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

«РУССКИЙ ЯЗЫК. СИНТАКСИС И

ПУНКТУАЦИЯ»

Л109. 25КП01. 002 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки

программного обеспечения

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

Архангельск 2025

СОДЕРЖАНИЕ

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем техническом отчете применяются следующие сокращения и обозначения:

БД – база данных

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

ERD – диаграмма «сущность-связь»

ORM – объектно-реляционное отображение

SQL – язык структурированных запросов

WPF – платформа представления Windows

XAML – расширяемый язык разметки приложений

MVVM – модель–представление–модель представления

ВВЕДЕНИЕ

Настольные образовательные приложения являются востребованной технологией в современном учебном процессе, обеспечивая студентам и преподавателям удобный доступ к учебным материалам и возможность проверки усвоенных знаний. В условиях цифровизации образования такие приложения играют ключевую роль в повышении эффективности обучения, систематизации информации и организации контроля знаний. Развитие информационных технологий и рост потребности в качественных образовательных ресурсах создают необходимость разработки решений, которые позволяют студентам изучать материал в удобной форме, а преподавателям – организовывать и контролировать учебный процесс.

Таким образом, разработка оконного приложения для изучения разделов «Синтаксис» и «Пунктуация» представляет собой важный проект, направленный на повышение эффективности образовательного процесса в колледжах и филологических образовательных учреждениях. Внедрение такого программного обеспечения позволит студентам получать структурированный доступ к учебным материалам, закреплять знания с помощью тестовых заданий и облегчить работу преподавателей по контролю успеваемости.

Актуальность разрабатываемого курсового проекта заключается в ускорении усвоения учебного материала, решении проблемы систематизации знаний, автоматизации процесса предоставления учебной информации и обеспечении пользователя удобными инструментами для изучения и проверки знаний.

Целью курсового проектирования является разработка обучающей программы, предоставляющей доступ к учебным материалам по разделам «Синтаксис» и «Пунктуация» и возможность прохождения тестовых заданий для проверки усвоенного материала.

Для достижения поставленной цели требуется решить следующие задачи:

- проанализировать предметную область;
- определить требования к разрабатываемому программному средству;
- спроектировать структуру БД для предметной области;
- разработать БД для хранения информации о пользователях, тестах, результатах тестов и лекционных материалах;
- разработать интуитивно понятный пользовательский интерфейс;
- реализовать разграничение прав доступа пользователей;
- провести тестирование разработанного программного продукта.

Реализация указанных задач позволит создать удобный инструмент, который помогает организовать учебный процесс, делает изучение материала более структурированным и обеспечивает эффективный контроль усвоенных знаний.

1 Сбор и анализ требований

1.1 Назначение и область применения

Назначение разрабатываемого ПО заключается в предоставлении возможности изучения лекционных материалов по разделам «Синтаксис» и «Пунктуация» и прохождения тестов для закрепления пройденного материала.

ПО предназначено для использования в колледжах и филологических образовательных учреждениях.

Основными категориями пользователей ПО будут студенты и преподаватели русского языка.

1.2 Постановка задачи

Необходимо разработать ПО, предоставляющее доступ к следующей функциональности:

- авторизация и регистрация пользователей;
- просмотр лекций;
- прохождение тестов по темам лекций;
- экспорт данных об успеваемости студентов в формате *.xlsx;
- фильтрация и сортировка результатов тестов пользователей;
- добавление и удаление тестов;
- разграничение прав доступа пользователей;
- просмотр пользователей и изменение данных о них.

Интерфейс разработанного ПО должен быть интуитивно понятен.

В приложении должна быть реализована авторизация для разграничения прав доступа пользователей.

Гость может только просматривать лекции и иметь возможность авторизоваться или зарегистрироваться.

«Студент» может проходить тесты, отслеживать результаты своих пройденных тестов и просматривать таблицу лидеров.

«Преподаватель» может все то, что и «Студент», кроме прохождения тестов, а также просматривать результаты пройденных тестов пользователей и формировать отчёты по этим результатам.

«Администратор» имеет полный доступ ко всем функциям системы, включая просмотр, удаление и редактирование пользователей, добавление преподавателя, а также добавление и удаление теста, кроме прохождения тестов. На рисунке 1 представлена диаграмма вариантов использования приложения.

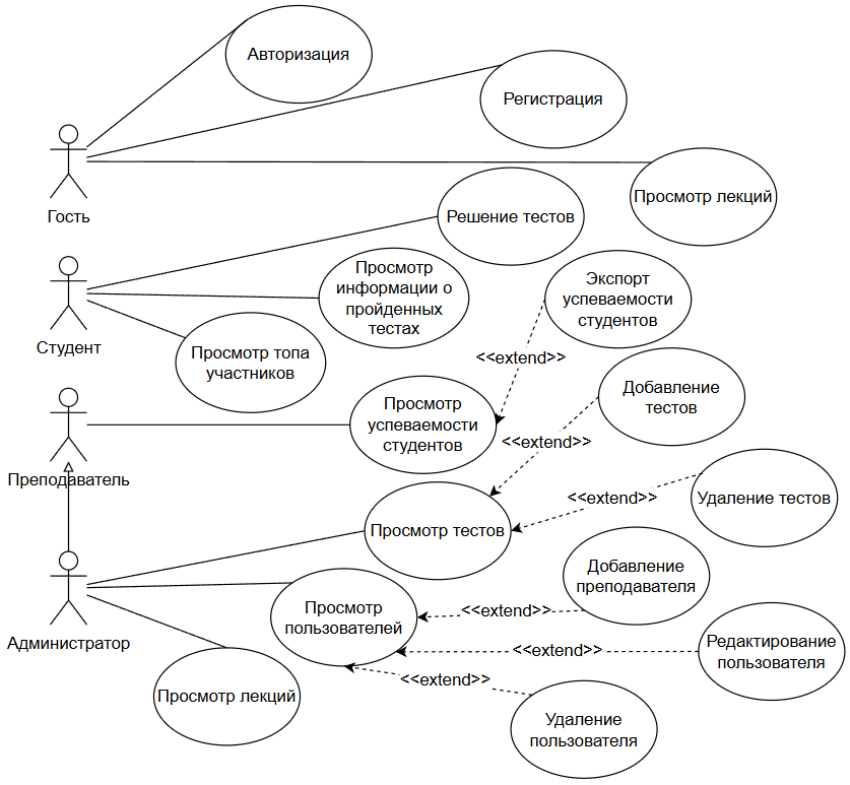

Рисунок 1 – Диаграмма вариантов использования

1.3 Выбор состава программных и технических средств

Согласно цели проекта требуется создать оконное приложение для изучения разделов русского языка «Синтаксис» и «Пунктуация».

