МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2022 г.

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ ДЛЯ ОБУЧЕНИЯ ДЕТЕЙ ПИСЬМУ НА ОСНОВЕ ЗРИТЕЛЬНЫХ ОБРАЗОВ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2022.203 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____Е.С. Ярош
«___»_____2022 г.

Автор работы,
студент группы КЭ-405
_____К.Е. Бакунина
«___»_____2022 г.

Нормоконтролёр,
к.пед.н., доцент каф. ЭВМ
_____ М.А. Алтухова
«___»_____2022 г.

Челябинск-2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2022 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Бакуниной Ксении Егоровне
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Разработка веб-приложения для обучения детей письму на основе зрительных образов» утверждена приказом по университету от 12 декабря 2021 г. №308/141.

2. **Срок сдачи студентом законченной работы**: 1 июня 2022 г.

3. **Исходные данные к работе:**

– система должна быть реализована на языке высокого уровня C#;

– средства реализации: среда разработки Visual Studio, БД SQL Server Management Studio;

– время проверки и входа в систему не должно превышать 10 секунд;

– вход в систему должен быть организован на основе предварительной регистрации.

4. **Перечень подлежащих разработке вопросов:**

– рассмотрение существующих методов обучения детей письму;

– анализ современных программных средств для обучения детей письму;

– разработка собственного программного продукта для обучения детей письму;

– оценка работоспособности разработанного программного продукта в различных режимах.

5. **Дата выдачи задания:** 1 декабря 2021 г.

Руководитель работы _____/ *Е.С. Ярош* /

Студент _____/ *К.Е. Бакунина* /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	10.03.2022	
Разработка модели, проектирование		
Разработка структуры входных и выходных данных	15.03.2022	
Выбор критериев эффективности и качества системы	20.03.2022	
Согласование и утверждение эскизного проекта	30.03.2022	
Реализация системы		
Разработка структурной схемы системы	05.04.2022	
Разработка структуры базы данных	10.04.2022	
Разработка программного обеспечения	15.04.2022	
Разработка интерфейса	20.04.2022	
Согласование и утверждение технического проекта	25.04.2022	
Тестирование, отладка, эксперименты		
Предварительные испытания	01.05.2022	
Опытная эксплуатация	06.05.2022	

Этап	Срок сдачи	Подпись руководителя
Компоновка текста работы и сдача на нормоконтроль	26.05.2022	
Подготовка презентации и доклада	01.06.2022	

Руководитель работы _____/ *Е.С. Ярош* /

Студент  _____/ *К.Е. Бакунина* /

АННОТАЦИЯ

К.Е. Бакунина. Разработка веб-приложения для обучения детей письму на основе зрительных образов. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШЭКН; 2022, 102 с., 37 ил., библиогр. список – 24 наим.

В рамках выпускной квалификационной работы выполнены проектирование и реализация программного обеспечения для обучения детей письму при помощи зрительных образов. Проведено исследование рынка обучающих программ с целью выявления и анализа аналогов. Рассмотрены преимущества и недостатки выбранных средств разработки. Доказана способность предлагаемой архитектуры к обеспечению успешного решения задач обучения письму.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во все времена образование играло ключевую роль в жизни человека. Образование является важным процессом, во время которого человек получает знания, приобщается к культуре и ценностям общества. Оно влияет на формирование мировоззрения человека, обеспечивает преемственность языка, традиций, тем самым формируя социально-культурные навыки [1].

Наиболее значимой из проблем образования является проблема освоения базовых навыков, таких как письмо и чтение.

Актуальность темы, связанной с образовательным процессом в области обучения письму, заключается в том, что письмо является базовым школьным навыком, без эффективного владения которым дальнейшее обучение затруднено или невозможно.

Также недостаточная сформированность данных навыков является серьезной социальной проблемой, поскольку может привести к функциональной неграмотности [2].

В настоящее время наблюдается значительное увеличение количества детей с нарушениями речи, испытывающих трудности в усвоении программы по родному языку [3]. Основную проблему при этом составляет овладение навыком письма.

Таким образом, проблема обучения письму детей младшего школьного возраста весьма значима и актуальна.

Целью представленной выпускной квалификационной работы является разработка программного комплекса, обеспечивающего повышение эффективности работы по обучению детей письму.

Для достижения поставленной цели необходимо решить следующие задачи:

1) рассмотреть существующие методики обучения детей письму;

2) провести анализ современных программных систем для организации обучения письму в контексте выбранной методики обучения;

3) спроектировать архитектуру собственного программного продукта и представить программную реализацию;

4) оценить работоспособность разработанного программного продукта.

Данная работа состоит из нескольких этапов:

1) обзор литературы – изучение и анализ программных систем по тематике обучения детей письму;

2) разработка модели, проектирование – разработка структуры системы;

3) реализация системы – разработка структурной схемы системы, базы данных, программной реализации и интерфейса системы;

4) тестирование, отладка, эксперименты – предварительные испытания.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Письмо – это продуктивный вид деятельности, при котором человек записывает речь для передачи другим в бумажном или электронном виде. Продуктом данного вида деятельности является речевое произведение или текст, предназначенный для прочтения.

Обучение письму – это в первую очередь выработка графического навыка. Как и всякий навык, он формируется в результате обучения – формирования умений и выполнения ряда упражнений на их основе.

Существуют два метода обучения письму [4]:

– аналитический;

– синтетический.

Аналитический метод – целостный метод, в котором обучение начинается с восприятия образов, слов и коротких предложений. Затем выделяются отдельные элементы. Этот метод был разработан в 1778 году Э.К. Транном.

Синтетический метод был известен под названием «буквенная метода» (буквы – слоги – слова). В 18 веке произошло усовершенствование этой методики в «Звуко-буквенную» методику обучения письму, разработанную Экельзамером. В ней отведена большпя роль произношения в обучении письму.

Как синтетический метод обучения письму, так и аналитический метод имеют свои положительные и отрицательные стороны.

Достоинства и недостатки аналитического метода:

– достоинство: обучение, основанное на восприятии зрительных образов;

– недостаток: отсутствие целостного образа написания букв.

Достоинства и недостатки синтетического метода:

– достоинство: воспроизведение звукового сопровождения букв при написании слова;

– недостаток: потеря ребенком концентрации при написании.

Аналитический метод рассматривает буквы как отдельные элементы начертания, что может вызвать затруднение у детей из-за отсутствия восприятия целостности начертания букв.

Синтетический метод опирается на обучение письму на основе звуков и звукоразложения, то есть отсутствии образа.

Исходя из представленной выше информации, по отдельности оба эти подхода являются неэффективными.

Кроме этого, традиционное обучение письму происходит без применения информационных технологий, которые значительно упрощают процесс обучения детей.

Для эффективного обучения необходимо совместить положительные стороны описанных ранее подходов и информационные ресурсы. Поэтому в разрабатываемой системе должны быть совмещены восприятие образов написания букв и звуковое сопровождение при написании слова. Также для улучшения графического восприятия слова система должна сопровождать написание слова иллюстрацией (картинкой) изучаемого слова.

1.1 Обзор аналогов

В настоящее время существует ряд программах продуктов, направленных на обучение детей письму [5]. Рассмотрим некоторые из них ниже.

«Letter School» – это приложение для изучения алфавита и тренировки рукописного ввода, предназначенное для школьников младшего возраста [6]. Приложение предлагает детям вводить строчные и заглавные буквы, а также цифры, используя указательный палец и формируя правильное графическое представление написания предмета изучения.

Перед непосредственным вводом приложение показывает детям, где они должны начинать и заканчивать каждую цифру или букву. Если обучающиеся

хотят перейти к следующему элементу обучения, они должны удовлетворительно написать текущий.

Демонстрация работы приложения в 3 вариантах оформления представлена на рисунке 1.1.

Рисунок 1.1 – Демонстрация работы приложения «Letter School»

Приложение привлекает детей интересными звуковыми эффектами и анимацией, которые побуждают их регулярно практиковаться в прописывании цифр и букв.

Также в приложении есть четыре увлекательные игры для каждой буквы и цифры, которые проверяют знания ребенка в написании элементов по памяти.

Приложение доступно для устройств на платформах Android и iOS.

«iTrace» – это приложение для детей, которое делает изучение чисел и букв веселым и интересным [7]. Данное приложение оказывает непосредственное влияние на то, как будут сформированы навыки письма ребенка, оно предоставляет возможность изучать не только буквы, но и цифры.

Родители могут выббрать один из трех различных стилей написания букв, а также – стандартное или курсивное написание. По мимо этого приложение обладает функцией переключения ориентации для левшей.

Демонстрация работы приложения представлена на рисунке 1.2.

Рисунок 1.2 – Демонстрация работы приложения «iTrace»

Для развития мышечной памяти ребенка приложение предлагает ему писать одну и ту же букву несколько раз. Также в качестве поощрения система предлагает по меньшей мере 10 уникальных призов за удовлетворительное написание элемента обучения.

Кроме того, родители могут следить за успехами детей, поскольку приложение хранит историю тренировок ребенка.

Приложение доступно для устройств на платформах MacOS, iOS.

«Мастер письма для детей» – приложение, предназначенное для обучения детей путем повторения графических образов цифр и букв с помощью написания пальцем [8]. Также одновременно с этим дети изучают фонетические звуки и названия букв.

Если ребенок борется с почерком, плохой мелкой моторикой и дисграфией, это приложение поможет ему научиться писать.

Приложение имеет прекрасную интерактивную анимацию: от появляющейся радуги до танцующих кексов, подпрыгивающих сердец, тигриных мордочек и многого другого – которая мотивирует ребенка продолжать заниматься.

Демонстрация работы приложения представлена на рисунке 1.3.

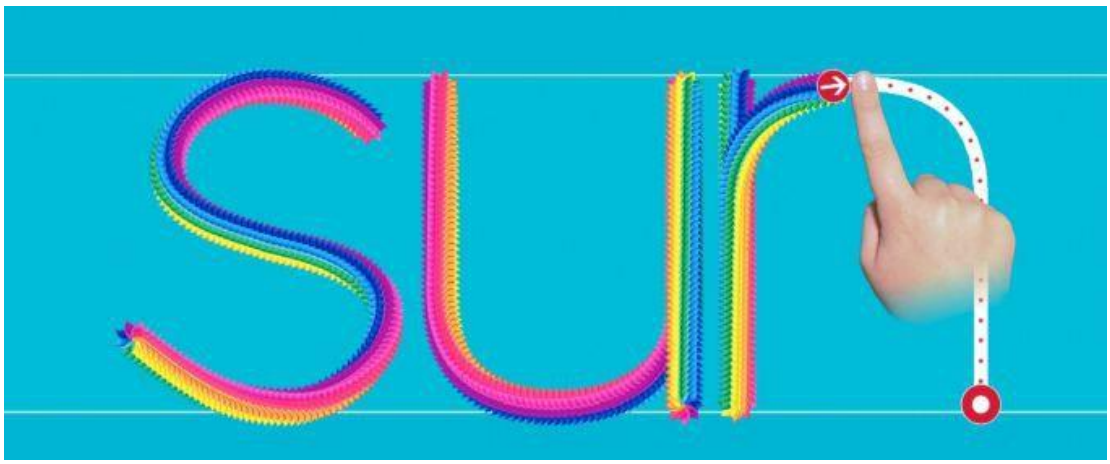

Рисунок 1.3 – Демонстрация работы приложения «Мастер письма для детей»

Яркая игривая форма помогает детям практиковать навык написания фигур, цифр, прописных или строчных букв и слов. После каждой правильной попытки система приложения выдает уникальные награды. Также приложение позволяет настроить графический ввод, выбрав один из различных стилей написания.

Приложение доступно для устройств на платформах Android и iOS.

«iWriteWords» – приложение, предназначенное для практики рукописного ввода, чтобы помочь детям младшего возраста практиковаться в написании букв и цифр в своем собственном темпе [9].

Каждая буква сопровождается картинкой для лучшей ассоциации букв со словами и слов с изображениями. Также дети узнают, как правильно выводить строчные и прописные буквы, простые слова и цифры от 0 до 20.

Забавный мультипликационный персонаж, Мистер Краб, сопровождает детей, пока они прописывают буквы и цифры, переходя от одной точки начертания предмета к другой. Если ребенок отклоняется от линии начертания, он должен начать написание сначала, а после окончания ввода каждого предмета изучения он получает картинку, которая помогает ему вспомнить, что он написал.

Демонстрация работы приложения представлена на рисунке 1.4.

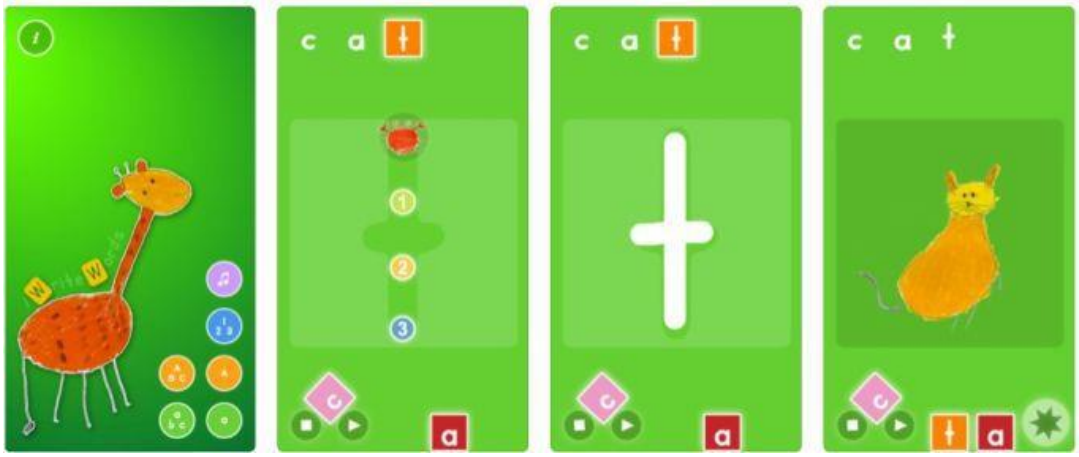

Рисунок 1.4 – Демонстрация работы приложения «iWriteWords»

Родители могут настроить приложение под потребности детей и выбрать отдельные слова или буквы для практики письма. Также можно расширить действие за кадром, помогая ребенку идентифицировать и рисовать изображения, которые он видит и которые соответствуют выученным буквам.

Приложение доступно для устройств на платформе iOS.

«Буквонямки. Учим малыша писать» – приложение, которое предоставляет набор из заданий для написания каждой буквы и забавных мини-игр в качестве поощрения за работу ребенка [10].

В данном приложении обучающемуся предоставляется выбор персонажа – снежинка или гусеница, в зависимости от того, на чем ребенок хочет учиться писать: на стекле с морозными узорами или на листочках и сочном яблоке. Выбор персонажа в приложении представлен на рисунке 1.5.

Рисунок 1.5 – Персонажи приложения

Во время обучения диктор называет букву и соответствующий ей звук, затем на экране устройства демонстрируется написание буквы. Для каждой буквы предлагается три упражнения, в которых последовательно повышается сложность.

В первом упражнении ребенок обводит буквы из начальной в конечную точку, указанную персонажем. При этом, если оторвать палец от линии или отклониться от траектории написания, персонаж остановится и будет ждать, когда ребенок продолжит писать букву. Демонстрация работы приложения представлена на рисунке 1.6.

Рисунок 1.6 – Демонстрация работы приложения «Буквонямки. Учим малыша писать»

Второе упражнение – более сложное: необходимо повторить траекторию ровно от начала до конца, не отрывая палец. Если ребенок собьется при вводе, то ему придется начать сначала.

В третьем упражнении ребенок должен написать букву самостоятельно: на экране появится только контур буквы без указания точек. Если ребенок не запомнил порядок написания элементов буквы, приложение еще раз продемонстрирует написание буквы.

Если ребенок только начинает учиться писать буквы, то можно установить «легкий» режим, в котором системой предлагается обвести букву 3 раза, но при этом сложность упражнения не будет повышаться.

За старания в любом режиме обучения ученик получит бонус – веселую мини-игру «Укрась снежинку» или «Раскрась бабочку». При этом каждый раз будут появляться новые элементы для украшения или раскрашивания. Благодаря этому у ребенка каждый раз будет получаться новая оригинальная картинка, которую он сможет сохранить на память. Демонстрация мини-игры приложения представлена на рисунке 1.7.

Рисунок 1.7 – Демонстрация мини-игры в приложении

Для родителей и учителей приложение предлагает широкие функциональные возможности: индивидуальные настройки обучения и игр для каждого ученика, включения/выключения звука, настройки голосового сопровождения и звука системы, выбор режима, а также отчеты по освоенным буквам и допущенным ошибкам ученика.

Приложение доступно для устройств на платформе iOS (iPhone и iPad).

«Буковки: учимся читать» – это приложение, которое помогает ребенку составлять слова из отдельных букв и читать первые слова по слогам [11].

Верным спутником обучения для ребенка будет персонаж – забавный единорог Лучик. Он подскажет в трудной ситуации, расскажет, как звучит та или иная буква, а также поможет прочесть первое слово.

Демонстрация работы приложения представлена на рисунке 1.8.

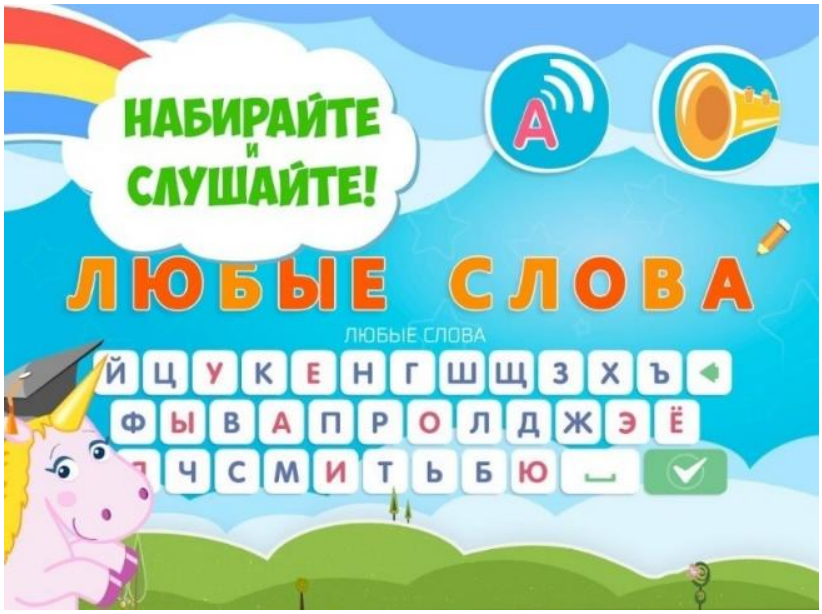

Рисунок 1.8 – Демонстрация работы приложения «Учимся читать по слогам»

В приложении ребенка ожидает:

– озвучивание букв и слогов;

– написание слов из букв;

– обучение делению слов на слоги;

– интерактивный игровой городок с увлекательными заданиями по чтению;

– интерактивные панорамы для изучения без помощи взрослых.

Приложение доступно для устройств на платформах Android и iOS.

«Kids Learn to Write» – приложение, которое помогает детям научиться писать веселым и интерактивным способом [12]. Дети учатся писать вертикальные и горизонтальные линии, фигуры, числа, английский алфавит.

Данное приложение развивает координацию рук и глаз ребенка, учит его рисовать линии, кривые и геометрические фигуры.

Приложение начинает обучение с самых простых элементов: вертикальных, горизонтальных и косых линий – и постепенно переходит к более сложным формам: кривые, круги, квадраты и т. д.

Демонстрация работы приложения представлена на рисунке 1.9.

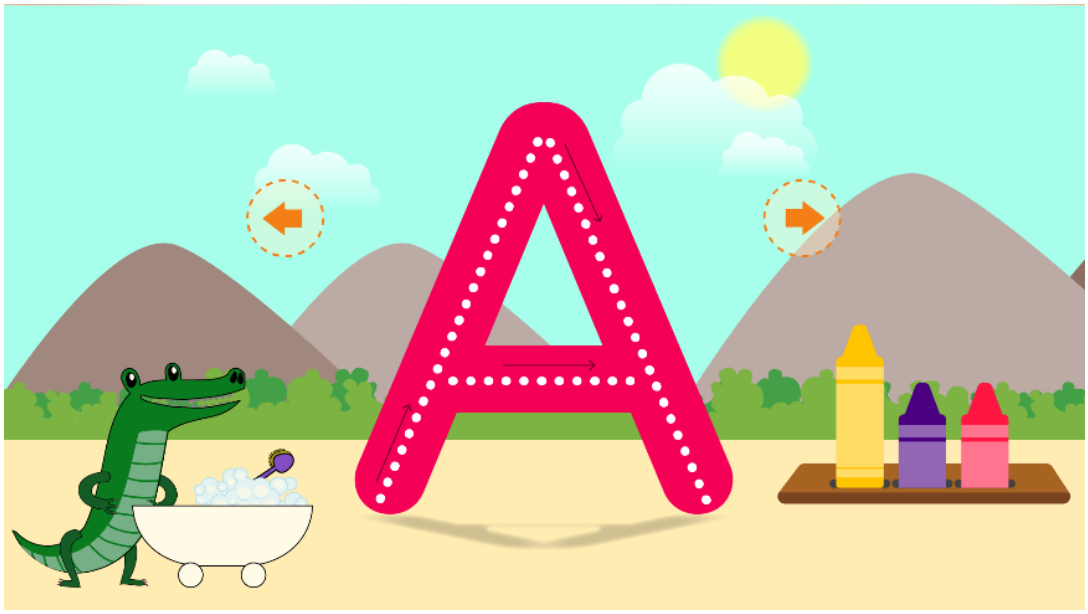

Рисунок 1.9 – Демонстрация работы «Kids Learn to Write»

В качестве развлекательного элемента приложение предоставляет игровой режим обучения. В этом режиме ребенок соревнуется с системой, которая присуждает очки за каждый завершенный игровой этап, очки отображаются на табло после каждой игры. Когда ребенок набирает достаточное количество очков, он может выбрать стикер по своему усмотрению в качестве награды. Все собранные наклейки ребенка хранятся в коллекции, в которую он может зайти в любое время.

Приложение доступно для устройств на платформах Windows и Android.

Для сравнения программных систем для обучения детей письму сформулируем следующие критерии:

1) программная платформа;

2) операционная система програмного средства (ПС);

3) способ ввода данных при обучении;

4) предмет обучения, предлагаемый ПС;

5) язык обучения;

6) наличие статистики обучения;

7) наличие звукового сопровождения при обучении;

8) дополнительные возможности, представленные в ПС.

Для дальнейшего анализа аналогов разрабатываемой системы приведем таблицу А.1 (приложение А) с характеристиками систем в соответствии с критериями, приведенными выше.

Согласно данным таблицы А.1 можно выделить следующие *недостатки аналогичных систем*:

1) платформа: большинство приложений разработано под мобильные платформы;

2) способ ввода: в представленных выше приложениях ввод осуществляется с помощью пальца или стилуса;

3) статистика: большинство приложений не предоставляют подробную статистику использования и прогресса в обучении ребенка.

Достоинствами анализируемых приложений являются:

1) звуковое сопровождение: большинство приложении озвучивают предмет изучения (буквы, слова, фигуры, цифры) до начертания;

2) поощрение: некоторые приложения представляют ребенку различные поощрения после успешного прохождения обучения (стикеры, мини-игры);

3) ассоциативные картинки: в некоторых представленных приложениях ввод предмета изучения (буквы, слова, фигуры, цифры) сопровождается картинками, ассоциативными с этим предметом;

4) пользовательские настройки: часть приложений предоставляет родителям возможность пользовательской настройки (добавление и редактирование слов, изменение ассоциативной картинки и настройка параметров системы).

1.2 Анализ основных технологических решений

В данном разделе описаны технические средства, которые были выбраны для реализации системы. Ниже приведены общие сведения о них, а также обоснована необходимость использования именно этих программных сред, средств и технологий.

Приведём список используемых технологии и программных средств, выбранных для программной реализации:

1) среда разработки Visual Studio Community 2022;

2) тип приложения ASP.NET Core MVC 3.1;

3) сервер БД SQL Server Express 2019;

4) технология Entity Framework Core (Migrations + Identity);

5) язык программирования C#;

6) язык с динамической типизацией JavaScript (jQuery);

7) язык разметки HTML5;

8) метаязык Sass (CSS).

В качестве *среды разработки для ПС* была выбрана среда Microsoft Visual Studio (MS Visual Studio, Visual Studio).

Интегрированная среда разработки MS Visual Studio является одной из самых распространённых площадок для написания, отладки, сборки кода и дальнейшей публикации ПС. Помимо основных возможностей разработки для мобильных и десктоп-платформ, Visual Studio предоставляет возможности для разработки под веб-платформу [13].

