ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА»
(СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ
ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ
(АКТ (ф) СПбГУТ)

КУРСОВОЙ ПРОЕКТ

НА ТЕМУ

РАЗРАБОТКА ОБУЧАЮЩЕЙ ПРОГРАММЫ

«ТРЕНАЖЕР ИЗУЧЕНИЯ ИНОСТРАННЫХ СЛОВ»

Л109. 25КП01. 026 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

Студент	ИСПП-21		08.12.2025	А.В. Речицкий
	(Группа)	(Подпись)	(Дата)	(И.О. Фамилия)
Преподаватель			09.12.2025	Ю.С. Маломан
		(Подпись)	(Дата)	(И.О. Фамилия)

Архангельск 2025

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М.А. БОНЧ-БРУЕВИЧА» (СПбГУТ)

АРХАНГЕЛЬСКИЙ КОЛЛЕДЖ ТЕЛЕКОММУНИКАЦИЙ ИМ. Б.Л. РОЗИНГА (ФИЛИАЛ) СПбГУТ (АКТ (ф) СПбГУТ)

УТВЕРЖДАЮ
Зав. отделением
_____ Ю.В. Солодкая
(Подпись) (И.О. Фамилия)
24 октября 2025

Задание
для курсового проектирования по МДК.02.01
Технология разработки программного обеспечения

студенту группы ИСПП-21 **,** 4 **курса**

Фамилия, имя, отчество Речицкому Александру Валентиновичу

1 Тема курсового проекта
Разработка обучающей программы «Тренажер изучения иностранных слов»

2 Исходные данные к проекту
Разработать серверную и клиентскую части многопользовательской информационной системы, автоматизирующей хранение, передачу, обработку и представление информации для организации процесса изучения иностранных языков с помощью карточек, грамматическими справочниками и мониторинг прогресса пользователя.

3 Содержание пояснительной записки
Введение
1 Анализ и разработка требований
2 Проектирование программного обеспечения
3 Разработка и интеграция модулей программного обеспечения
4 Тестирование и отладка программного обеспечения
5 Инструкция по эксплуатации программного обеспечения
Заключение
Список использованных источников

4 Перечень графического материала

5 Календарный график работы над проектом на весь период проектирования
24.10-31.10.2025 – анализ поставленной задачи; 01.11-07.11.2025 – проектирование ПО; 08.11-28.11.2025 – разработка и интеграция модулей ПО; 29.11-05.12.2025 – тестирование и отладка ПО; 24.10-07.12.2025 – написание и проверка программной документации, оформление пояснительной записки; 08.12.2025 – сдача курсового проекта на проверку; 09.12.2025 - защита курсового проекта

6 Срок сдачи студентом законченного курсового проекта 08 декабря 2025

7 Рекомендуемая литература, интернет источники

7.1 Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

7.2 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – URL: https://znanium.ru/catalog/product/2178802. – Режим доступа: по подписке. – Текст : электронный.

7.3 Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2024. – 368 с. – URL: https://znanium.ru/catalog/product/2096940. – Режим доступа: по подписке. – Текст : электронный.

7.4 Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – URL: https://ibooks.ru/bookshelf/386796/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

7.5 Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. – Москва : КУРС : ИНФРА-М, 2024. – 336 с. – URL: https://znanium.ru/catalog/product/2083407. – Режим доступа: по подписке. – Текст : электронный.

Преподаватель и руководитель проектирования _____ Ю.С. Маломан
(Подпись) (И.О. Фамилия)

24 октября 2025

Задание принял к исполнению 24 октября 2025

Студент группы ИСПП-21, 4 курса, очной формы обучения _____
(Подпись)

Заключение по курсовому проекту

Преподаватель и руководитель проектирования _____Ю.С. Маломан
(Подпись) (И.О. Фамилия)

_____ _____20 ____

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем техническом отчете применяются следующие сокращения и обозначения:

ИС – информационная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

ЯП – язык программирования

API – интерфейс прикладного программирования

EF Core – Entity Framework Core

GUI – графический пользовательский интерфейс

JSON – JavaScript Object Notation

MS SQL – Microsoft Structured Query Language

SSMS - SQL Server Microsoft

UML – унифицированный язык моделирования

WPF – Windows Presentation Foundation

ВВЕДЕНИЕ

В современных условиях для эффективного освоения языковых навыков важны доступность информации и регулярность повторений. Изучающие язык и преподаватели сталкиваются с проблемами, связанными с разрозненностью учебных материалов, сложностью организации персональных словарей и отсутствием удобных инструментов для проверки знаний.

Разработка подсистемы (или программного средства) для поддержки изучения английского языка позволит значительно упростить процесс пополнения словарного запаса, структурировать грамматический материал и повысить общую эффективность обучения за счет использования интерактивных методов.

Актуальность разрабатываемого проекта заключается в автоматизации процессов изучения иностранного языка, систематизации учебных материалов и контроля прогресса обучающихся.

Целью курсового проекта является разработка подсистемы, обеспечивающей возможность комплексного управления процессом обучения, включая ведение персональных словарей, работу с грамматическими правилами и проверку знаний с использованием метода интервальных повторений.

Для достижения поставленной цели требуется решить следующие задачи:

- провести сбор и анализ требований целевой аудитории (студенты, изучающие язык, преподаватели, лингвистические центры);
- проанализировать информационные источники по предметной области (методики запоминания, форматы электронных словарей);
- изучить существующие решения в области приложений для изучения языков (Quizlet, Anki и аналоги);
- спроектировать архитектуру подсистемы (MVVM);
- спроектировать диаграмму использования подсистемы;

- выбрать состав программных и технических средств для реализации проекта (WPF, .NET, SQLite);
- спроектировать БД для хранения словарей, слов, правил и статистики;
- создать БД в выбранной СУБД;
- разработать слой доступа к данным для взаимодействия клиентского приложения с БД;
- реализовать разграничение прав доступа пользователей;
- обеспечить защиту данных и безопасное хранение пользовательской информации;
- разработать пользовательский интерфейс (с поддержкой темной темы и адаптивного дизайна);
- реализовать функциональность управления словарями и добавления новых слов;
- реализовать функциональность создания и просмотра грамматических правил (с поддержкой Markdown);
- реализовать функциональность режима тренировки (система флеш-карточек);
- выполнить структурное тестирование ПО;
- выполнить функциональное тестирование ПО;
- разработать программную документацию;
- разработать эксплуатационную документацию.

В результате выполнения поставленных задач будет создана подсистема для автоматизации изучения английского языка, которая значительно упростит процесс пополнения словарного запаса и повысит качество усвоения материала за счет удобного доступа к теории и практике.

1 Анализ и разработка требований

1.1 Назначение и область применения

Основным назначением разрабатываемого ПО "Learning Trainer" является автоматизация и упрощение процесса изучения и запоминания иностранной лексики и грамматических правил. Это позволит снизить временные затраты пользователей на организацию и повторение учебных материалов, а также минимизировать несистематичность в подходе к обучению.

ПО предназначено для использования в двух основных областях:

1) индивидуальное использование в целях самообучения и повышения личной эффективности в изучении языков;

2) образовательные учреждения, где ПО может применяться в качестве вспомогательного инструмента для студентов и преподавателей.