Приложение будет разрабатываться на языке C#, который предоставляет широкий набор инструментов для создания стабильных, удобных и функциональных оконных приложений с современным интерфейсом, с использованием MVVM-архитектуры, что обеспечит удобство и расширяемость системы.

В качестве среды разработки будет использоваться Visual Studio 2022, обеспечивающая удобные средства проектирования, отладки и управления кодом.

В качестве СУБД выбрана MySQL 8.0, обеспечивающая высокую производительность, масштабируемость и надежность.

Для функционирования системы на стороне клиента необходимы следующие программные и технические средства:

- .NET 8.0;
- операционная система Windows 10/11 64-bit;
- процессор с частотой 1 ГГц;
- доступная оперативная память 2 ГБ;
- свободное место на диске 500 Мб.

Для функционирования системы на стороне сервера необходимы следующие программные технические и средства:

- операционная система Windows x86 64-бит или Linux x86 64-бит;
- MySQL Server не ниже 8.0;
- процессор с 2 ядрами и частотой 2 ГГц;
- доступная оперативная память 2 ГБ;
- минимальный объем дискового пространства 10 ГБ.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В рамках разработки ПО с использованием Figma спроектирован пользовательский интерфейс в виде wireframe [4], который демонстрирует структуру приложения, его основные элементы и доступную функциональность. На рисунке 2 в виде wireframe представлены следующие страницы ПО:

- главная страница;
- страница профиля пользователя;
- страница создания преподавателя.

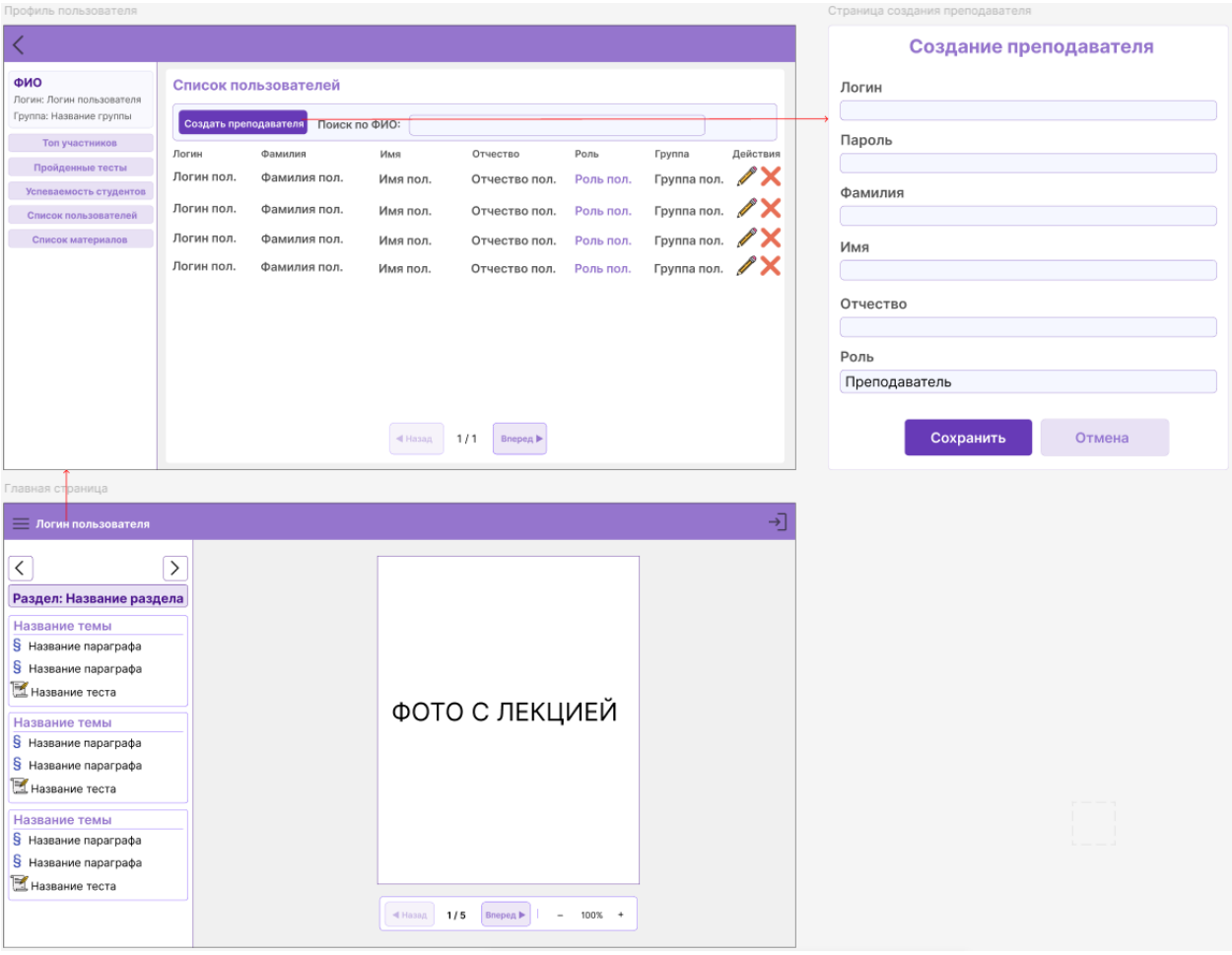

Рисунок 2 – Wireframe главной, профиля и страницы создания преподавателя

Для интерфейса оконного приложения выбрана следующая цветовая палитра:

- #9575CD – основной акцентный цвет;
- #F0F0F0 – основной фон;
- #FFFFFF– фон карточек, контейнеров и основного содержимого;
- #000000 – основной цвет текста;
- #673AB7 – цвет фона акцентных кнопок;
- #EDE7F6 – фон вторичных кнопок;
- #D8000C – цвет текста ошибок;
- #D1C4E9 – вторичный акцентный цвет.

2.2 Разработка архитектуры программного обеспечения

Архитектура ПО построена на основе клиент-серверной модели [5]. Диаграмма развертывания компонентов представлена на рисунке 3.