Компания Microsoft представляет следующие выпуски среды разработки MS Visual Studio: Community, Professional, Enterprise.

Для выполнения ПС был выбран выпуск Community, поскольку Visual Studio Community является бесплатной полнофункциональной интегрированной средой разработки для учащихся, участников проектов с открытым кодом и индивидуальных разработчиков.

В качестве *типа приложения для ПС* был выбран ASP.NET Core MVC.

ASP.NET MVC – это многофункциональная платформа для создания веб-приложений и API-интерфейсов с помощью структуры проектирования Model-View-Controller (MVC) [14].

Архитектура типа MVC разделяет структуру приложения на три основных группы компонентов: модели, представлении и контроллеры. Это позволяет реализовать принципы разделения задач.

Данная платформа была выбрана из-за высокой функциональности и предоставлении встроенных шаблонов для создания динамических веб-приложений с четким разделением задач. Также она обеспечивает полный контроль разметки веб-страниц и поддерживает новейшие стандарты веб-разработки.

В качестве *сервера базы данных* был выбран Microsoft SQL Server Express (MS SQL Server, SQL Server).

Microsoft SQL Server – это система управления реляционными базами данных, разработанная корпорацией Microsoft [15].

MS SQL Server является одной из наиболее популярных систем управления базами данных (СУБД). Эта СУБД предоставляет поддержку проектов различной сложности: от небольших приложений до больших высоконагруженных проектов.

Особенностями SQL Server являются:

– производительность;

– простота использования;

– безопасность (система предоставляет шифрование данных).

Компания Microsoft представляет следующие выпуски СУБД MS SQL Server: Developer, Express.

Для выполнения ПС был выбран выпуск Express, поскольку SQL Server Express является бесплатным выпуском SQL Server, который идеально подходит при разработке приложений для использования на настольных компьютерах, веб-серверах и других небольших серверах.

Для *реализации взаимодействия с СУБД* была выбрана технология Entity Framework Core (EF Core).

Entity Framework Core – это простая в использовании кроссплатформенная и расширяемая версия другой популярной технологии доступа к данным Entity Framework с открытым исходным кодом [16].

Особенностями EF Core являются:

– работа с базой данных с помощью объектов .NET;

– отсутствие необходимости в большей части кода для доступа к данным;

– поддержика множества систем баз данных.

Технология Entity Framework Core была выбрана для разработки ПС, поскольку позволяет осуществлять гибкую работу со структурой базы данных через код и структуру приложения.

В частности, в работе будут использоваться две конкретных платформы: Migrations и Identity.

Migrations – платформа для управления и хранения данных о изменениях базы данных в ASP.NET Core приложениях. Migrations используется для обновления структуры СУБД через код приложения.

В EF Core реализуется подход «Code First», при котором сначала происходит разработка классов приложения, а уже после с помощью миграций формируется представление базы данных проекта [17].

Identity – платформа для управления и хранения учетных записей пользователей в ASP.NET Core приложениях. Identity используется для аутентификации отдельных учетных записей пользователей.

В качестве *языка программирования для ПС* был выбран язык C#.

C# – это современный объектно-ориентированный язык программирования. Он позволяет создавать разные типы безопасных и надежных приложений, выполняющихся на платформах .NET [18].

Основым критерием выбора данного языка является то, что он ориентирован на разработку отдельных компонентов системы. Также немаловажной причиной выбора данного языка является то, что он относится к семейству языков C, что значительно упрощает работу с синтаксисом при наличии базовых навыков программирования на объектно-ориентированных языка.

В качестве *языка с динамической типизацией* был выбран язык JavaScript (JS).

JavaScript – это объектно-ориентированный язык программирования, который в основном используют для написания frontend- и backend-частей сайтов.

При разработке frontend-частей сайтов JS используют для создания интерактива (анимации, всплывающих форм, автозаполнения), так как он связан с языками HTML и CSS и может ими манипулировать. Во время разработки backend-частей работа с языком JS осуществляется на платформе Node.js, с помощью которой разрабатывают серверные веб-приложения и подключают библиотеки [19].

При работе с JS, в частности, будет использоваться библиотека jQuery, которая обеспечивает взаимодействие JavaScript, HTML и CSS.

Данный язык программирования был выбран как язык с динамической типизацией, поскольку через него можно осуществлять работу языками HTML и CSS, использующимися при веб-разработке. Также одной из причин выбора данного языка является то, что в настоящее время все популярные браузеры поддерживают JS.

В качестве *языка разметки для ПС* был выбран язык HTML5.

HTML5 – это 5 версия языка разметки HTML, который используется для разметки гипертекста контента на страницах браузера. Данная версия языка HTML является новой открытой платформой, которая предназначена для создания и отображения веб-приложений, использующих графику, анимацию, аудио и видео на своих страницах [20].

Данный язык был выбран для реализации разметки страниц проектируемого приложения, поскольку он является одним из основных языков для веб-средств и осуществляет отображение широкого спектра интерактивных элементов веб-страниц.

В качестве *метаязыка для ПС* был выбран язык Sass (CSS).

Sass – это метаязык, который упрощает и ускоряет написание CSS-кода. В основе Sass лежит то, он с помощью собственного внутреннего синтаксиса языка генерируется CSS-код, который понятен любому браузеру [21].

CSS (Cascading Style Sheets) представляет из себя каскадные таблицы стилей. Он отвечает за описание внешнего вида HTML-страниц.

Данный язык был выбран для реализации внешнего вида страниц проектируемого приложения, поскольку является основным языком для описания стилей веб-страниц.

1.3 Вывод

В данной части работы был проведен анализ существующих аналогов разрабатываемой системы – системы обучения детей письму.

Результатом проведенного анализа является определение параметров разрабатываемой системы, которые сформированы на основе анализа достоинств и недостатков рассмотренных систем-аналогов.

Основным недостатком систем-аналогов является то, что большинство из них разработаны под мобильную платформу. Мобильная платформа ограничивает использование системы только мобильными устройствами, что значительно сужает круг пользователей.

Привязка к платформе определяет специфичность способа ввода. Для мобильной платформы основным способом ввода информации является ввод с помощью касаний пальцами. Непрерывный ввод элементов слова с помощью прорисовки пальцем отдельных букв является менее эффективным по сравнению с побуквенной печатью, поскольку вызывает постоянное напряжение в мышцах руки, что приводит к ограничению времени использования приложения.

Отслеживание результатов обучения является наиболее значимой задачей со стороны родителя или учителя. Для этого лучше всего подходит графическое представление результатов, так как оно является самым наглядным для отслеживания динамики обучения на различных временных промежутках.

Звуковое сопровождение написания слова является частью синтетического метода обучения письму. В то время как сопровождение слова ассоциативными картинками является частью аналитического метода обучения письму. Использование этих компонентов позволит ребенку сформировать целостное представление об объекте изучения для улучшения дальнейшего запоминания.

Для удержания внимания ребенка во время обучения необходимо совместить учебный и развлекательный процесс. В системах обучения используются различные поощрения, которые могут представлять из себя как различные награды, так и мини-игры.

Возможность пользовательской настройки позволяет лучше адаптировать процесс обучения под конкретного ребенка, сделать его не только практичным, но и удобным.

Исходя из приведенных выше описаний параметров систем-аналогов, сформулируем параметры *разрабатываемой системы*:

1) платформа: проектируемое приложение будет разработано под веб-платформу [22];

2) способ ввода: ввод данных в приложение будет осуществляться с помощью клавиатуры (виртуальная или физическая клавиатура в зависимости от типа устройства);

3) статистика: приложение будет предоставлять родителю или учителю статистику по выученным и невыученным словам в виде списка;

4) звуковое сопровождение: проектируемое приложение будет поддерживать озвучивание вводимого слова до ввода;

5) поощрение: в результате успешного прохождения задания приложение предложит ребенку поощрение в картинки;

6) ассоциативные картинки: во время выполнения задания система будет предоставлять картинку вводимого слова для лучшего восприятия изучаемого предмета;

7) пользовательские настройки: родителям и учителям приложение будет предоставлять возможность пользовательской настройки, а именно добавление и редактирование пользовательских слов.

2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

В данном разделе работы представлены требования к разрабатываемой системе. Требования к структуре и функционированию системы представлены в разделе функциональные требования. Требования к реализации функций системы – в разделе нефункциональные требования.

2.1 Функциональные требования

В системе предлагается выделить следующие функциональные подсистемы:

– подсистема сбора, обработки и хранения данных, которая предназначена для реализации процессов сбора и хранения данных;

– подсистема режимов обучения, которая предназначена для реализации обучения;

– подсистема формирования и визуализации статистики обучения.

Система должна поддерживать следующие режимы функционирования:

– *основной режим*, в котором пользователь типа «ребенок-ученик» выполняет основные задачи обучающей системы;

– *статистический режим*, в котором пользователь типа «родитель-учитель» работает со статистикой использования системы пользователем «ребенок-ученик»;

– *режим пользовательских настроек*, в котором пользователь типа «родитель-учитель» выполняет настройку пользовательских параметров.

В *основном режиме* функционирования система должна обеспечивать обучение письму на основе зрительных образов в режиме вывода озвучивания слова и картинки для написания слова по буквам.

В *статистическом режиме* система должна обеспечивать возможность просмотра статистики по конкретному пользователю типа «ребенок-ученик».

В *режиме пользовательской настройки* система должна обеспечивать возможность редактирования пользовательских слов.

2.2 Нефункциональные требования

Подсистема входа и регистрации должна позволять:

1) учитывать следующие сведения:

– логин;

– пароль;

– адрес электронной почты;

– тип пользователя.

2) регистрировать пользователя вручную;

3) при регистрации в системе нового пользователя запрашивать:

– логин;

– пароль;

– адрес электронной почты.

Также время проверки вводимых данных и входа в систему не должно превышать 10 секунд.

Подсистема поощрений ребенка должна позволять:

1) отображать следующие виды поощрений:

– анимированный вывод сообщения об успешном прохождении слова (картинка с поздравлением);

– звуковое сопровождение поздравления (веселая мелодия).

2) длительность воспроизведения поощрения не должна превышать 10 секунд.

Подсистема сбора и вывода статистики выполнения должна выводить данные:

1) о выученных словах. Выученными словами являются слова, которые были введены без ошибок за последние 3 дня использования системы;

2) о невыученных словах. Невыученными словами считаются слова, которые были введены с ошибками за последние 3 дня использования системы.

Статистика должна формироваться и выводиться не более, чем за 10 секунд.

Подсистема управления пользователями должна:

1) разграничивать функции типов пользователей:

– для типа пользователей «ребенок-ученик» должен быть доступен только режим обучения;

– для типа пользователей «родитель-учитель» должен быть доступен режим отображения статистики и настройки.

Подсистема *режима обучения* должна давать пользователю типа «ребенок-ученик» возможность выбора режима обучения. К режимам обучения относится режим вывода озвучивания слова и картинки для написания слова по буквам.

Подсистема *режима пользовательских настроек* должна давать пользователю типа «родитель-учитель» возможность:

1) добавления новых элементов в режим обучения: нового слова, его визуального (изображения) и аудио сопровождения.

2) редактирования элементов в режиме обучения: редактирование слова и/или его визуального и аудио сопровождения;

3) просмотра статистики пользователя «ребенок-ученик».

Графические изображения должны быть представлены в следующих форматах:

– png;

– jpg.

Аудио сопровождение должно быть представлено в формате mp3.

3 ПРОЕКТИРОВАНИЕ

В этом разделе описана архитектура предлагаемого решения с графическим представлением основных модулей системы, описанием всех модулей системы, функциональной структурой системы и описанием пользовательского интерфейса. Также в этом разделе представлено описание вводимых и выводимых данных системы.

3.1 Архитектура предлагаемого решения

Представим архитектуру предполагаемого решения в модели TO BE – как должен выглядеть объект информатизации после разработки.

Для этого используем модель IDEF0. IDEF0 – это нотация графического моделирования, служащая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальные объекты, связывающие эти функции. Протопитирование модели выполнено в программном средстве Ramus [23].

Основные модули системы представлены на рисунках рисунки 3.1–3.5.

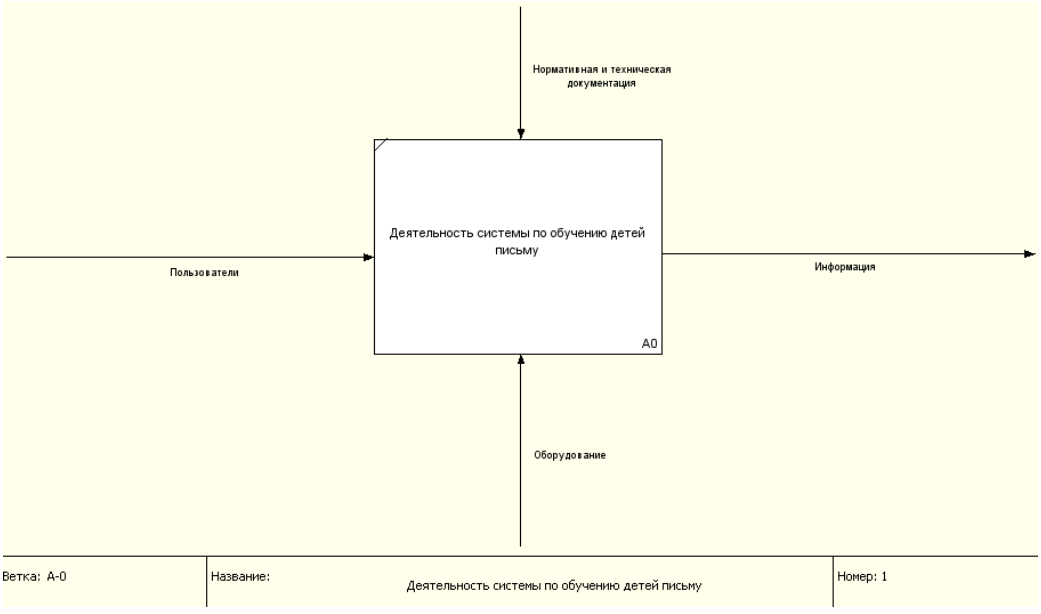

Рисунок 3.1 – Функционирование системы

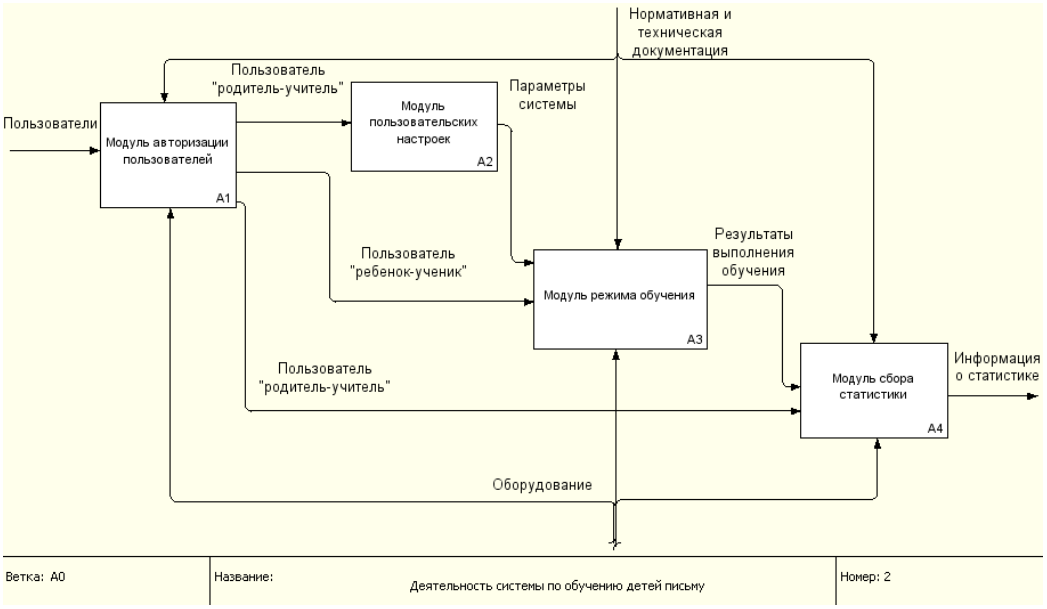

Рисунок 3.2 – Декомпозиция блока «Деятельность системы по обучению детей письму»

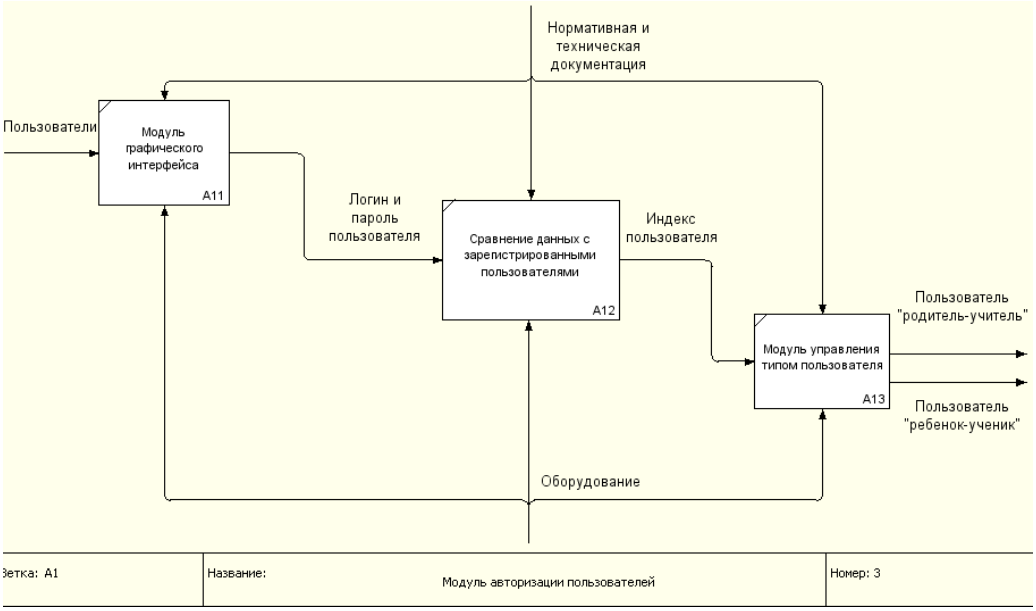

Рисунок 3.3 – Декомпозиция блока «Модуль авторизации пользователя»

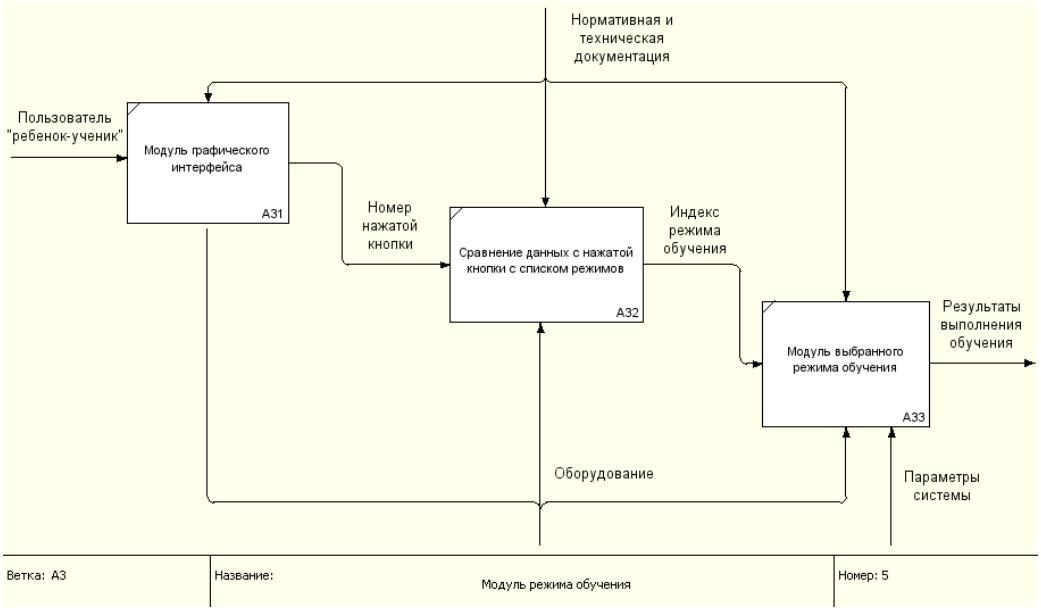

Рисунок 3.4 – Декомпозиция блока «Модуль режима обучения»

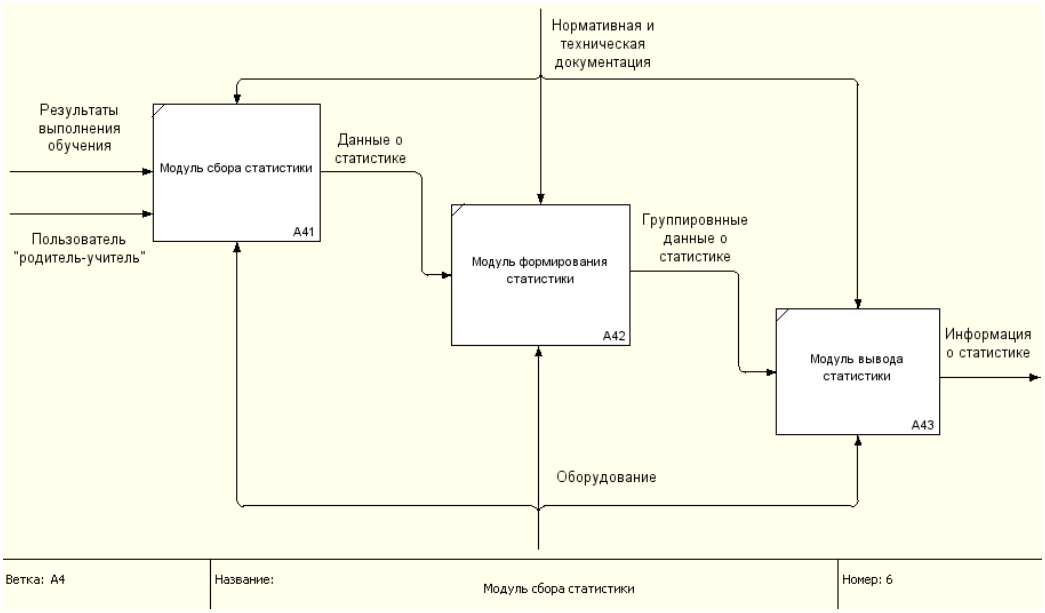

Рисунок 3.5 – Декомпозиция блока «Модуль сбора статистики»

В системе выделены следующие *функциональные подсистемы*:

– подсистема сбора, обработки и хранения данных, которая предназначена для реализации процессов работы с данными;

– подсистема режимов обучения, которая предназначена для реализации процессов обучения;

– подсистема формирования и визуализации статистики обучения.

Схема функциональной структуры системы представлена на рисунке 3.6.

