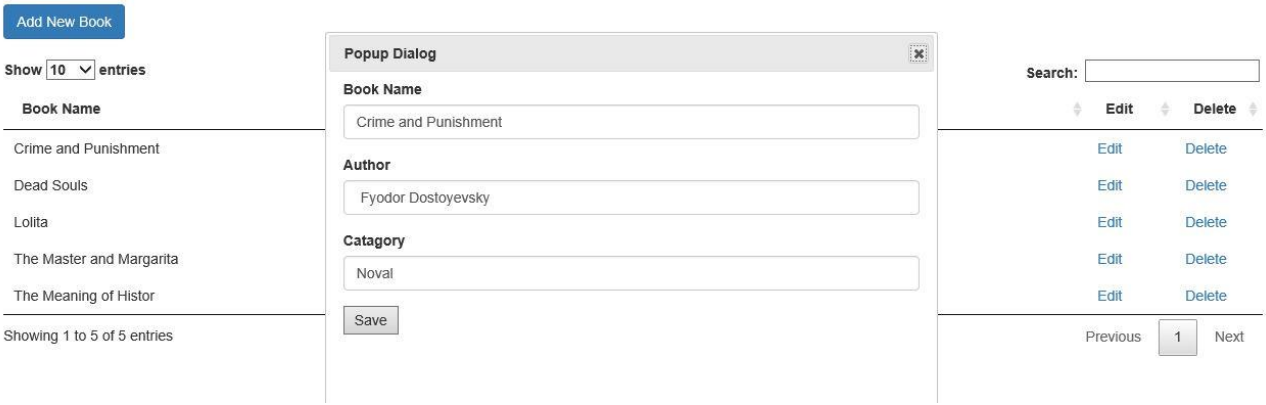


Рисунок 38 – Форма для редактирования книги

Здесь администратор изменяет информацию о книге, система проверяет действительность возможности изменения и, если имеются полномочия

администратора, то система может изменить информацию о книге.

На рисунке 40 представлено окно удаления книги.

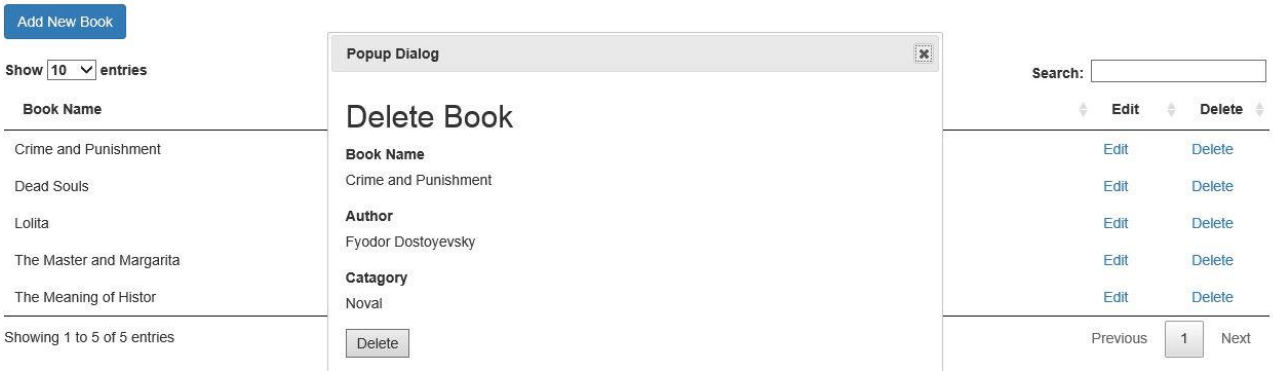


Рисунок 39 – Удаление книги

Администратор удаляет книгу, и система проверяет достоверность удаления.

Вывод по третьему разделу

В третьем разделе описаны требования для создания ИС поддержки изучение русского языка как иностранного для студентов НИУ БЕЛГУ, разработана диаграмма «сущность-связь», отражена реализация программного обеспечения.

ЗАКЛЮЧЕНИЕ

В процессе выполнения магистерской диссертационной работы были сформированы требования к информационной системе составления рабочих программ дисциплин. Подробно изучив процесс разработки рабочих программ в НИУ «БелГУ» для разработки информационной системы, был выбран следующий ряд программных продуктов: СУБД SQL Server, среда программирования Visual Studio 2017 и язык программирования C# и Asp.net.

Исходя из выявленных требований, была создана информационная система поддержки изучения русского языка, как иностранного для студентов НИУ БелГУ.

Данная система имеет простой пользовательский интерфейс и не требует какой-либо специфической подготовки и знаний в области работы с базами данных или глубоких знаний в работе с офисными программами.

В ходе выполнения выпускной квалификационной работы были получены следующие результаты:

- проанализирована предметная область;
- составлено описание бизнес-процессов в выбранных нотациях;
- спроектирована и разработана UML-модель;
- осуществлен выбор инструментальных средств разработки информационной системы;
- спроектирована и разработана база данных информационной системы;
- разработана и протестирована информационная система.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Преимущества обучения в России для иностранцев [Электронный ресурс]. – Режим доступа: https://studyinrussia.ru/why-russia/advantages/.
2. Google Trends [Электронный ресурс]. – Режим доступа: https://trends.google.com/trends/.
3. Keyword statistics – Yandex [Электронный ресурс]. – Режим доступа: https://wordstat.yandex.com/.
4. Second Language Learning Difficulties [Электронный ресурс]. – Режим доступа: https://www.myenglishpages.com/blog/second-language-learning-difficulties/.
5. Лернер, И. Я. Дидактические основы методов обучения/ И. Я. Лернер. – Педагогика. – 1981. – С. 132.
6. Беляев, Б. В. Очерки по психологии обучения иностранным языкам / Б. В. Беляев. – Просвещение: Москва. – 1965. – С. 211.
7. Трудности, возникающие у арабских студентов при изучении падежной системы русского [Электронный ресурс]. – Режим доступа: http://scjournal.ru/articles/issn_1997-2911_2016_11-1_60.pdf.
8. Онлайн-образование. История, текущее положение дел. MOOC-платформы [Электронный ресурс]. – Режим доступа: https://spark.ru/startup/ewave/blog/10890/onlajn-obrazovanie-istoriya-tekuschee-polozhenie-del-mooc-platformi.
9. MIT OpenCourseWare | Free Online Course Materials [Электронный ресурс]. – Режим доступа: https://ocw.mit.edu/about/.
10. Tarek Abdel Raouf Mohamed Education and e-learning/ Mohamed Tarek Abdel Raouf. – Al Yazori Publishing House-Amman. – 2018. – С. 12.
11. Sherif Al-Attrabi E-learning and information services/ Al-Attrabi Sherif. – Al Arabi Publishing and Distributing-Cairo. – 2015. – С. 124.
12. Диаграмма классов (class diagram) [Электронный ресурс]. – Режим доступа: http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl5/gl5.html.

13. Программа развития Института «Высшая школа экономики и менеджмента Уральского федерального университета им. Первого Президента России Б.Н. Ельцина (ВШЭМ УрФУ) в 2011-2016 гг. и на период 2020 г. – 2011. – С. 3-16.

14. Моделирование на UML. Назначение UML [Электронный ресурс]. – Режим доступа: https://book.uml3.ru/sec_1_2.

15. Петров, В. Н. Информационные системы/ В. Н. Петров. – СПб: Питер. – 2007. – С. 688.

16. ГОСТ 34.602-89 Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – М.: Госстандарт СССР. – 1990. – С. 7.

17. Конституция Российской Федерации (принята всенародным голосованием 12.12.1993) (с учетом поправок, внесенных Законами РФ о поправках к Конституции РФ от 30.12.2008 № 6-ФКЗ, от 30.12.2008 № 7-ФКЗ, от 05.02.2014 № 2 - ФКЗ) // Собрание законодательства РФ, 14.04.2014. № 15. – ст. 1691.

18. Суэринг, С. PHP и MySQL. Библия программиста, 2-е изд. / С. Суэринг, Т. Конверс, Дж. Парк: пер. с англ. – М.: ООО «И.Д. Вильямс». – 2010. – С. 912.

19. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. – М.: Госстандарт СССР. – 1990. – С. 11.

20. Информационные технологии в профессиональной [Электронный ресурс]. – Режим доступа: https://studref.com/336133/informatika/arhitektura.

21. Классификация по способу организации [Электронный ресурс]. – Режим доступа: https://helpiks.org/8-17500.html.

22. Компонент SQL Server Database Engine [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/sql/database-engine/sql-server-database-engine-overview?view=sql-server-2017.

23. Основные сведения о СУБД Microsoft SQL Server [Электронный

ресурс]. – Режим доступа: https://studfiles.net/preview/4418102/.

24. История языка C# [Электронный ресурс]. – Режим доступа: https://it-black.ru/istoriya-yazyka-ci_sharp/.

25. Общие сведения о платформе .NET Framework [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview.

26. ASP.NET [Электронный ресурс]. – Режим доступа: https://life-prog.ru/view_zam2.php?id=130&cat=5&page=1.

27. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://dic.academic.ru/dic.nsf/ruwiki/56677.

28. Модель сущность-атрибут-связь (er) [Электронный ресурс]. – Режим доступа: http://csaa.ru/model-sushhnost-atribut-svjaz-er/.

29. Белгородский государственный национальный исследовательский университет. Университет сегодня [Электронный ресурс]. – Режим доступа: https://www.bsu.edu.ru/bsu/info/today/index.php.

30. Маторин, С. И. Теория систем и системный анализ. Учебное пособие / С. И. Маторин, О. А. Зимовец. – Белгород: Белгородский государственный университет.