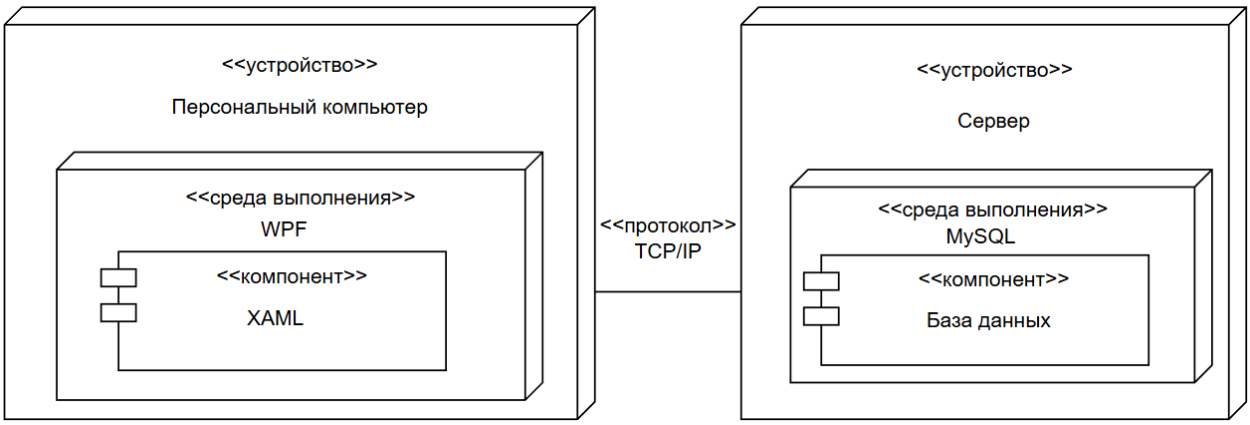

Рисунок 3 – Диаграмма развертывания компонентов

2.3 Проектирование базы данных

В рамках курсового проектирования требуется разработать БД для обучающей программы по русскому языку [3]. На рисунке 4 в виде ERD

показана физическая модель предметной области, созданная при помощи MySQL Workbench.

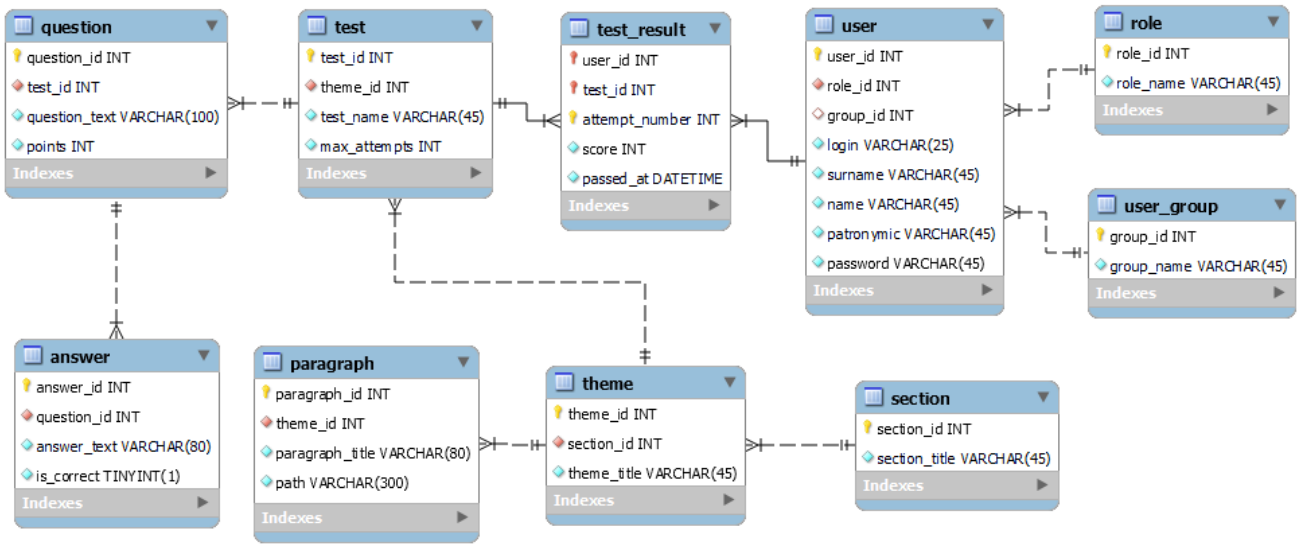

Рисунок 4 – Физическая модель данных

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

Взаимодействие с БД осуществляется непосредственно из клиентского WPF-приложения. В качестве ORM используется Entity Framework Core, обеспечивающий объектно-ориентированное взаимодействие с реляционной БД. Для работы с СУБД MySQL применяется провайдер Pomelo.EntityFrameworkCore.MySql, позволяющий корректно выполнять миграции и формировать SQL-запросы для MySQL.

В WPF-приложении реализован сервис DataAccessService, который обеспечивает доступ к данным приложения и инкапсулирует взаимодействие с БД.

Код для получения списка разделов представлен листингом 1.

Листинг 1 – Код метода получения списка разделов

```
        /// <summary>
        /// Возвращает полный список разделов (Sections) с
включёнными связанными сущностями.
        /// </summary>
        /// <returns>
        /// Список разделов с вложенными темами, параграфами и
тестами.
        /// </returns>
        public async Task<List<Section>> GetSections() => await
_context.Sections
            .AsNoTracking()
            .Include(s => s.Themes)
                .ThenInclude(t => t.Paragraphs)
            .Include(s => s.Themes)
                .ThenInclude(t => t.Tests)
            .ToListAsync();
```

3.2 Реализация интерфейса пользователя

Пользовательский интерфейс WPF-приложения реализован с использованием архитектурного шаблона MVVM [2], что обеспечивает разделение логики представления и данных. Визуальная часть создаётся на основе XAML, а управление состоянием осуществляется во ViewModel-классах.

Навигация между экранами построена на собственном сервисе NavigationService, который работает через механизм Dependency Injection и предоставляет централизованный способ смены активной ViewModel. Отображение соответствующего представления выполняется автоматически благодаря системе DataTemplate: каждому классу ViewModel сопоставлено собственное View, которое WPF подставляет в элемент ContentControl главного окна.

Главная ViewModel (MainViewModel) отслеживает изменение текущей ViewModel и обновляет заголовок окна, подстраивая интерфейс под активный раздел приложения. Все ViewModel и сервисы регистрируются в контейнере зависимостей при старте приложения, что обеспечивает слабую связность компонентов и удобство расширения проекта.

Для отображения темы используется контейнер Border с заголовком и визуальным разделителем. Параграфы и тесты представлены списками (ItemsControl), где каждый элемент – кнопка с иконкой и названием, привязанная к соответствующей команде (SelectParagraphCommand или StartTestCommand).

Код реализации карточки с темой и ее содержимым представлен листингом 2.