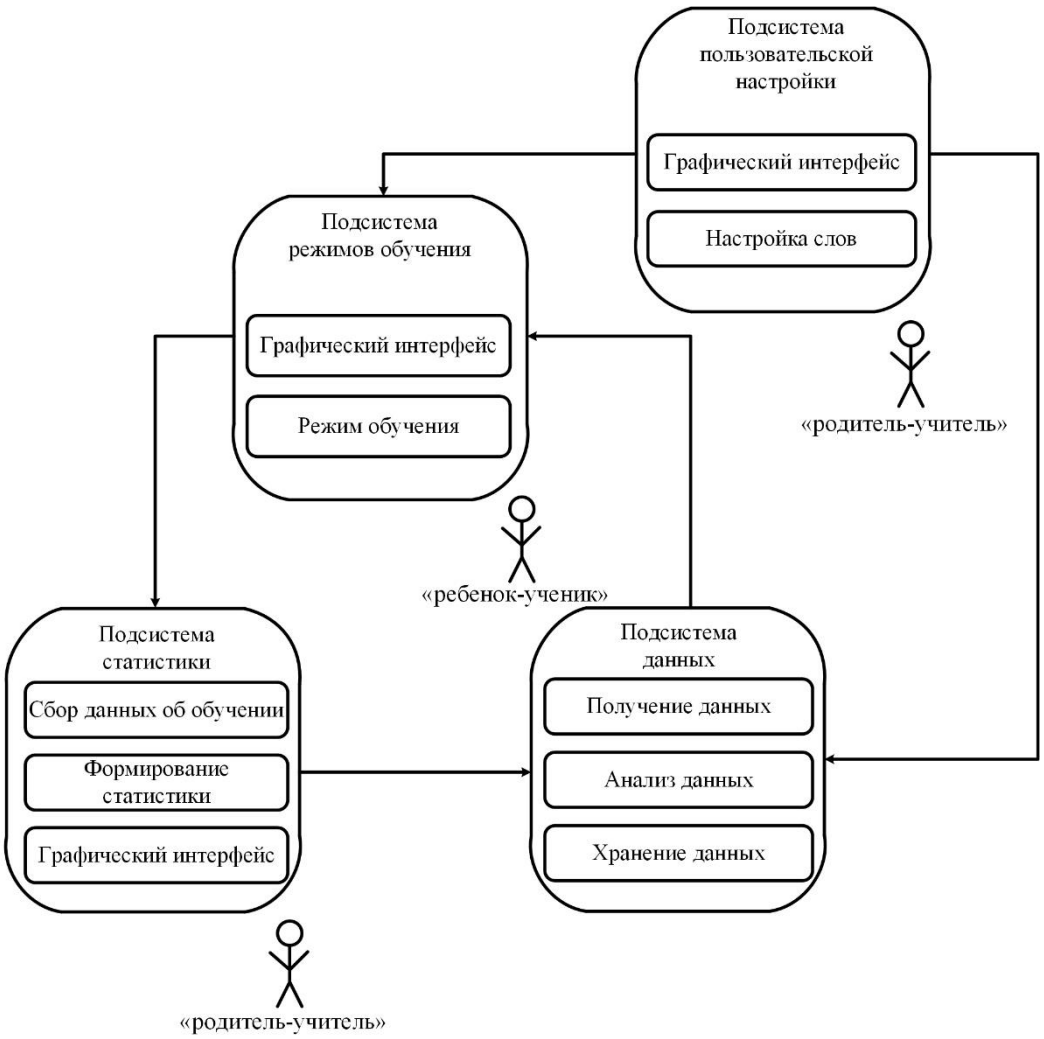

Рисунок 3.6 – Схема функциональной структуры системы

Система состоит из следующих компонентов, реализующих подсистемы:

– модуль режимов обучения;

– модуль авторизации пользователя;

– модуль выбора пользователя;

– модуль сбора статистики использования;

– модуль редактирования базы данных;

– модуль поощрения;

– модуль графического интерфейса.

Все компоненты взаимосвязаны и выполняют чётко определённую функцию в рамках всей системы.

Модуль режимов обучения представляет собой набор режимов для обучения, находящихся в главном окне графического интерфейса программы пользователя «ребенок-ученик».

В его состав входит режим озвучивания и показа картинки для дальнейшего набора слова на клавиатуре.

Модуль авторизации пользователя представляет собой набор проверок совпадения вводимых данных пользователя с данными, находящимися в базе пользователей, таких как «логин» и «пароль».

Суть этого модуля заключается в обеспечении безопасности доступа к данным и разграничении пользовательских данных в статистике.

Модуль выбора пользователя отвечает за определение типа пользователя и выдачу соответствующих прав.

В данной системе существует несколько типов пользователей:

– родитель-учитель;

– ребенок-ученик.

Тип пользователя «родитель-учитель» имеет права администратора: может просматривать статистику, редактировать режим, добавлять данные в базу данных слов.

Тип пользователя «ребенок-ученик» имеет права пользователя: может использовать режимы обучения и проходить задания.

Модуль сбора статистики пользователя представляет собой инструмент выдачи статистики по результатам использования системы пользователем «ребенок-ученик».

С его помощью можно просмотреть списки выученных и невыученных слов.

Модуль редактирования базы данных предназчен для упрощения процессов удаления, изменения и добавления новых элементов. Данные, поддающиеся редактированию: база слов, изображений и аудио, соответствующих этим словам.

Модуль поощрения предназначен для выдачи поощрения пользователю типа «ребенок-ученик».

Модуль графического интерфейса отвечает за визуальное представление пользовательского интерфейса.

Приведем *описание возможного сценария* для последующей реализации подсистем (таблицы 3.1 – 3.3).

Таблица 3.1 – Описание сценариев использования подсистем сбора, обработки и хранения данных

Подзадача	**Действие**
Создание записи	– пользователь типа «родитель-учитель» заполняет поля в графическом интерфейсе; – подсистема обрабатывает введенные данные и систематизирует их под шаблон данных в базе данных; – подсистема сохраняет данные в базу данных системы.
Редактирование записи	– подсистема извлекает выбранные данные из базы данных; – подсистема заполняет графический интерфейс извлеченными данными; – пользователь типа «родитель-учитель» изменяет поля в графическом интерфейсе; – подсистема обрабатывает введенные данные и систематизирует их под шаблон данных в базе данных; – подсистема сохраняет обновленные данные в базу данных системы.
Удаление записи	– пользователь типа «родитель-учитель» через графический интерфейс выбирает запись данных, выбранную к удалению; – подсистема обрабатывает полученную команду и находит запись данных в базе данных; – подсистема удаляет определенную запись из базы данных.
Создание записи статистики	– подсистема получает данные о результатах работы модуля режимов обучения; – подсистема обрабатывает введенные данные и систематизирует их под шаблон данных в базе данных; – подсистема сохраняет данные в базу данных системы.

Таблица 3.2 – Описание сценариев использования подсистемы режимов обучения

Подзадача	Действие
Выбор режима	– пользователь типа «ребенок-ученик» выбирает режим обучения в графическом интерфейсе; – подсистема обрабатывает полученную команду и находит выбранный режим; – подсистема запускает выбранный пользователем режим.
Выполнение обучения	– подсистема случайным образом выбирает запись с данными о картинке и слове; – подсистема выводит данные в графический интерфейс; – пользователь типа «ребенок-ученик» выполняет задание в соответствии с режимом; – подсистема сравнивает полученное значение из формы с правильным значением; – после неправильного ввода слова подсистема возвращает страницу ввода и выдает сообщение об ошибке; – после правильного введения слова подсистема выдаёт поощрение и переходит на следующие слово, не меняя режим.
Получение данных о результате прохождения слова	– после окончания слова, если были допущены ошибки, система заносит слово в список невыученных слов; – после окончания слова, если не были допущены ошибки, система заносит слово в список выученных слова.

Таблица 3.3 – Описание сценариев использования подсистемы статистики

Подзадача	Действие
Обработка данных об обучении	– подсистема считает число невыученных слов и считывает их список; – подсистема считает число выученных слов и считывает их список.
Представление графической реализации статистики	– подсистема представляет данные статистики в графическом интерфейсе в виде списков.

Приведем *решения по пользовательскому интерфейсу*.

Организация диалогового взаимодействия с пользователями происходит в таких подсистемах, как:

1) вход, регистрация;

2) основное меню;

3) режимы обучения;

4) средство вывода статистики;

5) меню пользовательских настроек.

Перечислим элементы пользовательского интерфейса, входящего в эти подсистемы.

Подсистема входа:

– поле ввода логина;

– поле ввода пароля;

– чек-бокс для определения типа пользователя;

– кнопка входа;

– кнопка регистрации.

Подсистема регистрации:

– поле ввода логина;

– поле ввода пароля;

– поле ввода электронной почты;

– кнопка регистрации.

Обязательными полями для заполнения являются: логин, пароль, электронная почта.

Основное меню различается в зависимости от типа пользователя.

Для меню пользователя ребенок-ученик отражаются следующие элементы: кнопки выбора режимов, кнопка выхода из профиля пользователя.

Для меню пользователя родитель-учитель отражаются следующие элементы: кнопка входа в статистику, кнопка для редактирования пользовательских слов.

Режимы обучения:

– картинка, соответствующая выбранному слову;

– аудио дорожка, соответствующая выбранному слову;

– текстовое отображение, выбранного слова;

– поле для ввода слова;

– кнопка для проверки введенного слова;

– кнопка выхода из режима.

Средство редактирования пользовательских слов:

– кнопка добавления нового слова;

– список всех пользовательских слов;

– ссылки на удаление, редактирование, расположенные со словом из списка.

Средство вывода статистики:

– список выученных пользователем слов;

– список невыученных пользователем слов.

Приведем уточнения по реализации пользовательского интерфейса.

Подсистема регистрации:

– при вводе пароля подсистема должна предложить сгенерировать случайный пароль;

– подсистема не должна позволять завершить регистрацию без введения всех обязательных полей.

Подсистема входа:

– подсистема должна выполнить проверку логина и пароля для пользователя;

– подсистема должна обработать чек-бокс для определения типа пользователя.

Подсистема режимов обучения:

– выбор отображаемого слова должен происходить случайным образом;

– после неправильного введения слова подсистема должна вернуть страницу с данным словом и выдать сообщение об ошибке;

– после успешного выполнения задания должен произойти переход на страницу поощерения, а после переход к следующему слову.

3.2 Описание данных

В качестве входных данных в модулях выступают вводимые слова, их ассоциативные картинки и аудио сопровождение.

В качестве выходных данных в модулях выступает статистика данных.

Структура хранения данных в системе должна состоять из следующих основных областей:

– область временного хранения данных;

– область постоянного хранения данных.

В область временного хранения заносятся данные полей «логин» и «пароль» при входе пользователя в систему.

В область постоянного хранения заносятся данные о пользователях, данные о статистике, база данных слов для режимов обучения.

В системе предполагаются следующие подсистемы для работы с базой данных:

1) подсистема сбора, обработки и загрузки данных – подсистема, предназначенная для сбора, обработки и загрузки данных в систему;

2) подсистема хранения данных – подсистема, предназначенная для хранения данных;

3) подсистема управления статистикой – подсистема, предназначенная для сбора, обработки и загрузки статистики обучения.

Информационный обмен между компонентами системы будет реализован согласно таблице 3.4.

Таблица 3.4 – Информационный обмен между компонентами системы

	Подсистема сбора, обработки и загрузки данных	**Подсистема хранения данных**	**Подсистема управления статистикой**
Подсистема сбора, обработки и загрузки данных		X	
Подсистема хранения данных	X		X
Подсистема управления статистикой		X	

Далее опишем таблицы, предполагаемые в базе данных приложения.

«*Пользователь*» – таблица для хранения данных пользователя:

– ID пользователя: uniqueidentifier (первичный ключ);

– логин: nvarchar;

– хеш-пароль: nvarchar;

– E-mail: nvarchar.

«*Типы пользователей*» – таблица для хранения данных о типе пользователя:

– ID пользователя: nvarchar (первичный ключ);

– ID типа: nvarchar (первичный ключ).

«*Список слов*» – таблица для хранения списка слов, в том числе слов пользователя:

– ID слова: uniqueidentifier (первичный ключ);

– слово: nvarchar;

– путь аудиофайла: nvarchar;

– путь файла с картинкой: nvarchar.

«*Выученные слова*» – таблица для хранения списка выученных слов пользователя**:**

– ID слова: nvarchar (первичный ключ);

– ID пользователя: nvarchar (первичный ключ).

«*Невыученные слова*» – таблица для хранения списка невыученных слов пользователя:

– ID слова: nvarchar (первичный ключ);

– ID пользователя: nvarchar (первичный ключ).

«*Текстовые поля*» – таблица для хранения текстовых полей приложения (для оформления графического интерфейса страниц приложения):

– ID: uniqueidentifier (первичный ключ);

– заглавие: nvarchar;

– подзаголовок: nvarchar;

– текст: nvarchar;

– дата добавления: datatime.

4 РЕАЛИЗАЦИЯ

В этом разделе представлена реализация проектируемой системы, разработанной на веб-платформе, для обучения детей письму на основе зрительных образов. Далее будут представлены интерфейсы приложения в виде скриншотов и описаны программные элементы, реализующие данные интерфейсы.

4.1 Основные программные решения

Для описания реализации структуры разработанного веб-приложения приведем список модулей, используемых в проекте приложения:

1) wwwroot – корневая папка приложения, содержащая аудио файлы и файлы изображений, а также файлы языков CSS, Sass, JS, которые используются для работы с графическим оформлением сайта;

2) areas – папка, разделяющая пространства типов пользователей, в ней содержатся контроллеры и представления для пользователей admin (администратор приложения) и user (пользователи типов «родитель-учитель», «ребенок-ученик»);

3) controllers – папка, содержащая файлы основных котроллеров для веб-приложения;

4) domain – папка, содержащая файлы реализации взаимодействия форм прилоежения с базой данных;

5) migrations – папка, содержащая файлы миграций по созданию и заполенению таблиц базы данных;

6) models – папка, содержащая файлы моделей для реализации ввода данных пользователями в клиентской части приложения;

7) service – папка, содержащая файлы описания функции общений к системе;

8) views – папка, содержащая файлы представлений приложения. реализующих вывод контента на страницы приложения.

Поскольку выбранным типом реализации приложения является ASP.NET Core MVC с структурой проектирования Model-View-Controller, ниже представим таблицы, описывающие данную структуру внутри приложения (таблицы 4.1-4.7).

Таблица 4.1 – Контроллеры для области admin

Название контроллера	**Назначение**
HomeController	реализует поведение пользователя admin после входа в систему
ServiceItemsController	реализует поведение пользователя admin при редактировании базы слов приложения
TextFieldsController	реализует поведение пользователя admin при редактировании страниц прилоежния

Таблица 4.2 – Представления для области admin

Название представления	**Назначение**
Index	реализует визуализацию HomeController на странице «Панель администратора»
Edit	реализует визуализацию ServiceItemsController на странице «Редактировать запись» для базы слов
Edit	реализует визуализацию TextFieldsController на странице «Редактировать запись» для страниц приложения
_ViewImports	реализует подключение различных модулей для области admin
_ViewStart	реализует подключение представлений для области admin

Таблица 4.3 – Контроллеры для области user

Название контроллера	**Назначение**
ChildController	реализует поведение пользователей «ребенок-ученик» после входа в систему
HomeController	реализует поведение пользователя «родитель-учитель» после выбора следующего раздела на странице «Меню родителя»
ParentController	реализует поведение пользователей «родитель-учитель» после входа в систему
ServiceItemsController	реализует поведение пользователя «родитель-учитель» при редактировании базы слов приложения

Таблица 4.4 – Представления для области user

Название представления	**Назначение**
Index	реализует визуализацию ChildController на странице «Меню ребенка»
Index	реализует визуализацию HomeController на странице «Пользовательские слова»
Index	реализует визуализацию ParentController на странице «Меню родителя»
Edit	реализует визуализацию ServiceItemsController на странице «Редактировать запись» для базы слов приложения
_ViewImports	реализует подключение различных модулей для области user
_ViewaStart	реализует подключение представлений для области user

Таблица 4.5 – Основные контроллеры приложения

Название контроллера	**Назначение**
AccountController	реализует поведение пользователя при входе, выходе и регистрации в приложении
HomeController	реализует поведение пользователя на страницах «О нас», «Контакты»
ServiceController	реализует поведение пользователя «родитель-учитель» при отображении списков слов, реализации статистки
StudyController	реализует поведение пользователя «ребенок-ученик» в режиме обучения

Таблица 4.6 – Основные представления приложения

Название представления	**Назначение**
Login	реализует визуализацию AccountController на странице «Вход в личный кабинет»
Registration	реализует визуализацию AccountController на странице «Регистрация»
Contacts	реализует визуализацию HomeController на странице «Контакты»
Index	реализует визуализацию HomeController на странице «О нас»
Index	реализует визуализацию ServiceController для списка слов
Show	реализует визуализацию ServiceController для одного слова
_Layout	реализует использование приведенных ниже представлений в различных частях приложения
CssPartial	реализует использование файлов языка CSS для отображения стилей интерфейса клиентской части приложения
FooterPartial	реализует визуализацию «подвала» приложения на страницах приложения
HeaderPartial	реализует визуализацию «шапки» приложения на страницах приложения
MetatagsPartial	реализует использование метатегов внутри приложения

Продолжение таблицы 4.6

ScriptPartial	реализует использование скриптов языка JS внутри приложения
Gift	реализует визуализацию StudyController на странице пощерения
Index	реализует визуализацию StudyController на странице режима обучения
OutputImg	реализует визуализацию вывода визуального и аудио сопровождения слова
_ViewaStart	реализует подключение представлений для основной части приложения
_ViewImports	реализует подключение различных модулей для основной части приложения

Таблица 4.7 – Модели приложения

Название контроллера	**Назначение**
LoginViewModel	реализует формы ввода данных на странице «Вход в личный кабинет»
RegistrationViewModel	реализует формы ввода данных на странице «Регистрация»
Mode1ViewModel	реализует форму ввода слова в режиме обучения для пользователя «ребенок-ученик»

Исходный код реализации веб-приложения представлен в приложении Б.

Программные файлы языков Sass и CSS, используемые для оформления интерфейса веб-приложения, не приводятся в приложении Б в виду большого числа строк (более 1600).

Програмный код, описывающий таблицы и поля базы данных приложения, приведен в листинге Б.33 (приложение Б).

4.2 Реализация интерфейсов

На главной странице разработанного веб-приложения представлен раздел «о нас», в котором приведена информация о названии приложения, его цели и функциональных возможностях (рисунок 4.1).

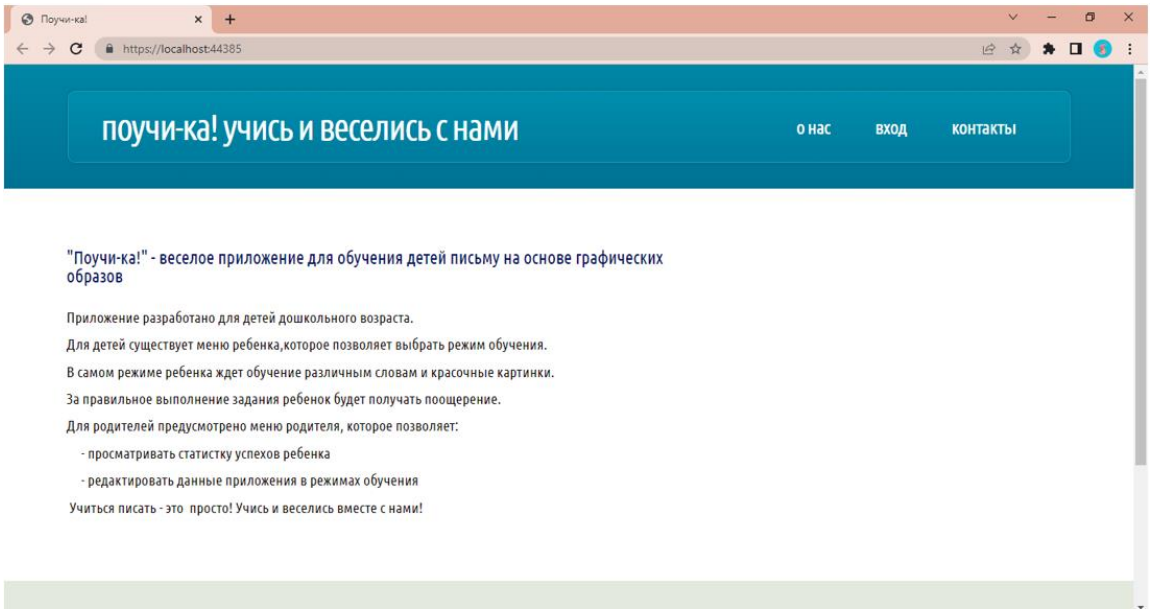

Рисунок 4.1 – Главная страница

Помимо раздела «о нас» в шапке (header) веб-приложения представлены такие разделы, как «вход» и «контакты».

На странице раздела «контакты» разработанного веб-приложения приведена информация о разработчике и контактах (рисунок 4.2). Также на данном рисунке представлен подвал (Footer) сайта, в котором указана информация о названии проекта, авторских правах и шаблоне дизайна сайта.

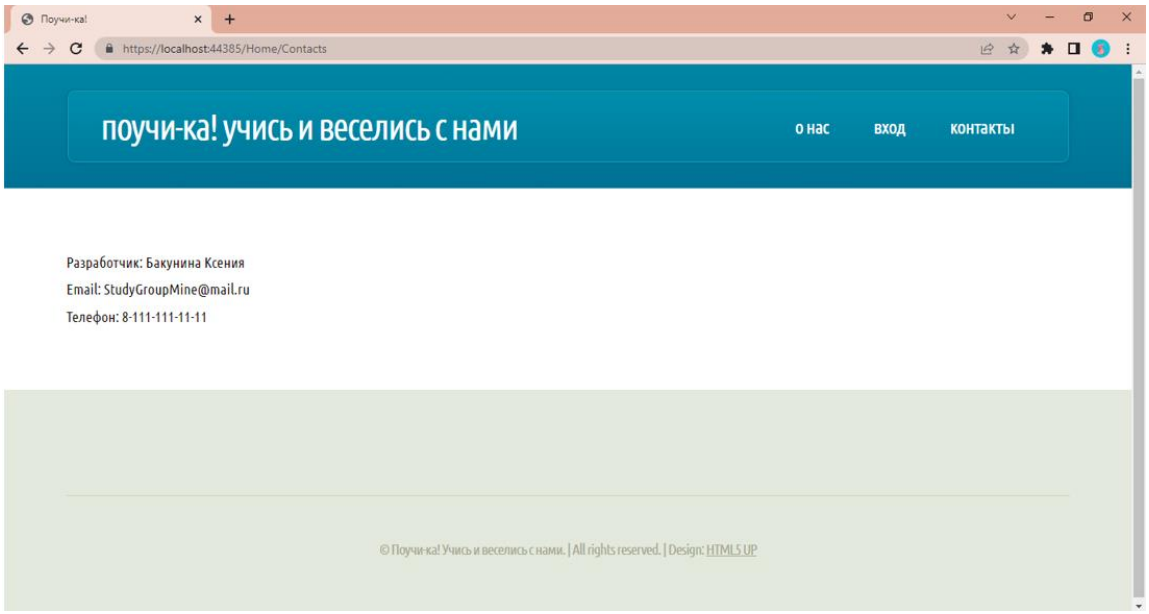

Рисунок 4.2 – Страница раздела «контакты»

На странице раздела «вход» разработанного веб-приложения приведена форма ввода пользовательских данных для аутентификации и входа в систему.

Также на этой странице реализован механизм разделения типов пользователей по средствам чек-бокса с заголовком «Родитель?» (рисунок 4.3).

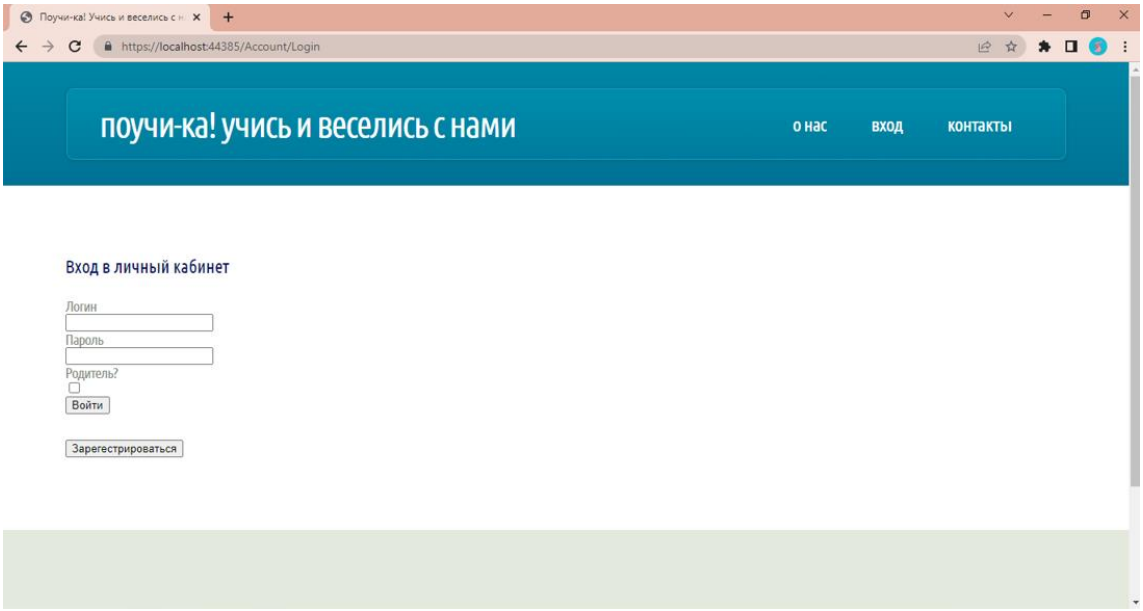

Рисунок 4.3 – Страница раздела «вход»

Для незарегистрированных пользователей ПС в разделе «вход» представлена кнопка перехода на раздел «регистрация». На странице данного раздела представлена форма для регистрации новых пользователей ПС, содержащая поля ввода данных нового пользователя (рисунок 4.4).