31. Блумфилд, Л. Краткое руководство по практическому изучению иностранных языков. Методика преподавания иностранных языков за рубежом / Л. Блумфилд. – Москва. – 1967. – С. 25.

32. C# programming guide [Электронный ресурс]. – Режим доступа: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx.

33. Getting Started with ASP.NET MVC 5 [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/arsa/aspnet/mvc/overview/gettingstarted/introduction/getting-started.

34. Самоучитель по языку SQL (SQLDML) [Электронный ресурс]. – Режим доступа: http://www.sql-ex.ru/help.

35. C# Station [Электронный ресурс]. – Режим доступа: http://csharp-

station.com/.

36. Learn to Build Great C# Apps [Электронный ресурс]. – Режим доступа: https://mva.microsoft.com/colleges/c_cert.

37. Stack Overflow Business Solutions [Электронный ресурс]. – Режим доступа: https://stackoverflow.com/.

38. HTML и Web дизайн для начинающих [Электронный ресурс]. – Режим доступа: http://sovet.h1.ru.

39. Создание шаблона сайта с помощью CSS [Электронный ресурс]. – Режим доступа: http://ruseller.com.

40. Создание страниц сайта с помощью HTML [Электронный ресурс]. – Режим доступа: http://roumik.ru.

41. Создание сайтов: HTML [Электронный ресурс]. – Режим доступа: http://www.codeharmony.ru.

42. Основы CSS и HTML [Электронный ресурс]. – Режим доступа: http://www.web–lesson.ru.

ПРИЛОЖЕНИЕ А

Программный код информационной системы

AccountController.cs

```
using System;
using System.Globalization;
using System.Linq;
using System.Security.Claims;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using QuizbeePlus.ViewModels;
using QuizbeePlus.Data;
using QuizbeePlus.Entities;
using QuizbeePlus.Commons;

namespace QuizbeePlus.Controllers
{
   [Authorize]
   public class AccountController : Controller
   {
      private QuizbeeSignInManager _signInManager;
      private QuizbeeUserManager _userManager;

      public AccountController()
      {
      }

      public AccountController(QuizbeeUserManager userManager, QuizbeeSignInManager signInManager )
      {
         UserManager = userManager;
         SignInManager = signInManager;
      }

      public QuizbeeSignInManager SignInManager
      {
         get
         {
            return _signInManager ?? HttpContext.GetOwinContext().Get<QuizbeeSignInManager>();
         }
         private set
         {
            _signInManager = value;
         }
      }

      public QuizbeeUserManager UserManager
      {
         get
         {
            return _userManager ?? HttpContext.GetOwinContext().GetUserManager<QuizbeeUserManager>();
         }
         private set
         {
            _userManager = value;
```

```
        }
    }

    //
    // GET: /Account/Login
    [AllowAnonymous]
    public ActionResult Login(string returnUrl)
    {
        ViewBag.ReturnUrl = returnUrl;

        return View("Login", "_LayoutEmpty", new LoginViewModel());
    }

    //
    // POST: /Account/Login
    [HttpPost]
    [AllowAnonymous]
    [ValidateAntiForgeryToken]
    public async Task<ActionResult> Login(LoginViewModel model, string returnUrl)
    {
        if (!ModelState.IsValid)
        {
            return View("Login", "_LayoutEmpty", model);
        }

        // This doesn't count login failures towards account lockout
        // To enable password failures to trigger account lockout, change to shouldLockout: true
        var result = await SignInManager.PasswordSignInAsync(model.Username, model.Password,
model.RememberMe, shouldLockout: false);
        switch (result)
        {
            case SignInStatus.Success:
                return RedirectToLocal(returnUrl);
            case SignInStatus.LockedOut:
                return View("Lockout");
            case SignInStatus.RequiresVerification:
                return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe =
model.RememberMe });
            case SignInStatus.Failure:
            default:
                ModelState.AddModelError("", "Invalid login attempt.");
                return View("Login", "_LayoutEmpty", model);
        }
    }

    //
    // GET: /Account/VerifyCode
    [AllowAnonymous]
    public async Task<ActionResult> VerifyCode(string provider, string returnUrl, bool rememberMe)
    {
        // Require that the user has already logged in via username/password or external login
        if (!await SignInManager.HasBeenVerifiedAsync())
        {
            return View("Error");
        }
        return View(new VerifyCodeViewModel { Provider = provider, ReturnUrl = returnUrl, RememberMe =
rememberMe });
    }

    //
    // POST: /Account/VerifyCode
    [HttpPost]
```

```
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> VerifyCode(VerifyCodeViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }

            // The following code protects for brute force attacks against the two factor codes.
            // If a user enters incorrect codes for a specified amount of time then the user account
            // will be locked out for a specified amount of time.
            // You can configure the account lockout settings in IdentityConfig
            var result = await SignInManager.TwoFactorSignInAsync(model.Provider, model.Code, isPersistent:
model.RememberMe, rememberBrowser: model.RememberBrowser);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(model.ReturnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.Failure:
                default:
                    ModelState.AddModelError("", "Invalid code.");
                    return View(model);
            }
        }

        //
        // GET: /Account/Register
        [AllowAnonymous]
        public ActionResult Register()
        {
            return View("Register", "_LayoutEmpty", new RegisterViewModel());
        }

        //
        // POST: /Account/Register
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> Register(RegisterViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = new QuizbeeUser { UserName = model.Username, Email = model.Email, RegisteredOn =
DateTime.Now};
                var result = await UserManager.CreateAsync(user, model.Password);
                if (result.Succeeded)
                {
                    UserManager.AddToRole(user.Id, Variables.UserRole);

                    await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);

                    return RedirectToAction("IndexUser", "Home");
                }
                AddErrors(result);
            }

            return View("Register", "_LayoutEmpty", model);
        }
```

```
        //
        // GET: /Account/ConfirmEmail
        [AllowAnonymous]
        public async Task<ActionResult> ConfirmEmail(string userId, string code)
        {
            if (userId == null || code == null)
            {
                return View("Error");
            }
            var result = await UserManager.ConfirmEmailAsync(userId, code);
            return View(result.Succeeded ? "ConfirmEmail" : "Error");
        }

        //
        // GET: /Account/ForgotPassword
        [AllowAnonymous]
        public ActionResult ForgotPassword()
        {
            return View();
        }

        //
        // POST: /Account/ForgotPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ForgotPassword(ForgotPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var user = await UserManager.FindByNameAsync(model.Email);
                if (user == null || !(await UserManager.IsEmailConfirmedAsync(user.Id)))
                {
                    // Don't reveal that the user does not exist or is not confirmed
                    return View("ForgotPasswordConfirmation");
                }

                // For more information on how to enable account confirmation and password reset please visit
http://go.microsoft.com/fwlink/?LinkID=320771
                // Send an email with this link
                // string code = await UserManager.GeneratePasswordResetTokenAsync(user.Id);
                // var callbackUrl = Url.Action("ResetPassword", "Account", new { userId = user.Id, code = code },
protocol: Request.Url.Scheme);
                // await UserManager.SendEmailAsync(user.Id, "Reset Password", "Please reset your password by
clicking <a href=\"" + callbackUrl + "\">here</a>");
                // return RedirectToAction("ForgotPasswordConfirmation", "Account");
            }

            // If we got this far, something failed, redisplay form
            return View(model);
        }

        //
        // GET: /Account/ForgotPasswordConfirmation
        [AllowAnonymous]
        public ActionResult ForgotPasswordConfirmation()
        {
            return View();
        }

        //
        // GET: /Account/ResetPassword
```

```
        [AllowAnonymous]
        public ActionResult ResetPassword(string code)
        {
            return code == null ? View("Error") : View();
        }

        //
        // POST: /Account/ResetPassword
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ResetPassword(ResetPasswordViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var user = await UserManager.FindByNameAsync(model.Email);
            if (user == null)
            {
                // Don't reveal that the user does not exist
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            var result = await UserManager.ResetPasswordAsync(user.Id, model.Code, model.Password);
            if (result.Succeeded)
            {
                return RedirectToAction("ResetPasswordConfirmation", "Account");
            }
            AddErrors(result);
            return View();
        }

        //
        // GET: /Account/ResetPasswordConfirmation
        [AllowAnonymous]
        public ActionResult ResetPasswordConfirmation()
        {
            return View();
        }

        //
        // POST: /Account/ExternalLogin
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public ActionResult ExternalLogin(string provider, string returnUrl)
        {
            // Request a redirect to the external login provider
            return new ChallengeResult(provider, Url.Action("ExternalLoginCallback", "Account", new { ReturnUrl =
returnUrl }));
        }

        //
        // GET: /Account/SendCode
        [AllowAnonymous]
        public async Task<ActionResult> SendCode(string returnUrl, bool rememberMe)
        {
            var userId = await SignInManager.GetVerifiedUserIdAsync();
            if (userId == null)
            {
                return View("Error");
            }
```

```
            var userFactors = await UserManager.GetValidTwoFactorProvidersAsync(userId);
            var factorOptions = userFactors.Select(purpose => new SelectListItem { Text = purpose, Value = purpose
}).ToList();
            return View(new SendCodeViewModel { Providers = factorOptions, ReturnUrl = returnUrl, RememberMe =
rememberMe });
        }

        //
        // POST: /Account/SendCode
        [HttpPost]
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SendCode(SendCodeViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View();
            }

            // Generate the token and send it
            if (!await SignInManager.SendTwoFactorCodeAsync(model.SelectedProvider))
            {
                return View("Error");
            }
            return RedirectToAction("VerifyCode", new { Provider = model.SelectedProvider, ReturnUrl =
model.ReturnUrl, RememberMe = model.RememberMe });
        }

        //
        // GET: /Account/ExternalLoginCallback
        [AllowAnonymous]
        public async Task<ActionResult> ExternalLoginCallback(string returnUrl)
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync();
            if (loginInfo == null)
            {
                return RedirectToAction("Login");
            }

            // Sign in the user with this external login provider if the user already has a login
            var result = await SignInManager.ExternalSignInAsync(loginInfo, isPersistent: false);
            switch (result)
            {
                case SignInStatus.Success:
                    return RedirectToLocal(returnUrl);
                case SignInStatus.LockedOut:
                    return View("Lockout");
                case SignInStatus.RequiresVerification:
                    return RedirectToAction("SendCode", new { ReturnUrl = returnUrl, RememberMe = false });
                case SignInStatus.Failure:
                default:
                    // If the user does not have an account, then prompt the user to create an account
                    ViewBag.ReturnUrl = returnUrl;
                    ViewBag.LoginProvider = loginInfo.Login.LoginProvider;
                    return View("ExternalLoginConfirmation", new ExternalLoginConfirmationViewModel { Email =
loginInfo.Email, Username = loginInfo.DefaultUserName });
            }
        }

        //
        // POST: /Account/ExternalLoginConfirmation
        [HttpPost]
```

```
        [AllowAnonymous]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ExternalLoginConfirmation(ExternalLoginConfirmationViewModel model,
string returnUrl)
        {
            if (User.Identity.IsAuthenticated)
            {
                return RedirectToAction("Index", "Manage");
            }

            if (ModelState.IsValid)
            {
                // Get the information about the user from the external login provider
                var info = await AuthenticationManager.GetExternalLoginInfoAsync();
                if (info == null)
                {
                    return View("ExternalLoginFailure");
                }
                var user = new QuizbeeUser { UserName = model.Username, Email = model.Email, RegisteredOn =
DateTime.Now };
                var result = await UserManager.CreateAsync(user);
                if (result.Succeeded)
                {
                    UserManager.AddToRole(user.Id, Variables.UserRole);

                    result = await UserManager.AddLoginAsync(user.Id, info.Login);
                    if (result.Succeeded)
                    {
                        await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                        return RedirectToLocal(returnUrl);
                    }
                }
                AddErrors(result);
            }

            ViewBag.ReturnUrl = returnUrl;
            return View(model);
        }

        //
        // POST: /Account/LogOff
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LogOff()
        {
            AuthenticationManager.SignOut(DefaultAuthenticationTypes.ApplicationCookie);
            return RedirectToAction("Index", "HomePage");
        }

        //
        // GET: /Account/ExternalLoginFailure
        [AllowAnonymous]
        public ActionResult ExternalLoginFailure()
        {
            return View();
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                if (_userManager != null)
```

```
        {
            _userManager.Dispose();
            _userManager = null;
        }

        if (_signInManager != null)
        {
            _signInManager.Dispose();
            _signInManager = null;
        }
    }

    base.Dispose(disposing);
}

#region Helpers
// Used for XSRF protection when adding external logins
private const string XsrfKey = "XsrfId";

private IAuthenticationManager AuthenticationManager
{
    get
    {
        return HttpContext.GetOwinContext().Authentication;
    }
}

private void AddErrors(IdentityResult result)
{
    foreach (var error in result.Errors)
    {
        ModelState.AddModelError("", error);
    }
}

private ActionResult RedirectToLocal(string returnUrl)
{
    if (Url.IsLocalUrl(returnUrl))
    {
        return Redirect(returnUrl);
    }
    return RedirectToAction("IndexUser", "Home");
}

internal class ChallengeResult : HttpUnauthorizedResult
{
    public ChallengeResult(string provider, string redirectUri)
        : this(provider, redirectUri, null)
    {
    }

    public ChallengeResult(string provider, string redirectUri, string userId)
    {
        LoginProvider = provider;
        RedirectUri = redirectUri;
        UserId = userId;
    }

    public string LoginProvider { get; set; }
    public string RedirectUri { get; set; }
    public string UserId { get; set; }
```

```
        public override void ExecuteResult(ControllerContext context)
        {
            var properties = new AuthenticationProperties { RedirectUri = RedirectUri };
            if (UserId != null)
            {
                properties.Dictionary[XsrfKey] = UserId;
            }
            context.HttpContext.GetOwinContext().Authentication.Challenge(properties, LoginProvider);
        }
    }
    #endregion
  }
}
```