Листинг 2 – Код компонента для отображения карточки упражнений

```
<Border Background="White" BorderBrush="{StaticResource
AccentBrush}" BorderThickness="1"
```

```
        CornerRadius="6" Padding="12" Margin="0,0,0,12">
    <StackPanel>
        <!-- Заголовок темы -->
        <TextBlock Text="{Binding ThemeTitle}"
                   FontSize="16" FontWeight="SemiBold"
                   Margin="0,0,0,12"
                   Foreground="{StaticResource PrimaryBrush}" />
        <!-- Разделитель: визуально отделяет заголовок от
содержимого -->
        <Border Height="1" Background="{StaticResource
AccentBrush}" Margin="0,0,0,12" />
        <!-- Список параграфов темы -->
        <ItemsControl ItemsSource="{Binding Paragraphs}">
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <!-- Кнопка для выбора параграфа -->
                    <Button Command="{Binding
DataContext.SelectParagraphCommand,
RelativeSource={RelativeSource AncestorType=UserControl}}"
                            CommandParameter="{Binding}"
                            Style="{StaticResource
TransparentButtonStyle}"
                            Margin="0,0,0,8" Padding="8,6"

HorizontalContentAlignment="Stretch">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <!-- Колонка для иконки -->
                                <ColumnDefinition Width="40"/>
                                <!-- Колонка для текста -->
                                <ColumnDefinition Width="*"/>
                            </Grid.ColumnDefinitions>
                            <!-- Иконка параграфа -->
                            <Image Grid.Column="0"

Source="pack://application:,,,/Resources/Images/paragraph.png"
                                   Width="40" Height="40"
                                   VerticalAlignment="Center" />
                            <!-- Название параграфа -->
                            <TextBlock Grid.Column="1"
                                       Text="{Binding
ParagraphTitle}"
                                       FontSize="15"

VerticalAlignment="Center"

TextTrimming="CharacterEllipsis"/>
                        </Grid>
                    </Button>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
        <!-- Список тестов темы -->
```

```
        <ItemsControl ItemsSource="{Binding Tests}"
Margin="0,4,0,0">
            <ItemsControl.ItemTemplate>
                <DataTemplate>
                    <!-- Кнопка для запуска теста -->
                    <Button Command="{Binding
DataContext.StartTestCommand, RelativeSource={RelativeSource
AncestorType=UserControl}}"
                            CommandParameter="{Binding}"
                            Style="{StaticResource
TransparentButtonStyle}"
                            Margin="0,0,0,8" Padding="8,6"

HorizontalContentAlignment="Stretch">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <!-- Колонка для иконки -->
                                <ColumnDefinition Width="40"/>
                                <!-- Колонка для текста -->
                                <ColumnDefinition Width="*"/>
                            </Grid.ColumnDefinitions>

                            <!-- Иконка теста -->
                            <Image Grid.Column="0"

Source="pack://application:,,,/Resources/Images/tests.png"
                                   Width="40" Height="40"
                                   VerticalAlignment="Center" />
                            <!-- Название теста -->
                            <TextBlock Grid.Column="1"
                                       Text="{Binding TestName}"
                                       FontSize="15"

VerticalAlignment="Center"

TextTrimming="CharacterEllipsis"/>
                        </Grid>
                    </Button>
                </DataTemplate>
            </ItemsControl.ItemTemplate>
        </ItemsControl>
    </StackPanel>
</Border>
```

3.3 Разграничение прав доступа пользователей

В приложении реализована система разграничения прав доступа на уровне пользовательского интерфейса. После авторизации пользователя

определяется его роль и в зависимости от неё активируются или скрываются соответствующие элементы интерфейса.

Код метода авторизации представлен листингом 3.

Листинг 3 – Код метода авторизации

```
        /// <summary>
        /// Аутентифицирует пользователя по логину и паролю.
        /// </summary>
        /// <param name="login">Логин пользователя.</param>
        /// <param name="password">Пароль пользователя.</param>
        /// <returns>
        /// <c>true</c>, если пользователь с указанными учетными
данными найден; иначе <c>false</c>.
        /// </returns>
        /// <remarks>
        /// Если аутентификация успешна, свойство <see
cref="CurrentUser"/> устанавливается на найденного пользователя.
        /// Метод использует <see cref="DataAccessService"/> для
получения списка пользователей.
        /// </remarks>
        public async Task<bool> Login(string login, string
password)
        {
            // Получаем всех пользователей из базы
            var user = (await _dataAccessService.GetUsers())
                // Ищем пользователя с совпадающим логином и
паролем
                .FirstOrDefault(u => u.Login == login &&
u.Password == password);
            // Сохраняем найденного пользователя как текущего
            CurrentUser = user;
            // Возвращаем true, если пользователь найден
            return user != null;
        }
```

На экране профиля пользователя размещена кнопка для перехода к разделу «Успеваемость студентов», которая видна только пользователям с ролью «Преподаватель» или «Администратор».

Код реализации отображения кнопки для перехода к разделу «Успеваемость студентов» представлен листингом 4.

Листинг 4 – Код отображения кнопки для перехода к успеваемости студентов

```
<!-- Кнопка для перехода к разделу "Успеваемость студентов" -->
<Button
    Content="Успеваемость студентов" <!-- Текст кнопки,
отображаемый пользователю -->
    Command="{Binding SelectProfileSectionCommand}" <!--
Команда, выполняемая при нажатии кнопки -->
    CommandParameter="Успеваемость" <!-- Параметр, передаваемый
в команду -->
    Visibility="{Binding IsStudent, Converter={StaticResource
InverseBooleanConverter}}"
        <!-- Видимость кнопки: скрыта для студентов, видна для
других ролей (обратное булево значение) -->
    Style="{StaticResource SecondaryButtonStyle}" <!--
Применяемый стиль кнопки из ресурсов -->
    Height="44" <!-- Высота кнопки -->
    FontSize="16" <!-- Размер текста на кнопке -->
    Margin="0 0 0 10" <!-- Отступ снизу -->
/>
```

3.4 Экспорт и импорт данных

Для реализации возможности экспорта результатов тестирования пользователей и импорта тестов используется библиотека ClosedXML. Основная логика экспорта и импорта размещена в статическом классе ExcelService и реализована методами ExportToExcel и ImportFromExcel, код которых представлен листингами 5 и 6. Вызов этих методов осуществляется из модели представления UserProfileViewModel в методах ExportResults и ImportTest , код которых представлен листингами 7 и 8.