1.2 Постановка задачи

Необходимо разработать гибридное оконное приложение с клиент-серверной архитектурой для операционной системы Windows. В приложении должны быть реализованы следующие функциональные возможности:

- аутентификация и авторизация пользователей;
- разграничение прав доступа на основе ролей;
- управление словарями: создание, просмотр и удаление словарей;
- управление словами: добавление и удаление слов в рамках выбранного словаря;
- управление правилами: создание, просмотр и удаление грамматических правил;
- реализация интерактивного тренажёра для изучения слов в режиме "flashcards" (карточек);

- поддержка онлайн-режима;
- поддержка автономного режима;
- автоматическая "Pull-синхронизация" данных при входе в онлайн-режим;
- реализация механизма импорта и экспорта данных в формате *.json;
- реализация связи "Учитель-Ученик" на уровне API и базы данных, что позволит назначать словари и отслеживать прогресс учеников.

Интерфейс должен быть интуитивно понятен для пользователя и реализован с использованием современных подходов.

В системе должны быть реализованы следующие роли пользователей:

- student: может просматривать список доступных словарей и правил, а также запускать интерактивный тренажёр для их изучения. Не имеет доступа к функциям создания, редактирования или удаления контента;
- user: обладает полным доступом ко всем функциям управления контентом, включая создание, редактирование и удаление словарей, слов и правил;
- teacher: обладает полными правами, в том числе управлением контента учеников привязанных к нему.

На рисунке 1 изображена диаграмма вариантов использования подсистемы.

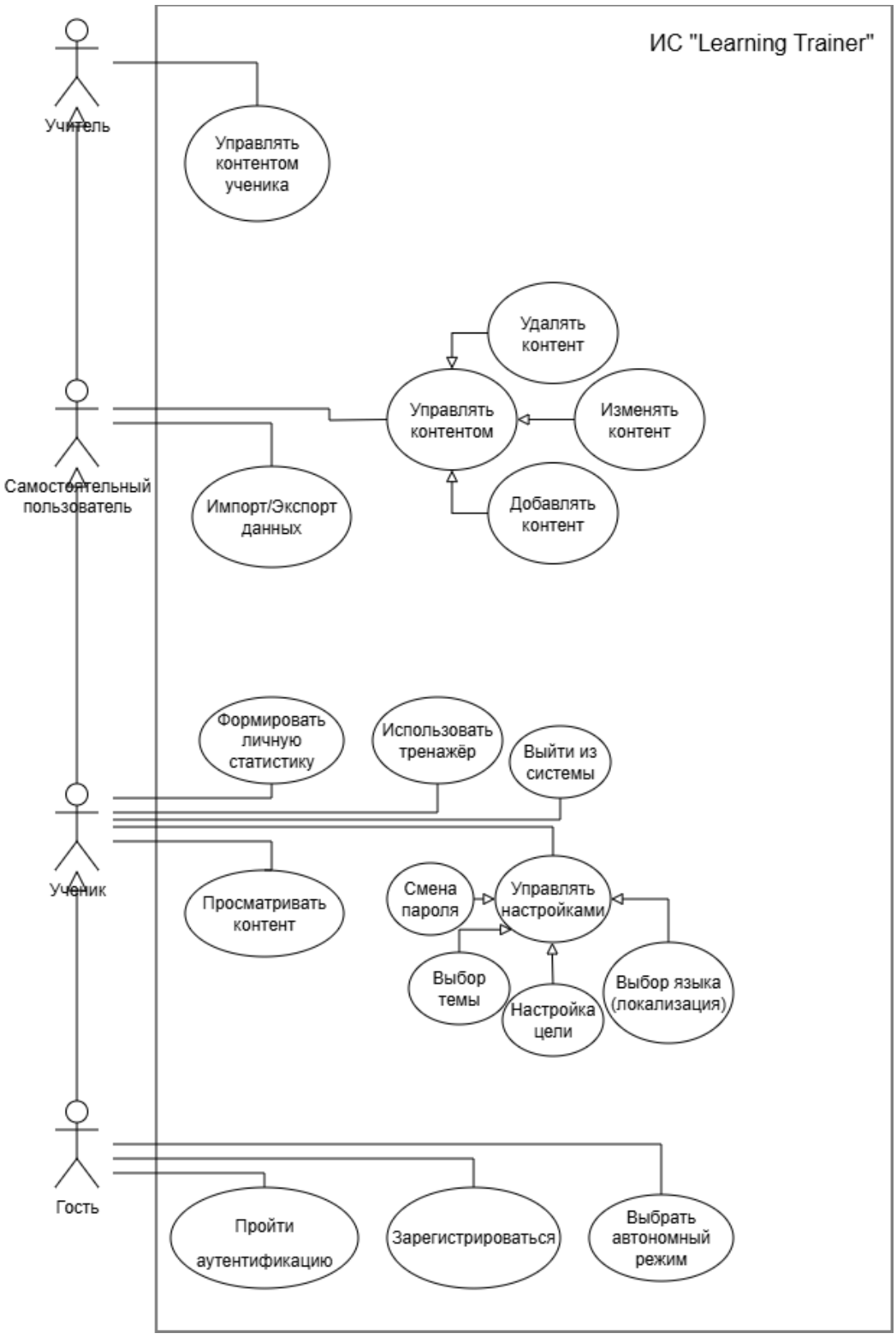

Рисунок 1 – Диаграмма вариантов использования

Работа с системой осуществляется по следующему алгоритму:

Пользователь запускает приложение и видит окно аутентификации (LoginView).

Функционирование системы будет основано на гибридной клиент-серверной архитектуре. Клиентское приложение будет обеспечивать взаимодействие с пользователем, в то время как серверная часть будет отвечать за централизованное хранение данных и бизнес-логику.

Алгоритм работы пользователя с системой следующий:

1) аутентификация: при запуске приложения пользователю будет предложено пройти аутентификацию. Система будет поддерживать два режима:

- онлайн-режим: Пользователь вводит логин и пароль. Клиентское приложение будет отправлять запрос на серверный API для проверки учетных данных в центральной базе данных (MS SQL Server);

- автономный режим: Пользователь сможет выбрать вход без подключения к интернету. Доступ к данным будет осуществляться на основе локально сохраненного кэша (SQLite).

2) загрузка и синхронизация данных:

- при успешном онлайн-входе будет инициироваться процесс "Pull-синхронизации". Система будет запрашивать актуальные данные (словари, правила) с сервера и обновлять ими локальный кэш (SQLite). Это обеспечит пользователя актуальной информацией для последующих автономных сессий;

- при автономном входе приложение будет загружать данные непосредственно из локального кэша SQLite.

3) работа с системой:

- после входа пользователю будет отображаться основной интерфейс («Дашборд»), предоставляющий доступ к функционалу в соответствии с правами доступа его роли;

- учитель сможет управлять учебным контентом: создавать, редактировать и удалять словари и правила;

- ученик сможет просматривать доступные материалы и запускать интерактивный тренажёр для изучения слов.

4) завершение работы: Пользователь сможет выйти из своей учетной записи, после чего система вернется к экрану аутентификации.

1.3 Выбор состава программных и технических средств

Работа с оконным приложением будет осуществляться на ПК и ноутбуках с ОС Windows (Windows 10 и новее), так как клиентская часть разрабатывается на платформе WPF.

В качестве серверной СУБД выбрана SSMS. Эта СУБД обеспечивает высокую производительность, надежность и масштабируемость, необходимые для централизованного хранения данных и реализации логики "Учитель-Ученик". В качестве локальной (кэширующей) СУБД выбрана SQLite, так как она является легковесной, встраиваемой и не требует отдельной установки, что идеально для автономного режима.

Клиентская и серверная части приложения будут разработаны на языке C#. Данный язык программирования и платформа .NET версии 8.0 позволяют эффективно создавать современные приложения с использованием:

- WPF для разработки клиентской части с гибким и современным UI, следуя паттерну MVVM;
- ASP.NET Core Web API для разработки серверной части, обеспечивая быструю и безопасную обработку HTTP-запросов.