AttemptQuizController.cs

```
using Microsoft.AspNet.Identity;
using QuizbeePlus.Commons;
using QuizbeePlus.Entities;
using QuizbeePlus.Services;
using QuizbeePlus.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;

namespace QuizbeePlus.Controllers
{
    public class AttemptQuizController : BaseController
    {
        [HttpGet]
        public ActionResult Attempt(int QuizID)
        {
            var quiz = QuizzesService.Instance.GetQuiz(QuizID);

            if (quiz == null) return HttpNotFound();

            QuizDetailViewModel model = new QuizDetailViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = string.Format("Quiz : {0}", quiz.Name),
                PageDescription = quiz.Description
            };

            model.ID = quiz.ID;
            model.Name = quiz.Name;
            model.Description = quiz.Description;
            model.TimeDuration = quiz.TimeDuration;
            model.EnableQuizTimer = quiz.EnableQuizTimer;

            return View(model);
        }

        [HttpPost]
        public async Task<ActionResult> Attempt(AttemptQuizViewModel model)
        {
            var quiz = QuizzesService.Instance.GetQuiz(model.QuizID);
```

```
            if (quiz == null) return HttpNotFound();

            StudentQuiz studentQuiz = new StudentQuiz();

            studentQuiz.StudentID = User.Identity.GetUserId();
            studentQuiz.QuizID = quiz.ID;
            studentQuiz.StartedAt = DateTime.Now;
            studentQuiz.ModifiedOn = DateTime.Now;

            if (await StudentQuizzesService.Instance.NewStudentQuiz(studentQuiz))
            {
                model.QuizType = quiz.QuizType;
                model.StudentQuizID = studentQuiz.ID;
                model.TotalQuestions = quiz.Questions.Count;
                model.Question = quiz.Questions.FirstOrDefault();
                model.QuestionIndex = 0;

                model.Options = new List<Option>();
                model.Options.AddRange(model.Question.Options);
                model.Options.Shuffle();

                model.EnableQuestionTimer = quiz.EnableQuestionTimer;
                model.Seconds = Calculator.CalculateAllowedQuestionTime(quiz);

                return PartialView("_QuizQuestion", model);
            }
            else
            {
                return new HttpStatusCodeResult(500);
            }
        }

        [HttpPost]
        public async Task<ActionResult> AnswerQuestion(AttemptQuizViewModel model)
        {
            Thread.Sleep(2000);

            var studentQuiz =
StudentQuizzesService.Instance.GetStudentQuiz(model.StudentQuizID);

            if (studentQuiz == null) return new HttpStatusCodeResult(500);

            if (model.TimerExpired)
            {
                studentQuiz.CompletedAt = DateTime.Now;
                studentQuiz.ModifiedOn = DateTime.Now;

                if (await StudentQuizzesService.Instance.UpdateStudentQuiz(studentQuiz))
                {
                    StudentQuizViewModel studentQuizModel = new StudentQuizViewModel();

                    studentQuizModel.StudentQuiz = studentQuiz;
                    studentQuizModel.TimerExpired = model.TimerExpired;

                    return PartialView("_AttemptDetails", studentQuizModel);
                }
                else return new HttpStatusCodeResult(500);
            }
            else
            {
                var quiz = QuizzesService.Instance.GetQuiz(model.QuizID);

                if (quiz == null) return HttpNotFound();
```

```
                var question = QuestionsService.Instance.GetQuizQuestion(quiz.ID,
model.QuestionID);

                if (question == null) return HttpNotFound();

                var selectedOptions =
QuestionsService.Instance.GetOptionsByIDs(model.SelectedOptionIDs.CSVToListInt());

                if (selectedOptions == null) return HttpNotFound();

                AttemptedQuestion attemptedQuestion = new AttemptedQuestion();

                attemptedQuestion.StudentQuizID = studentQuiz.ID;
                attemptedQuestion.QuestionID = question.ID;
                attemptedQuestion.SelectedOptions = selectedOptions.Select(x => new
AttemptedOption() { AttemptedQuestionID = attemptedQuestion.ID, Option = x, OptionID =
x.ID }).ToList();
                attemptedQuestion.Score =
Calculator.CalculateAttemptedQuestionScore(question.Options,
attemptedQuestion.SelectedOptions);
                attemptedQuestion.AnsweredAt = DateTime.Now;
                attemptedQuestion.ModifiedOn = DateTime.Now;

                if (await
StudentQuizzesService.Instance.NewAttemptedQuestion(attemptedQuestion))
                {
                    if (model.QuestionIndex != quiz.Questions.Count() - 1)
                    {
                        model.QuizType = quiz.QuizType;

                        model.TotalQuestions = quiz.Questions.Count;
                        model.Question =
quiz.Questions.ElementAtOrDefault(model.QuestionIndex + 1);
                        model.QuestionIndex = model.QuestionIndex + 1;

                        model.Options = new List<Option>();
                        model.Options.AddRange(model.Question.Options);
                        model.Options.Shuffle();

                        model.EnableQuestionTimer = quiz.EnableQuestionTimer;
                        model.Seconds = Calculator.CalculateAllowedQuestionTime(quiz);

                        return PartialView("_QuizQuestion", model);
                    }
                    else //this was the Last question so display the result
                    {
                        studentQuiz.CompletedAt = DateTime.Now;

                        if (!await
StudentQuizzesService.Instance.UpdateStudentQuiz(studentQuiz))
                            return new HttpStatusCodeResult(500);

                        return RedirectToAction("AttemptDetails", new { studentQuizID =
studentQuiz.ID, isPartial = true, timerExpired = model.TimerExpired });
                    }
                }
                else return new HttpStatusCodeResult(500);
            }
        }

            [HttpGet]
            public ActionResult AttemptDetails(int studentQuizID, bool? timerExpired,
bool? isPartial = false)
            {
```

```
                    var studentQuiz =
StudentQuizzesService.Instance.GetStudentQuiz(studentQuizID);

            if (studentQuiz == null) return HttpNotFound();

                    StudentQuizViewModel model = new StudentQuizViewModel();

            model.TimerExpired = timerExpired.HasValue ? timerExpired.Value : false;

                    model.PageInfo = new PageInfo()
                    {
                           PageTitle = "Quiz Attempt Details",
                           PageDescription = "Details of Attempted Quiz"
                    };

                    model.StudentQuiz = studentQuiz;

            if(isPartial.HasValue && isPartial.Value)
            {
                return PartialView("_AttemptDetails", model);
            }
            else
            {
                return View(model);
            }
              }

        [HttpGet]
        public ActionResult MyResults(string search, int? page, int? items, bool?
isPartial)
        {
            StudentQuizListViewModel model = new StudentQuizListViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "My Quiz Results",
                PageDescription = "Results of My Quiz Attempts"
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 10;

            var resultsSearch =
StudentQuizzesService.Instance.GetUserQuizAttempts(User.Identity.GetUserId(),
model.searchTerm, model.pageNo, model.pageSize);

            model.StudentQuizzes = resultsSearch.StudentQuizzes;
            model.TotalCount = resultsSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            if (isPartial.HasValue && isPartial.Value)
            {
                model.isPartialView = true;
                return PartialView(model);
            }
            else return View(model);
        }

        [HttpGet]
        public ActionResult PrintResult(int studentQuizID)
        {
```

```
            var studentQuiz =
StudentQuizzesService.Instance.GetStudentQuiz(studentQuizID);

            if (studentQuiz == null) return HttpNotFound();

            StudentQuizViewModel model = new StudentQuizViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Quiz Attempt Details",
                PageDescription = "Details of Attempted Quiz"
            };

            model.StudentQuiz = studentQuiz;

            return PartialView("_PrintResult", model);
        }
    }
}
```