Листинг 5 – Код метода формирования Excel-документа

```
    /// <summary>
    /// Экспортирует любую коллекцию объектов в Excel.
    /// </summary>
    public static void ExportToExcel<T>(IEnumerable<T> data,
string defaultFileName = "data.xlsx")
    {
        if (!CheckDataExists(data)) return;
        var fileName = ShowSaveFileDialog(defaultFileName);
```

```
        if (fileName == null) return;
        var wb = new XLWorkbook();
        var ws = wb.Worksheets.Add("Лист1");
        FillHeaders<T>(ws);
        FillData(ws, data);
        ws.Columns().AdjustToContents();
        wb.SaveAs(fileName);
        ShowInformationWindow("Данные успешно экспортированы.");
    }
```

Листинг 7 – Код метода импорта из Excel-документа

```
    /// <summary>
    /// Импорт данных из Excel в список объектов типа T.
    /// </summary>
    public static List<T> ImportFromExcel<T>() where T : new()
    {
        var fileName = ShowOpenFileDialog();
        if (fileName == null) return new List<T>();
        var wb = new XLWorkbook(fileName);
        var ws = wb.Worksheet(1);
        if (ws.RowsUsed().Count() < 2)
            throw new Exception("Файл Excel пустой!");
        var headers = GetHeaders(ws);
        var result = new List<T>();
        int rowNumber = 2;
        while (!ws.Cell(rowNumber, 1).IsEmpty())
        {
            var row = ws.Row(rowNumber);
            var item = ParseRow<T>(row, headers);
            result.Add(item);
            rowNumber++;
        }
        return result;
    }
```

Листинг 8 – Код метода для создания Excel-документа с успеваемостью студентов

```
/// <summary>
/// Команда экспорта результатов тестирования в Excel-файл.
/// Вызывает универсальный сервис экспорта и обрабатывает
возможные ошибки.
/// </summary>
private void ExportResults()
{
    try
    {
```

```
        // Передаём отфильтрованную коллекцию результатов и имя
создаваемого файла
        ExcelService.ExportToExcel(_filteredResultsForExport,
"Результаты_студентов.xlsx");
    }
    catch (System.IO.IOException ioEx)
    {
        // Обработка ошибки: файл занят или открыт другой
программой
        if (ioEx.Message.Contains("занят другим процессом") ||
            ioEx.HResult == -2147024864 || ioEx.HResult == 32)
        {
            ShowInformationWindow("Ошибка экспорта: Файл открыт
в другом приложении. Пожалуйста, закройте его и повторите
попытку.");
        }
        else
        {
            // Любые другие ошибки ввода-вывода
            ShowInformationWindow($"Ошибка ввода-вывода при
экспорте: {ioEx.Message}");
        }
        return;
    }
    catch (Exception ex)
    {
        // Общий обработчик на случай всех непредвиденных
исключений
        ShowInformationWindow($"Общая ошибка экспорта:
{ex.Message}");
        return;
    }
}
```

Листинг 6 – Код метода импорта теста из Excel-документа

```
/// <summary>
/// Импортирует тестовые вопросы из Excel-файла и создаёт новый
тест.
/// Выполняет предварительную валидацию данных, проверяет
корректность импортированных строк
/// и сохраняет тест вместе с вопросами в базу данных.
/// </summary>
private async Task ImportTest()
{
    // Проверяем корректность введённых пользователем параметров
    if (!ValidateImportInput())
        return;
    try
    {
```

```
        // Импортируем вопросы из Excel-файла в виде списка DTO-
объектов
        var importedQuestions =
ExcelService.ImportFromExcel<TestImportDto>();
        // Проверка: если файл пустой или содержит неверные
заголовки — прекращаем импорт
        if (importedQuestions.Count == 0)
        {
            ShowError("Файл пустой или не содержит допустимых
данных!");
            return;
        }
        // Проверяем, есть ли строки с некорректными или
неполными данными
        var invalidRows = importedQuestions
            .Where(q =>
string.IsNullOrWhiteSpace(q.QuestionText) ||
                       string.IsNullOrWhiteSpace(q.AnswerText)
||
                       q.Points <= 0)
            .ToList();
        // Если найдены ошибочные вопросы — информируем
пользователя и прерываем импорт
        if (invalidRows.Any())
        {
            ShowError("Файл содержит пустые или некорректные
строки. Проверьте данные.");
            return;
        }
        // Создание нового теста и сохранение его вопросов в
базе данных
        await _dataAccessService.CreateTestWithQuestionsAsync(
            NewTestName,
            SelectedImportTheme.ThemeId,
            NewTestMaxAttempts,
            importedQuestions
        );
        // Обновляем список материалов после успешного импорта
        await LoadMaterialsManagementAsync();
        // Сбрасываем состояние формы создания/импорта теста
        CancelTestCreation();
        // Уведомляем пользователя об успешном завершении
операции
        ShowInformationWindow($"Тест '{NewTestName}' успешно
импортирован и создан.");
    }
    catch (Exception ex)
    {
        // Обработка любых непредвиденных ошибок на этапе
импорта или сохранения
        ShowError($"Ошибка при импорте: {ex.Message}");
    }
}
```

4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

В ходе курсового проектирования было выполнено структурное тестирование метода DeleteUserAsync с использованием фреймворка xUnit [1]. Для имитации работы БД применялась встроенная in-memory БД на основе библиотеки Microsoft.EntityFrameworkCore.InMemory, что позволило проводить тестирование без обращения к основной БД. Для проверки корректности работы метода был разработан тестовый сценарий DeleteUserAsync_UserExists_ReturnsTrue, код которого представлен листингом 9.

Листинг 9 – Код метода DeleteUserAsync_UserExists_ReturnsTrue

```
[Fact]
public async Task DeleteUserAsync_UserExists_ReturnsTrue()
{
    // Создаём in-memory базу данных
    var context = await GetInMemoryDbContextAsync();
    // Создаём роль
    var role = new Role { RoleId = 1, RoleName = "Student" };
    context.Roles.Add(role);
    // Создаём пользователя, который должен быть удалён
    var user = new User
    {
        UserId = 1,
        RoleId = 1,
        Login = "alice123",
        Name = "Alice",
        Surname = "Smith",
        Patronymic = "A.",
        Password = "password123"
    };
    // Добавляем пользователя в in-memory базу
    context.Users.Add(user);
    // Сохраняем изменения, имитируя работу реальной БД
    await context.SaveChangesAsync();
    // Создаём сервисы, как в реальном приложении
    var dataAccess = new DataAccessService(context);
    var service = new UserService(dataAccess);
```

```
    // Пытаемся удалить пользователя с ID = 1
    var result = await service.DeleteUserAsync(1);
    // Метод должен вернуть true — пользователь найден и удалён
    Assert.True(result);
    // В базе больше не должно быть пользователя с UserId = 1
    Assert.Null(await context.Users.FindAsync(1));
}
```

Результат тестирования метода DeleteUserAsync представлен на рисунке 5

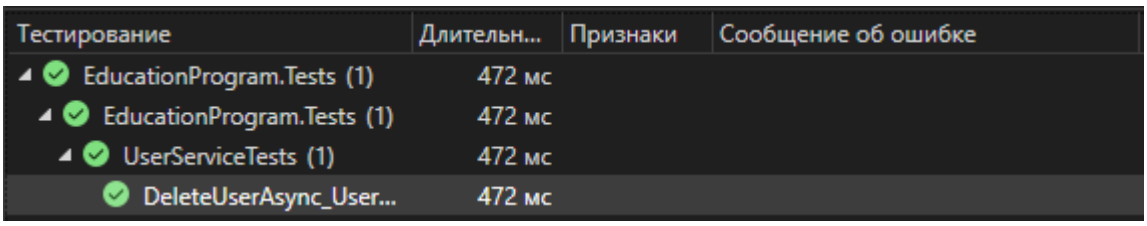

Рисунок 5 – Visual Studio 2022. Вид вкладки обозревателя тестов

4.2 Функциональное тестирование

В ходе курсового проектирования проведено функциональное тестирование ключевых пользовательских сценариев для авторизованных и неавторизованных пользователей. Тестирование главного окна выполнено методом «чёрного ящика». Результаты тестирования представлены в таблице 1.