Для разработки будет использоваться IDE Visual Studio 2022 , так как эта среда предлагает удобные инструменты для комплексной работы с C#, WPF, ASP.NET Core, EF Core и SQL Server, включая встроенные средства отладки и управления версиями Git.

Для функционирования системы на стороне сервера необходимы:

- ОС: Windows Server 2016 / Ubuntu 22.04 или новее;
- сервер БД: Microsoft SQL Server 2019 или новее;

- .NET 8.0 Runtime или новее;
- процессор: 2 ГГц или выше;
- ОЗУ: 4 ГБ;
- свободное место на диске: не менее 5 ГБ.

Для функционирования системы на стороне клиента необходимы:

- ОС: Windows 10 (версии 1809) или новее;
- .NET 8.0 Desktop Runtime;
- процессор: 1.6 ГГц или выше (рекомендуется 2 ГГц);
- ОЗУ: 2 ГБ (рекомендуется 4 ГБ);
- свободное место на диске: не менее 500 МБ.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В рамках разработки оконного приложения создан интерфейс пользователя в виде Wireframe при помощи инструмента draw.io. Эти визуальные представления позволяют представить структуру приложения, его основные элементы и функциональность.

Wireframe страниц «Авторизация», «Правила и словари», «Просмотр правила», «Изучение слов» представлен на рисунке 2

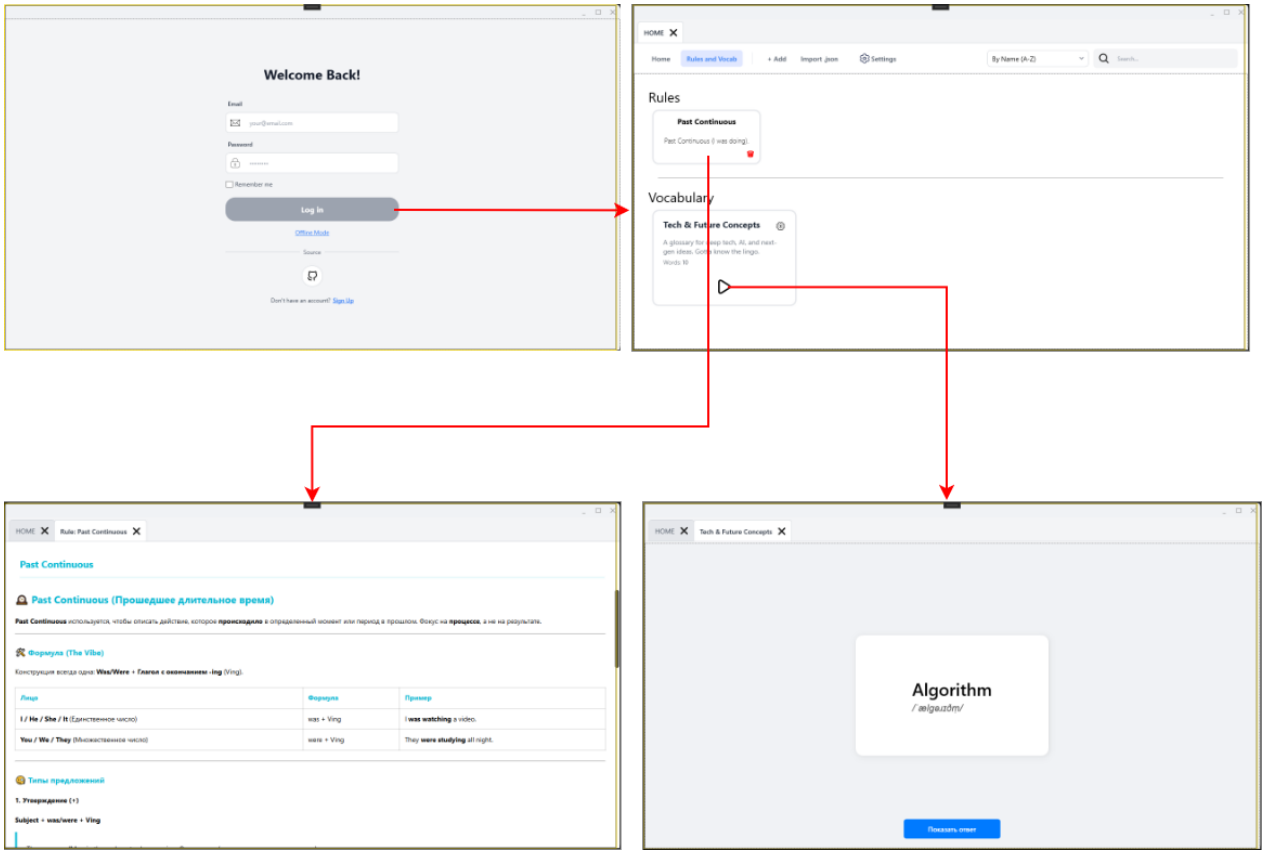

Рисунок 2 –Wireframe основных окон подсистемы

Для оконного приложения выбраны следующие цвета:

- основной цвет: #FF000000;
- цвет фона: #FFF3F4F6;

- основной цвет текста: #FF3B82F6;
- вторичный цвет текста: #FF000000.

2.2 Разработка архитектуры программного обеспечения

Архитектура построена на основе клиент-серверной модели и включает в себя несколько ключевых компонентов: оконное приложение, БД, API, позволяющий клиенту взаимодействовать с сервером. Диаграмма развертывания изображена на рисунке 3.

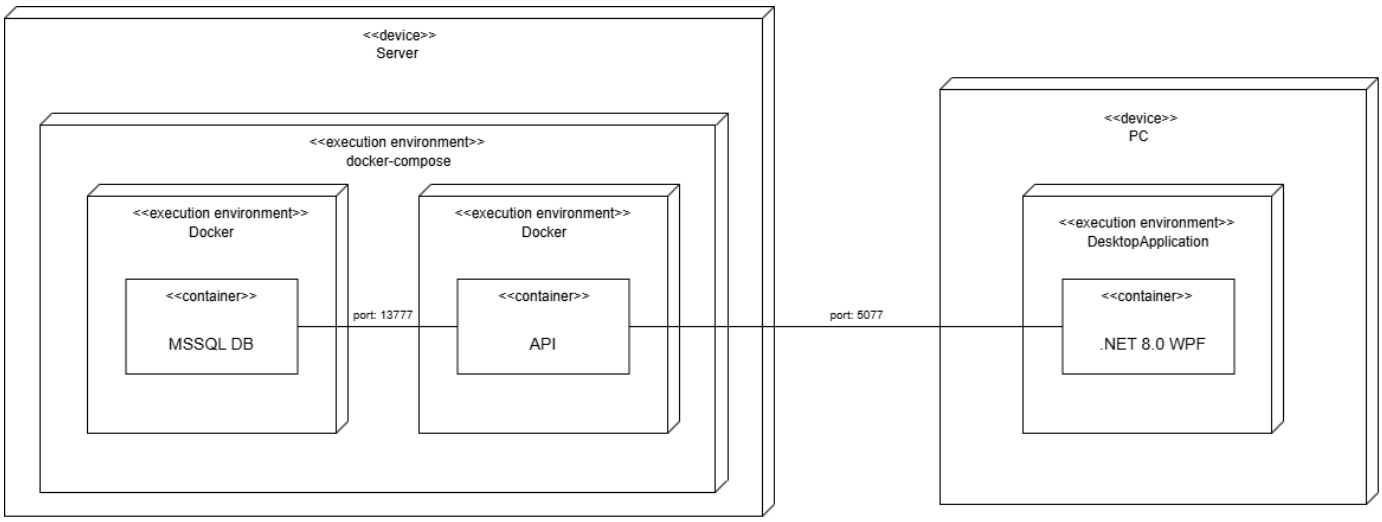

Рисунок 3 – Диаграмма развертывания

2.3 Проектирование базы данных

В рамках проектирования информационной системы требуется разработать базу данных для централизованного хранения информации о пользователях и их ролях, личных и общих словарях, лексических единицах (словах), грамматических правилах, а также данных о прогрессе обучения и статистике успеваемости [3]. На рисунке 4 в виде ERD показана физическая модель предметной области, спроектированная в среде SQL Server Management Studio.