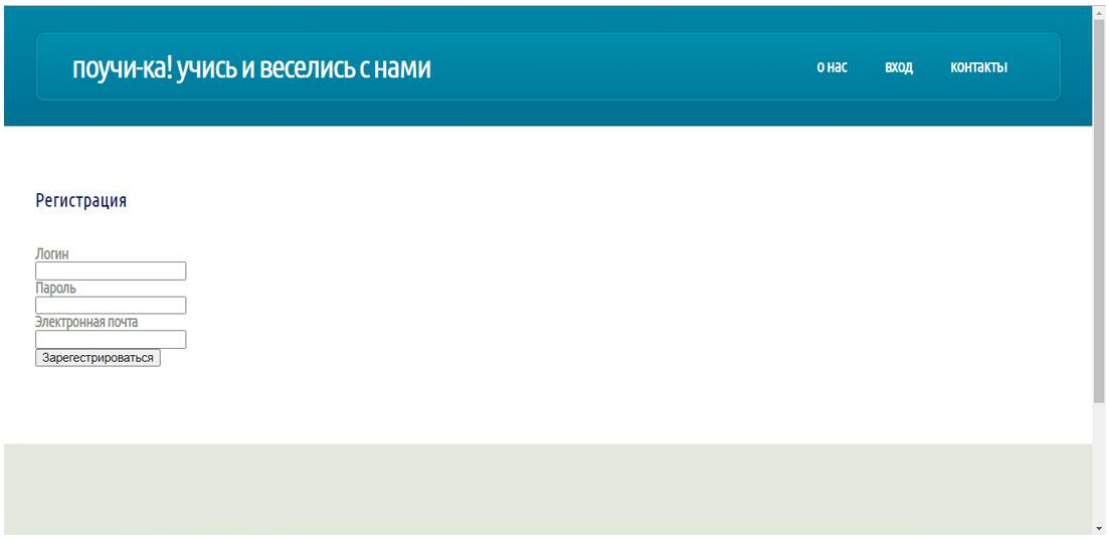

Рисунок 4.4 – Страница раздела «регистрация»

После входа в систему пользователя типа «ребенок-ученик» происходит переход в раздел «меню ребенка», который реализует взаимодействие ребенка с ПС. На странице раздела «меню ребенка» представлены кнопки выбора режимов обучения и кнопка выхода из аккаунта (рисунок 4.5).

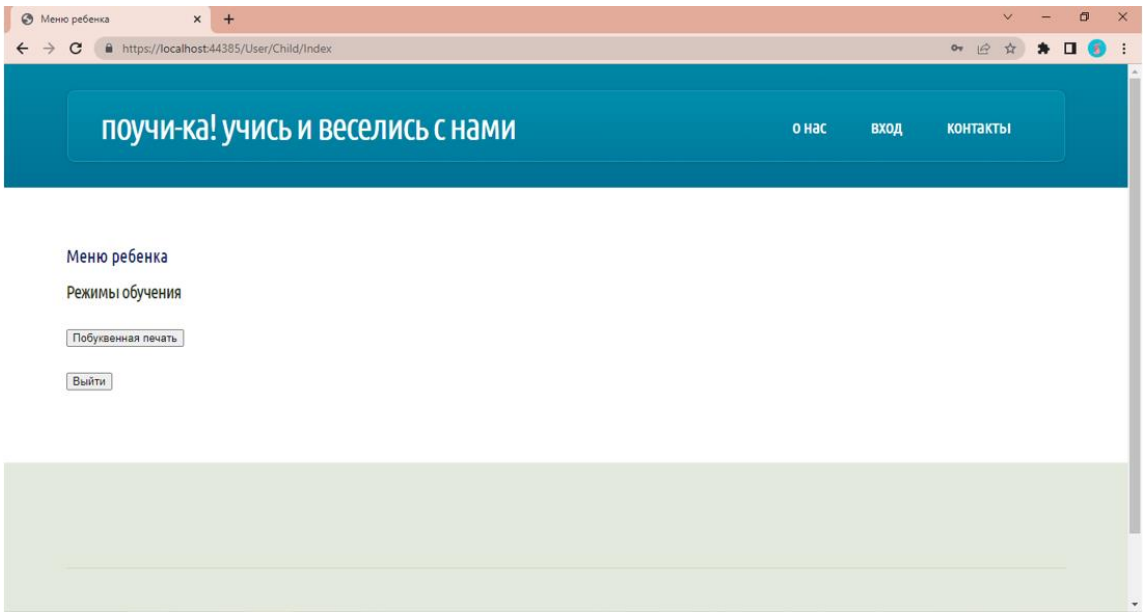

Рисунок 4.5 – Страница раздела «меню ребенка»

При нажатии кнопки «побуквенная печать» система приложение переведет ребенка на страницу режима обучения «вывод озвучивания слова и картинки для написания слова по буквам». На данной странице представлено слово для изучения, его сопроводительная картинка и панель управления аудиоэлементом озвучивания слова. После этого представлено окно для ввода ребенком указанного слова и кнопка для проверки правильности ввода (рисунок 4.6).

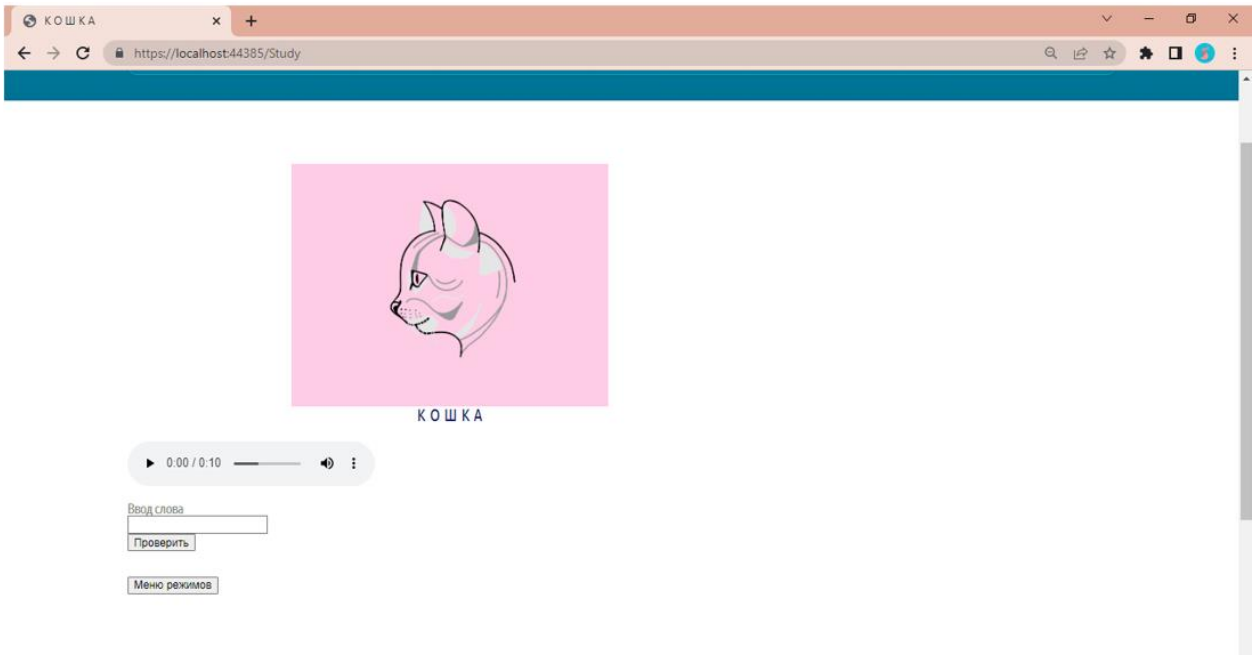

Рисунок 4.6 – Страница режима обучения

При правильном вводе слова система переведет пользователя на страницу поощрения. На данной странице отображена веселая картинка, также при переходе на данную страницу система автоматически включает мелодию, свидетельствующую о правильном введении слова. Также на этой странице представлена кнопка для перехода к изучению нового слова (рисунок 4.7).

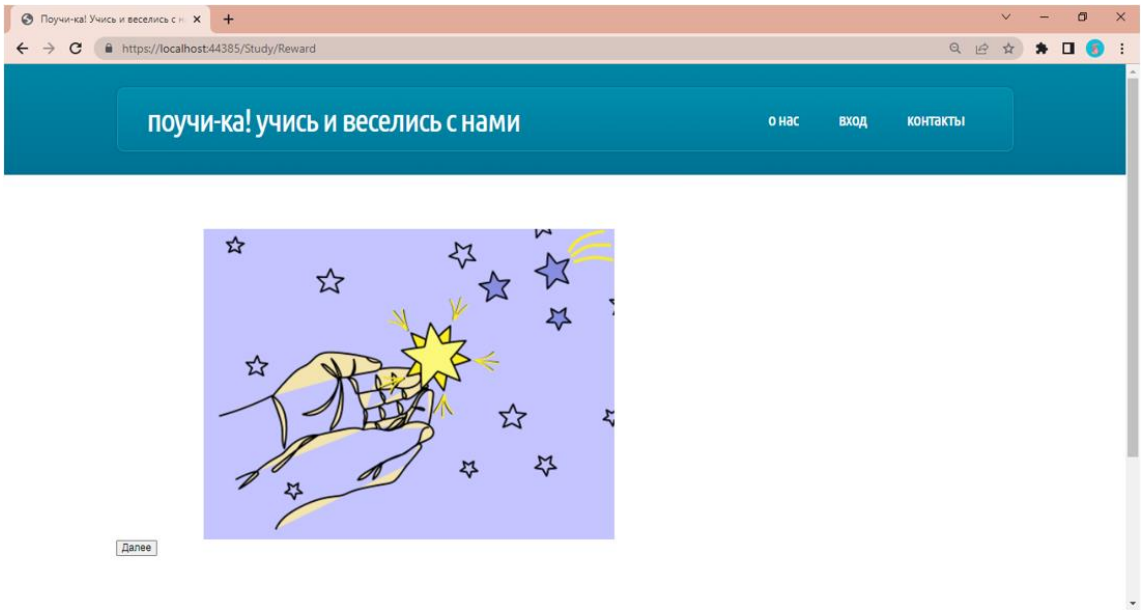

Рисунок 4.7 – Страница поощрения

После входа в систему пользователя типа «родитель-учитель» происходит переход в раздел «меню родителя». Данный раздел реализует взаимодействие родителя с ПС. На странице раздела «меню родителя» представлены кнопки выбора для настройки базы слов системы и просмотра статистики ребенка, а также кнопка выхода из аккаунта (рисунок 4.8).

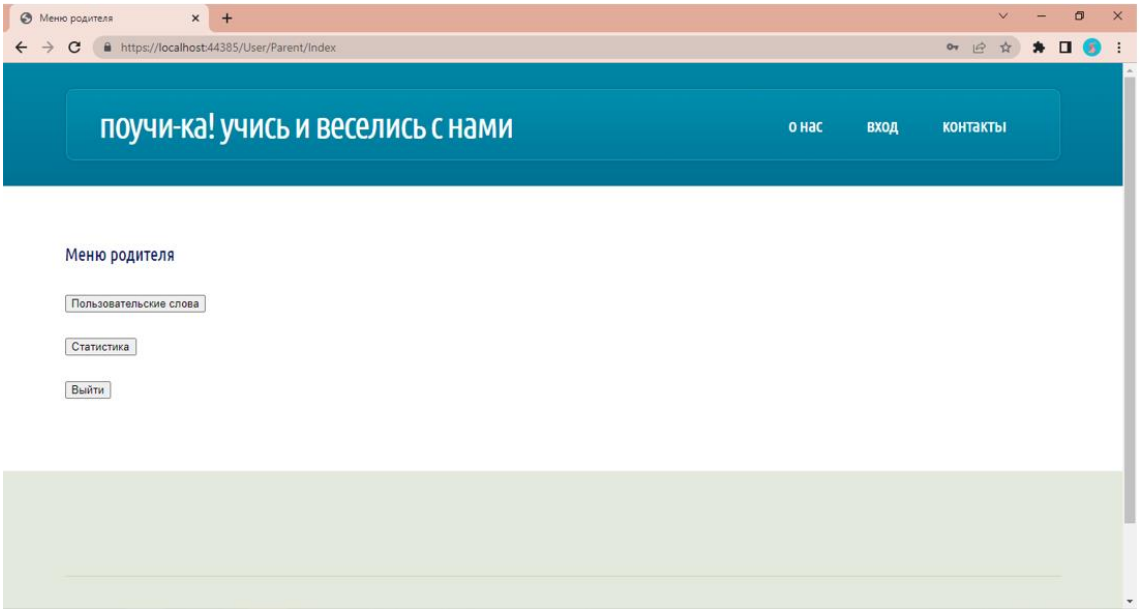

Рисунок 4.8 – Страница раздела «меню родителя»

При нажатии кнопки «пользовательский слова» система приложение переведет пользователя на соответствующую страницу. На данной странице представлены список пользовательских слов с возможностью просмотра при нажатии и инструменты его редактирования, такие как добавление нового слова, редактирования и удаления выбранного слова (рисунок 4.9).

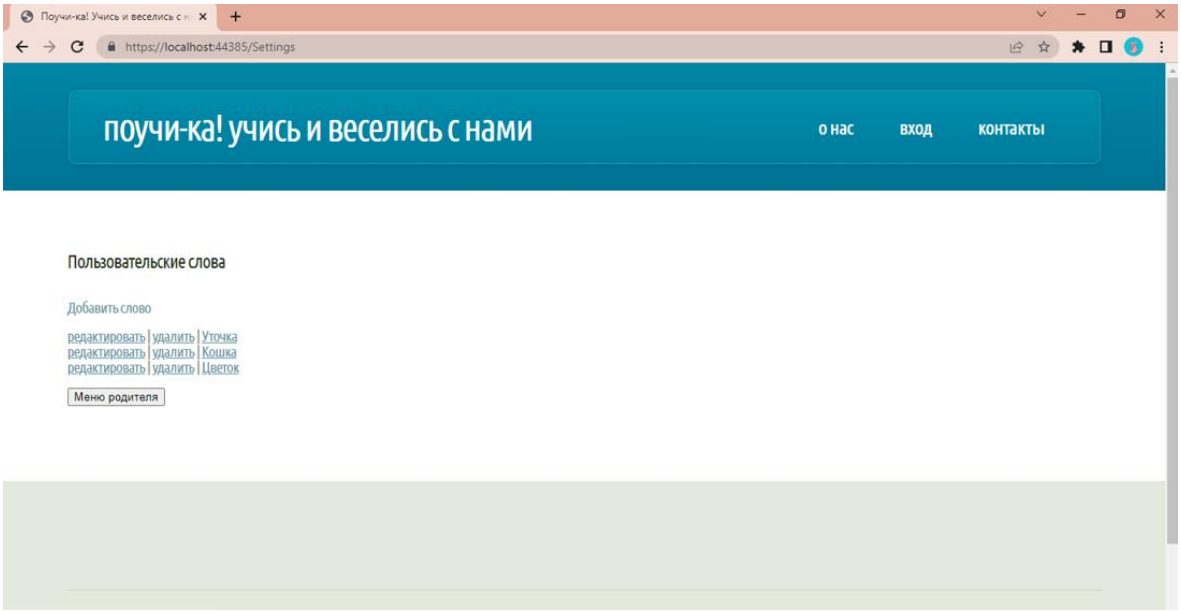

Рисунок 4.9 – Страница раздела «пользовательские слова»

При нажатии кнопки «статистика» система приложение переведет пользователя на страницу «статистика ребенка». На данной странице представлены списки выученных и невыученных слов ребенка, также представлена кнопка возврата в меню родителя (рисунок 4.10).

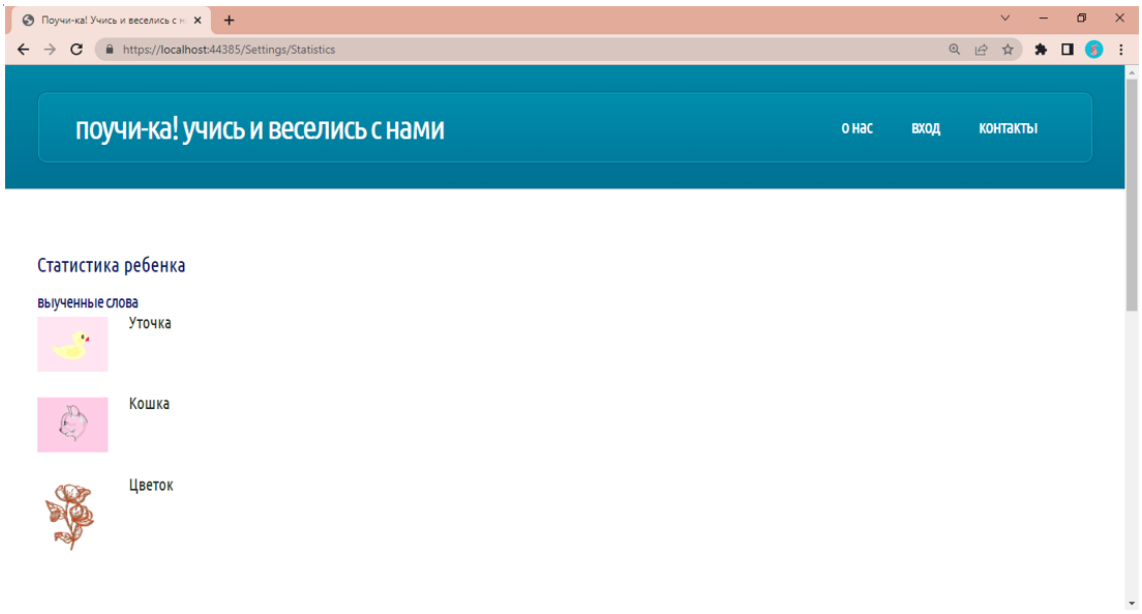

Рисунок 4.10 – Страница раздела «статистика ребенка»

Помимо пользователей типа «ребенок-ученик» и «родитель-учитель» для технического обслуживания сайта был предусмотрен пользователь типа «администратор», обладающий правами для редактирования контента страниц сайта.

Переход на страницу раздела «панель администратора» осуществляется после авторизации и при введении раздела «admin» в адресной строке сайта. В данном разделе представлены инструменты для редактирования базы слов веб-приложения и страниц сайта. Также предусмотрена кнопка выхода их аккаунта (рисунок 4.11).

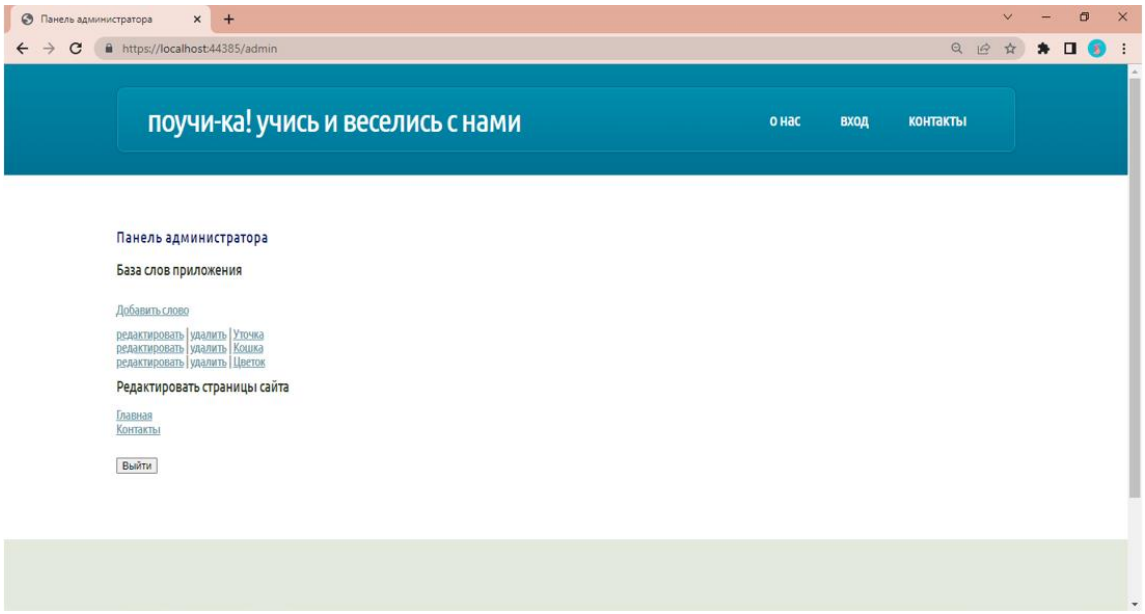

Рисунок 4.11 – Страница раздела «панель администратора»

5 ТЕСТИРОВАНИЕ

В данном разделе работы представлено тестирование разработанного программного средства.

Программное средство было полностью отлажено, а также прошло альфа-тестирование со стороны разработчика.

Ниже в подразделе «Методология тестирования» приведена информация о методологии, использованной в ходе тестирования ПС.

В подразделе «Проведение процедуры тестирования» представлена проверка правильности работы системы.

5.1 Методология тестирования

В качестве модели тестирования была выбрана каскадная модель.

Каскадная модель (Waterfall Model) – это одна из наиболее старых моделей, которая применяется для разработки и тестирования программного обеспечения [24].

Базовым принципом данной модели является последовательный характер выполнения задач. Это означает, что переход к следующему шагу разработки или тестирования осуществляется только после того, как предыдущий был успешно завершен.

В контексте данной модели процесс тестирования программного продукта начинается только после завершения процесса разработки.

Каскадная модель применяется при работе с небольшими проектами, в которых четко определены требования.

Данная модель тестирования была выбрана исходя из того, что этапы её проведения максимально приближены к этапам написания ВКР.

5.2 Проведение процедуры тестирования

Далее проведем тестирование различных разделов сайта путем ввода различных данных, в том числе некорректных.

При введении некорректных данных пользователя на странице «вход» система возвращает форму ввода данных для аутентификации и выдает ошибку о некорректности ввода (рисунок 5.1).

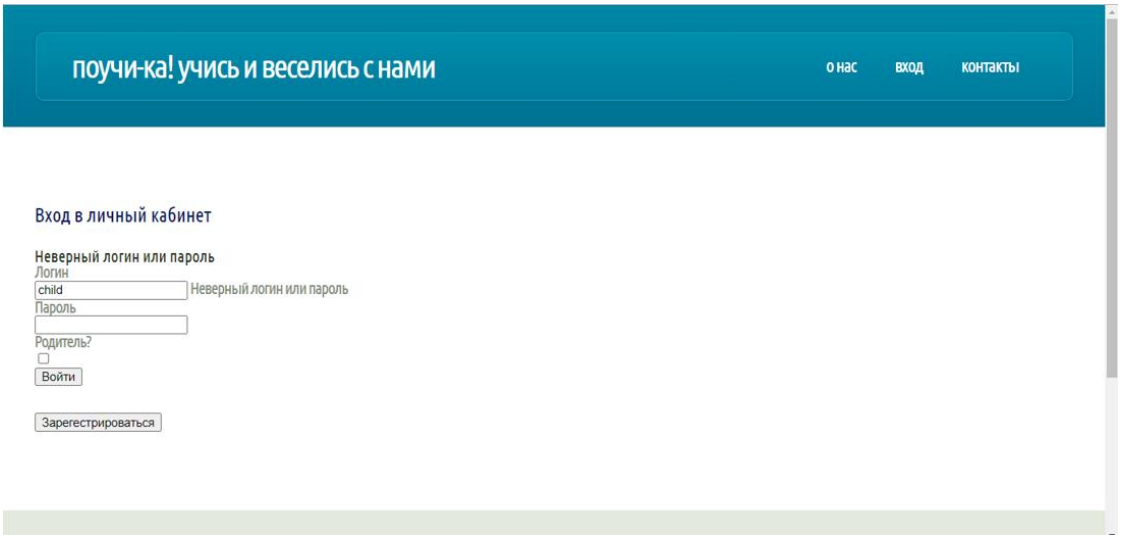

Рисунок 5.1 – Некорректный ввод данных

При введении корректных данных пользователя на странице «вход» и отсутствии выставленного значения в чек-боксе «Родитель?» система совершит переход на страницу «меню ребенка» (рисунок 5.2).