Таблица 1 – Набор тестов главной страницы

Действие	Ожидаемый результат	Полученный результат
Нажатие на иконку входа	Перенаправление пользователя на страницу авторизации	Совпадает с ожидаемым
Нажатие на кнопку теста будучи неавторизованным	Появление диалогового окна с текстом «Требуется авторизация для прохождения теста!»	Совпадает с ожидаемым

Продолжение таблицы 1

Действие	**Ожидаемый результат**	**Полученный результат**
Нажатие на кнопку с надписью «Гость»	Появление диалогового окна с текстом «Вы не авторизованы!»	Совпадает с ожидаемым
Нажатие на иконку выхода	Появление диалогового окна с текстом «Вы точно хотите выйти из аккаунта?» и примечанием «При выходе вы потеряете возможность проходить тесты, отслеживать прогресс и накапливать очки»	Совпадает с ожидаемым
Нажатие на кнопку теста будучи авторизованным	Появление на главной панели информации о тесте, а также кнопка для прохождения теста	Совпадает с ожидаемым
Нажатие на кнопку теста будучи авторизованным, но при этом исчерпаны все попытки для прохождения теста	Появление диалогового окна с текстом «Вы исчерпали все попытки для этого теста!	Совпадает с ожидаемым
Нажатие на кнопку с надписью в виде логина пользователя	Перенаправление пользователя на страницу профиля	Совпадает с ожидаемым

По результатам тестирования можно сделать вывод, что разработанное приложение функционирует корректно и соответствует ожидаемому поведению.

5 Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

Для функционирования системы на стороне клиента необходимы следующие программные и технические средства:

- .NET 8.0;
- операционная система Windows 10/11 64-bit;
- процессор с частотой 1 ГГц;
- доступная оперативная память 2 ГБ;
- свободное место на диске 500 Мб.

Для функционирования системы на стороне сервера необходимы следующие программные технические и средства:

- операционная система Windows x86 64-бит или Linux x86 64-бит;
- MySQL Server не ниже 8.0;
- процессор с 2 ядрами и частотой 2 ГГц;
- доступная оперативная память 2 ГБ;
- минимальный объем дискового пространства 10 ГБ.

В качестве учётных данных используются следующие данные:

- имя пользователя: angel;
- пароль: 12345678.

5.2 Инструкция по работе

При открытии оконного приложения пользователя в качестве гостя сразу встречает главный экран с лекциями и тестами. Главный экран представлен рисунком 6.

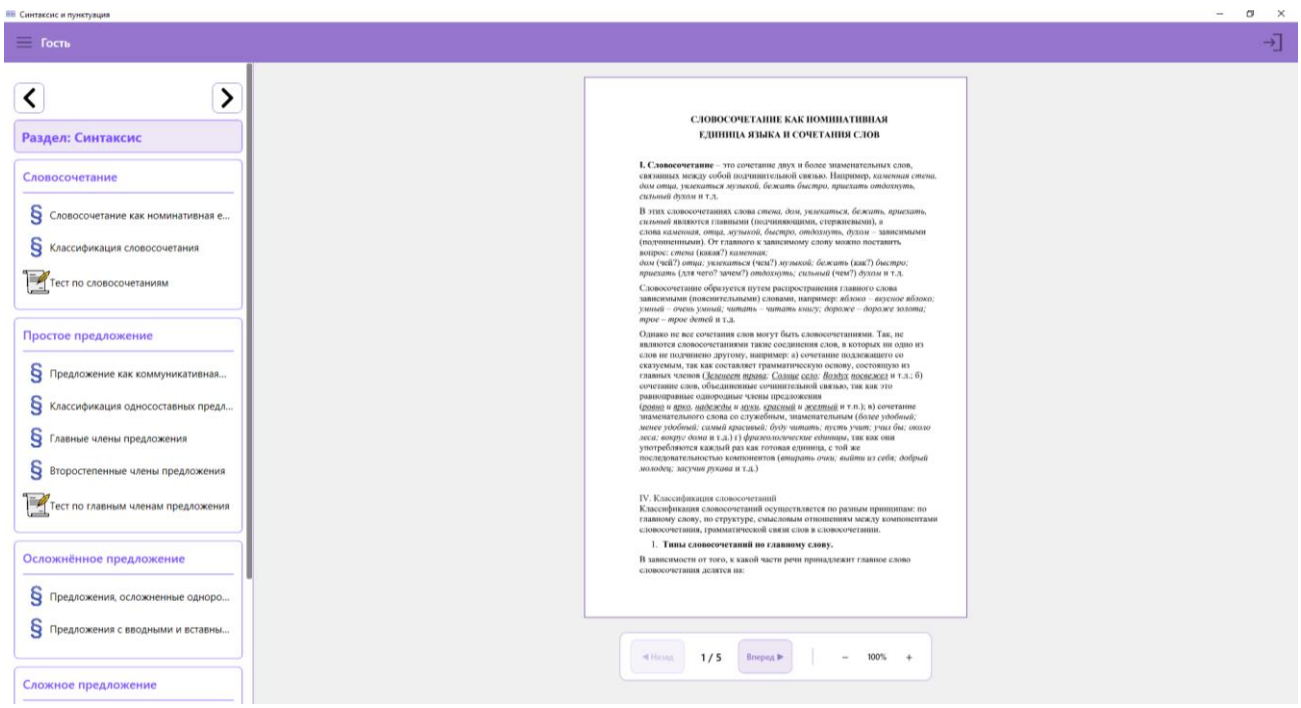

Рисунок 6 – Синтаксис и пунктуация. Вид главной страницы

Для того, чтобы начать проходить тесты и отслеживать свои результаты тестирования, пользователю требуется пройти авторизацию. Перейти к авторизации можно, нажав на главной странице в правом верхнему углу иконку входа. На странице авторизации пользователю требуется ввести свои учетные данные. После успешной авторизации пользователя вернет на главную страницу. Страница авторизации представлена рисунком 7.