Рисунок 4 – Физическая модель БД

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

В ходе курсового проектирования разработаны: клиентское приложение на языке программирования C# с использованием технологии WPF и серверная часть на платформе .NET с применением ORM Entity Framework Core [2].

Взаимодействие клиентского приложения с сервером реализовано посредством HTTP-запросов к API, обмен данными осуществляется в формате JSON. Реализация метода для создания записи словаря путем отправки POST-запроса представлена листингом 1.

Листинг 1 – Код метода для отправки POST-запроса на сервер.

```
// POST: /api/dictionaries
[HttpPost]
public async Task<IActionResult> AddDictionary([FromBody]
CreateDictionaryRequest requestDto)
{
    var userIdString =
User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
    if (!int.TryParse(userIdString, out var userId)) return
Unauthorized();

    if (requestDto == null ||
string.IsNullOrWhiteSpace(requestDto.Name))
        return BadRequest("Name is required.");

    var newDictionary = new Dictionary
    {
        Name = requestDto.Name,
        Description = requestDto.Description,
        LanguageFrom = requestDto.LanguageFrom,
        LanguageTo = requestDto.LanguageTo,
        Words = new List<Word>(),
        UserId = userId
    };
```

```
    _context.Dictionaries.Add(newDictionary);
    await _context.SaveChangesAsync();

    return CreatedAtAction(nameof(GetDictionaries), new { id =
newDictionary.Id }, newDictionary);
}
```

Для получения списка словарей и правил в приложении разработана функция, представленная листингом 2.

Листинг 2 – Код функции получения списка словарей и правил

```
private async void LoadDataAsync()
{
    try
    {
        // Установка начальной сортировки по имени
        SelectedSortKey = SortKey.NameAsc;
        List<Dictionary> dictionaries;
        List<Rule> rules;

        // Логика загрузки данных в зависимости от роли
пользователя
        if (_currentUser?.Role?.Name == "Student")
        {
            // Если пользователь студент — загружаем доступные
ему (расшаренные) словари и правила
            dictionaries = await
_dataService.GetAvailableDictionariesAsync();
            rules = await _dataService.GetAvailableRulesAsync();
        }
        else
        {
            // Если пользователь учитель - загружаем его личные
словари и правила
            dictionaries = await
_dataService.GetDictionariesAsync();
            rules = await _dataService.GetRulesAsync();
        }

        // Очистка локальных коллекций перед заполнением новыми
данными
        Dictionaries.Clear();
        Rules.Clear();

        // Заполнение коллекции словарей, оборачивая модели в
ViewModel
        foreach (var dict in dictionaries)
```

```
            {
                Dictionaries.Add(new DictionaryViewModel(dict));
            }

            // Заполнение коллекции правил
            foreach (var rule in rules) Rules.Add(rule);

            // Обновление отображаемой коллекции (для привязки в UI)
            DisplayDictionaries.Clear();
            foreach (var dict in Dictionaries)
            {
                DisplayDictionaries.Add(dict);
            }

            // Уведомление UI об изменении данных
            OnPropertyChanged(nameof(Dictionaries));
        }
        catch (HttpRequestException httpEx)
        {
            // Обработка ошибки авторизации (истек токен доступа)
            if (httpEx.StatusCode ==
System.Net.HttpStatusCode.Unauthorized)
            {
                // Публикация события для принудительного выхода из
системы
                EventAggregator.Instance.Publish(new
LogoutRequestedMessage());
            }
            else
            {
                System.Diagnostics.Debug.WriteLine($"!!!ОШИБКА HTTP
в Dashboard: {httpEx.Message}");
            }
        }
        catch (Exception ex)
        {
            // Логирование прочих критических ошибок
            System.Diagnostics.Debug.WriteLine($"!!!КРИТИЧЕСКАЯ
ОШИБКА в Dashboard.LoadData: {ex.Message}");
        }
}
```

Изменение информации в БД осуществляется посредством Web-API приложения, код изменения информации о словаре по идентификатору представлен листингом 3.

Листинг 3 – Код функции изменения информации о словаре по идентификатору

```
[HttpPut("{id:int}")]
public async Task<IActionResult> UpdateDictionary(int id,
[FromBody] Dictionary dictionary)
{
    if (id != dictionary.Id) return BadRequest("ID mismatch");

    _context.Entry(dictionary).State = EntityState.Modified;

    try
    {
        await _context.SaveChangesAsync();
    }
    catch (DbUpdateConcurrencyException)
    {
        if (!_context.Dictionaries.Any(e => e.Id == id)) return
NotFound();
        else throw;
    }

    return NoContent();
}
```

3.2 Реализация интерфейса пользователя

Интерфейс приложения реализован с использованием технологии WPF в рамках главного окна-контейнера (Shell), поддерживающего навигацию между функциональными модулями посредством динамических вкладок. В приложении разработаны унифицированные элементы управления и стили оформления (поддержка темной темы) для обеспечения эргономичности и визуальной целостности. Навигация по основным разделам системы осуществляется через главную панель управления (Dashboard), предоставляющую доступ к библиотеке материалов. Для отображения краткой информации о созданных словарях разработан специализированный элемент интерфейса – карточка словаря, которая представлена на рисунке 5.