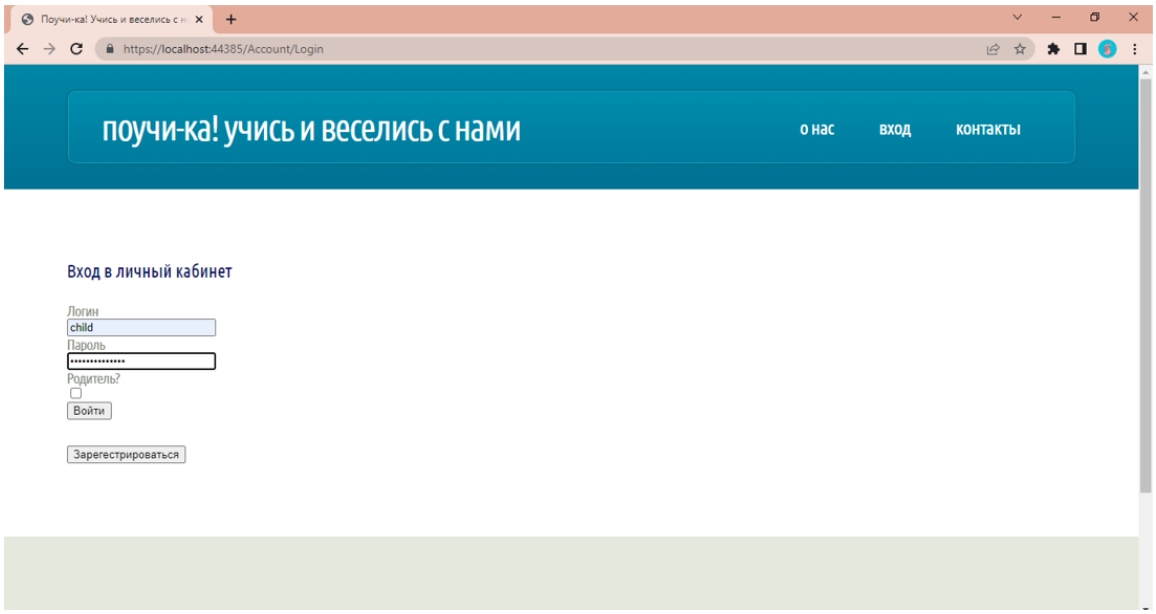

Рисунок 5.2 – Заполнение формы входа для пользователя «ребенок-ученик»

При введении корректных данных пользователя на странице «вход» и наличии выставленного значения в чек-боксе «Родитель?» система совершит переход на страницу «меню родителя» (рисунок 5.3).

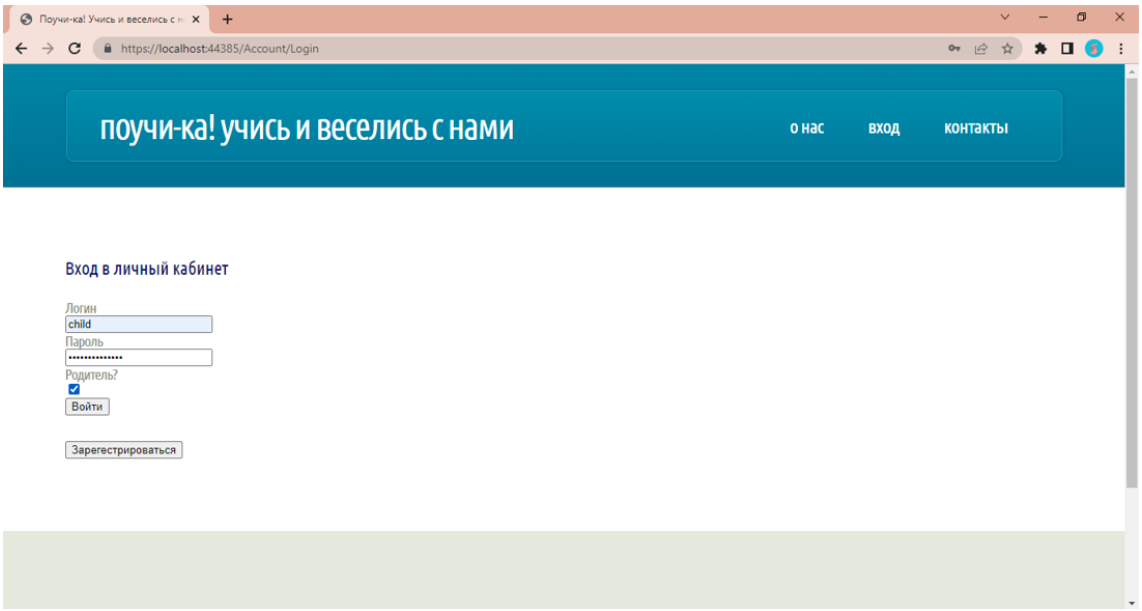

Рисунок 5.3 – Заполнение формы входа для пользователя «родитель-учитель»

Для перехода на страницу «панель администратора» необходимо корректно ввести данные администратора в форме входа и выполнить переход с помощью адресной строки (рисунок 5.4).

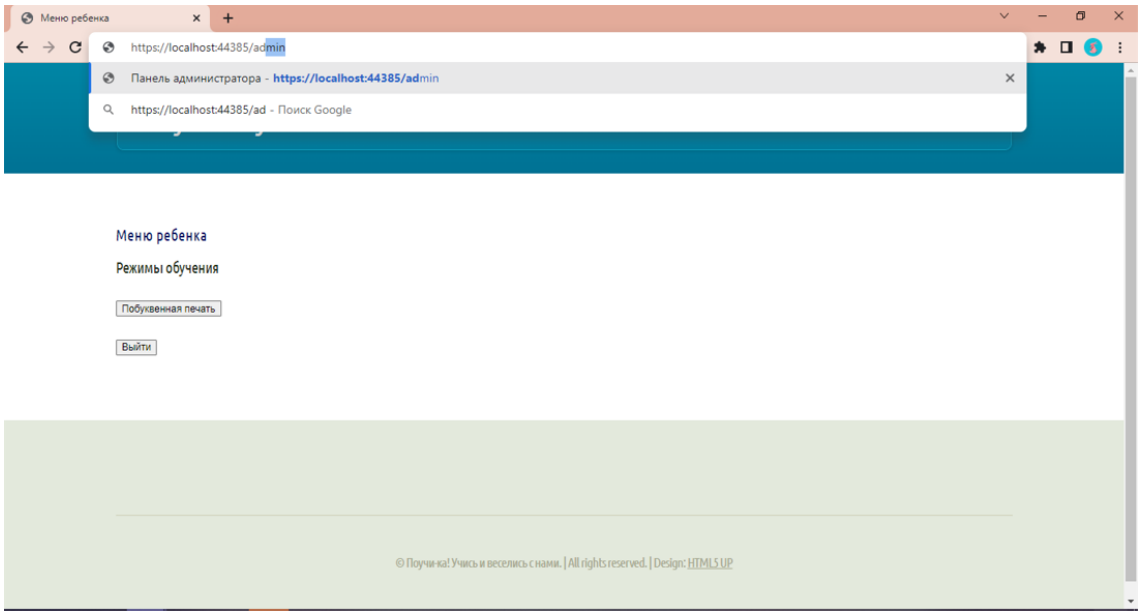

Рисунок 5.4 – Осуществление перехода на страницу «панель администратора»

При введении неправильного написания слова пользователем типа «ребенок-ученик» в поле ввода слова в режиме обучения система возвращает

форму ввода слова для изучения и выдает ошибку о некорректности ввода (рисунок 5.5).

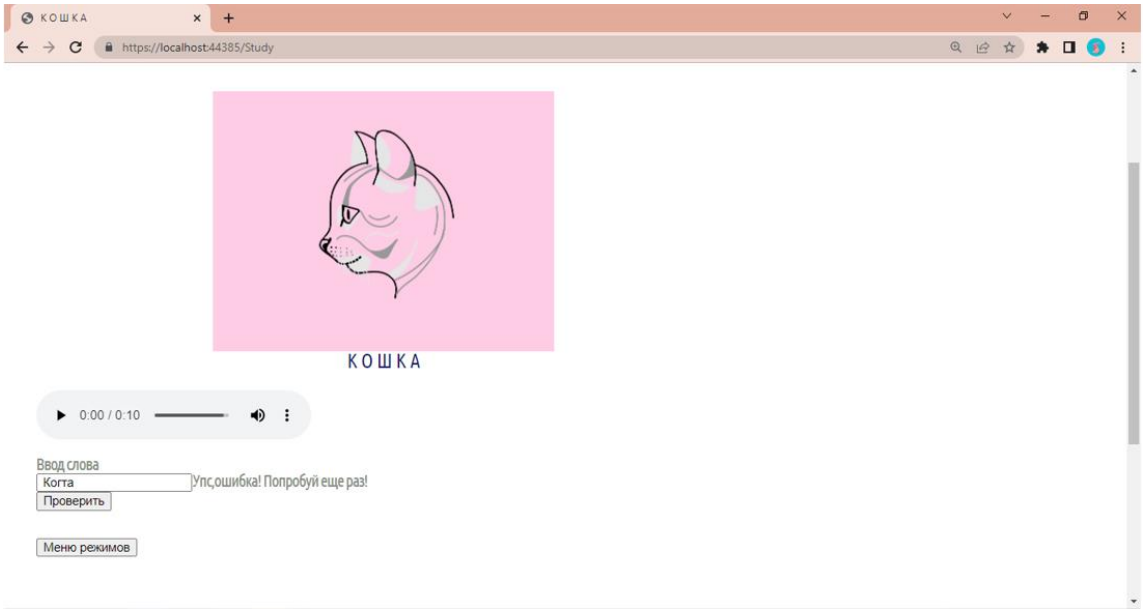

Рисунок 5.5 – Ошибка при неправильном вводе слова в режиме обучения

При введении правильного написания слова пользователем типа «ребенок-ученик» в поле ввода слова в режиме обучения система переходит на страницу поощрения, а после нажатия кнопки «далее» система переводит ребенка на новое случайное слово (рисунок 5.6-5.8).

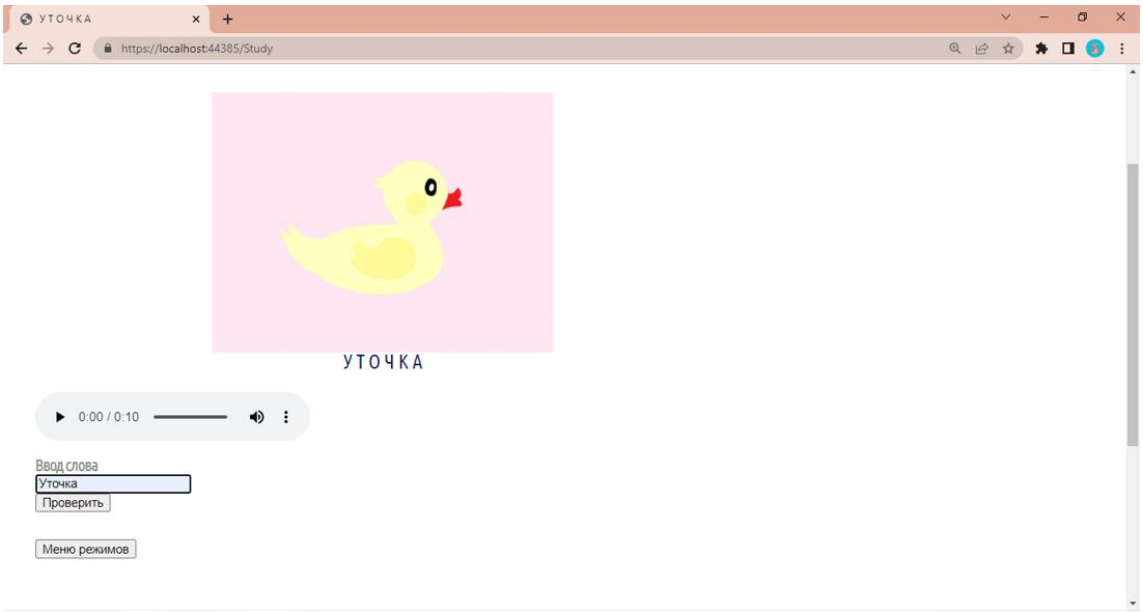

Рисунок 5.6 – Правильный ввод слова в режиме обучения

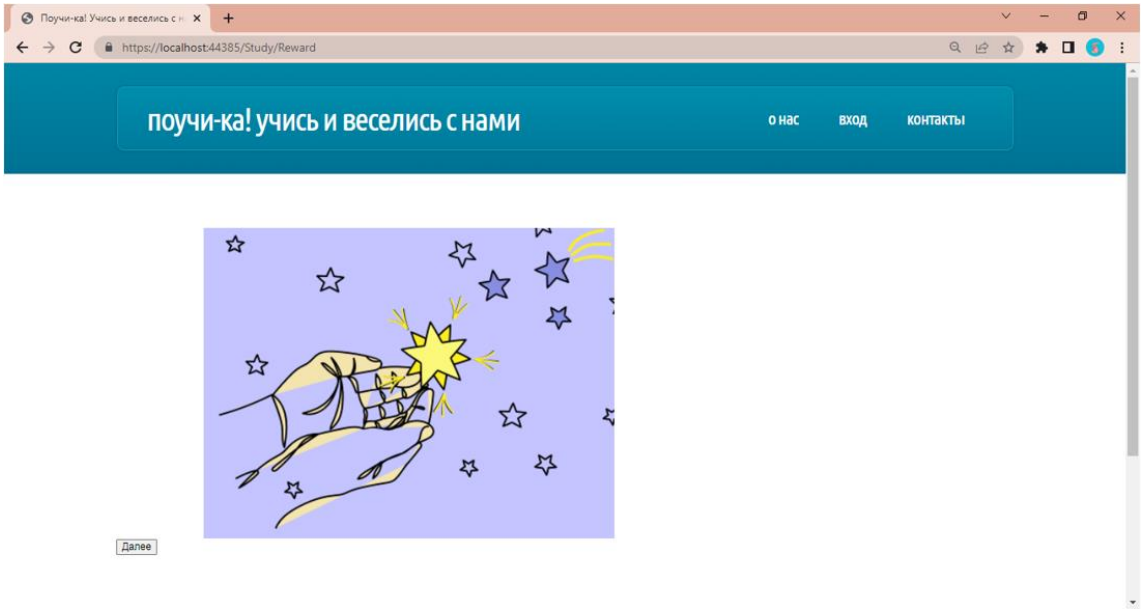

Рисунок 5.7 – Страница поощрения после правильного ввода слова в режиме обучения

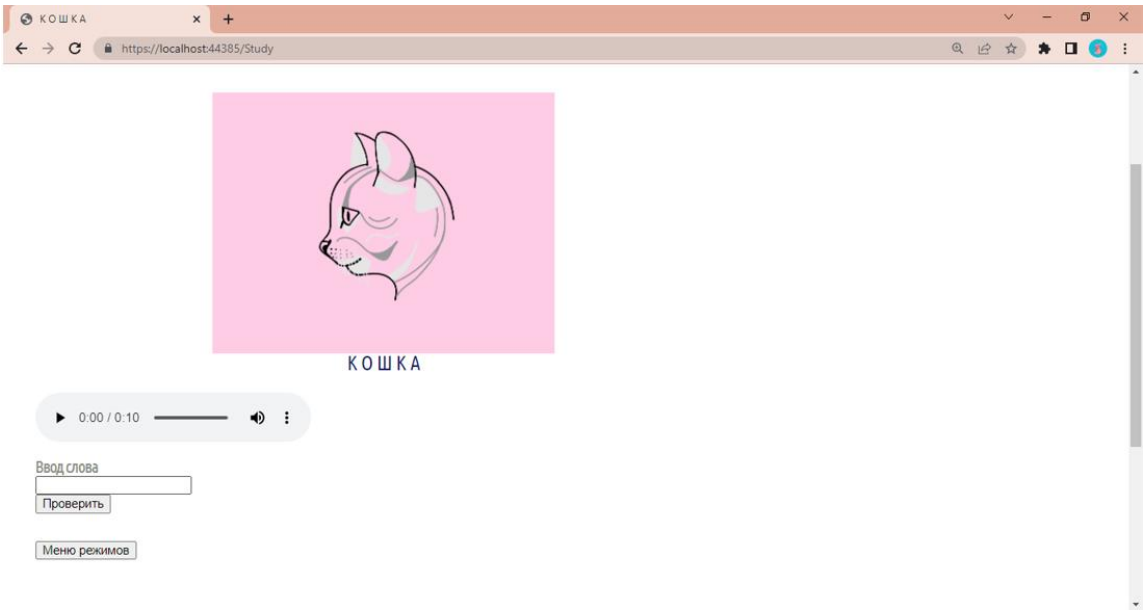

Рисунок 5.8 – Страница нового изучаемого слова после перехода со страницы поощрения

Веб-приложение предоставляет пользователям типа «родитель-учитель» и «администратор» инструменты взаимодействия с базой слов системы.

Одним из элементов взаимодействия является возможность добавление нового слова. При нажатии кнопки «добавление нового слова» на странице «пользовательские слова» для пользователя типа «родитель-учитель» или на странице «панель администратора» для пользователя типа «администратор» система переходит на страницу формы для заполнения данных о новом слове. На рисунке 5.9 показана страница добавления нового слова.

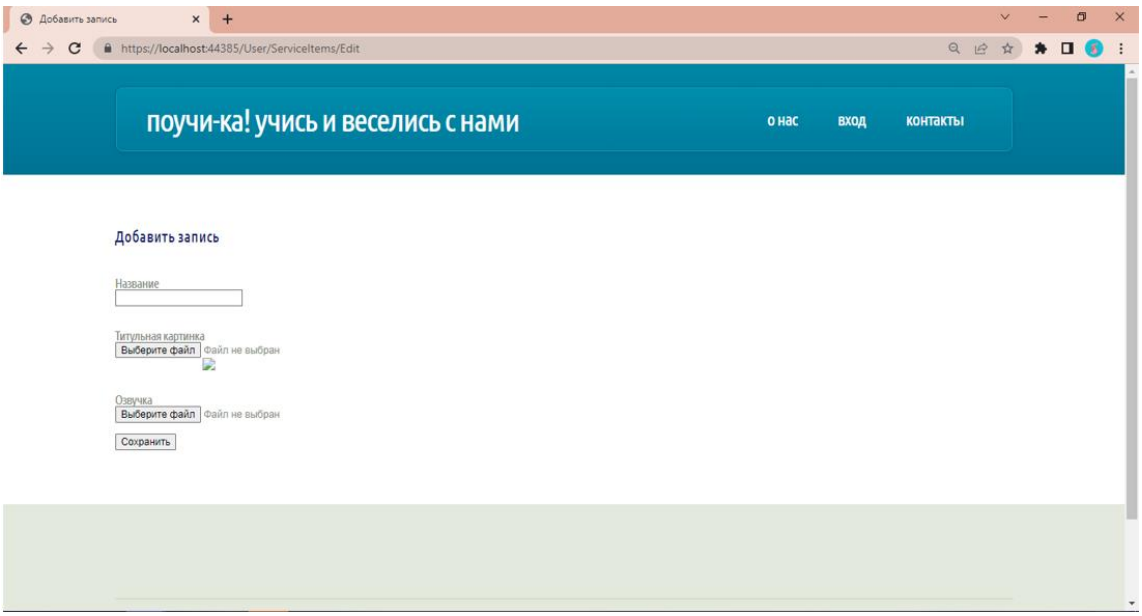

Рисунок 5.9 – Страница добавления нового слова в систему

При нажатии кнопки «редактирование» на странице «пользовательские слова» для пользователя типа «родитель-учитель» или на странице «панель администратора» для пользователя типа «администратор» система переходит на страницу формы с заполненными данными выбранного слова. На рисунке 5.10 показана страница редактирования слова «Уточка».

Рисунок 5.10 – Страница редактирования слова «Уточка»

Дополнительно пользователь типа «администратор» имеет возможность редактирования контента на страницах «о нас» и «контакты».

При выборе соответствующей страницы в панели администратора система переходит на страницу формы с заполненными данными требуемой

страницы сайта. На рисунке 5.11 показана страница редактирования контента раздела «о нас».

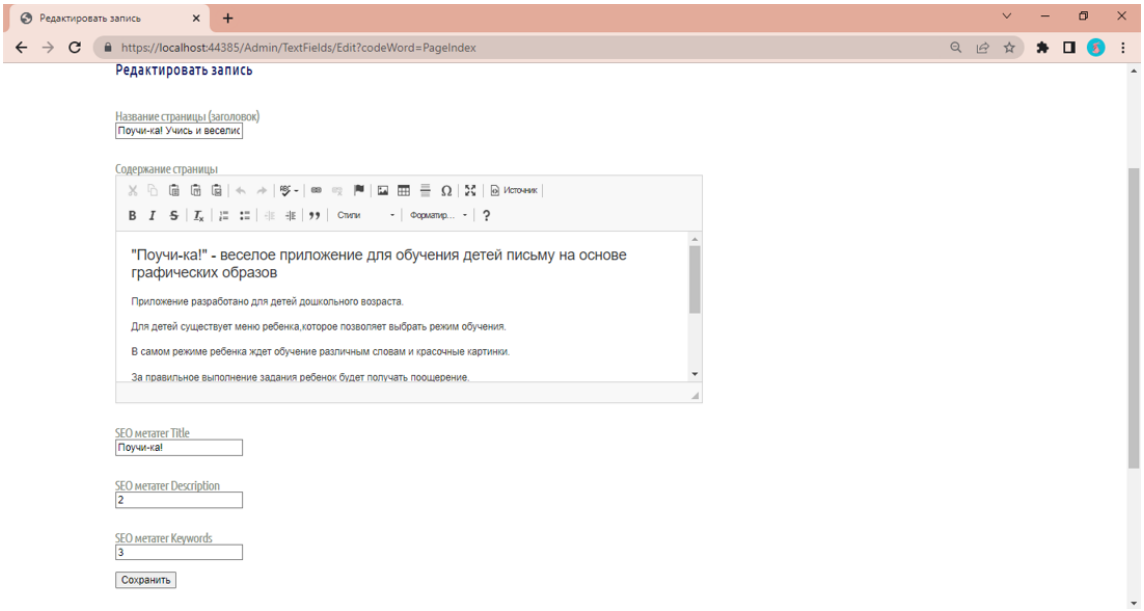

Рисунок 5.11 – Страница редактирования контента «о нас»

ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной-квалификационной работы были выполнены следующие задачи:

1) рассмотрены существующие методики обучения детей письму, выделены достоинства выбранных методик для дальнейший реализации;

2) проведен анализ современных программных систем для организации обучения письму в контексте выбранной методики обучения;

3) разработана архитектура собственного программного продукта и представлена её программная реализация;

4) оценена работоспособность разработанного программного продукта.

Результатом выполнения выпускной-квалификационной работы является полное решение основной поставленной задачи – повышение эффективности обучения детей письму. Для решения данной задачи было разработано программное средство, направленное на обучение детей письму на основе зрительных образов.

Спроектированное программное средство содержит в себе все достоинства аналогичных программных средств и учитывает их недостатки.

Данное программное средство может быть применено как для домашнего использования, так для использования в учебных заведения и специализированных центрах детского развития.

В ходе дальнейшего развития системы могут быть добавлены новые режимы обучения в меню пользователя «ребенок-ученик», такие как:

– набор букв слова по образу букв на виртуальной клавиатуре;

– вывод слов на одну заданную букву для написания слова по буквам.

А также может быть расширена база поощрений для ребенка.

Для пользователя типа «родитель-учитель» могут быть добавлены различные формы представления статистики обучения пользователя «ребенок-ученик», такие как:

– график зависимости ошибок при обучении пользователя «ребенок-ученик» от времени;

– диаграмма процентного разделения выученных и невыученных слов от общего числа.

Также может быть улучшено взаимодействие пользователя «родитель-учитель» с системными настройками, такими как работа с поощрением, звуковым сопровождением, аккаунтом пользователя «ребенок-ученик».

В ходе выполнения работы были получены следующие результаты:

– получено программное средство на веб-платформе для обучения детей письму на основе графических образов;

– объем созданного программного средства составляет 47 Мб;

– организована база данных на платформе сервера MS SQL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Безруких, М. М. Материалы курса «Трудности обучения письму и чтению в начальной школе» : лекции 1–4. / М. М. Безруких. – Москва : Педагогический университет «Первое сентября», 2009. – 84 с.

2. Justice, L. M. Engaging Children with Print: Building Early Literacy Skills through Quality Read-Alouds / L. M. Justice, A. E. Sofka. – New York : The Guilford Press, 1968. – 224 p. – ISBN: ISBN 978-1-60623-535-5.

3. Макаров, И. В. Нарушение речевого развития у детей / И. В. Макаров, Д. А. Емелина. – Текст : электронный // CYBERLENINKA : Научная электронная библиотека. – 2017. – №4. – URL : https://cyberleninka.ru/article/n/narusheniya-rechevogo-razvitiya-u-detey (дата обращения : 01.03.2022).

4. Агаркова, Н. Г. Основы формирования графического навыка у младших школьников / Н. Г. Агаркова // Начальная школа. – 1999. – №4. – С. 36 – 39.

5. 4 лучших приложения для письма для детей. – Текст : электронный // WebSetNet : [сайт]. – 2020. – URL: https://websetnet.net/ru/4-best-writing-apps-for-kids/ (дата обращения: 02.03.2022).

6. Letter School : [официальный сайт] / Letter School. – URL: https://www.letterschool.com/ (дата обращения: 02.03.2022). – Текст : электронный.