BaseController.cs

```
using QuizbeePlus.Commons;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace QuizbeePlus.Controllers
{
    [Authorize]
    public class BaseController : Controller
    {
        protected override void OnException(ExceptionContext filterContext)
        {
            Exception ex = filterContext.Exception;

            #if DEBUG
            throw ex;
            #endif

            //Log Exception e
            filterContext.ExceptionHandled = true;
            filterContext.Result = new ViewResult()
            {
                ViewName = "Error"
            };
        }

        public bool UserHasRights()
        {
            return User.IsInRole(Variables.Administrator);
        }

        public ActionResult UnAuthorized()
        {
            Response.StatusCode = 403;

            return View();
        }
    }
}
```

ControlPanelController.cs

```
using QuizbeePlus.Data;
using QuizbeePlus.Services;
using QuizbeePlus.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using System.Globalization;
using System.Security.Claims;
using System.Threading.Tasks;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using QuizbeePlus.Entities;
using QuizbeePlus.Commons;
using Microsoft.AspNet.Identity.EntityFramework;

namespace QuizbeePlus.Controllers
{
   [CustomAuthorize(Roles = "Administrator")]
   public class ControlPanelController : BaseController
   {
      public ActionResult Index(bool? isPartial = false)
      {
         ControlPanelViewModel model = new ControlPanelViewModel();

         model.PageInfo = new PageInfo()
         {
            PageTitle = "Control Panel",
            PageDescription = "Manage Quizbee."
         };

         return View(model);
      }

      public ActionResult Users(string search, int? page, int? items)
      {
         UsersListingViewModel model = new UsersListingViewModel();

         model.PageInfo = new PageInfo()
         {
            PageTitle = "Users",
            PageDescription = "Quizbee Users."
         };

         model.searchTerm = search;
         model.pageNo = page ?? 1;
         model.pageSize = items ?? 10;

         var usersSearch = UsersService.Instance.GetUsersWithRoles(model.searchTerm, model.pageNo,
model.pageSize);

         model.Users = usersSearch.Users;
         model.TotalCount = usersSearch.TotalCount;

         model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);
```

```
        return PartialView("_Users", model);
    }

    public ActionResult UserDetails(string ID)
    {
        UserDetailsViewModel model = new UserDetailsViewModel();

        model.PageInfo = new PageInfo()
        {
            PageTitle = "User Details",
            PageDescription = "User Details"
        };

        model.User = UsersService.Instance.GetUserWithRolesByID(ID);
        model.AvailableRoles = ControlPanelService.Instance.GetAllRoles();

        //remove roles from dropdown which are already with the user
        foreach (var userRole in model.User.Roles)
        {
            var availableRole = model.AvailableRoles.Where(x => x.Id.Equals(userRole.Id)).FirstOrDefault();

            if(availableRole != null)
            {
                model.AvailableRoles.Remove(availableRole);
            }
        }

        return PartialView("_UserDetails", model);
    }

    public ActionResult Roles(string search, int? page, int? items)
    {
        RolesListingViewModel model = new RolesListingViewModel();

        model.PageInfo = new PageInfo()
        {
            PageTitle = "Roles",
            PageDescription = "Quizbee Roles."
        };

        model.searchTerm = search;
        model.pageNo = page ?? 1;
        model.pageSize = items ?? 10;

        var rolesSearch = ControlPanelService.Instance.GetRoles(model.searchTerm, model.pageNo,
model.pageSize);

        model.Roles = rolesSearch.Roles;
        model.TotalCount = rolesSearch.TotalCount;

        model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

        return PartialView("_Roles", model);
    }

    public ActionResult NewRole(string ID)
    {
        NewRoleViewModel model = new NewRoleViewModel();

        return PartialView("_NewRole", model);
    }
```

```
        [HttpPost]
        public async Task<JsonResult> NewRole(NewRoleViewModel model)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            if (!ModelState.IsValid)
            {
                var Errors = new List<string>();

                foreach (ModelState modelState in ViewData.ModelState.Values)
                {
                    foreach (ModelError error in modelState.Errors)
                    {
                        Errors.Add(error.ErrorMessage.ToString());
                    }
                }

                result.Data = new { Success = false, Errors = Errors };

                return result;
            }
            else
            {
                try
                {
                    result.Data = new { Success = await ControlPanelService.Instance.NewRole(new IdentityRole() {
Name = model.Name })};
                }
                catch (Exception ex)
                {
                    result.Data = new { Success = false, Errors = ex.InnerException.Message };
                }
            }

            return result;
        }

        [HttpPost]
        public async Task<JsonResult> AddUserRole(string userID, string roleID)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            try
            {
                result.Data = new { Success = await ControlPanelService.Instance.AddUserRole(userID, roleID) };
            }
            catch (Exception ex)
            {
                result.Data = new { Success = false, Errors = ex.InnerException.Message };
            }

            return result;
        }

        [HttpPost]
        public async Task<JsonResult> RemoveUserRole(string userID, string roleID)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;
```

```
            try
            {
                result.Data = new { Success = await ControlPanelService.Instance.RemoveUserRole(userID, roleID) };
            }
            catch (Exception ex)
            {
                result.Data = new { Success = false, Errors = ex.InnerException.Message };
            }

            return result;
        }

        public ActionResult RoleDetails(string ID)
        {
            RoleDetailsViewModel model = new RoleDetailsViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Role Details",
                PageDescription = "Role Details"
            };

            model.Role = ControlPanelService.Instance.GetRoleByID(ID);

            return PartialView("_RoleDetails", model);
        }

        [HttpPost]
        public async Task<JsonResult> UpdateRole(UpdateRoleViewModel model)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            if (!ModelState.IsValid)
            {
                var Errors = new List<string>();

                foreach (ModelState modelState in ViewData.ModelState.Values)
                {
                    foreach (ModelError error in modelState.Errors)
                    {
                        Errors.Add(error.ErrorMessage.ToString());
                    }
                }

                result.Data = new { Success = false, Errors = Errors };

                return result;
            }
            else
            {
                try
                {
                    await ControlPanelService.Instance.UpdateRole(new IdentityRole() { Id = model.ID, Name =
model.Name });

                    result.Data = new { Success = true };
                }
                catch (Exception ex)
                {
                    result.Data = new { Success = false, Errors = ex.InnerException };
                }
```

```
        }

        return result;
    }

    [HttpPost]
    public async Task<JsonResult> DeleteRole(string ID)
    {
        JsonResult result = new JsonResult();
        result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

        if (string.IsNullOrEmpty(ID))
        {
            result.Data = new { Success = false, Errors = "Role can not be identified." };

            return result;
        }
        else
        {
            try
            {
                await ControlPanelService.Instance.DeleteRole(ID);

                result.Data = new { Success = true };
            }
            catch (Exception ex)
            {
                result.Data = new { Success = false, Errors = ex.Message };
            }
        }

        return result;
    }


    public ActionResult Quizzes(string search, int? page, int? items)
    {
        QuizListViewModel model = new QuizListViewModel();

        model.PageInfo = new PageInfo()
        {
            PageTitle = "Quizzes",
            PageDescription = "List of Quizzes."
        };

        model.searchTerm = search;
        model.pageNo = page ?? 1;
        model.pageSize = items ?? 10;

        var quizzesSearch = QuizzesService.Instance.GetQuizzes(model.searchTerm, model.pageNo,
model.pageSize);

        model.Quizzes = quizzesSearch.Quizzes;
        model.TotalCount = quizzesSearch.TotalCount;

        model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

        return PartialView("_Quizzes", model);
    }

    public ActionResult Attempts(string search, int? page, int? items)
    {
```

```
        QuizAttemptsListViewModel model = new QuizAttemptsListViewModel();

        model.searchTerm = search;
        model.pageNo = page ?? 1;
        model.pageSize = items ?? 10;

        var resultsSearch = StudentQuizzesService.Instance.GetQuizAttempts(model.searchTerm, model.pageNo,
model.pageSize);

        model.StudentQuizzes = resultsSearch.StudentQuizzes;
        model.TotalCount = resultsSearch.TotalCount;

        model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

        return PartialView("_Attempts", model);
    }
  }
}
```