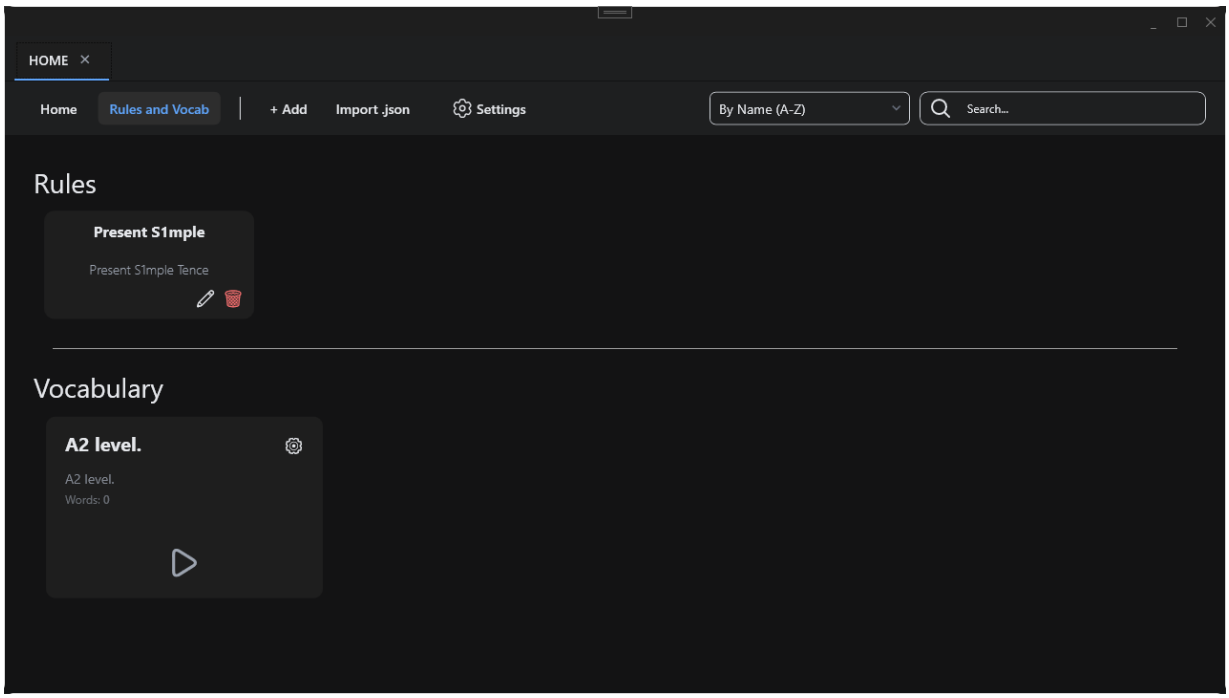

Рисунок 5 – Карточки правил

Ключевым элементом приложения является режим тренировки («Карточки»), реализованный в представлении LearningView. Визуализация карточки выполнена с использованием Border и привязок к состоянию ViewModel. Для отображения перевода по требованию используется конвертер BooleanToVisibilityConverter.

Фрагмент XAML-разметки карточки слова представлен листингом 4.

Листинг 4 – Фрагмент кода карточки правила

```
      <Border Grid.Row="1" Margin="20"
Background="{DynamicResource CardBackgroundBrush}"
        CornerRadius="15" Effect="{DynamicResource CardShadow}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="Auto"/>
        </Grid.RowDefinitions>

        <StackPanel VerticalAlignment="Center"
HorizontalAlignment="Center">
            <TextBlock Text="{Binding CurrentWord.Original}"
```

```
                        FontSize="36" FontWeight="Bold"
                        Foreground="{DynamicResource
TextPrimaryBrush}"
                        HorizontalAlignment="Center"
TextWrapping="Wrap"/>

            <TextBlock Text="{Binding
CurrentWord.Transcription}"
                        FontSize="18"
Foreground="{DynamicResource TextSecondaryBrush}"
                        HorizontalAlignment="Center"
Margin="0,5,0,0"/>

            <StackPanel Visibility="{Binding
IsTranslationVisible, Converter={StaticResource
BooleanToVisibilityConverter}}"
                         Margin="0,20,0,0">
                <TextBlock Text="{Binding
CurrentWord.Translation}"
                           FontSize="28"
Foreground="{DynamicResource AccentBrush}"
                           HorizontalAlignment="Center"/>
                <TextBlock Text="{Binding CurrentWord.Example}"
                           FontSize="16" FontStyle="Italic"
                           Foreground="{DynamicResource
TextSecondaryBrush}"
                           HorizontalAlignment="Center"
Margin="0,10,0,0" TextWrapping="Wrap"/>
            </StackPanel>
        </StackPanel>

        <Button Grid.Row="1" Content="{DynamicResource
Lang.ShowAnswer}"
                Command="{Binding FlipCardCommand}"
                Style="{DynamicResource OutlineButtonStyle}"
Margin="0,0,0,20" HorizontalAlignment="Center"
                Visibility="{Binding IsTranslationVisible,
Converter={StaticResource
InvertedBooleanToVisibilityConverter}}"/>
    </Grid>
</Border>
```

3.3 Разграничение прав доступа пользователей

Разграничение доступа к данным реализовано в методе загрузки данных LoadDataAsync во ViewModel главной панели (DashboardViewModel). Система проверяет роль текущего пользователя и выбирает соответствующий метод сервиса данных: для студентов загружаются доступные (расшаренные)

словари, для преподавателей — их личные. Логика определения доступности функций администратора представлено листингом 5

Листинг 5 – Логика определения доступности функций администратора

```
private async void LoadDataAsync()
{
    try
    {
        List<Dictionary> dictionaries;

        // Проверка роли пользователя
        if (_currentUser?.Role?.Name == "Student")
        {
            // Для студента загружаем только доступные ему
словари
            dictionaries = await
_dataService.GetAvailableDictionariesAsync();
        }
        else
        {
            // Для учителя/админа загружаем все личные словари
            dictionaries = await
_dataService.GetDictionariesAsync();
        }

        // Заполнение коллекции для отображения
        Dictionaries.Clear();
        foreach (var dict in dictionaries)
        {
            Dictionaries.Add(new DictionaryViewModel(dict));
        }
    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine($"КРИТИЧЕСКАЯ ОШИБКА
в Dashboard.LoadData: {ex.Message}");
    }
}
```

3.4 Экспорт и импорт данных

Для обмена учебными материалами реализована функция импорта словарей из формата JSON. Метод ImportDictionary выполняет чтение файла, десериализацию объекта с сохранением ссылок (ReferenceHandler.Preserve) и

последовательное сохранение словаря и слов в базу данных через API, обнуляя идентификаторы для создания новых записей.

Реализация метода импорта словаря в JSON-файл представлена листингом 6.

Листинг 6 – Логика метода импорта словаря в JSON-файл

```
private async Task ImportDictionary()
{
    if (_dialogService.ShowOpenDialog(out string filePath))
    {
        try
        {
            string json = await File.ReadAllTextAsync(filePath);

            var options = new JsonSerializerOptions
            {
                ReferenceHandler = ReferenceHandler.Preserve,
                PropertyNameCaseInsensitive = true
            };

            var newDictionary =
JsonSerializer.Deserialize<Dictionary>(json, options);

            // Сброс ID для создания новой копии словаря
            newDictionary.Id = 0;
            var wordsToImport = newDictionary.Words.ToList();
            newDictionary.Words.Clear();

            var savedDictionary = await
_dataService.AddDictionaryAsync(newDictionary);
            foreach (var word in wordsToImport)
            {
                word.Id = 0;
                word.DictionaryId = savedDictionary.Id;
                await _dataService.AddWordAsync(word);
            }

            // Обновление интерфейса
            EventAggregator.Instance.Publish(new
DictionaryAddedMessage(savedDictionary));
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка импорта: {ex.Message}");
        }
    }
}
```

Реализация метода экспорта словаря из JSON-файл представлена листингом 7.

Листинг 7 – Логика метода экспорта словаря из JSON-файл

```
private void ExportDictionary()
{
    string defaultName = $"dictionary-
{_dictionary.Name.Replace(" ", "-")}.json";

    if (_dialogService.ShowSaveDialog(defaultName, out string
filePath))
    {
        try
        {
            var options = new JsonSerializerOptions
            {
                ReferenceHandler = ReferenceHandler.Preserve,
                WriteIndented = true,
                Encoder =
JavaScriptEncoder.Create(UnicodeRanges.All)
            };

            string json = JsonSerializer.Serialize(_dictionary,
options);

            File.WriteAllText(filePath, json);

            MessageBox.Show(
                $"Словарь '{_dictionary.Name}' успешно
экспортирован!",
                "Экспорт завершен",
                MessageBoxButton.OK,
                MessageBoxImage.Information);
        }
        catch (Exception ex)
        {
            System.Diagnostics.Debug.WriteLine($"Ошибка
экспорта: {ex.Message}");
            MessageBox.Show(
                $"Произошла ошибка: {ex.Message}",
                "Ошибка экспорта",
                MessageBoxButton.OK,
                MessageBoxImage.Error);
        }
    }
}
```

4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

Во время проектирования проведено тестирование методом белого ящика метода редактирования информации о категории в Web-API, результаты которого представлены в таблице 1 [1].