7. iTrace – обучение письму. – Текст : электронный // App Store : [сайт]. – URL: https://apps.apple.com/ru/app/itrace/id645416621 (дата обращения: 02.03.2022).

8. Мастер письма для детей. – Текст : электронный // App Store : [сайт]. – URL: https://apps.apple.com/us/app/writing-wizard-for-kids/id631446426 (дата обращения: 02.03.2022).

9. iWriteWords. – Текст : электронный // App Store : [сайт]. – URL: https://apps.apple.com/ru/app/iwritewords-handwriting-game/id307025309 (дата обращения: 02.03.2022).

10. Буквонямки. – Текст : электронный // KinderMatica : [сайт]. – URL: https://ru.kindermatica.com/our-apps/bukvonyamki.html (дата обращения: 02.03.2022).

11. Буковки: Дети учатся читать. – Текст : электронный // Bukovki : [сайт]. – URL: https://bukovki.su/ (дата обращения: 02.03.2022).

12. Kids Learn to Write. – Текст : электронный // MicrosoftStore : [сайт]. – URL: https://www.microsoft.com/ru-ru/p/kids-learn-to-write/9nl4mgg11rb6?activetab=pivot:regionofsystemrequirementstab (дата обращения: 02.03.2022).

13. Visual Studio Community. – Текст : электронный // Microsoft : [сайт]. – URL: https://visualstudio.microsoft.com/ru (дата обращения: 05.03.2022).

14. Общие сведения ASP.NET Core MVC. – Текст : электронный // Microsoft : [сайт]. – 2022. – URL: https://docs.microsoft.com/ru-ru/aspnet/core/mvc (дата обращения: 05.03.2022).

15. SQL Server. – Текст : электронный // Microsoft : [сайт]. – URL: https://www.microsoft.com/ru-ru/sql-server (дата обращения: 05.03.2022).

16. Entity Framework Core. – Текст : электронный // Microsoft : [сайт]. – 2022. – URL: https://docs.microsoft.com/ru-ru/ef/core/ (дата обращения: 05.03.2022).

17. Что такое Code First?. – Текст : электронный // SolutionsIT : [сайт]. – 2021. – URL: https://solutions-it.co.il/code-first/ (дата обращения: 05.03.2022).

18. Краткий обзор языка C#. – Текст : электронный // Microsoft : [сайт]. – 2022. – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/ (дата обращения: 07.03.2022).

19. JavaScript. – Текст : электронный // SkillFactory.Блог : [сайт]. – 2021. – URL: https://blog.skillfactory.ru/glossary/javascript/ (дата обращения: 07.03.2022).

20. HTML5. – Текст : электронный // htmlbook.ru : [сайт]. – URL: http://htmlbook.ru/html5 (дата обращения: 07.03.2022).

21. Sass. – Текст : электронный // SkillFactory.Блог : [сайт]. – 2022. – URL: https://blog.skillfactory.ru/glossary/sass/ (дата обращения: 07.03.2022).

22. Нильсен, Я. Web-дизайн: удобство использования Web-сайтов / Я. Нильсен, Х. Лоранжер. – Москва : Вильямс, 2007. – 368 с. – ISBN: 978-5-8459-1222-0.

23. Ramus Educational. – Текст : электронный // software.informer : [сайт]. – URL: https://ramus-educational.software.informer.com (дата обращения: 15.03.2022).

24. Жизненный цикл ПО. Каскадная модель (Waterfall). – Текст : электронный // XBSoftware : [сайт]. – 2014. – URL: https://xbsoftware.ru/blog/zhiznennyj-tsykl-po-kaskadnaya-model-waterfall/ (дата обращения: 30.04.2022).

ПРИЛОЖЕНИЕ А
Сравнительная таблица аналогов

Таблица А.1 – Сравнительный анализ программных систем для обучения письму

	Платформа	**ОС**	**Способ ввода**	**Предмет обучения**	**Язык**	**Статистика**	**Звуковое сопровождение**	**Дополнительные возможности**
Letter School	мобильное	Android, iOS	пальцем, стилусом	буквы, цифры, фигуры	английский	отсутствует	присутствует	курсивное начертание, выбор стиля начертания
iTrace	мобильное, десктопное	iOS, MacOS	пальцем, пальцами, стилусом	буквы, слова, цифры	английский, русский	присутствует	присутствует	ориентация для левшей, курсивное начертание, выбор цвета, стиля начертания, призы за задание
Мастер письма для детей	мобильное	Android, iOS	пальцем, стилусом	буквы, слова, фигуры, цифры	английский	неполная: история ввода	присутствует	ориентация для левшей, наклейки, игры, выбор стиля начертания
iWriteWords	мобильное	iOS	пальцем, стилусом	буквы, слова, цифры	английский	отсутствует	присутствует	ассоциативные картинки, пользовательская настройка букв, слов, картинок
Буквоямки	мобильное	iOS	пальцем, стилусом	буквы	русский	присутствует	присутствует	выбор персонажа, мини-игры, индивидуальные настройки обучения
Буковки	мобильное	Android, iOS	пальцами	буквы, слова	русский	отсутствует	присутствует	стикеры, установка времени использования
Kids Learn to Write	десктопное, мобильное	Android, Windows	палацем, стилусом, мышкой	буквы, числа, фигуры	английский	отсутствует	присутствует	стикеры, курсивное начертание, выбор цвета начертания

ПРИЛОЖЕНИЕ Б
Исходный код программы

Листинг Б.1 – Файл Areas/Admin/Controllers/HomeController.css

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;

namespace diplom.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class HomeController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными приложения
        public HomeController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        // объявление метода для реализации панели администратора
        public IActionResult Index()
        {
            return View(dataManager.ServiceItems.GetServiceItems());
        }
    }
}
```

Листинг Б.2 – Файл Areas/Admin/Controllers/ServiceItemsController.css

```
using System;
using System.IO;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;
using diplom.Domain.Entities;
using diplom.Service;

namespace diplom.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class ServiceItemsController : Controller
    {
        private readonly DataManager dataManager;
        private readonly IWebHostEnvironment hostingEnvironment;
        public ServiceItemsController(DataManager dataManager, IWebHostEnvironment
hostingEnvironment)
        {
            this.dataManager = dataManager;
            this.hostingEnvironment = hostingEnvironment;
        }
        // объявление метода редактиования записей базы слов приложения
        public IActionResult Edit(Guid id)
        {
            var entity = id == default ? new ServiceItem() :
dataManager.ServiceItems.GetServiceItemById(id);
            return View(entity);
        }
        [HttpPost]
        public IActionResult Edit(ServiceItem model, IFormFile titleImageFile)
        {
            if (ModelState.IsValid)
            {
                if (titleImageFile != null)
                {
                    model.TitleImagePath = titleImageFile.FileName;
                    using (var stream = new
FileStream(Path.Combine(hostingEnvironment.WebRootPath, "images/",
titleImageFile.FileName), FileMode.Create))
```

```
                    {
                        titleImageFile.CopyTo(stream);
                    }
                }
                dataManager.ServiceItems.SaveServiceItem(model);
                return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());
            }
            return View(model);
        }
        // объявление метода для удаления записей из базы данных слов
        [HttpPost]
        public IActionResult Delete(Guid id)
        {
            dataManager.ServiceItems.DeleteServiceItem(id);
            return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());
        }
    }
}
```

Листинг Б.3 – Файл Areas/Admin/Controllers/TextFieldsController.css

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;
using diplom.Domain.Entities;
using diplom.Service;

namespace diplom.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class TextFieldsController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными приложения
        public TextFieldsController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        // объявление метода редактироввания страниц приложения
        public IActionResult Edit(string codeWord)
        {
            var entity = dataManager.TextFields.GetTextFieldByCodeWord(codeWord);
            return View(entity);
        }

        [HttpPost]
        public IActionResult Edit(TextField model)
        {
            if (ModelState.IsValid)
            {
                dataManager.TextFields.SaveTextField(model);
                return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());
            }
            return View(model);
        }
    }
}
```

Листинг Б.4 – Файл Areas/Admin/Views/Home/Index.cshtml

```
@model IQueryable<ServiceItem>
@{
    string strTitle = "Панель администратора";
    ViewBag.Title = strTitle;
}

<div>
    <h2>@strTitle</h2>
    @*отображение блока редактирования базы слов приложения*@
```

```
    <div>
        <h3>База слов приложения</h3>
        <div class="div-box">
            <a asp-area="Admin" asp-controller="ServiceItems" asp-action="Edit" asp-
route-id="">Добавить слово</a>
        </div>
        @if (Model.Any())
        {
            <div>
                @foreach (ServiceItem entity in Model)
                {
                    <div>
                        <a asp-area="Admin" asp-controller="ServiceItems" asp-
action="Edit" asp-route-id="@entity.Id">редактировать</a>
                        |
                        <form style="display: inline-block;" id="form-@entity.Id" asp-
area="Admin" asp-controller="ServiceItems" asp-action="Delete" method="post">
                            <input type="hidden" name="id" value="@entity.Id">
                            <a href="#" onclick="document.getElementById('form-
@entity.Id').submit();">удалить</a>
                        </form>
                        |
                        <a asp-area="" asp-controller="Services" asp-action="Index"
asp-route-id="@entity.Id">
                            @($"{entity.Title}")
                        </a>
                    </div>
                }
            </div>
        }
    </div>
    @*отображение блока редактирования страниц приложения*@
    <div class="div-box">
        <h3>Редактировать страницы сайта</h3>
        <a asp-area="Admin" asp-controller="TextFields" asp-action="Edit" asp-route-
codeWord="PageIndex">Главная</a>
        <a asp-area="Admin" asp-controller="TextFields" asp-action="Edit" asp-route-
codeWord="PageContacts">Контакты</a>
    </div>
    @*отображение кнопки выхода из аккаунта*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Logout" method="post">
            <input type="submit" value="Выйти" />
        </form>


    </div>
</div>
```

Листинг Б.5 – Файл Areas/Admin/Views/ServiceItem/Edit.cshtml

```
@model ServiceItem
@{
    string strTitle = "Редактировать запись";
    ViewBag.Title = strTitle;
}
@*подключение скрипта js*@
<script src="~/js/ckeditor/ckeditor.js"></script>

<div>
    <h2>@strTitle</h2>
    @*отображение форм для редактирования записей базы слов*@
    <div>
        <form asp-area="Admin" asp-controller="ServiceItems" asp-action="Edit"
method="post" enctype="multipart/form-data">
            <input type="hidden" asp-for="Id" />
            <input type="hidden" asp-for="DateAdded" />
            <input type="hidden" asp-for="TitleImagePath" />

            <div asp-validation-summary="All"></div>
```

```
            <div class="div-box">
                <label asp-for="Title"></label>
                <input asp-for="Title" />
                <span asp-validation-for="Title"></span>
            </div>
            <div class="div-box">
                <label asp-for="Subtitle"></label>
                <input asp-for="Subtitle" />
                <span asp-validation-for="Subtitle"></span>
            </div>
            <div class="div-box">
                <label asp-for="Text"></label>
                <textarea asp-for="Text"></textarea>
                <span asp-validation-for="Text"></span>
            </div>
            <div class="div-box">
                <label asp-for="TitleImagePath"></label>
                <input type="file" name="titleImageFile" id="titleImageFile" />
                <div>
                    <img class="img-block" src="~/images/@Model.TitleImagePath" />
                </div>
            </div>
            <div class="div-box">
                <label asp-for="MetaTitle"></label>
                <input asp-for="MetaTitle" />
                <span asp-validation-for="MetaTitle"></span>
            </div>
            <div class="div-box">
                <label asp-for="MetaDescription"></label>
                <input asp-for="MetaDescription" />
                <span asp-validation-for="MetaDescription"></span>
            </div>
            <div class="div-box">
                <label asp-for="MetaKeywords"></label>
                <input asp-for="MetaKeywords" />
                <span asp-validation-for="MetaKeywords"></span>
            </div>
            @*отображение кнопки сохранения*@
            <input type="submit" value="Сохранить" />
        </form>
    </div>
</div>


@*блок динамического заполнения форм данными*@
<script>
    window.onload = function() {
        var newCKEdit = CKEDITOR.replace('@Html.IdFor(x=>x.Text)');
        newCKEdit.updateElement();
    }
</script>
```

Листинг Б.6 – Файл Areas/Admin/Views/TextFields/Edit.cshtml

```
@model TextField
@{
    string strTitle = "Редактировать запись";
    ViewBag.Title = strTitle;
}
@*подключение скрипта js*@
<script src="~/js/ckeditor/ckeditor.js"></script>

<div>
    <h2>@strTitle</h2>
    <div>
        @*отображение форм текстовых полей для страниц приложения*@
        <form asp-area="Admin" asp-controller="TextFields" asp-action="Edit"
method="post" enctype="multipart/form-data">
            <input type="hidden" asp-for="Id" />
            <input type="hidden" asp-for="DateAdded" />
```

```
            <input type="hidden" asp-for="CodeWord" />

            <div asp-validation-summary="All"></div>
            <div class="div-box">
                <label asp-for="Title"></label>
                <input asp-for="Title" />
                <span asp-validation-for="Title"></span>
            </div>
            <div class="div-box">
                <label asp-for="Text"></label>
                <textarea asp-for="Text"></textarea>
                <span asp-validation-for="Text"></span>
            </div>
            <div class="div-box">
                <label asp-for="MetaTitle"></label>
                <input asp-for="MetaTitle" />
                <span asp-validation-for="MetaTitle"></span>
            </div>
            <div class="div-box">
                <label asp-for="MetaDescription"></label>
                <input asp-for="MetaDescription" />
                <span asp-validation-for="MetaDescription"></span>
            </div>
            <div class="div-box">
                <label asp-for="MetaKeywords"></label>
                <input asp-for="MetaKeywords" />
                <span asp-validation-for="MetaKeywords"></span>
            </div>
            @*отображение кнопки сохраения*@
            <input type="submit" value="Сохранить" />
        </form>
    </div>
</div>


@*блок динамического заполнения форм данными*@
<script>
    window.onload = function() {
        var newCKEdit = CKEDITOR.replace('@Html.IdFor(x=>x.Text)');
        newCKEdit.updateElement();
    }
</script>
```

Листинг Б.7 – Файл Areas/Admin/Views/_ViewImport.cshtml

```
@*подключение частей приложения*@
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using diplom.Models
@using diplom.Service
@using diplom.Domain.Entities
@using diplom.Models.ViewComponents
```

Листинг Б.8 – Файл Areas/Admin/Views/_ViewStart.cshtml

```
@*подключение представления Layout*@
@{
    Layout = "_Layout";
}
```

Листинг Б.9 – Файл Areas/User/Controllers/ChildController.cs

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;

namespace diplom.Areas.User.Controllers
{
    [Area("User")]
    public class ChildController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public ChildController(DataManager dataManager)
```

```
        {
            this.dataManager = dataManager;
        }
        //объявление метода для страницы меню ребенка
        public IActionResult Index()
        {
            return View(dataManager.ServiceItems.GetServiceItems());
        }
    }
}
```

Листинг Б.10 – Файл Areas/User/Controllers/HomeController.cs

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;

namespace diplom.Areas.User.Controllers
{
    [Area("User")]
    public class HomeController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public HomeController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        //объявление метода для страницы пользовательских слов
        public IActionResult Index()
        {
            return View(dataManager.ServiceItems.GetServiceItems());
        }
        //объявление метода для перехода на страницу меню ребенка
        public IActionResult Statistics()
        {
            return RedirectPermanent("~/User/Child/Index");
        }
    }
}
```

Листинг Б.11 – Файл Areas/User/Controllers/ParentController.cs

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;

namespace diplom.Areas.User.Controllers
{
    [Area("User")]
    public class ParentController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public ParentController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        //объявление метода для страницы меню родителя
        public IActionResult Index()
        {
            return View(dataManager.ServiceItems.GetServiceItems());
        }
    }
}
```

Листинг Б.12 – Файл Areas/User/Controllers/ServiseItemsController.cs

```
using System;
using System.IO;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;
```

```
using diplom.Domain.Entities;
using diplom.Service;

namespace diplom.Areas.User.Controllers
{
    [Area("User")]
    public class ServiceItemsController : Controller
    {
        private readonly DataManager dataManager;
        private readonly IWebHostEnvironment hostingEnvironment;
        public ServiceItemsController(DataManager dataManager, IWebHostEnvironment
hostingEnvironment)
        {
            this.dataManager = dataManager;
            this.hostingEnvironment = hostingEnvironment;
        }
        // объявление метода редактиования записей базы слов приложения
        public IActionResult Edit(Guid id)
        {
            var entity = id == default ? new ServiceItem() :
dataManager.ServiceItems.GetServiceItemById(id);
            return View(entity);
        }
        [HttpPost]
        public IActionResult Edit(ServiceItem model, IFormFile titleImageFile,
IFormFile titleAudioFile)
        {
            if (ModelState.IsValid)
            {
                if (titleImageFile != null)
                {
                    model.TitleImagePath = titleImageFile.FileName;
                    using (var stream = new
FileStream(Path.Combine(hostingEnvironment.WebRootPath, "images/",
titleImageFile.FileName), FileMode.Create))
                    {
                        titleImageFile.CopyTo(stream);
                    }
                }
                dataManager.ServiceItems.SaveServiceItem(model);
                return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());

                if (titleAudioFile != null)
                {
                    model.AudioPath = titleAudioFile.FileName;
                    using (var stream = new
FileStream(Path.Combine(hostingEnvironment.WebRootPath, "audio/",
titleAudioFile.FileName), FileMode.Create))
                    {
                        titleAudioFile.CopyTo(stream);
                    }
                }
                dataManager.ServiceItems.SaveServiceItem(model);
                return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());
            }
            return View(model);
        }
        // объявление метода удаления записей базы слов приложения
        [HttpPost]
        public IActionResult Delete(Guid id)
        {
            dataManager.ServiceItems.DeleteServiceItem(id);
            return RedirectToAction(nameof(HomeController.Index),
nameof(HomeController).CutController());
        }
    }
}
```

Листинг Б.13 – Файл Areas/User/Views/Child/Index.cshtml

```
@model IQueryable<ServiceItem>
@{
    string strTitle = "Меню ребенка";
    ViewBag.Title = strTitle;
}

<div>
    <h2>@strTitle</h2>
    @*блок отображения кнопок для выбора режимов обучения*@
    <div>
        <h3>Режимы обучения</h3>
       <div class="div-box">
        <form asp-area="" asp-controller="Study" asp-action="Index" method="post">
            <input type="submit" value="Побуквенная печать" />
        </form>
    </div>
    </div>
    @*блок отображения кнопки выхода из аккаунта*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Logout" method="post">
            <input type="submit" value="Выйти" />
        </form>
    </div>
</div>
```

Листинг Б.14 – Файл Areas/User/Views/Home/Index.cshtml

```
@model IQueryable<ServiceItem>

<div>

    <div>
        @*блок отображения элементов редактирования базы слов приложения*@
        <h3>Пользовательские слова</h3>
        <div class="div-box">
            <a asp-area="User" asp-controller="ServiceItems" asp-action="Edit" asp-
route-id="">Добавить слово</a>
        </div>
        @if (Model.Any())
        {
            <div>
                @foreach (ServiceItem entity in Model)
                {
                    <div>
                        <a asp-area="User" asp-controller="ServiceItems" asp-
action="Edit" asp-route-id="@entity.Id">редактировать</a>
                        |
                        <form style="display: inline-block;" id="form-@entity.Id" asp-
area="User" asp-controller="ServiceItems" asp-action="Delete" method="post">
                            <input type="hidden" name="id" value="@entity.Id">
                            <a href="#" onclick="document.getElementById('form-
@entity.Id').submit();">удалить</a>
                        </form>
                        |
                        <a asp-area="" asp-controller="Services" asp-action="Index"
asp-route-id="@entity.Id">
                            @($"{entity.Title}")
                        </a>
                    </div>
                }
            </div>
        }
    </div>
    @*блок отображения кнопки возврата в меню родителя*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="BackParent"
method="post">
            <input type="submit" value="Меню родителя" />
```

```
        </form>
    </div>
</div>
```

Листинг Б.15 – Файл Areas/User/Views/Parent/Index.cshtml

```
@model IQueryable<ServiceItem>
@{
    string strTitle = "Меню родителя";
    ViewBag.Title = strTitle;
}

<div>
    <h2>@strTitle</h2>
    @*блок отображения кнопок выбора действия на странице меню родителя*@
    <div>
       <div class="div-box">
        <form asp-area="" asp-controller="Settings" asp-action="Index" method="post">
            <input type="submit" value="Пользовательские слова" />
        </form>
    </div>
    <div class="div-box">
        <form asp-area="" asp-controller="Settings" asp-action="Statistics"
method="post">
            <input type="submit" value="Статистика" />
        </form>
    </div>
    </div>
    @*блок отображения кнопки выхода из аккаунта*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Logout" method="post">
            <input type="submit" value="Выйти" />
        </form>
    </div>
</div>
```

Листинг Б.16 – Файл Areas/User/Views/ServiceItems/Edit.cshtml

```
@model ServiceItem
@{
    string strTitle = "Редактировать запись";
    ViewBag.Title = strTitle;
}
@*подключение скрипта js*@
<script src="~/js/ckeditor/ckeditor.js"></script>

<div>
    <h2>@strTitle</h2>
    @*отображение форм для редактирования записей базы слов*@
    <div>
        <form asp-area="User" asp-controller="ServiceItems" asp-action="Edit"
method="post" enctype="multipart/form-data">
            <input type="hidden" asp-for="Id" />
            <input type="hidden" asp-for="DateAdded" />
            <input type="hidden" asp-for="TitleImagePath" />

            <div asp-validation-summary="All"></div>
            <div class="div-box">
                <label asp-for="Title"></label>
                <input asp-for="Title" />
                <span asp-validation-for="Title"></span>
            </div>
            <div class="div-box">
                <label asp-for="TitleImagePath"></label>
                <input type="file" name="titleImageFile" id="titleImageFile" />
                <div>
                    <img class="img-block" src="~/images/@Model.TitleImagePath" />
                </div>
            </div>
            <div class="div-box">
                <label asp-for="AudioPath"></label>
```

```
                <input type="file" name="titleAudioFile" id="titleAudioFile" />
                <div>
                    <audio class="src" src="~/audio/@Model.AudioPath" />
                </div>
            </div>
            @*отображение кнопки сохранения*@
            <input type="submit" value="Сохранить" />
        </form>
    </div>
</div>


@*блок динамического заполнения форм данными*@
<script>
    window.onload = function() {
        var newCKEdit = CKEDITOR.replace('@Html.IdFor(x=>x.Text)');
        newCKEdit.updateElement();
    }
</script>
```

Листинг Б.17 – Файл Areas/User/Views/_ViewImport.cshtml

```
@*подключение частей приложения*@
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using diplom.Models
@using diplom.Service
@using diplom.Domain.Entities
@using diplom.Models.ViewComponents
```

Листинг Б.18 – Файл Areas/User/Views/_ViewStart.cshtml