HomeController.cs

```
using QuizbeePlus.Services;
using QuizbeePlus.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace QuizbeePlus.Controllers
{
    [AllowAnonymous]
    public class HomeController : BaseController
    {
        public ActionResult Index(string search, int? page, int? items)
        {
            HomeViewModel model = new HomeViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "изучение русского языка",
                PageDescription = "Quizbee helps you to create scalable and dynamic
quizzes with any number of questions and related options. Creating and attempting Quizzes
have never been this easy. Try it now!"
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 9;

            var quizzesSearch =
QuizzesService.Instance.GetQuizzesForHomePage(model.searchTerm, model.pageNo,
model.pageSize);

            model.Quizzes = quizzesSearch.Quizzes;
            model.TotalCount = quizzesSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
        }
        public ActionResult IndexUser(string search, int? page, int? items)
        {
            HomeViewModel model = new HomeViewModel();
```

```
            model.PageInfo = new PageInfo()
            {
                PageTitle = "изучение русского языка",
                PageDescription = "Quizbee helps you to create scalable and dynamic
quizzes with any number of questions and related options. Creating and attempting Quizzes
have never been this easy. Try it now!"
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 9;

            var quizzesSearch =
QuizzesService.Instance.GetQuizzesForHomePage(model.searchTerm, model.pageNo,
model.pageSize);

            model.Quizzes = quizzesSearch.Quizzes;
            model.TotalCount = quizzesSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
        }
        public ActionResult Quiz(string search, int? page, int? items)
        {
            HomeViewModel model = new HomeViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "изучение русского языка",
                PageDescription = "Quizbee helps you to create scalable and dynamic
quizzes with any number of questions and related options. Creating and attempting Quizzes
have never been this easy. Try it now!"
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 9;

            var quizzesSearch =
QuizzesService.Instance.GetQuizzesForHomePage(model.searchTerm, model.pageNo,
model.pageSize);

            model.Quizzes = quizzesSearch.Quizzes;
            model.TotalCount = quizzesSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
        }
    }


}
```