Таблица 1 – Результаты тестирования метода полученной информации о категории

Действие	Ожидаемый результат	Фактический результат

1) Проверить, что прогресс для пары User-Word не существует.
2) Выполнить POST запрос в API с телом:
{
"wordId": 2,
"quality": 3
}
	Создание новой записи LearningProgress в БД. Установка KnowledgeLevel = 4, TotalAttempts = 2. Ответ 200 OK.	Совпадает с ожидаемым

1) Проверить, что прогресс существует и KnowledgeLevel > 0.
2) Выполнить POST запрос в API с телом:
{
"wordId": 2,
"quality": 0
}
	Сброс KnowledgeLevel до 0. Установка NextReview через 5 минут. Ответ 200 OK	Совпадает с ожидаемым

1) Проверить, что прогресс существует.
2) Выполнить POST запрос в API с телом:
{
"wordId": 2,
"quality": 3
}
	Увеличение KnowledgeLevel на 2. Расчет NextReview с коэффициентом 1.5. Ответ 200 OK.	Совпадает с ожидаемым

4.2 Функциональное тестирование

Во время разработки проведено функциональное тестирование приложения методом черного ящика [5], результаты тестирования представлены в таблице 2.

Таблица 2 – Результаты тестирования приложения методом черного ящика

Действие	**Ожидаемый результат**	**Фактический результат**
Нажать на кнопку «Settings» в верхнем меню	Отображение списка настроек для приложения	Совпадает с ожидаемым
Нажать на кнопку «Home» в навигационном меню	Отображение личной статистики пользователя	Совпадает с ожидаемым
Нажать на кнопку «Add Rule» из выпадающего списка «+ Add»	Переход на экран «Создание правила»	Совпадает с ожидаемым
Нажать на кнопку «Add Vocabulary» из выпадающего списка «+ Add»	Переход на экран «Создание карточки»	Совпадает с ожидаемым
Нажать на кнопку «Import .json» в навигационном меню и выбрать подходящий файл	В списке «Vocabulary» появился словарь со словами	Совпадает с ожидаемым
В меню настроек написать в поле «Old Password» текущий пароль от аккаунта, затем написать будущий пароль в поле «New Password», нажать на кнопку «Save»	Пароль должен быть изменён на новый	Совпадает с ожидаемым
В меню настроек нажать на кнопку «Become Teacher» на аккаунте с ролью «Admin»	Появляется поздравительное уведомление, появляется пригласительный код для создания класса	Совпадает с ожидаемым
Создать аккаунт с логином «student@gmail.com», паролем «password» и сгенерированным кодом «TR-dtlRcu»	Аккаунт успешно создан, на аккаунте с ролью «учитель» доступа способность делиться карточками с учениками в классе	Совпадает с ожидаемым
В меню настроек нажать на кнопку «Log out»	Переход на экран «Авторизация»	Совпадает с ожидаемым

По результатам тестирования можно сделать вывод, что разработанное приложение работает корректно и согласно ожиданиям.

5 Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

Архитектура системы предполагает раздельное развертывание серверной части (в контейнерах Docker) и клиентского приложения (на рабочей станции пользователя). Требования к серверной части:

- ОС: Windows 10/11 (с поддержкой WSL2) или Linux (Ubuntu 20.04+);
- процессор частотой 2 ГГц;
- свободная оперативная память 4 ГБ;
- установленное ПО: Docker Desktop (для Windows) или Docker Engine + Docker Compose (для Linux).

Требования к клиентской части:

- ОС: Windows 10 версии 1809 и выше;
- .NET Desktop Runtime 8.0;
- свободное место на диске не менее 200 МБ.

Инструкция по развертыванию сервера:

1) установить Docker Desktop и убедиться, что служба Docker запущена;
2) разместить файлы Dockerfile, docker-compose.yml и исходный код API в папку проекта на сервере;
3) открыть терминал (PowerShell или CMD) в папке с файлом docker-compose.yml;
4) выполнить команду сборки и запуска контейнеров docker-compose up --build -d;
5) дождаться окончания сборки. Система автоматически развернет два контейнера: mssql и learning-api.

Параметры подключения к БД заданы в файле конфигурации контейнеров представлены листингом 8.

Листинг 8 – Конфигурация среды (фрагмент docker-compose.yml)

```
environment:
  - DB_HOST=mssql
  - DB_NAME=EnglishTrainerDB
  - DB_SA_PASSWORD=Your_password123
  - ASPNETCORE_URLS=http://+:8080
```

Инструкция по установке клиента:

1) установить пакет .NET 8.0 Desktop Runtime;

2) разместить папку с исполняемым файлом LearningTrainer.exe на компьютере пользователя;

3) в файле настроек клиента appsettings.json проверить адрес подключения к API. Он должен соответствовать порту, открытому в Docker представленном листингом 9.

Листинг 9 – Настройка подключения клиента

```
{
  "ApiSettings": {
    "BaseUrl": "http://localhost:5077"
  }
}
```

5.2 Инструкция по работе

При запуске приложения пользователь попадает на экран авторизации. Если у пользователя нет учетной записи, он может перейти к регистрации, нажав соответствующую ссылку.

Окно авторизации представлено на рисунке 6. В системе предусмотрены роли: «Ученик», «Учитель» и «Администратор».

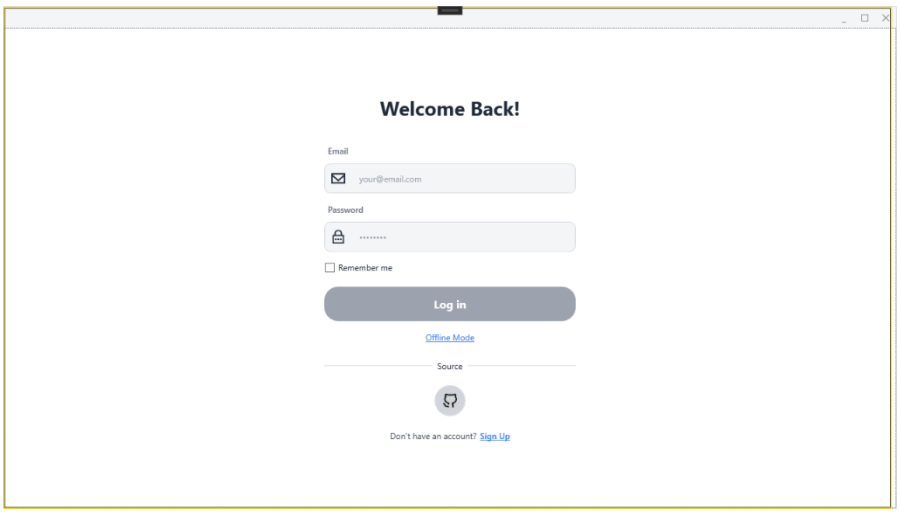

Рисунок 6 – Окно авторизации пользователя

После успешного входа открывается главная панель (Dashboard), предоставляющая доступ к словарям и правилам. Интерфейс главной панели представлен на рисунке 7. В верхней части окна расположены элементы навигации и настройки профиля. В центральной части отображается список доступных словарей.