```
@*подключение представления Layout*@
@{
    Layout = "_Layout";
}
```

Листинг Б.19 – Файл Controllers/AccountController.cs

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.AspNetCore.Mvc;
using diplom.Models;
using diplom.Domain;


namespace diplom.Controllers
{
    [Authorize]
    public class AccountController : Controller
    {

        private readonly UserManager<IdentityUser> userManager;
        private readonly SignInManager<IdentityUser> signInManager;
        // объявление экземпляров класса для работы с аутентификацией
        public AccountController(UserManager<IdentityUser> userMgr,
SignInManager<IdentityUser> signinMgr)
        {
            userManager = userMgr;
            signInManager = signinMgr;
        }
        // объявление метода для создания новой пустой формы для входа в систему
        [AllowAnonymous]
        public IActionResult Login(string returnUrl)
        {
            ViewBag.returnUrl = returnUrl;
            return View(new LoginViewModel());
        }
        [HttpPost]

        // объявление метода для создания новой пустой формы для регистрации
```

```
        [AllowAnonymous]
        public IActionResult Registration(string returnUrl)
        {
            ViewBag.returnUrl = returnUrl;
            return View(new RegistrationViewModel());
        }
        [HttpPost]
        // объявление метода процедуры входа в систему
        [AllowAnonymous]
        public async Task<IActionResult> Login(LoginViewModel model, string returnUrl)
        {

            bool checkParent = model.Parent;
            if (ModelState.IsValid)
            {
                //поиск пользователя по введенному в форму имени
                IdentityUser user = await userManager.FindByNameAsync(model.UserName);
                if (user != null)
                {
                    await signInManager.SignOutAsync();
                    //попытка входа в аккаунт пользователя с данными пароля из формы
                    Microsoft.AspNetCore.Identity.SignInResult result = await
signInManager.PasswordSignInAsync(user, model.Password, model.RememberMe, false);

                    if (result.Succeeded)
                    {
                        // переход на разные меню в зависимости от значения чек-бокса
родитель?
                        if (checkParent == true)
                        {
                            return RedirectPermanent("~/User/Parent/Index");
                        }
                        else { return RedirectPermanent("~/User/Child/Index"); }
                    }
                }
                //вывод ошибки при неправильном заполнении данных формы
                ModelState.AddModelError(nameof(LoginViewModel.UserName), "Неверный
логин или пароль");
            }
            return View(model);
        }


        // объявление метода процедуры регистрации в системе
        public async Task<IActionResult> Registration(RegistrationViewModel model,
string returnUrl)
        {
            //получение нового id от системы guid
            string id = Guid.NewGuid().ToString();

            //объявление значения guid для типа пользователя user
            string IdentityUserRole = "26a4b2e5-5cbc-4f99-8989-01f54f897a36";

            // проверки ввхода данных логина и пароля при регистрации
                IdentityUser user = await userManager.FindByNameAsync(model.UserName);
                if (user != null)
                {
                    ViewBag.Error = "Данный логин уже занят!";
                }

                string checkPassword = model.Password;
                if (checkPassword.Length < 6)
                {
                    ViewBag.Conditions = "Пароль должен содержать не менее 6
символов";
                }

            // регистрация
```

```
            if (ModelState.IsValid)
            {

                    // добавление нового пользователя
                    IdentityUser Newuser = new IdentityUser {id, IdentityUserRole,
UserName = model.UserName, Email = model.Email, EmailConfirmed = true };

                    // проверка входа в аккаунт пользователя регистрации пользователя
                    var result = await userManager.CreateAsync(Newuser,
model.Password);
                    if (result.Succeeded)
                    {
                        // установка куки и переход на главную страницу приложения
                        await signInManager.SignInAsync(user, false);
                        return RedirectToAction("Index", "Home");
                    }
            }
            return View(model);
        }
        // объявление метода выхода из системы
        [Authorize]
        public async Task<IActionResult> Logout()
        {
            await signInManager.SignOutAsync();
            return RedirectToAction("Index", "Home");
        }
        // объявление метода выхода в меню для ребенка
        [Authorize]
        public IActionResult Back()
        {
            return RedirectPermanent("~/User/Child/Index");
        }
        // объявление метода выхода в меню для родителя
        [Authorize]
        public IActionResult BackParent()
        {
            return RedirectPermanent("~/User/Parent/Index");
        }
    }
}
```

Листинг Б.20 – Файл Controllers/HomeController.cs

```
namespace diplom.Controllers
{
    public class HomeController : Controller
    {
        private readonly DataManager dataManager;

        // объявление экземпляра класса для работы с данными
        public HomeController (DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        // объявление метода работы с текстовыми полями страницы о нас
        public IActionResult Index()
        {
            return View(dataManager.TextFields.GetTextFieldByCodeWord("PageIndex"));
        }

        // объявление метода работы с текстовыми полями страницы контакты
        public IActionResult Contacts()
        {
            return
View(dataManager.TextFields.GetTextFieldByCodeWord("PageContacts"));
        }
    }
}
```

Листинг Б.21 – Файл Controllers/ServiceController.cs

```
namespace diplom.Controllers
{
    public class ServicesController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public ServicesController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        // объявление метода для вывода списков слов или отдельного слова
        public IActionResult Index(Guid id)
        {
            if (id != default)
            {
                return View("Show", dataManager.ServiceItems.GetServiceItemById(id));
            }

            ViewBag.TextField =
dataManager.TextFields.GetTextFieldByCodeWord("PageServices");
            return View(dataManager.ServiceItems.GetServiceItems());
        }
        // объявление метода для вывода списка выученных слов
        public IActionResult Learned(Guid id)
        {
            if (id != default)
            {
                return View("Show", dataManager.ServiceItems.GetServiceItemById(id));
            }

            ViewBag.TextField =
dataManager.TextFields.GetTextFieldByCodeWord("PageServices2");
            return View(dataManager.ServiceItems.GetServiceItems());
        }

        // объявление метода для вывода списка невыученных слов
        public IActionResult Unlearned(Guid id)
        {
            if (id != default)
            {
                return View("Show", dataManager.ServiceItems.GetServiceItemById(id));
            }

            ViewBag.TextField =
dataManager.TextFields.GetTextFieldByCodeWord("PageServices3");
            return View(dataManager.ServiceItems.GetServiceItems());
        }
    }
}
```

Листинг Б.22 – Файл Controllers/SettingsController.cs

```
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;

namespace diplom.Controllers
{
    public class SettingsController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public SettingsController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }
        // объявление метода перехода на страницу пользовательских слов
        public IActionResult Index()
        {
```

```
            return View(dataManager.ServiceItems.GetServiceItems());
        }
        // объявление метода перехода на страницу статистики
        public IActionResult Statistics()
        {
            return View(dataManager.ServiceItems.GetServiceItems());
        }
    }
}
```

Листинг Б.23 – Файл Controllers/StudyController.cs

```
using System;
using Microsoft.AspNetCore.Mvc;
using diplom.Domain;
using diplom.Domain.Entities;
using diplom.Models;

namespace diplom.Controllers
{
    public class StudyController : Controller
    {
        private readonly DataManager dataManager;
        // объявление экземпляра класса для работы с данными
        public StudyController(DataManager dataManager)
        {
            this.dataManager = dataManager;
        }

        // объявление метода рандомного выбора id
        public IActionResult Choise()
        {
            return RedirectToAction("Index",
dataManager.ServiceItems.RandtServiceItemById());
        }

        // объявление метода вывода страницы поощерения
        public IActionResult Reward()
        {
           return View("Gift");
        }


        public IActionResult Index(Mode1ViewModel model, ServiceItem model1, Guid id)
        {
            // запись вводимого в форму слова
            string checkWord = model.Word;

            if (checkWord != null)
            {
                // поиск слова по базе данных
                ViewBag.Myword =
dataManager.ServiceItems.GetServiceItemByTitle(checkWord);

                if (ViewBag.Myword != null)
                {
                    // вывод поощерения
                    return RedirectToAction("Reward", "Study");
                }
                else {
                    //вывод сообщения о ошибке
                    ViewBag.Mistake = "Упс,ошибка! Попробуй еще раз!";
                }
            }

            return View();
        }

        // объявление метода перехода к следующему слову
        public IActionResult Next()
```

```
        {
            return RedirectToAction("Index", "Choise");
        }
    }
}
```

Листинг Б.24 – Файл Domain/Entities/EntityBase.cs

```
using System;
using System.ComponentModel.DataAnnotations;


namespace diplom.Domain.Entities
{
    public abstract class EntityBase
    {
        //подключение считывания времени
        protected EntityBase() => DateAdded = DateTime.UtcNow;
        //запрос нового id
        [Required]
        public Guid Id { get; set; }
        //отображение форм полей базы ванных
        [Display(Name = "Название (заголовок)")]
        public virtual string Title { get; set; }

        [Display(Name = "Краткое описание")]
        public virtual string Subtitle { get; set; }

        [Display(Name = "Полное описание")]
        public virtual string Text { get; set; }

        [Display(Name = "Титульная картинка")]
        public virtual string TitleImagePath { get; set; }

        [Display(Name = "Озвучка")]
        public virtual string AudioPath { get; set; }

        [Display(Name = "SEO метатег Title")]
        public string MetaTitle { get; set; }

        [Display(Name = "SEO метатег Description")]
        public string MetaDescription { get; set; }

        [Display(Name = "SEO метатег Keywords")]
        public string MetaKeywords { get; set; }

        [DataType(DataType.Time)]
        public DateTime DateAdded { get; set; }
    }
}
```

Листинг Б.25 – Файл Domain/Entities/ServiceItem.cs

```
using System.ComponentModel.DataAnnotations;

namespace diplom.Domain.Entities
{ // вывод форм заполения базы данных слов
    public class ServiceItem : EntityBase
    {
        [Required(ErrorMessage = "Заполните слово")]
        [Display(Name = "Название")]
        public override string Title { get; set; }

    }
}
```

Листинг Б.26 – Файл Domain/Entities/TextField.cs

```
using System.ComponentModel.DataAnnotations;

namespace diplom.Domain.Entities
{
```

```
    public class TextField : EntityBase
    {
        [Required]
        public string CodeWord { get; set; }
        // вывод форм для заполения страниц приложения
        [Display(Name = "Название страницы (заголовок)")]
        public override string Title { get; set; } = "Заглавие страницы";

        [Display(Name = "Содержание страницы")]
        public override string Text { get; set; } = "Содержание страницы будет
заполняться администратором приложения";
    }
}
```

Листинг Б.27 – Файл Domain/Repositories/Abstract/IServiceItemsRepository.cs

```
using System;
using System.Linq;
using diplom.Domain.Entities;

namespace diplom.Domain.Repositories.Abstract
{
    public interface IServiceItemsRepository
    {
        // объявление методов работы с базой данных слов
        IQueryable<ServiceItem> GetServiceItems();
        ServiceItem GetServiceItemById(Guid id);
        ServiceItem RandtServiceItemById();
        ServiceItem GetServiceItemByTitle(string title);
        void SaveServiceItem(ServiceItem entity);
        void DeleteServiceItem(Guid id);
    }
}
```

Листинг Б.28 – Файл Domain/Repositories/Abstract/ITextFieldsRepository.cs

```
using System;
using System.Linq;
using diplom.Domain.Entities;

namespace diplom.Domain.Repositories.Abstract
{
    public interface ITextFieldsRepository
    { // объявление модов работы с текстовыми полями страниц приложения
        IQueryable<TextField> GetTextFields();
        TextField GetTextFieldById(Guid id);
        TextField GetTextFieldByCodeWord(string codeWord);
        void SaveTextField(TextField entity);
        void DeleteTextField(Guid id);
    }
}
```

Листинг Б.29 – Файл Domain/Repositories/EntityFramework/ EFServiceItemsRepository.cs

```
using diplom.Domain.Entities;
using diplom.Domain.Repositories.Abstract;

namespace diplom.Domain.Repositories.EntityFramework
{ // описание реализации методов работы с базой слов приложения
    public class EFServiceItemsRepository : IServiceItemsRepository
    {
        private readonly AppDbContext context;
        public EFServiceItemsRepository(AppDbContext context)
        {
            this.context = context;
        }

        public IQueryable<ServiceItem> GetServiceItems()
        {
            return context.ServiceItems;
```

```
        }
        //поиск слова по id
        public ServiceItem GetServiceItemById(Guid id)
        {
            return context.ServiceItems.FirstOrDefault(x => x.Id == id);
        }
        //рандомный выбор слова
        public ServiceItem RandtServiceItemById()
        {
            int CountId = context.ServiceItems.Count();
           return (context.ServiceItems.[new Random().Next(CountId).ToString()](x =>
x.Id));

        }
        //поиск слова по названию
        public ServiceItem GetServiceItemByTitle(string title)
        {
            return context.ServiceItems.FirstOrDefault(x => x.Title == title);
        }
        //сохранение
        public void SaveServiceItem(ServiceItem entity)
        {
            if (entity.Id == default)
                context.Entry(entity).State = EntityState.Added;
            else
                context.Entry(entity).State = EntityState.Modified;
            context.SaveChanges();
        }
        // удаление слова
        public void DeleteServiceItem(Guid id)
        {
            context.ServiceItems.Remove(new ServiceItem() { Id = id });
            context.SaveChanges();
        }
    }
}
```

Листинг Б.30 – Файл Domain/Repositories/EntityFramework/EFTextFieldsRepository.cs

```
using System;
using System.Linq;
using Microsoft.EntityFrameworkCore;
using diplom.Domain.Entities;
using diplom.Domain.Repositories.Abstract;

namespace diplom.Domain.Repositories.EntityFramework
{ // описание реализации методов работы с текстовыми полями страниц прилодения
    public class EFTextFieldsRepository : ITextFieldsRepository
    {
        private readonly AppDbContext context;
        public EFTextFieldsRepository(AppDbContext context)
        {
            this.context = context;
        }
        //получить текстовое поле
        public IQueryable<TextField> GetTextFields()
        {
            return context.TextFields;
        }
        //получить текстовое поле по id
        public TextField GetTextFieldById(Guid id)
        {
            return context.TextFields.FirstOrDefault(x => x.Id == id);
        }
        //получить текстовое поле по ключевому слову
        public TextField GetTextFieldByCodeWord(string codeWord)
        {
            return context.TextFields.FirstOrDefault(x => x.CodeWord == codeWord);
        }
```

```
        //сохранить текстовое поле
        public void SaveTextField(TextField entity)
        {
            if (entity.Id == default)
                context.Entry(entity).State = EntityState.Added;
            else
                context.Entry(entity).State = EntityState.Modified;
            context.SaveChanges();
        }
        //удалить текстовое поле
        public void DeleteTextField(Guid id)
        {
            context.TextFields.Remove(new TextField() { Id = id });
            context.SaveChanges();
        }
    }
}
```

Листинг Б.31 – Файл Domain/AppDbContext.cs

```
using System;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;
using diplom.Domain.Entities;

namespace diplom.Domain
{
    public class AppDbContext : IdentityDbContext<IdentityUser>
    {
        public AppDbContext(DbContextOptions<AppDbContext> options) : base(options) {
}

        public DbSet<TextField> TextFields { get; set; }
        public DbSet<ServiceItem> ServiceItems { get; set; }

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            base.OnModelCreating(modelBuilder);

            modelBuilder.Entity<IdentityRole>().HasData(new IdentityRole
            {
                Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",
                Name = "admin",
                NormalizedName = "ADMIN"
            });

            modelBuilder.Entity<IdentityRole>().HasData(new IdentityRole
            {
                Id = "14769d9a-e6da-41d9-b21b-8980e64d9624",
                Name = "user",
                NormalizedName = "USER"
            });

            modelBuilder.Entity<IdentityUser>().HasData(new IdentityUser
            {
                Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",
                UserName = "admin",
                NormalizedUserName = "ADMIN",
                Email = "my@email.com",
                NormalizedEmail = "MY@EMAIL.COM",
                EmailConfirmed = true,
                PasswordHash = new PasswordHasher<IdentityUser>().HashPassword(null,
"adminpassword"),
                SecurityStamp = string.Empty
            });

            modelBuilder.Entity<IdentityUser>().HasData(new IdentityUser
            {
                Id = "26a4b2e5-5cbc-4f99-8989-01f54f897a36",
```

```
                UserName = "child",
                NormalizedUserName = "CHILD",
                Email = "parent@email.com",
                NormalizedEmail = "PARENT@EMAIL.COM",
                EmailConfirmed = true,
                PasswordHash = new PasswordHasher<IdentityUser>().HashPassword(null,
"parentpassword"),
                SecurityStamp = string.Empty
            });

            modelBuilder.Entity<IdentityUserRole<string>>().HasData(new
IdentityUserRole<string>
            {
                RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab",
                UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8"
            });

            modelBuilder.Entity<IdentityUserRole<string>>().HasData(new
IdentityUserRole<string>
            {
                RoleId = "14769d9a-e6da-41d9-b21b-8980e64d9624",
                UserId = "26a4b2e5-5cbc-4f99-8989-01f54f897a36"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),
                CodeWord = "PageIndex",
                Title = "Главная"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),
                CodeWord = "PageServices",
                Title = "Список слов"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("0178b641-1f30-4cd9-b84d-d740c2bdde69"),
                CodeWord = "PageServices2",
                Title = "Выученые слова"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("15085f29-9df3-4e6b-98ee-6ce33d49b2e9"),
                CodeWord = "PageServices3",
                Title = "Невыученные слова"
            });

            modelBuilder.Entity<TextField>().HasData(new TextField
            {
                Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),
                CodeWord = "PageContacts",
                Title = "Контакты"
            });
        }
    }
}
```

Листинг Б.32 – Файл Domain/ DataManager.cs

```
using diplom.Domain.Repositories.Abstract;

namespace diplom.Domain
{
    public class DataManager
    {
```

```
        //реализация метода работы с данными приложения
        public ITextFieldsRepository TextFields { get; set; }
        public IServiceItemsRepository ServiceItems { get; set; }

        public DataManager(ITextFieldsRepository textFieldsRepository,
IServiceItemsRepository serviceItemsRepository)
        {
            TextFields = textFieldsRepository;
            ServiceItems = serviceItemsRepository;
        }
    }
}
```

Листинг Б.33 – Файл AppDbContextModelSnapshot.cs

```
// <auto-generated />
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Infrastructure;
using Microsoft.EntityFrameworkCore.Metadata;
using Microsoft.EntityFrameworkCore.Storage.ValueConversion;
using diplom.Domain;

namespace diplom.Migrations
{
    [DbContext(typeof(AppDbContext))]
    partial class AppDbContextModelSnapshot : ModelSnapshot
    {
        protected override void BuildModel(ModelBuilder modelBuilder)
        {
#pragma warning disable 612, 618
            modelBuilder
                .HasAnnotation("ProductVersion", "3.1.1")
                .HasAnnotation("Relational:MaxIdentifierLength", 128)
                .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>
                {
                    b.Property<string>("Id")
                        .HasColumnType("nvarchar(450)");

                    b.Property<string>("ConcurrencyStamp")
                        .IsConcurrencyToken()
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("Name")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);

                    b.Property<string>("NormalizedName")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);

                    b.HasKey("Id");

                    b.HasIndex("NormalizedName")
                        .IsUnique()
                        .HasName("RoleNameIndex")
                        .HasFilter("[NormalizedName] IS NOT NULL");

                    b.ToTable("AspNetRoles");

                    b.HasData(
                        new
                        {
                            Id = "44546e06-8719-4ad8-b88a-f271ae9d6eab",
                            ConcurrencyStamp = "ccfd5022-02bf-4b4d-b513-bae9efaa896d",
                            Name = "admin",
                            NormalizedName = "ADMIN"
```

```
                        },
                        new
                        {
                            Id = "14769d9a-e6da-41d9-b21b-8980e64d9624",
                            ConcurrencyStamp = "6e8460a3-72ea-4af1-b75b-dbfff4b80af3",
                            Name = "user",
                            NormalizedName = "USER"
                        });
                });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("ClaimType")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("ClaimValue")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("RoleId")
                        .IsRequired()
                        .HasColumnType("nvarchar(450)");

                    b.HasKey("Id");

                    b.HasIndex("RoleId");

                    b.ToTable("AspNetRoleClaims");
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUser", b =>
                {
                    b.Property<string>("Id")
                        .HasColumnType("nvarchar(450)");

                    b.Property<int>("AccessFailedCount")
                        .HasColumnType("int");

                    b.Property<string>("ConcurrencyStamp")
                        .IsConcurrencyToken()
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("Email")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);

                    b.Property<bool>("EmailConfirmed")
                        .HasColumnType("bit");

                    b.Property<bool>("LockoutEnabled")
                        .HasColumnType("bit");

                    b.Property<DateTimeOffset?>("LockoutEnd")
                        .HasColumnType("datetimeoffset");

                    b.Property<string>("NormalizedEmail")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);

                    b.Property<string>("NormalizedUserName")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);
```

```
                    b.Property<string>("PasswordHash")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("PhoneNumber")
                        .HasColumnType("nvarchar(max)");

                    b.Property<bool>("PhoneNumberConfirmed")
                        .HasColumnType("bit");

                    b.Property<string>("SecurityStamp")
                        .HasColumnType("nvarchar(max)");

                    b.Property<bool>("TwoFactorEnabled")
                        .HasColumnType("bit");

                    b.Property<string>("UserName")
                        .HasColumnType("nvarchar(256)")
                        .HasMaxLength(256);

                    b.HasKey("Id");

                    b.HasIndex("NormalizedEmail")
                        .HasName("EmailIndex");

                    b.HasIndex("NormalizedUserName")
                        .IsUnique()
                        .HasName("UserNameIndex")
                        .HasFilter("[NormalizedUserName] IS NOT NULL");

                    b.ToTable("AspNetUsers");

                    b.HasData(
                        new
                        {
                            Id = "3b62472e-4f66-49fa-a20f-e7685b9565d8",
                            AccessFailedCount = 0,
                            ConcurrencyStamp = "25ccbf1f-a019-402a-8d33-6bfe14447b7b",
                            Email = "my@email.com",
                            EmailConfirmed = true,
                            LockoutEnabled = false,
                            NormalizedEmail = "MY@EMAIL.COM",
                            NormalizedUserName = "ADMIN",
                            PasswordHash =
"AQAAAAEAACcQAAAAEIuIujKrRV+VNwho1mtrpSnqYoiOevrMDuRAwJn3ATrRJpxZRoo5ntGjFxcLjzlFAg=="
,
                            PhoneNumberConfirmed = false,
                            SecurityStamp = "",
                            TwoFactorEnabled = false,
                            UserName = "admin"
                        },
                        new
                        {
                            Id = "26a4b2e5-5cbc-4f99-8989-01f54f897a36",
                            AccessFailedCount = 0,
                            ConcurrencyStamp = "da66c50f-70d5-4f39-9f39-2892a198e9ac",
                            Email = "parent@email.com",
                            EmailConfirmed = true,
                            LockoutEnabled = false,
                            NormalizedEmail = "PARENT@EMAIL.COM",
                            NormalizedUserName = "CHILD",
                            PasswordHash =
"AQAAAAEAACcQAAAAEKUaU27+jnn+Nm+bV5wjrn/x4UyyGXrXFpwWkKCkR4EZwZCBOOim6pNvdAjhSgH9fg=="
,
                            PhoneNumberConfirmed = false,
                            SecurityStamp = "",
                            TwoFactorEnabled = false,
                            UserName = "child"
                        });
```

```
                });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>
                {
                    b.Property<int>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("int")
                        .HasAnnotation("SqlServer:ValueGenerationStrategy",
SqlServerValueGenerationStrategy.IdentityColumn);

                    b.Property<string>("ClaimType")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("ClaimValue")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("UserId")
                        .IsRequired()
                        .HasColumnType("nvarchar(450)");

                    b.HasKey("Id");

                    b.HasIndex("UserId");

                    b.ToTable("AspNetUserClaims");
                });


modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>
                {
                    b.Property<string>("LoginProvider")
                        .HasColumnType("nvarchar(450)");

                    b.Property<string>("ProviderKey")
                        .HasColumnType("nvarchar(450)");

                    b.Property<string>("ProviderDisplayName")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("UserId")
                        .IsRequired()
                        .HasColumnType("nvarchar(450)");

                    b.HasKey("LoginProvider", "ProviderKey");

                    b.HasIndex("UserId");

                    b.ToTable("AspNetUserLogins");
                });


modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>
                {
                    b.Property<string>("UserId")
                        .HasColumnType("nvarchar(450)");

                    b.Property<string>("RoleId")
                        .HasColumnType("nvarchar(450)");

                    b.HasKey("UserId", "RoleId");

                    b.HasIndex("RoleId");

                    b.ToTable("AspNetUserRoles");

                    b.HasData(
                        new
                        {
                            UserId = "3b62472e-4f66-49fa-a20f-e7685b9565d8",
```

```
                        RoleId = "44546e06-8719-4ad8-b88a-f271ae9d6eab"
                    },
                    new
                    {
                        UserId = "26a4b2e5-5cbc-4f99-8989-01f54f897a36",
                        RoleId = "14769d9a-e6da-41d9-b21b-8980e64d9624"
                    });
            });

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>
            {
                b.Property<string>("UserId")
                    .HasColumnType("nvarchar(450)");

                b.Property<string>("LoginProvider")
                    .HasColumnType("nvarchar(450)");

                b.Property<string>("Name")
                    .HasColumnType("nvarchar(450)");

                b.Property<string>("Value")
                    .HasColumnType("nvarchar(max)");

                b.HasKey("UserId", "LoginProvider", "Name");

                b.ToTable("AspNetUserTokens");
            });

        modelBuilder.Entity("diplom.Domain.Entities.ServiceItem", b =>
            {
                b.Property<Guid>("Id")
                    .ValueGeneratedOnAdd()
                    .HasColumnType("uniqueidentifier");

                b.Property<string>("AudioPath")
                    .HasColumnType("nvarchar(max)");

                b.Property<DateTime>("DateAdded")
                    .HasColumnType("datetime2");

                b.Property<string>("MetaDescription")
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("MetaKeywords")
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("MetaTitle")
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("Subtitle")
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("Text")
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("Title")
                    .IsRequired()
                    .HasColumnType("nvarchar(max)");

                b.Property<string>("TitleImagePath")
                    .HasColumnType("nvarchar(max)");

                b.HasKey("Id");

                b.ToTable("ServiceItems");
            });

        modelBuilder.Entity("diplom.Domain.Entities.TextField", b =>
```

```
                {
                    b.Property<Guid>("Id")
                        .ValueGeneratedOnAdd()
                        .HasColumnType("uniqueidentifier");

                    b.Property<string>("AudioPath")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("CodeWord")
                        .IsRequired()
                        .HasColumnType("nvarchar(max)");

                    b.Property<DateTime>("DateAdded")
                        .HasColumnType("datetime2");

                    b.Property<string>("MetaDescription")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("MetaKeywords")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("MetaTitle")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("Subtitle")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("Text")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("Title")
                        .HasColumnType("nvarchar(max)");

                    b.Property<string>("TitleImagePath")
                        .HasColumnType("nvarchar(max)");

                    b.HasKey("Id");

                    b.ToTable("TextFields");

                    b.HasData(
                        new
                        {
                            Id = new Guid("63dc8fa6-07ae-4391-8916-e057f71239ce"),
                            CodeWord = "PageIndex",
                            DateAdded = new DateTime(2022, 5, 21, 13, 46, 19, 666,
DateTimeKind.Utc).AddTicks(2604),
                            Text = "Содержание заполняется администратором",
                            Title = "Главная"
                        },
                        new
                        {
                            Id = new Guid("70bf165a-700a-4156-91c0-e83fce0a277f"),
                            CodeWord = "PageServices",
                            DateAdded = new DateTime(2022, 5, 21, 13, 46, 19, 666,
DateTimeKind.Utc).AddTicks(5986),
                            Text = "Содержание заполняется администратором",
                            Title = "Список слов"
                        },
                        new
                        {
                            Id = new Guid("0178b641-1f30-4cd9-b84d-d740c2bdde69"),
                            CodeWord = "PageServices2",
                            DateAdded = new DateTime(2022, 5, 21, 13, 46, 19, 666,
DateTimeKind.Utc).AddTicks(6130),
                            Text = "Содержание заполняется администратором",
                            Title = "Выученые слова"
                        },
                        new
```

```
                        {
                            Id = new Guid("15085f29-9df3-4e6b-98ee-6ce33d49b2e9"),
                            CodeWord = "PageServices3",
                            DateAdded = new DateTime(2022, 5, 21, 13, 46, 19, 666,
DateTimeKind.Utc).AddTicks(6197),
                            Text = "Содержание заполняется администратором",
                            Title = "Невыученные слова"
                        },
                        new
                        {
                            Id = new Guid("4aa76a4c-c59d-409a-84c1-06e6487a137a"),
                            CodeWord = "PageContacts",
                            DateAdded = new DateTime(2022, 5, 21, 13, 46, 19, 666,
DateTimeKind.Utc).AddTicks(6256),
                            Text = "Содержание заполняется администратором",
                            Title = "Контакты"
                        });
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>
                {
                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)
                        .WithMany()
                        .HasForeignKey("RoleId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>
                {
                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)
                        .WithMany()
                        .HasForeignKey("UserId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>
                {
                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)
                        .WithMany()
                        .HasForeignKey("UserId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>
                {
                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)
                        .WithMany()
                        .HasForeignKey("RoleId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();

                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)
                        .WithMany()
                        .HasForeignKey("UserId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });

            modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>
                {
                    b.HasOne("Microsoft.AspNetCore.Identity.IdentityUser", null)
```

```
                        .WithMany()
                        .HasForeignKey("UserId")
                        .OnDelete(DeleteBehavior.Cascade)
                        .IsRequired();
                });
#pragma warning restore 612, 618
        }
    }
}
```