ManageController.cs

```
using System;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
```

```
using System.Web.Mvc;
using System.Web.UI;
using Microsoft.AspNet.Identity;
using Microsoft.AspNet.Identity.Owin;
using Microsoft.Owin.Security;
using QuizbeePlus.Data;
using QuizbeePlus.ViewModels;
using System.Collections.Generic;
using System.Net;

namespace QuizbeePlus.Controllers
{
    [Authorize]
    public class ManageController : Controller
    {
        private QuizbeeSignInManager _signInManager;
        private QuizbeeUserManager _userManager;

        public ManageController()
        {
        }

        public ManageController(QuizbeeUserManager userManager, QuizbeeSignInManager signInManager)
        {
            UserManager = userManager;
            SignInManager = signInManager;
        }

        public QuizbeeSignInManager SignInManager
        {
            get
            {
                return _signInManager ?? HttpContext.GetOwinContext().Get<QuizbeeSignInManager>();
            }
            private set
            {
                _signInManager = value;
            }
        }

        public QuizbeeUserManager UserManager
        {
            get
            {
                return _userManager ?? HttpContext.GetOwinContext().GetUserManager<QuizbeeUserManager>();
            }
            private set
            {
                _userManager = value;
            }
        }

        //
        // GET: /Manage/Index
        public async Task<ActionResult> Index(ManageMessageId? message)
        {
            ProfileViewModel pModel = new ProfileViewModel();

            pModel.PageInfo = new PageInfo()
            {
                PageTitle = "Profile",
                PageDescription = "My Profile."
```

```
            };

            pModel.User = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            pModel.UserName = pModel.User.UserName;
            pModel.Email = pModel.User.Email;

            return View(pModel);
        }

        [OutputCache(Duration = 3600, VaryByCustom = "User", Location = OutputCacheLocation.Client, NoStore =
true)]
        public ActionResult MyPhoto()
        {
            if (User.Identity.IsAuthenticated)
            {
                var user = UserManager.FindById(User.Identity.GetUserId());

                var imagePath = string.Empty;

                if (string.IsNullOrEmpty(user.Photo))
                {
                    imagePath = Path.Combine(Server.MapPath("~/Content/images/"), "user-default-avatar.png");
                }
                else imagePath = Path.Combine(Server.MapPath("~/Content/images/"), user.Photo);

                return new FileStreamResult(new FileStream(imagePath, FileMode.Open), "image/jpeg");
            }
            else return null;
        }

        [OutputCache(Duration = 3600, VaryByCustom = "User", Location = OutputCacheLocation.Client, NoStore =
true)]
        public ActionResult UserPhoto(string userID)
        {
            var user = UserManager.FindById(userID);

            var imagePath = string.Empty;

            if (string.IsNullOrEmpty(user.Photo))
            {
                imagePath = Path.Combine(Server.MapPath("~/Content/images/"), "user-default-avatar.png");
            }
            else imagePath = Path.Combine(Server.MapPath("~/Content/images/"), user.Photo);

            return new FileStreamResult(new FileStream(imagePath, FileMode.Open), "image/jpeg");
        }

        [HttpPost]
        public async Task<JsonResult> UpdateInfo(ProfileViewModel pModel)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            if (!ModelState.IsValid)
            {
                var Errors = new List<string>();

                foreach (ModelState modelState in ViewData.ModelState.Values)
                {
                    foreach (ModelError error in modelState.Errors)
                    {
                        Errors.Add(error.ErrorMessage.ToString());
```

```
            }
        }

        result.Data = new { Success = false, Errors = Errors };

        return result;
    }

    var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());

    if (!string.IsNullOrEmpty(pModel.Photo))
    {
        user.Photo = pModel.Photo;
    }

    user.UserName = pModel.UserName;
    user.Email = pModel.Email;

    var updateResult = await UserManager.UpdateAsync(user);

    if (updateResult.Succeeded)
    {
        result.Data = new { Success = true };
    }
    else
    {
        result.Data = new { Success = false, Errors = updateResult.Errors };
    }

    return result;
}

[HttpPost]
public async Task<JsonResult> UpdateUserInfo(string userID, ProfileViewModel pModel)
{
    JsonResult result = new JsonResult();
    result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

    if (!ModelState.IsValid)
    {
        var Errors = new List<string>();

        foreach (ModelState modelState in ViewData.ModelState.Values)
        {
            foreach (ModelError error in modelState.Errors)
            {
                Errors.Add(error.ErrorMessage.ToString());
            }
        }

        result.Data = new { Success = false, Errors = Errors };

        return result;
    }

    var user = await UserManager.FindByIdAsync(userID);

    if (!string.IsNullOrEmpty(pModel.Photo))
    {
        user.Photo = pModel.Photo;
    }
```

```
            user.UserName = pModel.UserName;
            user.Email = pModel.Email;

            var updateResult = await UserManager.UpdateAsync(user);

            if (updateResult.Succeeded)
            {
                result.Data = new { Success = true };
            }
            else
            {
                result.Data = new { Success = false, Errors = updateResult.Errors };
            }

            return result;
        }

        [HttpPost]
        public async Task<JsonResult> DeleteUser(string userID)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            if (userID == null)
            {
                result.Data = new { Success = false, Errors = new HttpStatusCodeResult(HttpStatusCode.BadRequest) };

                return result;
            }

            var user = await UserManager.FindByIdAsync(userID);
            var logins = user.Logins;
            var rolesForUser = await UserManager.GetRolesAsync(userID);

            foreach (var login in logins.ToList())
            {
                await UserManager.RemoveLoginAsync(login.UserId, new UserLoginInfo(login.LoginProvider,
login.ProviderKey));
            }

            if (rolesForUser.Count() > 0)
            {
                foreach (var item in rolesForUser.ToList())
                {
                    await UserManager.RemoveFromRoleAsync(user.Id, item);
                }
            }

            var deleteResult = await UserManager.DeleteAsync(user);

            if (deleteResult.Succeeded)
            {
                result.Data = new { Success = true };
            }
            else
            {
                result.Data = new { Success = false, Errors = deleteResult.Errors };
            }

            return result;
        }
```

```
        [HttpPost]
        public async Task<JsonResult> UpdatePassword(ChangePasswordViewModel model)
        {
            JsonResult result = new JsonResult();
            result.JsonRequestBehavior = JsonRequestBehavior.AllowGet;

            if (!ModelState.IsValid)
            {
                var Errors = new List<string>();

                foreach (ModelState modelState in ViewData.ModelState.Values)
                {
                    foreach (ModelError error in modelState.Errors)
                    {
                        Errors.Add(error.ErrorMessage.ToString());
                    }
                }

                result.Data = new { Success = false, Errors = Errors };

                return result;
            }

            var updationResult = await UserManager.ChangePasswordAsync(User.Identity.GetUserId(),
model.OldPassword, model.NewPassword);
            if (updationResult.Succeeded)
            {
                var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                }

                result.Data = new { Success = true };
            }
            else
            {
                result.Data = new { Success = false, Errors = updationResult.Errors };
            }

            return result;
        }

        //
        // POST: /Manage/RemoveLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> RemoveLogin(string loginProvider, string providerKey)
        {
            ManageMessageId? message;
            var result = await UserManager.RemoveLoginAsync(User.Identity.GetUserId(), new
UserLoginInfo(loginProvider, providerKey));
            if (result.Succeeded)
            {
                var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                }
                message = ManageMessageId.RemoveLoginSuccess;
            }
            else
```

```
            {
                message = ManageMessageId.Error;
            }
            return RedirectToAction("ManageLogins", new { Message = message });
        }

        //
        // GET: /Manage/AddPhoneNumber
        public ActionResult AddPhoneNumber()
        {
            return View();
        }

        //
        // POST: /Manage/AddPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> AddPhoneNumber(AddPhoneNumberViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            // Generate the token and send it
            var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
model.Number);
            if (UserManager.SmsService != null)
            {
                var message = new IdentityMessage
                {
                    Destination = model.Number,
                    Body = "Your security code is: " + code
                };
                await UserManager.SmsService.SendAsync(message);
            }
            return RedirectToAction("VerifyPhoneNumber", new { PhoneNumber = model.Number });
        }

        //
        // POST: /Manage/EnableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> EnableTwoFactorAuthentication()
        {
            await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), true);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }

        //
        // POST: /Manage/DisableTwoFactorAuthentication
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> DisableTwoFactorAuthentication()
        {
            await UserManager.SetTwoFactorEnabledAsync(User.Identity.GetUserId(), false);
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
```

```
            {
                await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
            }
            return RedirectToAction("Index", "Manage");
        }

        //
        // GET: /Manage/VerifyPhoneNumber
        public async Task<ActionResult> VerifyPhoneNumber(string phoneNumber)
        {
            var code = await UserManager.GenerateChangePhoneNumberTokenAsync(User.Identity.GetUserId(),
phoneNumber);
            // Send an SMS through the SMS provider to verify the phone number
            return phoneNumber == null ? View("Error") : View(new VerifyPhoneNumberViewModel { PhoneNumber
= phoneNumber });
        }

        //
        // POST: /Manage/VerifyPhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> VerifyPhoneNumber(VerifyPhoneNumberViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var result = await UserManager.ChangePhoneNumberAsync(User.Identity.GetUserId(),
model.PhoneNumber, model.Code);
            if (result.Succeeded)
            {
                var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message = ManageMessageId.AddPhoneSuccess });
            }
            // If we got this far, something failed, redisplay form
            ModelState.AddModelError("", "Failed to verify phone");
            return View(model);
        }

        //
        // POST: /Manage/RemovePhoneNumber
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> RemovePhoneNumber()
        {
            var result = await UserManager.SetPhoneNumberAsync(User.Identity.GetUserId(), null);
            if (!result.Succeeded)
            {
                return RedirectToAction("Index", new { Message = ManageMessageId.Error });
            }
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user != null)
            {
                await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
            }
            return RedirectToAction("Index", new { Message = ManageMessageId.RemovePhoneSuccess });
        }
```

```
        //
        // GET: /Manage/ChangePassword
        public ActionResult ChangePassword()
        {
            return View();
        }

        //
        // POST: /Manage/ChangePassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> ChangePassword(ChangePasswordViewModel model)
        {
            if (!ModelState.IsValid)
            {
                return View(model);
            }
            var result = await UserManager.ChangePasswordAsync(User.Identity.GetUserId(), model.OldPassword,
model.NewPassword);
            if (result.Succeeded)
            {
                var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                if (user != null)
                {
                    await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                }
                return RedirectToAction("Index", new { Message = ManageMessageId.ChangePasswordSuccess });
            }
            AddErrors(result);
            return View(model);
        }

        //
        // GET: /Manage/SetPassword
        public ActionResult SetPassword()
        {
            return View();
        }

        //
        // POST: /Manage/SetPassword
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<ActionResult> SetPassword(SetPasswordViewModel model)
        {
            if (ModelState.IsValid)
            {
                var result = await UserManager.AddPasswordAsync(User.Identity.GetUserId(), model.NewPassword);
                if (result.Succeeded)
                {
                    var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
                    if (user != null)
                    {
                        await SignInManager.SignInAsync(user, isPersistent: false, rememberBrowser: false);
                    }
                    return RedirectToAction("Index", new { Message = ManageMessageId.SetPasswordSuccess });
                }
                AddErrors(result);
            }

            // If we got this far, something failed, redisplay form
            return View(model);
```

```
        }

        //
        // GET: /Manage/ManageLogins
        public async Task<ActionResult> ManageLogins(ManageMessageId? message)
        {
            ViewBag.StatusMessage =
                message == ManageMessageId.RemoveLoginSuccess ? "The external login was removed."
                : message == ManageMessageId.Error ? "An error has occurred."
                : "";
            var user = await UserManager.FindByIdAsync(User.Identity.GetUserId());
            if (user == null)
            {
                return View("Error");
            }
            var userLogins = await UserManager.GetLoginsAsync(User.Identity.GetUserId());
            var otherLogins = AuthenticationManager.GetExternalAuthenticationTypes().Where(auth =>
userLogins.All(ul => auth.AuthenticationType != ul.LoginProvider)).ToList();
            ViewBag.ShowRemoveButton = user.PasswordHash != null || userLogins.Count > 1;
            return View(new ManageLoginsViewModel
            {
                CurrentLogins = userLogins,
                OtherLogins = otherLogins
            });
        }

        //
        // POST: /Manage/LinkLogin
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult LinkLogin(string provider)
        {
            // Request a redirect to the external login provider to link a login for the current user
            return new AccountController.ChallengeResult(provider, Url.Action("LinkLoginCallback", "Manage"),
User.Identity.GetUserId());
        }

        //
        // GET: /Manage/LinkLoginCallback
        public async Task<ActionResult> LinkLoginCallback()
        {
            var loginInfo = await AuthenticationManager.GetExternalLoginInfoAsync(XsrfKey,
User.Identity.GetUserId());
            if (loginInfo == null)
            {
                return RedirectToAction("ManageLogins", new { Message = ManageMessageId.Error });
            }
            var result = await UserManager.AddLoginAsync(User.Identity.GetUserId(), loginInfo.Login);
            return result.Succeeded ? RedirectToAction("ManageLogins") : RedirectToAction("ManageLogins", new {
Message = ManageMessageId.Error });
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing && _userManager != null)
            {
                _userManager.Dispose();
                _userManager = null;
            }

            base.Dispose(disposing);
        }
```

```
#region Helpers
        // Used for XSRF protection when adding external logins
        private const string XsrfKey = "XsrfId";

        private IAuthenticationManager AuthenticationManager
        {
            get
            {
                return HttpContext.GetOwinContext().Authentication;
            }
        }

        private void AddErrors(IdentityResult result)
        {
            foreach (var error in result.Errors)
            {
                ModelState.AddModelError("", error);
            }
        }

        private bool HasPassword()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PasswordHash != null;
            }
            return false;
        }

        private bool HasPhoneNumber()
        {
            var user = UserManager.FindById(User.Identity.GetUserId());
            if (user != null)
            {
                return user.PhoneNumber != null;
            }
            return false;
        }

        public enum ManageMessageId
        {
            AddPhoneSuccess,
            ChangePasswordSuccess,
            SetTwoFactorSuccess,
            SetPasswordSuccess,
            RemoveLoginSuccess,
            RemovePhoneSuccess,
            Error
        }

#endregion
    }
}
```