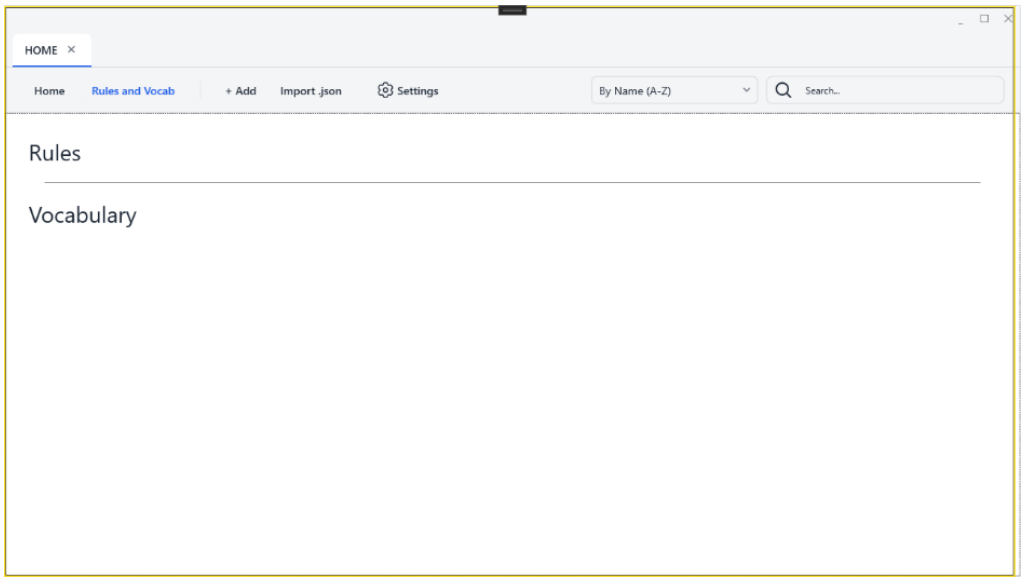

Рисунок 7 – Главное окно приложения

Для начала обучения необходимо выбрать словарь и нажать кнопку «▷». Откроется режим изучения слов (рисунок 8), где пользователю предлагаются карточки с иностранными словами. Нажатие кнопки «Show answer» демонстрирует перевод и пример использования слова.

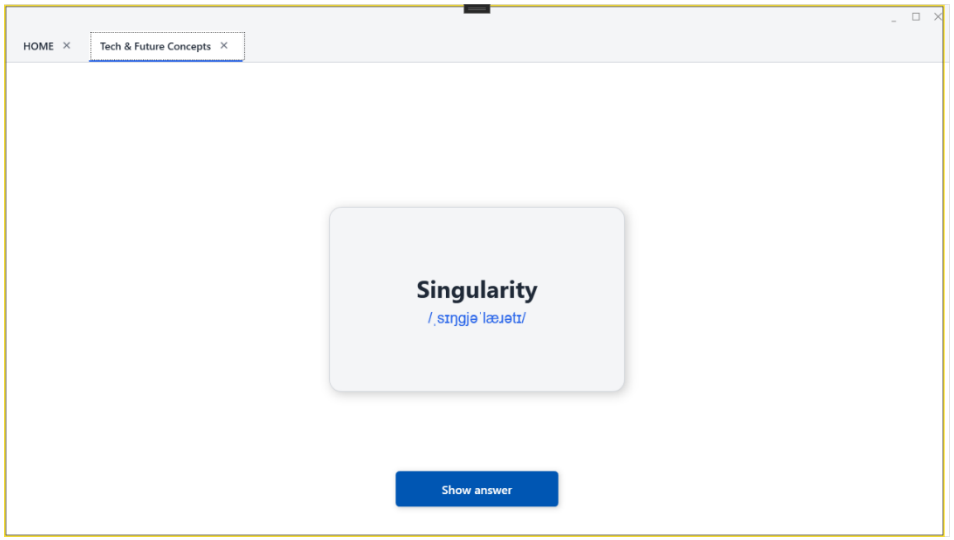

Рисунок 8 – Режим тренировки слов

Для добавления новых материалов пользователь с правами «Учитель» или «Администратор» может воспользоваться кнопкой «Add Rule» или «Add Vocabulary». Форма добавления нового словаря представлена на рисунке 9.

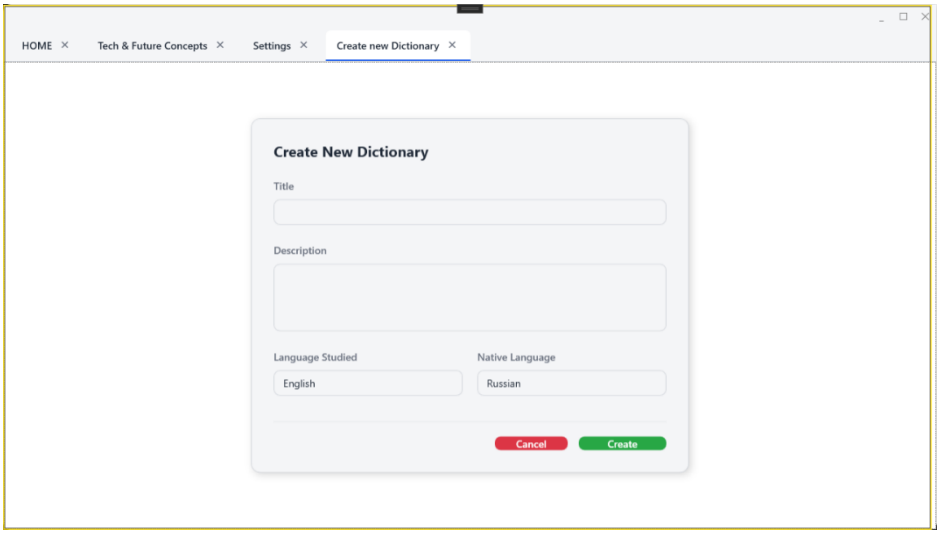

Рисунок 9 – Окно создания нового словаря

Пользователи с ролью «Учитель» имеют возможность назначать учебные материалы своим ученикам. Данный функционал позволяет гибко управлять учебным планом, открывая доступ к словарям или правилам по мере прохождения курса.

Для назначения материала необходимо нажать кнопку «Share Access» на карточке словаря. Откроется окно управления доступом (Рисунок 10).

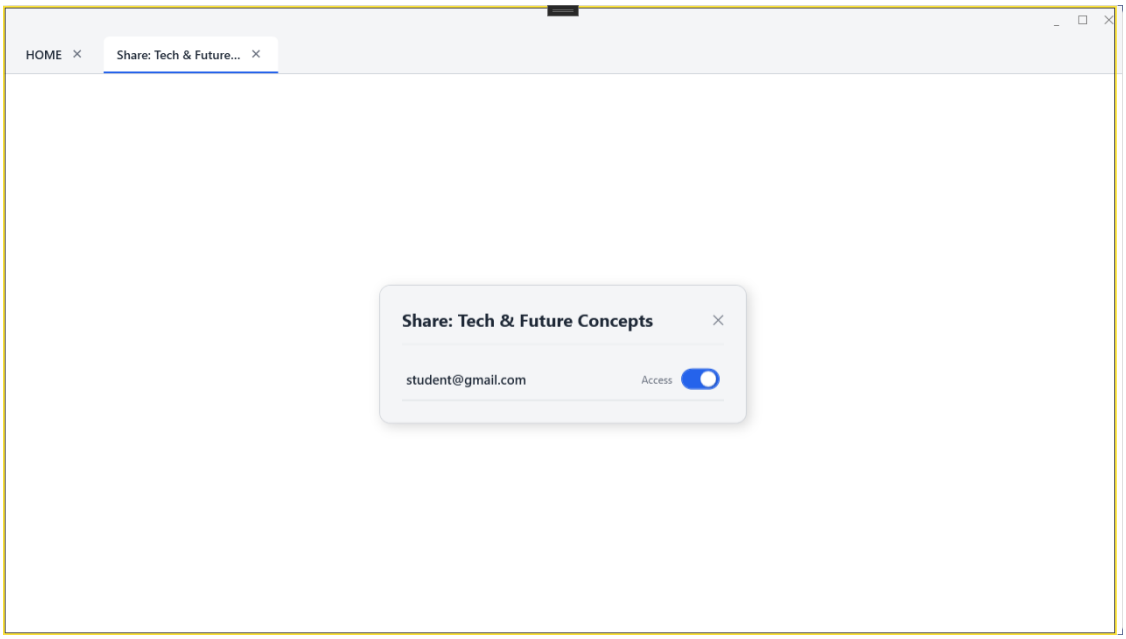

Рисунок 10 – Окно назначения доступа к словарю

Доступ к параметрам конфигурации приложения осуществляется через раздел меню «Settings». Данный модуль позволяет адаптировать интерфейс под индивидуальные предпочтения пользователя.

