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 ПОДСИСТЕМЫ ДЛЯ СУДЕЙСТВА СОРЕВНОВАНИЙ ПО ГРЕКО-РИМСКОЙ И ВОЛЬНОЙ БОРЬБЕ

Л109. 25КП01. 016 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки программного обеспечения

Студент	ИСПП-21		08.12.2025	Н.Ю. Махин
	(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 курсовом проекте применяются следующие сокращения и обозначения:

БД – база данных

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

CRUD – создание, чтение, обновление и удаление

ERD – диаграмма «сущность-связь»

IDE – интегрированная среда разработки

MSSQL – реляционная система управления базами данных от Microsoft

SQL – структурированный язык запросов

SSD – твердотельный накопитель

SSMS – студия управления SQL Server

WPF – платформа пользовательского интерфейса Windows

XAML – расширяемый язык разметки приложений

ВВЕДЕНИЕ

Традиционные методы судейства, основанные на бумажных протоколах и ручном подсчете, не позволяют обеспечить оперативную обработку данных и синхронизацию информации между членами судейской бригады, что может приводить к задержкам и спорным ситуациям. Существующие же специализированные решения часто являются узкопрофильными, дорогостоящими или не адаптированы под конкретные правила и особенности греко-римской и вольной борьбы, что снижает их практическую доступность для организаторов соревнований регионального и местного уровня.

Актуальность разработки подсистемы для судейства соревнований по греко-римской и вольной борьбе обусловлена необходимостью внедрения современных цифровых инструментов, которые обеспечивают оперативный, точный и прозрачный процесс оценивания спортивных поединков в условиях высокой динамики и строгого регламента соревнований. Внедрение автоматизированных технологий в судейскую практику позволяет не только фиксировать баллы и технические действия в реальном времени, но и минимизировать влияние человеческого фактора, обеспечивая беспристрастность и объективность судейства.

Целью курсового проектирования является разработка оконного приложения для судейства и проведения соревнований по греко-римской и вольной борьбе.

Для достижения поставленной цели требуется решить следующие задачи:

- изучить правила, регламенты и особенности судейства соревнований по греко-римской и вольной борьбе;
- спроектировать диаграмму вариантов использования;
- спроектировать физическую модель предметной области;
- спроектировать диаграмму развертывания компонентов;

- разработать БД для хранения информации о участниках соревнований и пользователях подсистемы;
- реализовать импорт и экспорт данных;
- реализовать разграничение прав доступа пользователей;
- реализовать фильтрацию, поиск и сортировку данных;
- реализовать защиту данных;
- выполнить тестирование и отладку ПО и проанализировать результаты тестирования;
- разработать программную и эксплуатационную документацию.

Требуется разработать подсистему, представляющую собой оптимальное, адаптированное под актуальные правила решение для автоматизации судейского процесса. Система должна иметь интуитивно понятный интерфейс с мощным инструментарием, что способствует повышению качества, скорости и объективности проведения соревнований.

1 Анализ и разработка требований

1.1 Назначение и область применения

Разрабатываемое оконное приложение предназначено для судейства и проведения соревнований по греко-римской и вольной борьбе. Область применения включает турниры регионального, городского и клубного уровня, где требуется точное, оперативное и прозрачное судейство.

Программа призвана заменить традиционные бумажные протоколы и ручной подсчет очков, минимизировать человеческие ошибки и обеспечить синхронизацию действий между членами судейской коллегии (главным судьей, боковыми судьями, секундантом). Автоматизация процесса ведения матча позволит сократить временные затраты на организацию соревнований и повысить качество их проведения.

1.2 Постановка задачи

Необходимо разработать оконное приложение, которое предоставит доступ к следующей функциональности:

- авторизация и добавление новых пользователей;
- ведение БД борцов с возможностью добавления, редактирования, удаления, фильтрации, поиска и импорта/экспорта данных;
- настройка параметров матча, а именно выбор борцов, установка весовой категории и стадии соревнований для текущего поединка;
- вывод информации о текущем матче.

При запуске приложения отображается экран авторизации, на котором пользователь вводит свои персональные данные либо входит как гость.

После успешной авторизации пользователю доступны окно судейство и таблица счёта.

Авторизованный пользователь может проводить матч, управляя всеми его параметрами в реальном времени, а пользователь с правами администратора управлять списком участников и учетными записями судей.

На рисунке 1 изображена диаграмма вариантов использования приложения.