QuestionController.cs

```
using Microsoft.AspNet.Identity;
using QuizbeePlus.Entities;
using QuizbeePlus.Services;
```

```
using QuizbeePlus.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;

namespace QuizbeePlus.Controllers
{
   public class QuestionController : BaseController
   {
      public ActionResult Index(int quizID, string search, int? page, int? items)
      {
         QuestionListViewModel model = new QuestionListViewModel();

         model.PageInfo = new PageInfo()
         {
            PageTitle = "Questions",
            PageDescription = "List of Questions for Selected Quiz."
         };

         model.searchTerm = search;
         model.pageNo = page ?? 1;
         model.pageSize = items ?? 10;

         var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(quizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), quizID);

         if (quiz != null)
         {
            model.QuizID = quizID;

            var result = QuestionsService.Instance.GetQuizQuestions(quiz.ID, model.searchTerm, model.pageNo,
model.pageSize);

            model.Questions = result.Questions;
            model.TotalCount = result.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
         }
         else return HttpNotFound();
      }

      [HttpGet]
      public ActionResult NewQuestion(int quizID)
      {
         NewQuestionViewModel model = new NewQuestionViewModel();

         var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(quizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), quizID);

         if (quiz == null) return HttpNotFound();

         model.PageInfo = new PageInfo()
         {
            PageTitle = "Add New Question",
            PageDescription = "Add questions to selected quiz."
         };
```

```
            model.QuizID = quiz.ID;
            model.QuizType = quiz.QuizType;

            return View(model);
        }

        [HttpPost]
        public async Task<ActionResult> NewQuestion(FormCollection collection)
        {
            NewQuestionViewModel model = new NewQuestionViewModel();

            model.QuizID = GetQuizIDFromCollection(collection);

            var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(model.QuizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), model.QuizID);

            if (quiz == null) return HttpNotFound();

            if (quiz.QuizType == QuizType.Image)
            {
                model = GetNewImageQuestionViewModelFromFormCollection(model, collection);
            }
            else
            {
                model = GetNewTextQuestionViewModelFromFormCollection(model, collection);
            }

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Add New Question",
                PageDescription = "Add questions to selected quiz."
            };

            if (string.IsNullOrEmpty(model.Title) || model.CorrectOptions.Count == 0 || model.Options.Count == 0)
            {
                if (string.IsNullOrEmpty(model.Title))
                {
                    ModelState.AddModelError("Title", "Please enter question title.");
                }

                if (model.CorrectOptions.Count == 0)
                {
                    ModelState.AddModelError("CorrectOptions", "Please enter some correct options.");
                }

                if (model.Options.Count == 0)
                {
                    ModelState.AddModelError("Options", "Please enter some other options.");
                }

                model.QuizType = quiz.QuizType;

                return View(model);
            }

            var question = new Question();

            question.QuizID = quiz.ID;
            question.Title = model.Title;

            question.Options = new List<Option>();
            question.Options.AddRange(model.CorrectOptions);
```

```
            question.Options.AddRange(model.Options);

            question.ModifiedOn = DateTime.Now;

            if (await QuestionsService.Instance.SaveNewQuestion(question))
            {
                return RedirectToAction("NewQuestion", new { quizID = question.QuizID });
            }
            else
            {
                return new HttpStatusCodeResult(500);
            }
        }

        [HttpGet]
        public ActionResult EditQuestion(int quizID, int ID)
        {
            EditQuestionViewModel model = new EditQuestionViewModel();

            var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(quizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), quizID);

            if (quiz == null) return HttpNotFound();

            var question = QuestionsService.Instance.GetQuizQuestion(quizID, ID);

            if (question == null) return HttpNotFound();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Modify Question",
                PageDescription = "Modify selected question."
            };

            model.ID = question.ID;
            model.QuizID = question.QuizID;
            model.QuizType = quiz.QuizType;
            model.Title = question.Title;
            model.CorrectOptions = question.Options.Where(q => q.IsCorrect).ToList();
            model.Options = question.Options.Where(q => !q.IsCorrect).ToList();

            return View(model);
        }

        [HttpPost]
        public async Task<ActionResult> EditQuestion(FormCollection collection)
        {
            EditQuestionViewModel model = new EditQuestionViewModel();

            model.QuizID = GetQuizIDFromCollection(collection);

            var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(model.QuizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), model.QuizID);

            if (quiz == null) return HttpNotFound();

            if (quiz.QuizType == QuizType.Image)
            {
                model = GetEditImageQuestionViewModelFromFormCollection(model, collection);
            }
            else
            {
```

```
            model = GetEditTextQuestionViewModelFromFormCollection(model, collection);
        }

        var question = QuestionsService.Instance.GetQuizQuestion(model.QuizID, model.ID);

        if (question == null) return HttpNotFound();

        model.PageInfo = new PageInfo()
        {
            PageTitle = "Modify Question",
            PageDescription = "Modify selected question."
        };

        if (model == null || string.IsNullOrEmpty(model.Title) || model.CorrectOptions.Count == 0 ||
model.Options.Count == 0)
        {
            if (string.IsNullOrEmpty(model.Title))
            {
                ModelState.AddModelError("Title", "Please enter question title.");
            }

            if (model.CorrectOptions.Count == 0)
            {
                ModelState.AddModelError("CorrectOptions", "Please enter some correct options.");
            }

            if (model.Options.Count == 0)
            {
                ModelState.AddModelError("Options", "Please enter some other options.");
            }

            model.QuizType = quiz.QuizType;

            return View(model);
        }

        question.QuizID = model.QuizID;
        question.Title = model.Title;

        question.Options = new List<Option>();
        question.Options.AddRange(model.CorrectOptions);
        question.Options.AddRange(model.Options);

        question.ModifiedOn = DateTime.Now;

        if (await QuestionsService.Instance.UpdateQuestion(question))
        {
            return RedirectToAction("Index", new { quizID = question.QuizID });
        }
        else
        {
            return new HttpStatusCodeResult(500);
        }
    }

    [HttpPost]
    public async Task<ActionResult> DeleteQuestion(FormCollection collection)
    {
        EditQuestionViewModel model = new EditQuestionViewModel();

        model.QuizID = GetQuizIDFromCollection(collection);
```

```
        var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(model.QuizID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), model.QuizID);

        if (quiz == null) return HttpNotFound();

        if (quiz.QuizType == QuizType.Image)
        {
           model = GetEditImageQuestionViewModelFromFormCollection(model, collection);
        }
        else
        {
           model = GetEditTextQuestionViewModelFromFormCollection(model, collection);
        }

        var question = QuestionsService.Instance.GetQuizQuestion(model.QuizID, model.ID);

        if (question == null) return HttpNotFound();

        if (await QuestionsService.Instance.DeleteQuestion(question))
        {
           return RedirectToAction("Index", new { quizID = question.QuizID });
        }
        else
        {
           return new HttpStatusCodeResult(500);
        }
     }

     private NewQuestionViewModel
GetNewImageQuestionViewModelFromFormCollection(NewQuestionViewModel model, FormCollection
collection)
     {
        model.Options = new List<Option>();
        model.CorrectOptions = new List<Option>();

        if (collection.AllKeys.Count() > 0)
        {
           foreach (string key in collection)
           {
              if (key == "Title")
              {
                 model.Title = collection[key];
              }
              else if (key.Contains("correctOption")) //this must be Correct Option
              {
                 if (!string.IsNullOrEmpty(collection[key]))
                 {
                    try
                    {
                       var correctOption = new Option();
                       correctOption.Image = ImagesService.Instance.GetImage(int.Parse(collection[key]));
                       correctOption.IsCorrect = true;
                       correctOption.ModifiedOn = DateTime.Now;

                       model.CorrectOptions.Add(correctOption);
                    }
                    catch (Exception ex)
                    {
                       //ignore this option
                    }
                 }
              }
```

```
            else if (key.Contains("option")) //this must be Option
            {
               if (!string.IsNullOrEmpty(collection[key]))
               {
                  try
                  {
                     var option = new Option();
                     option.Image = ImagesService.Instance.GetImage(int.Parse(collection[key]));
                     option.IsCorrect = false;
                     option.ModifiedOn = DateTime.Now;

                     model.Options.Add(option);
                  }
                  catch (Exception ex)
                  {
                     //ignore this option
                  }
               }
            }
         }
      }

      return model;
   }

   private NewQuestionViewModel
GetNewTextQuestionViewModelFromFormCollection(NewQuestionViewModel model, FormCollection collection)
   {
      model.Options = new List<Option>();
      model.CorrectOptions = new List<Option>();

      if (collection.AllKeys.Count() > 0)
      {
         foreach (string key in collection)
         {
            if (key == "Title")
            {
               model.Title = collection[key];
            }
            else if (key.Contains("correctOption")) //this must be Correct Option
            {
               if (!string.IsNullOrEmpty(collection[key]))
               {
                  var correctOption = new Option();
                  correctOption.Answer = collection[key];
                  correctOption.IsCorrect = true;
                  correctOption.ModifiedOn = DateTime.Now;

                  model.CorrectOptions.Add(correctOption);
               }
            }
            else if (key.Contains("option")) //this must be Option
            {
               if (!string.IsNullOrEmpty(collection[key]))
               {
                  var option = new Option();
                  option.Answer = collection[key];
                  option.IsCorrect = false;
                  option.ModifiedOn = DateTime.Now;

                  model.Options.Add(option);
               }
```

```
            }
        }
    }

    return model;
}

private EditQuestionViewModel
GetEditImageQuestionViewModelFromFormCollection(EditQuestionViewModel model, FormCollection
collection)
{
    model.Options = new List<Option>();
    model.CorrectOptions = new List<Option>();

    if (collection.AllKeys.Count() > 0)
    {
        foreach (string key in collection)
        {
            if (key == "ID")
            {
                model.ID = int.Parse(collection[key]);
            }
            else if (key == "Title")
            {
                model.Title = collection[key];
            }
            else if (key.Contains("correctOption")) //this must be Correct Option
            {
                if (!string.IsNullOrEmpty(collection[key]))
                {
                    try
                    {
                        var correctOption = new Option();
                        correctOption.Image = ImagesService.Instance.GetImage(int.Parse(collection[key]));
                        correctOption.IsCorrect = true;
                        correctOption.ModifiedOn = DateTime.Now;

                        model.CorrectOptions.Add(correctOption);
                    }
                    catch
                    {
                        //ignore this option
                    }
                }
            }
            else if (key.Contains("option")) //this must be Option
            {
                if (!string.IsNullOrEmpty(collection[key]))
                {
                    try
                    {
                        var option = new Option();
                        option.Image = ImagesService.Instance.GetImage(int.Parse(collection[key]));
                        option.IsCorrect = false;
                        option.ModifiedOn = DateTime.Now;

                        model.Options.Add(option);
                    }
                    catch
                    {
                        //ignore this option
                    }
```

```
                }
            }
        }
    }

    return model;
}

private EditQuestionViewModel
GetEditTextQuestionViewModelFromFormCollection(EditQuestionViewModel model, FormCollection collection)
{
    model.Options = new List<Option>();
    model.CorrectOptions = new List<Option>();

    if (collection.AllKeys.Count() > 0)
    {
        foreach (string key in collection)
        {
            if (key == "ID")
            {
                model.ID = int.Parse(collection[key]);
            }
            else if (key == "Title")
            {
                model.Title = collection[key];
            }
            else if (key.Contains("correctOption")) //this must be Correct Option
            {
                if (!string.IsNullOrEmpty(collection[key]))
                {
                    var correctOption = new Option();
                    correctOption.Answer = collection[key];
                    correctOption.IsCorrect = true;
                    correctOption.ModifiedOn = DateTime.Now;

                    model.CorrectOptions.Add(correctOption);
                }
            }
            else if (key.Contains("option")) //this must be Option
            {
                if (!string.IsNullOrEmpty(collection[key]))
                {
                    var option = new Option();
                    option.Answer = collection[key];
                    option.IsCorrect = false;
                    option.ModifiedOn = DateTime.Now;

                    model.Options.Add(option);
                }
            }
        }
    }

    return model;
}

private int GetQuizIDFromCollection(FormCollection collection)
{
    if (collection.AllKeys.Count() > 0)
    {
        foreach (string key in collection)
        {
```

```
            if (key == "QuizID")
            {
                return int.Parse(collection[key]);
            }
        }
    }

    return 0;
  }
 }
}
```