Окно настроек (Рисунок 11) разделено на несколько функциональных блоков:

Учетная запись: Отображает информацию о текущем пользователе (Логин, ID и Роль). Здесь же находится кнопка «Log out» для безопасного завершения сеанса.

Внешний вид (Themes): Приложение поддерживает смену тем оформления «на лету» без необходимости перезагрузки. Доступны следующие цветовые схемы:

- Light - светлая тема для работы в хорошо освещенных помещениях;
- Dark - темная тема для снижения нагрузки на зрение в темное время суток;
- Dracula и Forest - дополнительные контрастные темы.

Язык интерфейса (Language): Реализована полная локализация приложения. Пользователь может выбрать один из поддерживаемых языков: Русский, Английский, Немецкий, Испанский или Китайский. Смена языка происходит мгновенно, обновляя все текстовые метки в приложении.

Выбранные настройки автоматически сохраняются в локальном конфигурационном файле пользователя и восстанавливаются при следующем запуске.

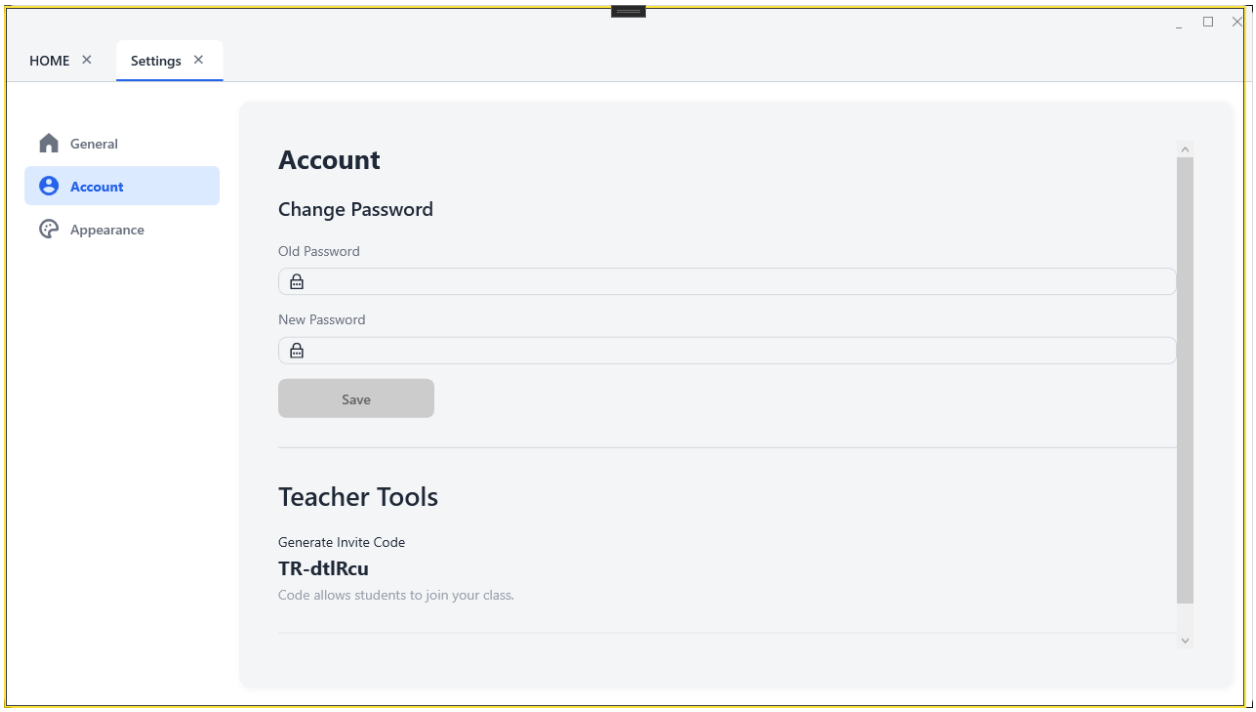

Рисунок 11 – Окно настроек

ЗАКЛЮЧЕНИЕ

Целью курсового проектирования являлась разработка подсистемы «Тренажер изучения иностранных слов», предназначенной для автоматизации процессов изучения иностранного языка, систематизации учебных материалов и контроля прогресса обучающихся.

В ходе курсового проектирования решены ключевые задачи, направленные на создание функционального и удобного в использовании программного средства. Разработанное приложение отвечает современным требованиям к образовательным сервисам и предоставляет необходимый набор функций для эффективного управления персональными словарями и грамматическими правилами.

Цель курсового проектирования достигнута, в процессе ее достижения решены следующие задачи:

- проведен сбор и анализ требований целевой аудитории, а также изучены существующие аналоги (Quizlet, Anki);
- спроектирована гибридная клиент-серверная архитектура подсистемы с использованием паттерна MVVM [4];
- выбран состав программных и технических средств (WPF, .NET 8.0, ASP.NET Core, MS SQL Server, SQLite);
- спроектирована и реализована база данных для хранения словарей, слов, правил и статистики;
- разработан интуитивно понятный пользовательский интерфейс с поддержкой темной темы;
- реализована функциональность управления словарями и грамматическими правилами;
- реализован режим тренировки с использованием системы флеш-карточек;
- реализован механизм импорта и экспорта данных в формате JSON;

- выполнено структурное и функциональное тестирование программного обеспечения;
- разработана программная и эксплуатационная документация.

В результате разработанная подсистема представляет собой не только инструмент для пополнения словарного запаса, но и эффективное средство для повышения качества усвоения учебного материала за счет использования интерактивных методик. Внедрение данного программного продукта способствует оптимизации процесса самообучения и может быть использовано в образовательных учреждениях в качестве вспомогательного инструмента для преподавателей и студентов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

2. Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л. Г. Гагарина, Е. В. Кокорева, Б. Д. Сидорова-Виснадул ; под ред. Л. Г. Гагариной. – Москва : ФОРУМ : ИНФРА-М, 2025. – 400 с. – URL: https://znanium.ru/catalog/product/2178802. – Режим доступа: по подписке. – Текст : электронный.

3. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем : учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. – Москва : ФОРУМ : ИНФРА-М, 2024. – 368 с. – URL: https://znanium.ru/catalog/product/2096940. – Режим доступа: по подписке. – Текст : электронный.

4. Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – URL: https://ibooks.ru/bookshelf/386796/reading. – Режим доступа: для зарегистрир. пользователей. – Текст : электронный.

5. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. – Москва : КУРС : ИНФРА-М, 2024. – 336 с. – URL: https://znanium.ru/catalog/product/2083407. – Режим доступа: по подписке. – Текст : электронный.