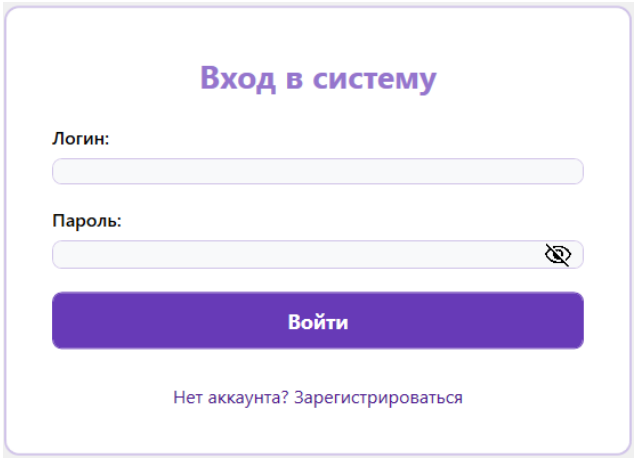

Рисунок 7 – Синтаксис и пунктуация. Вид главной страницы

Теперь пользователю доступно прохождение тестов. Для того, чтобы перейти к прохождению теста, требуется выбрать на боковой панели кнопку с тестом, после чего на главной панели отобразятся сведения о выбранном тесте. Отображение сведений о тесте на главной панели представлены рисунком 8.

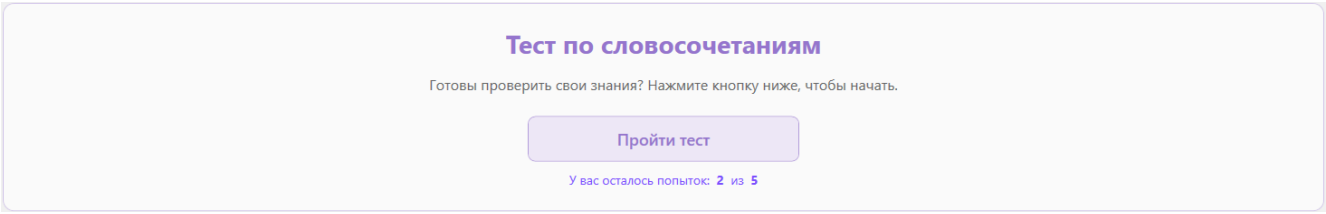

Рисунок 8 – Синтаксис и пунктуация. Вид сведений о тесте

Для того, чтобы перейти к тестированию, требуется нажать на кнопку с текстом «Пройти тест». Отображение экрана с тестированием представлено рисунком 9.

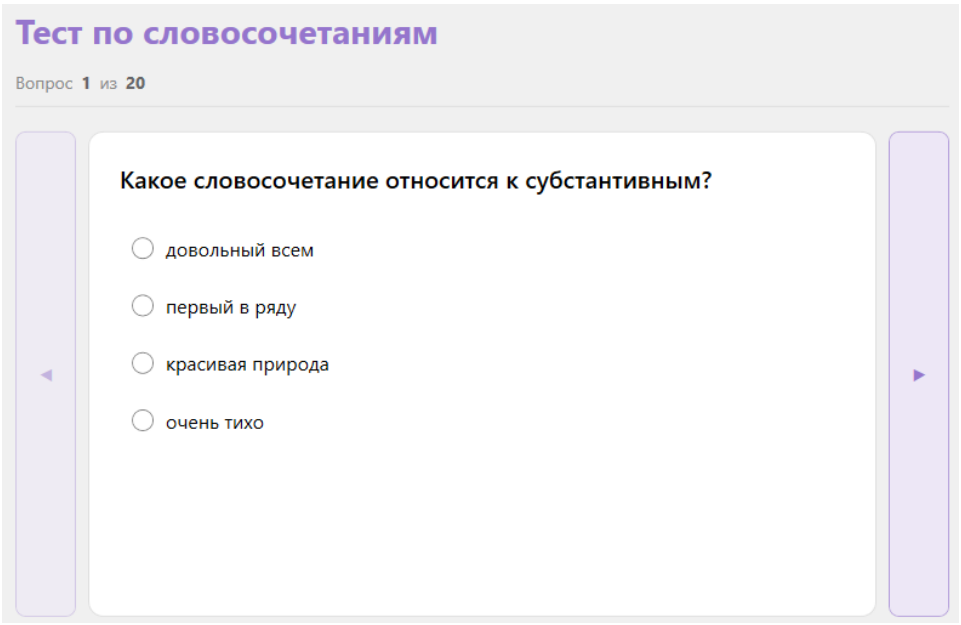

Рисунок 9 – Синтаксис и пунктуация. Вид экрана тестирования

После нажатия кнопки «Завершить тест» пользователю выводится диалоговое окно с его результатами тестирования, при закрытии которого

пользователя возвращает на главный экран. Вид окна с результатами тестирования представлен рисунком 10.

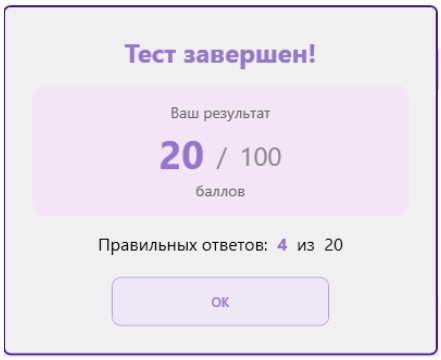

Рисунок 10 – Синтаксис и пунктуация. Вид окна результатов тестирования

Для того, чтобы посмотреть свои результаты за тесты, пользователю требуется перейти в профиль, где по умолчанию будет выбран раздел с пройденными тестами пользователя. Этот раздел виден только пользователям с ролью «Студент». Для этого ему на главном экране нужно нажать на свой логин в левом верхнем углу. Профиль пользователя представлен рисунком 11.