QuizController.cs

```
using Microsoft.AspNet.Identity;
using QuizbeePlus.Entities;
using QuizbeePlus.Services;
using QuizbeePlus.ViewModels;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;

namespace QuizbeePlus.Controllers
{
    public class QuizController : BaseController
    {
        // GET: Quiz
        public ActionResult Index(string search, int? page, int? items)
        {
            QuizListViewModel model = new QuizListViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Quizzes",
                PageDescription = "List of Quizzes."
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 10;

            var quizzesSearch = UserHasRights() ?
QuizzesService.Instance.GetQuizzesForAdmin(model.searchTerm, model.pageNo,
model.pageSize) : QuizzesService.Instance.GetUserQuizzes(User.Identity.GetUserId(),
model.searchTerm, model.pageNo, model.pageSize);

            model.Quizzes = quizzesSearch.Quizzes;
            model.TotalCount = quizzesSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
        }

        [HttpGet]
        public ActionResult NewQuiz()
        {
            NewQuizViewModel model = new NewQuizViewModel();

            model.PageInfo = new PageInfo()
            {
```

```
            PageTitle = "Create New Quiz",
            PageDescription = "Create new quiz."
        };

        model.EnableQuizTimer = true; //set Default Quiz Timer to true

        model.QuizTypes = new List<QuizType>() { QuizType.Text, QuizType.Image };
        model.SelectedQuizType = QuizType.Text;

        return View(model);
    }

    [HttpPost]
    public async Task<ActionResult> NewQuiz(NewQuizViewModel model)
    {
        //check if Model is valid
        if (!ModelState.IsValid)
        {
            model.PageInfo = new PageInfo()
            {
                PageTitle = "Create New Quiz",
                PageDescription = "Create new quiz."
            };

            model.QuizTypes = new List<QuizType>() { QuizType.Text, QuizType.Image };

            return View(model);
        }

        //populate new Quiz from Model
        var quiz = new Quiz();
        quiz.OwnerID = User.Identity.GetUserId();
        quiz.Name = model.Name;
        quiz.Description = model.Description;

        quiz.QuizType = model.SelectedQuizType;

        quiz.EnableQuizTimer = model.EnableQuizTimer;

        if (quiz.EnableQuizTimer)
        {
            quiz.TimeDuration = new TimeSpan(model.Hours, model.Minutes, 0);
            quiz.EnableQuestionTimer = model.EnableQuestionTimer;
        }
        else
        {
            quiz.TimeDuration = new TimeSpan(0, 0, 0);
            quiz.EnableQuestionTimer = false;
        }

        quiz.ModifiedOn = DateTime.Now;

        if (await QuizzesService.Instance.SaveNewQuiz(quiz))
        {
            //now redirect to New Question screen for this new Quiz
            return RedirectToAction("NewQuestion", "Question", new { quizID = quiz.ID
});
        }
        else
        {
            return new HttpStatusCodeResult(500);
        }
    }
```

```
        [HttpGet]
        public ActionResult EditQuiz(int ID)
        {
            //get quiz by supplied ID and check if this User is its owner
            var quiz = UserHasRights() ? QuizzesService.Instance.GetQuizForAdmin(ID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), ID);

            if (quiz == null)
                return HttpNotFound();

            EditQuizViewModel model = new EditQuizViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "Modify Quiz",
                PageDescription = "Modify this quiz."
            };

            model.ID = quiz.ID;
            model.Name = quiz.Name;
            model.Description = quiz.Description;

            model.QuizTypes = new List<QuizType>() { QuizType.Text, QuizType.Image };
            model.SelectedQuizType = quiz.QuizType;

            model.EnableQuizTimer = quiz.EnableQuizTimer;

            if (model.EnableQuizTimer)
            {
                model.Hours = quiz.TimeDuration.Hours;
                model.Minutes = quiz.TimeDuration.Minutes;

                model.EnableQuestionTimer = quiz.EnableQuestionTimer;
            }
            else
            {
                model.Hours = 0;
                model.Minutes = 0;

                model.EnableQuestionTimer = false;
            }

            model.NoOfQuestions = quiz.Questions.Count;

            return View(model);
        }

        [HttpPost]
        public async Task<ActionResult> EditQuiz(EditQuizViewModel model)
        {
            //get quiz by supplied ID and check if this User is its owner
            var quiz = UserHasRights() ?
QuizzesService.Instance.GetQuizForAdmin(model.ID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), model.ID);

            if (quiz == null)
                return HttpNotFound();

            //check if Model is valid
            if (!ModelState.IsValid)
            {
                model.PageInfo = new PageInfo()
                {
                    PageTitle = "Modify Quiz",
```

```
                    PageDescription = "Modify this quiz."
                };

                return View(model);
            }

            //populate quiz from Model
            //quiz.OwnerID = User.Identity.GetUserId(); //<- Do not change owner of Quiz
on update. Reason: if the Super Admin updated the quiz, it will no longer show in
original owner
            quiz.Name = model.Name;
            quiz.Description = model.Description;

            //quiz type should not be changed because this will affect the display of
questions
            //quiz.QuizType = model.SelectedQuizType;

            quiz.EnableQuizTimer = model.EnableQuizTimer;

            if (quiz.EnableQuizTimer)
            {
                quiz.TimeDuration = new TimeSpan(model.Hours, model.Minutes, 0);
                quiz.EnableQuestionTimer = model.EnableQuestionTimer;
            }
            else
            {
                quiz.TimeDuration = new TimeSpan(0, 0, 0);
                quiz.EnableQuestionTimer = false;
            }
            quiz.ModifiedOn = DateTime.Now;

            if (await QuizzesService.Instance.UpdateQuiz(quiz))
            {
                //now redirect to Quiz listing Page
                return RedirectToAction("Index");
            }
            else
            {
                return new HttpStatusCodeResult(500);
            }
        }

        [HttpPost]
        public async Task<ActionResult> DeleteQuiz(EditQuizViewModel model)
        {
            //get quiz by supplied ID and check if this User is its owner
            var quiz = UserHasRights() ?
QuizzesService.Instance.GetQuizForAdmin(model.ID) :
QuizzesService.Instance.GetUserQuiz(User.Identity.GetUserId(), model.ID);

            if (quiz == null)
                return HttpNotFound();

            quiz.ModifiedOn = DateTime.Now;

            if (await QuizzesService.Instance.DeleteQuiz(quiz))
            {
                //now redirect to Quiz listing Page
                return RedirectToAction("Index");
            }
            else
            {
                return new HttpStatusCodeResult(500);
            }
```

```
        }

        public ActionResult QuizView(string search, int? page, int? items)
        {
            HomeViewModel model = new HomeViewModel();

            model.PageInfo = new PageInfo()
            {
                PageTitle = "изучение русского языка",
                PageDescription = "Quizbee helps you to create scalable and dynamic
quizzes with any number of questions and related options. Creating and attempting Quizzes
have never been this easy. Try it now!"
            };

            model.searchTerm = search;
            model.pageNo = page ?? 1;
            model.pageSize = items ?? 9;

            var quizzesSearch =
QuizzesService.Instance.GetQuizzesForHomePage(model.searchTerm, model.pageNo,
model.pageSize);

            model.Quizzes = quizzesSearch.Quizzes;
            model.TotalCount = quizzesSearch.TotalCount;

            model.Pager = new Pager(model.TotalCount, model.pageNo, model.pageSize);

            return View(model);
        }
    }
}
```

SharedController.cs

```
using Microsoft.AspNet.Identity;
using QuizbeePlus.Commons;
using QuizbeePlus.Entities;
using QuizbeePlus.Services;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading.Tasks;
using System.Web;
using System.Web.Mvc;
using System.Web.UI;

namespace QuizbeePlus.Controllers
{
    public class SharedController : BaseController
    {
        // GET: Shared
        public JsonResult UploadImage()
        {
            JsonResult result = new JsonResult();

            try
            {
                var file = Request.Files[0];

                var fileName = Guid.NewGuid() + Path.GetExtension(file.FileName);
                var path = Path.Combine(Server.MapPath("~/content/images/"), fileName);
```

```
                file.SaveAs(path);

                var newImage = new Image() { Name = fileName };

                if(ImagesService.Instance.SaveNewImage(newImage))
                {
                    result.Data = new { Success = true, Image= fileName, File =
newImage.ID, ImageURL = string.Format("{0}{1}", Variables.ImageFolderPath, fileName) };
                }
                else
                {
                    result.Data = new { success = false, Message = new
HttpStatusCodeResult(500) };
                }
            }
            catch (Exception ex)
            {
                result.Data = new { success = false, Message = ex.Message };
            }

            return result;
        }
    }
}
```