Листинг Б.34 – Файл Models/ LoginViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace diplom.Models
{
    public class LoginViewModel
    {
        //вывод форм полей для страницы входа
        [Required]
        [Display(Name = "Логин")]
        public string UserName { get; set; }

        [Required]
        [UIHint("password")]
        [Display(Name = "Пароль")]
        public string Password { get; set; }

        [Display(Name = "Запомнить меня?")]
        public bool RememberMe { get; set; }

        [Display(Name = "Родитель?")]
        public bool Parent { get; set; }

    }
}
```

Листинг Б.34 – Файл Models/Mode1ViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace diplom.Models
{ //форма поля для ввода слова на странице режима обучения
    public class Mode1ViewModel
    {
        [Required(ErrorMessage = "Введите слово!")]
        [Display(Name = "Ввод слова")]
        public string Word { get; set; }

    }

}
```

Листинг Б.35 – Файл Models/RegistrationViewModel.cs

```
using System.ComponentModel.DataAnnotations;

namespace diplom.Models
{
    public class RegistrationViewModel
    { // поля форм ввода данных на странице регистрации
        [Required(ErrorMessage = "Заполните поле Email")]
        [Display(Name = "Электронная почта")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Заполните поле логин")]
        [Display(Name = "Логин")]
        public string UserName { get; set; }

        [Required(ErrorMessage = "Заполните поле пароль")]
        [UIHint("password")]
        [Display(Name = "Пароль")]
```

```
        public string Password { get; set; }

    }
}
```

Листинг Б.36 – Файл Service/AdminAreaAuthorization.cs

```
using System;
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.ApplicationModels;
using Microsoft.AspNetCore.Mvc.Authorization;

namespace diplom.Service
{ // реализация доступа пользователя admin к внутренней системе приложения
    public class AdminAreaAuthorization : IControllerModelConvention
    {
        private readonly string area;
        private readonly string policy;

        public AdminAreaAuthorization(string area, string policy)
        {
            this.area = area;
            this.policy = policy;
        }

        public void Apply(ControllerModel controller)
        {
            if (controller.Attributes.Any(a =>
                    a is AreaAttribute && (a as AreaAttribute).RouteValue.Equals(area,
StringComparison.OrdinalIgnoreCase))
                || controller.RouteValues.Any(r =>
                    r.Key.Equals("area", StringComparison.OrdinalIgnoreCase) &&
r.Value.Equals(area, StringComparison.OrdinalIgnoreCase)))
            {
                controller.Filters.Add(new AuthorizeFilter(policy));
            }
        }
    }
}
```

Листинг Б.37 – Файл Service/Config.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;


namespace diplom.Service
{ //объявление метода хранения данных о приложении
    public class Config
    {
        //строка подключения к базе данных
        public static string ConnectionString { get; set; }
        //название компании
        public static string CompanyName { get; set; }
        //номер компании
        public static string CompanyPhone { get; set; }
        //номер компании(короткий формат)
        public static string CompanyPhoneShort { get; set; }
        //адрес электронной почты компании
        public static string CompanyEmail { get; set; }
    }
}
```

Листинг Б.38 – Файл Service/Extensions.cs

```
namespace diplom.Service
{
    public static class Extensions
    {
```

```
        //объявление метода отбрасывания слова контроллер при ображении к контроллерам
в других частях приложения
        public static string CutController(this string str)
        {
            return str.Replace("Controller", "");
        }
    }
}
```

Листинг Б.39 – Файл View/Account/Login.cshtml

```
@model LoginViewModel

<div>
    @*отображение данных на странице входа в систему*@
    <div class="div-box">
        <h2>Вход в личный кабинет</h2>
        <form asp-area="" asp-controller="Account" asp-action="Login" method="post"
asp-route-returnUrl="@ViewBag.returnUrl">
            <div asp-validation-summary="All"></div>
            <div>
                <label asp-for="UserName"></label>
                <input asp-for="UserName" />
                <span asp-validation-for="UserName"></span>
            </div>
            <div>
                <label asp-for="Password"></label>
                <input asp-for="Password" />
                <span asp-validation-for="Password"></span>
            </div>
            <div>
                <label asp-for="Parent"></label>
                <input asp-for="Parent" />
                <span asp-validation-for="Parent"></span>
            </div>

            @*отображение кнопки входа*@
            <div>
                <input type="submit" value="Войти" />
            </div>
        </form>
        </div>
        @*отображение кнопки регистрации*@
         <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Registration"
method="post" asp-route-returnUrl="@ViewBag.returnUrl">
            <div>
                <input type="submit" value="Зарегестрироваться" />
            </div>
        </form>
    </div>
</div>
```

Листинг Б.40 – Файл View/Account/Registration.cshtml

```
@model RegistrationViewModel

<div>
    @*отображение данных на странице регистрации*@
    <h2>Регистрация</h2>
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Registration"
method="post" asp-route-returnUrl="@ViewBag.returnUrl">
            <div asp-validation-summary="All"></div>
            <div>
                <label asp-for="UserName"></label>
                <input asp-for="UserName" />@ViewBag.Error
                <span asp-validation-for="UserName"></span>
            </div>
            <div>
                <label asp-for="Password"></label>
```

```
                <input asp-for="Password" />@ViewBag.Conditions
                <span asp-validation-for="Password"></span>
            </div>
            <div>
                <label asp-for="Email"></label>
                <input asp-for="Email" />
                <span asp-validation-for="Email"></span>
            </div>
            @*отображение кнопки окончания регистрации*@
            <div>
                <input type="submit" value="Зарегестрироваться" />
            </div>
        </form>
    </div>
</div>
```

Листинг Б.41 – Файл View/Home/Contacts.cshtml

```
@model TextField
@{
    ViewBag.Title = Model.MetaTitle;
    ViewBag.Description = Model.MetaDescription;
    ViewBag.Keywords = Model.MetaKeywords;
}
@*отображение данных на странице контакты*@
<div>
    @Html.Raw(Model.Text)
</div>
```

Листинг Б.42 – Файл View/Home/Index.cshtml

```
@model TextField
@{
    ViewBag.Title = Model.MetaTitle;
    ViewBag.Description = Model.MetaDescription;
    ViewBag.Keywords = Model.MetaKeywords;
}
@*отображение данных на странице о нас*@
<div>
    @Html.Raw(Model.Text)
</div>
```

Листинг Б.43 – Файл View/Service/Index.cshtml

```
@model IQueryable<ServiceItem>
@{
    TextField textField = ViewBag.TextField;

    ViewBag.Title = textField.MetaTitle;
    ViewBag.Description = textField.MetaDescription;
    ViewBag.Keywords = textField.MetaKeywords;
}
@*вывод списка слов с катинками*@
<div>
    @if (Model.Any())
    {
        <ul class="big-image-list">
            @foreach (ServiceItem entity in Model)
            {
                <li>
                    <a asp-area="" asp-controller="Services" asp-action="Index" asp-
route-id="@entity.Id">
                        <img src="~/images/@entity.TitleImagePath" alt="" class="left"
/>
                    </a>
                    <h4>@entity.Title</h4>
                    <p>@entity.Subtitle</p>
                </li>
            }
        </ul>
    }
</div>
```

Листинг Б.44 – Файл View/Service/Show.cshtml

```
@model ServiceItem
@{
    ViewBag.Title = Model.MetaTitle;
    ViewBag.Description = Model.MetaDescription;
    ViewBag.Keywords = Model.MetaKeywords;
}
@*вывод слова и его сопровождающей картики*@
<div>
    <h2>
        @Model.Title
    </h2>
    <h2>
        @Model.Subtitle
    </h2>
    <div>
        <img class="img-block" src="~/images/@Model.TitleImagePath" />
    </div>
</div>
<div>
    @Html.Raw(Model.Text)
</div>
```

Листинг Б.45 – Файл View/Settings/Index.cshtml

```
@model IQueryable<ServiceItem>

<div>
 @*вывод списка слов и ссылок на инсрументы работы с ними на странице пользовательские
слова*@
    <div>
        <h3>Пользовательские слова</h3>
        <div class="div-box">
            <a asp-area="User" asp-controller="ServiceItems" asp-action="Edit" asp-
route-id="">Добавить слово</a>
        </div>
        @if (Model.Any())
        {
            <div>
                @foreach (ServiceItem entity in Model)
                {
                    <div>
                        <a asp-area="User" asp-controller="ServiceItems" asp-
action="Edit" asp-route-id="@entity.Id">редактировать</a>
                        |
                        <form style="display: inline-block;" id="form-@entity.Id" asp-
area="User" asp-controller="ServiceItems" asp-action="Delete" method="post">
                            <input type="hidden" name="id" value="@entity.Id">
                            <a href="#" onclick="document.getElementById('form-
@entity.Id').submit();">удалить</a>
                        </form>
                        |
                        <a asp-area="" asp-controller="Services" asp-action="Index"
asp-route-id="@entity.Id">
                            @($"{entity.Title}")
                        </a>
                    </div>
                }
            </div>
        }
    </div>
     @*вывод кнопки возврата на страницу меню родителя*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="BackParent"
method="post">
            <input type="submit" value="Меню родителя" />
        </form>
    </div>
</div>
```

Листинг Б.46 – Файл View/Settings/Statistics.cshtml

```
@model IQueryable<ServiceItem>
@*вывод списков выученных и невыученных слов на странице статистика ребенка*@
<div>
  <h2>Статистика ребенка</h2>
    <div>

        @if (Model?.Any() == true)
          {
                    <section>
                        <h1>Выученные слова</h1>
                        <ul class="small-image-list">
                            @foreach (ServiceItem entity in Model)
                            {
                                <li>
                                    <a asp-area="" asp-controller="Services" asp-
action="Learned" asp-route-id="@entity.Id">
                                        <img src="~/images/@entity.TitleImagePath"
alt="" class="left" />
                                    </a>
                                    <h4>
                                        @entity.Title
                                    </h4>
                                    <p>
                                        @entity.Subtitle
                                    </p>
                                </li>
                            }
                        </ul>
                    </section>
          }


                @if (Model?.Any() == true)
          {
                    <section>
                        <h1>Невыученные слова</h1>
                        <ul class="small-image-list">
                            @foreach (ServiceItem entity in Model)
                            {
                                <li>
                                    <a asp-area="" asp-controller="Services" asp-
action="Unlearned" asp-route-id="@entity.Id">
                                        <img src="~/images/@entity.TitleImagePath"
alt="" class="left" />
                                    </a>
                                    <h4>
                                        @entity.Title
                                    </h4>
                                    <p>
                                        @entity.Subtitle
                                    </p>
                                </li>
                            }
                        </ul>
                    </section>
          }
    </div>
    @*вывод кнопки возврата на страницу меню пользователя*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="BackParent"
method="post">
            <input type="submit" value="Меню родителя" />
        </form>
    </div>
</div>
```

Листинг Б.47 – Файл View/Shared/_Layout.cshtml

```
<!DOCTYPE HTML>
<html>
    @*подключение представлений для вывода их на страницы приложения*@
<head>
    @await Html.PartialAsync("MetatagsPartial")
    @await Html.PartialAsync("CssPartial")
</head>
<body>
    <div id="page-wrapper">
        @await Html.PartialAsync("HeaderPartial")
        <div id="main">
            <div class="container">
                <div class="row main-row">
                    <div class="col-8 col-12-medium">
                        @RenderBody()
                    </div>
                </div>
            </div>
        </div>
        @await Html.PartialAsync("FooterPartial")
    </div>
    @await Html.PartialAsync("ScriptsPartial")
</body>
</html>
```

Листинг Б.48 – Файл View/Shared/CSSPartial.cshtml

```
@*подключение файла css для отображения стилей*@
<link rel="stylesheet" href="~/css/main.min.css" asp-append-version="true" />
```

Листинг Б.49 – Файл View/Shared/FooterPartial.cshtml

```
@*вывод оформления и данных в подвал страниц приложения*@
<div id="footer-wrapper">
    <div class="container">
        <div class="row">
            <div class="col-12">

                <div id="copyright">
                    &copy; @Config.CompanyName. | All rights reserved. | Design: <a
href="http://html5up.net">HTML5 UP</a>
                </div>

            </div>
        </div>
    </div>
</div>
```

Листинг Б.50 – Файл View/Shared/HeaderPartial.cshtml

```
@*вывод оформления шапки и вкладок о нас и контакты в шапке страниц приложения*@
<div id="header-wrapper">
    <div class="container">
        <div class="row">
            <div class="col-12">

                <header id="header">
                    <h1><a href="/" id="logo">@Config.CompanyName</a></h1>
                    <nav id="nav">
                        <a asp-area="" asp-controller="Home" asp-action="Index">О
нас</a>
                        <a asp-area="" asp-controller="Account" asp-action="Login"
asp-route-id="">Вход</a>
                        <a asp-area="" asp-controller="Home" asp-
action="Contacts">Контакты</a>
                    </nav>
                </header>

            </div>
        </div>
```

```
    </div>
</div>
```

Листинг Б.51 – Файл View/Shared/MetatagsPartial.cshtml

```
@using diplom.Service;
@*вывод метатегов на страницах приложения*@
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width, initial-scale=1, user-scalable=no"
/>
@if (ViewBag.Title != null)
{
    <title> @ViewBag.Title </title>
}
else
{
    <title> @Config.CompanyName </title>
}

@if (ViewBag.Description != null)
{
    <meta name="description" content="@ViewBag.Description">
}

@if (ViewBag.Keywords != null)
{
    <meta name="keywords" content="@ViewBag.Keywords">
}
```

Листинг Б.52 – Файл View/Shared/ScriptsPartial.cshtml

```
@*подключение скрипта JS*@
<script src="~/js/scripts.min.js" asp-append-version="true"></script>
```

Листинг Б.53 – Файл View/Study/Gift.cshtml

```
@*вывод картинки поощерения с аудио дорожкой*@
<div>
    <div>
        <img class="img-block" src="~/images/Gift.png" />
    </div>

    <audio autoplay>
    <source src="~/audio/поощерение.mp3" type="audio/mpeg">
    </audio>
    @*вывод кнопки далее*@
    <form asp-area="" asp-controller="Study" asp-action="Next" method="post" asp-
route-returnUrl="">
            <div>
                <input type="submit" value="Далее" />
            </div>
        </form>
</div>
```

Листинг Б.54 – Файл View/Study/Index.cshtml

```
@model ModelViewModel
@*подключение частичного представления для вывода картинки и озвучиваания в режиме
обучения*@
await Html.PartialAsync("~/Views/Study/OutputImg.cshtml")
@*вывод формы для ввода слоа на странице режима обучения*@
<div>
    <div class="div-box">
        <form asp-area="" asp-controller="Study" asp-action="Index" method="post">
            <div>
                <label asp-for="Word" ></label>
                <input asp-for="Word"  />@ViewBag.Mistake
            </div>
            @*вывод кнопки для проверки правильности введенного слова*@
            <div>
                <input type="submit" value="Проверить"  />
            </div>
        </form>
```

```
    </div>
    @*вывод кнопки для выхода в меню ребенка*@
    <div class="div-box">
        <form asp-area="" asp-controller="Account" asp-action="Back" method="post">
            <input type="submit" value="Меню режимов" />
        </form>
    </div>
</div>
```

Листинг Б.55 – Файл View/Study/OutputImg.cshtml

```
@model ServiceItem

@{
    ViewBag.Title = Model.MetaTitle;
    ViewBag.Description = Model.MetaDescription;
    ViewBag.Keywords = Model.MetaKeywords;
}
@*вывод слова,картинки и аудио для режима обучения*@
<div>
    <ul class="big-image-list">
        <div>
                <h2>
                    @Model.Title
                </h2>
                <div>
                    <img class="img-block" src="~/images/@Model.TitleImagePath" />
                </div>
        </div>
    </ul>

    <audio controls>
    <source src="~/audio/@Model.AudioPath" type="audio/mpeg">
    </audio>

</div>
```

Листинг Б.56– Файл View/Study/Show.cshtml

```
@model ServiceItem
@{

    ViewBag.Description = Model.MetaDescription;
    ViewBag.Keywords = Model.MetaKeywords;
}
@*вывод слова и картинки для режима обучения*@
<div>
    <h2>
        @Model.Title
    </h2>
    <div>
        <img class="img-block" src="~/images/@Model.TitleImagePath" />
    </div>
</div>
```

Листинг Б.57 – Файл View/_ViewImports.cshtml

```
@*подключение частей приложения*@
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
@using diplom.Models
@using diplom.Service
@using diplom.Domain.Entities
@using diplom.Models.ViewComponents
```

Листинг Б.58 – Файл View/_ViewStart.cshtml

```
@*подключение представления Layout*@
@{
    Layout = "_Layout";
}
```

Листинг Б.59 – Файл appsettings.json

```
{
  // заполение полей данных о приложении и строке подключения к БД
  "Project": {
    "ConnectionString": "Data Source=LAPTOPKSUSHA; Database=Diplom; Persist Security
Info=false; User ID='sa'; Password='sa'; MultipleActiveResultSets=True;
Trusted_Connection=False;",
    "CompanyName": "Поучи-ка! Учись и веселись с нами",
    "CompanyPhone": "+7 (951) 111-11-11",
    "CompanyPhoneShort": "+79511111111",
    "CompanyEmail": "KBakunina2001@mail.ru"
  }
}
```

Листинг Б.60 – Файл bundleconfig.json

```
[// настройка маршрутов для LS
  {
    "outputFileName": "wwwroot/js/scripts.js",
    "inputFiles": [
      "wwwroot/js/jquery.min.js",
      "wwwroot/js/browser.min.js",
      "wwwroot/js/breakpoints.min.js",
      "wwwroot/js/util.js",
      "wwwroot/js/main.js"
    ]
  }
]
```

Листинг Б.61 – Файл compilerconfig.json

```
[// настройка взаимодействия Css и Sass
  {
    "outputFile": "wwwroot/css/main.css",
    "inputFile": "wwwroot/sass/main.sass"
  }
]
```

Листинг Б.62 – Файл libman.json

```
{
  "version": "1.0",
  "defaultProvider": "cdnjs",
  "libraries": []
}
```

Листинг Б.63 – Файл Program.cs

```
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.Logging;

namespace diplom
{
    public class Program
    {
        //метод объявления создания веб-приложения
        public static void Main(string[] args)
        {
            CreateWebHostBuilder(args).Build().Run();
        }

        public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

Листинг Б.64 – Файл Startup.cs

```
using System;
using System.Collections.Generic;
```

```
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Builder;
using Microsoft.AspNetCore.Hosting;
using Microsoft.Extensions.Hosting;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.DependencyInjection;

using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Microsoft.Extensions.Configuration;
using diplom.Service;
using diplom.Domain;
using diplom.Domain.Repositories.Abstract;
using diplom.Domain.Repositories.EntityFramework;


namespace diplom
{
    public class Startup
    {
        public IConfiguration Configuration { get; }
        public Startup(IConfiguration configuration) => Configuration = configuration;

        public void ConfigureServices(IServiceCollection services)
        {
            //подключение конфига из appsetting.json
            Configuration.Bind("Project", new Config());

            //подключение сервисов
            services.AddTransient<ITextFieldsRepository, EFTextFieldsRepository>();
            services.AddTransient<IServiceItemsRepository,
EFServiceItemsRepository>();
            services.AddTransient<DataManager>();

            //подключение контекста БД
            services.AddDbContext<AppDbContext>(x =>
x.UseSqlServer(Config.ConnectionString));

            //настройка identity для системы
            services.AddIdentity<IdentityUser, IdentityRole>(opts =>
            {
                opts.User.RequireUniqueEmail = true;
                opts.Password.RequiredLength = 6;
                opts.Password.RequireNonAlphanumeric = false;
                opts.Password.RequireLowercase = false;
                opts.Password.RequireUppercase = false;
                opts.Password.RequireDigit = false;
            }).AddEntityFrameworkStores<AppDbContext>().AddDefaultTokenProviders();

            //настройка cookie для аутентификации
            services.ConfigureApplicationCookie(options =>
            {
                options.Cookie.Name = "myCompanyAuth";
                options.Cookie.HttpOnly = true;
                options.LoginPath = "/account/login";
                options.AccessDeniedPath = "/account/accessdenied";
                options.SlidingExpiration = true;
            });

            //настройка политики входа для Admin area
            services.AddAuthorization(x =>
            {
                x.AddPolicy("AdminArea", policy => { policy.RequireRole("admin"); });
            });

            //добавление сервисов для контроллеров и представлений (MVC)
            services.AddControllersWithViews(x =>
```

```
            {
                x.Conventions.Add(new AdminAreaAuthorization("Admin", "AdminArea"));
            })
                // совместимость с asp.net core 3.0

.SetCompatibilityVersion(CompatibilityVersion.Version_3_0).AddSessionStateTempDataProv
ider();
        }

        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {

            //вывод ошибок при разработке
            if (env.IsDevelopment())
                app.UseDeveloperExceptionPage();

            //поддержка статичных файлов в приложении (css, js и т.д.)
            app.UseStaticFiles();

            //подключение системы маршрутизации
            app.UseRouting();

            //подключение аутентификации и авторизации
            app.UseCookiePolicy();
            app.UseAuthentication();
            app.UseAuthorization();

            //регистрация маршрутов приложения
            app.UseEndpoints(endpoints =>
            {
                endpoints.MapControllerRoute("admin",
"{area:exists}/{controller=Home}/{action=Index}/{id?}");
                endpoints.MapControllerRoute("default",
"{controller=Home}/{action=Index}/{id?}");
            });
        }
    }
}
```