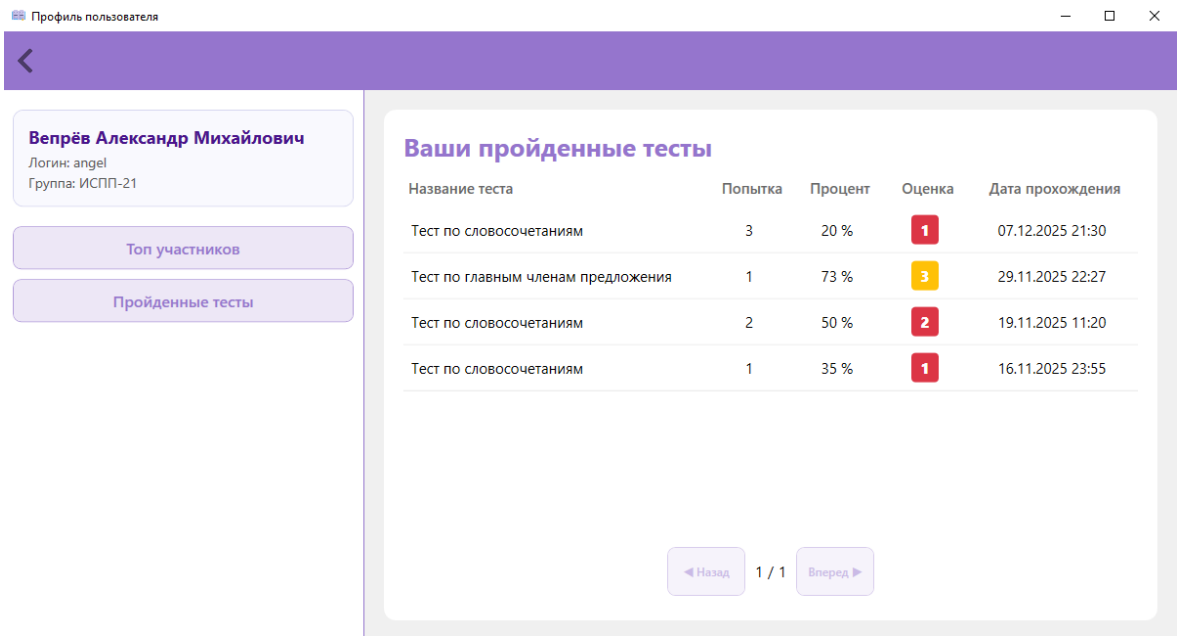

Рисунок 11 – Синтаксис и пунктуация. Вид профиля пользователя

Также в профиле пользователь может посмотреть таблицу лидеров, нажав кнопку с соответствующим текстом. Отображение таблицы лидеров представлено рисунком 12.

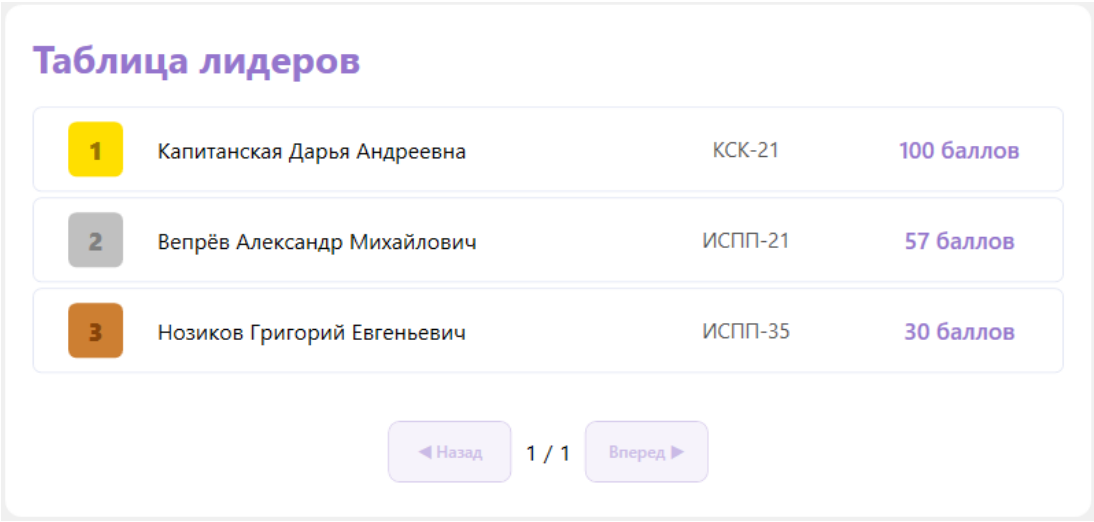

Рисунок 12 – Синтаксис и пунктуация. Вид таблицы лидеров

Пользователю с ролью «Администратор» доступны скрытые кнопки для просмотра пользователей и материалов и кнопка для отслеживания успеваемости студентов, которая ещё видна пользователю с ролью «Преподаватель». Отображение скрытых кнопок представлено рисунком 13.

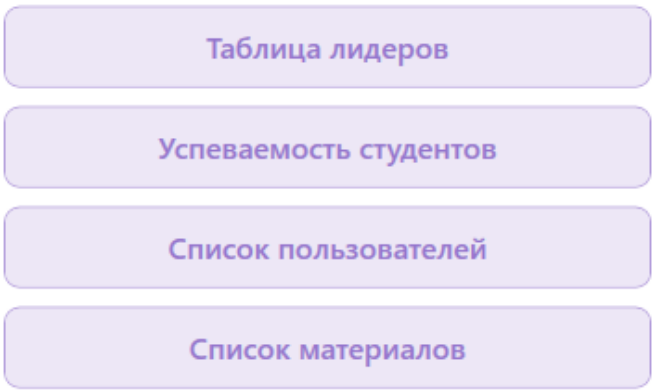

Рисунок 13 – Синтаксис и пунктуация. Вид скрытых кнопок

ЗАКЛЮЧЕНИЕ

Целью курсового проектирования являлась разработка оконного приложения для изучения разделов «Синтаксис» и «Пунктуация».

В ходе курсового проектирования решены ключевые задачи, направленные на создание функционального и удобного в использовании приложения. Разработанное программное обеспечение соответствует современным требованиям образовательного процесса и предоставляет необходимые функции для эффективного изучения учебного материала и проверки усвоенных знаний.

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

- проанализирована предметная область;
- определены требования к разрабатываемому программному средству;
- спроектирована структура БД для предметной области;
- разработана БД для хранения информации о пользователях, тестах, результатах тестов и лекционных материалах;
- разработан интуитивно понятный пользовательский интерфейс;
- реализовано разграничение прав доступа пользователей;
- проведено тестирование разработанного программного продукта.

В результате разработанное оконное приложение представляет собой не только удобный инструмент для систематизации и закрепления учебного материала, но и важный элемент повышения общей эффективности образовательного процесса. Оно способствует упорядочиванию знаний студентов, ускоряет усвоение материала и обеспечивает преподавателей необходимыми современными средствами для контроля успеваемости.

Полученные в ходе работы знания и практический опыт имеют большую практическую ценность и могут быть применены в дальнейшей профессиональной деятельности в сфере разработки образовательного программного обеспечения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бек, К. Экстремальное программирование: разработка через тестирование. – Санкт-Петербург : Питер, 2021. – 224 с. – URL: https://ibooks.ru/bookshelf/376974/reading (дата обращения: 21.11.2025). – 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 (дата обращения: 02.11.2025). – 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. – Текст : электронный. – URL: https://znanium.ru/catalog/product/2096940 (дата обращения: 28.10.2025). – Режим доступа: по подписке.

4. Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – Текст : электронный. – URL: https://ibooks.ru/bookshelf/386796/reading (дата обращения: 26.10.2025). – Режим доступа: для зарегистрир. пользователей.

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 (дата обращения: 01.11.2025). – Режим доступа: по подписке. – Текст : электронный.