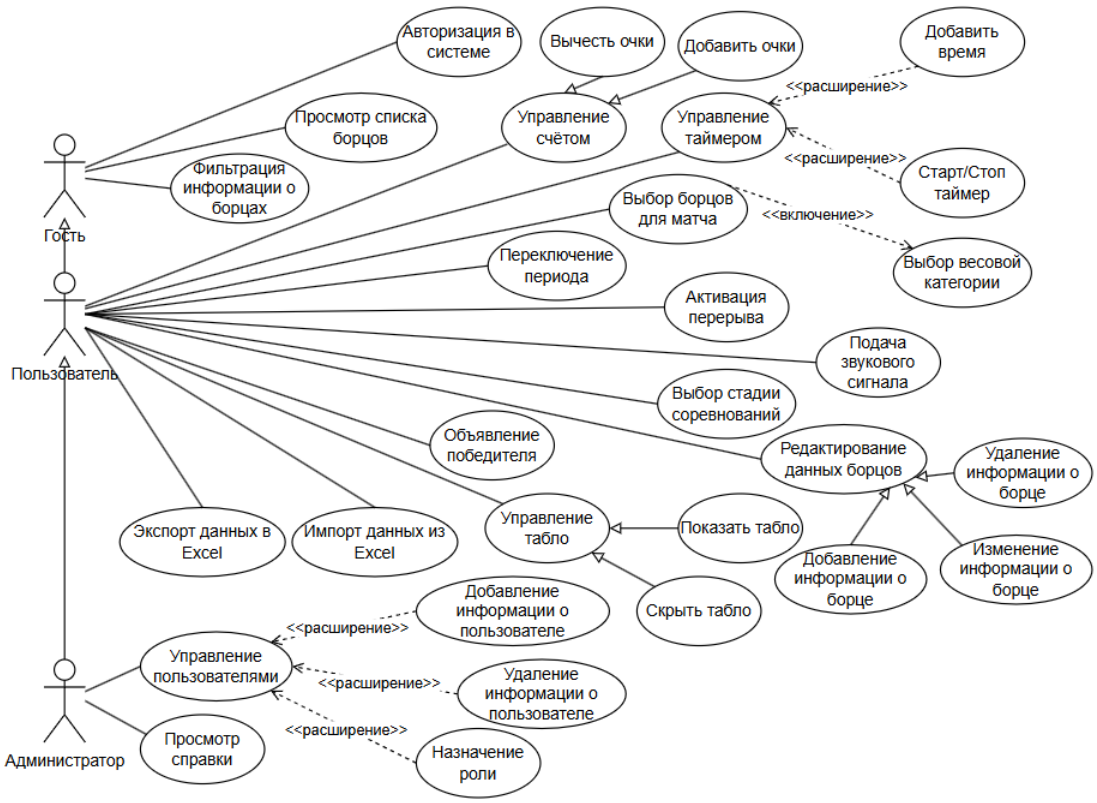

Рисунок 1 – Диаграмма вариантов использования

1.3 Выбор состава программных и технических средств

Приложение будет написано на языке программирования C#, так как он является основным языком платформы .NET и оптимально подходит для создания производительных оконных приложений с графическим интерфейсом под Windows. C# предоставляет строгую типизацию, современные языковые конструкции и обширную экосистему библиотек, что обеспечивает высокую надежность и скорость разработки.

Для создания пользовательского интерфейса используется технология WPF, которая является стандартным и наиболее мощным фреймворком для построения современных оконных приложений на платформе .NET. WPF поддерживает сложную векторную графику, стилизацию, привязку данных и шаблоны, что позволяет реализовать динамичный и интуитивно понятный интерфейс для судейства, включая таймеры, счет и панели управления.

Для разработки приложения будет использоваться IDE Visual Studio, так как она предоставляет полную поддержку WPF, встроенный визуальный редактор интерфейсов, мощные инструменты отладки, профилирования и тестирования, а также глубокую интеграцию с системами контроля версий

В качестве СУБД выбран Microsoft SQL Server. Она удобна в подключении к оконному приложению на C# через готовые библиотеки, поддерживает высокую производительность благодаря оптимизации запросов и проста в использовании.

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

- ОС Windows 10/11 (64-bit);
- .NET Desktop Runtime 8.0 или новее;
- 2-ядерный процессор частотой не ниже 2 ГГц;
- оперативная память в объеме 4 ГБ;
- свободное место в хранилище 1 ГБ;
- постоянное интернет-подключение.

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

- ОС Windows 10/11 (64-bit);
- 2-ядерный процессор частотой не ниже 2 ГГц;
- оперативная память объемом 8 ГБ;
- SSD объемом 25 ГБ;
- дополнительное ПО: SSMS.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В рамках разработки при помощи графического редактора draw.io спроектирован набор wireframe [4]. Эти визуальные представления показывают расположение и взаимодействие основных элементов.

Wireframe экранов приложения представлен на рисунке 2.

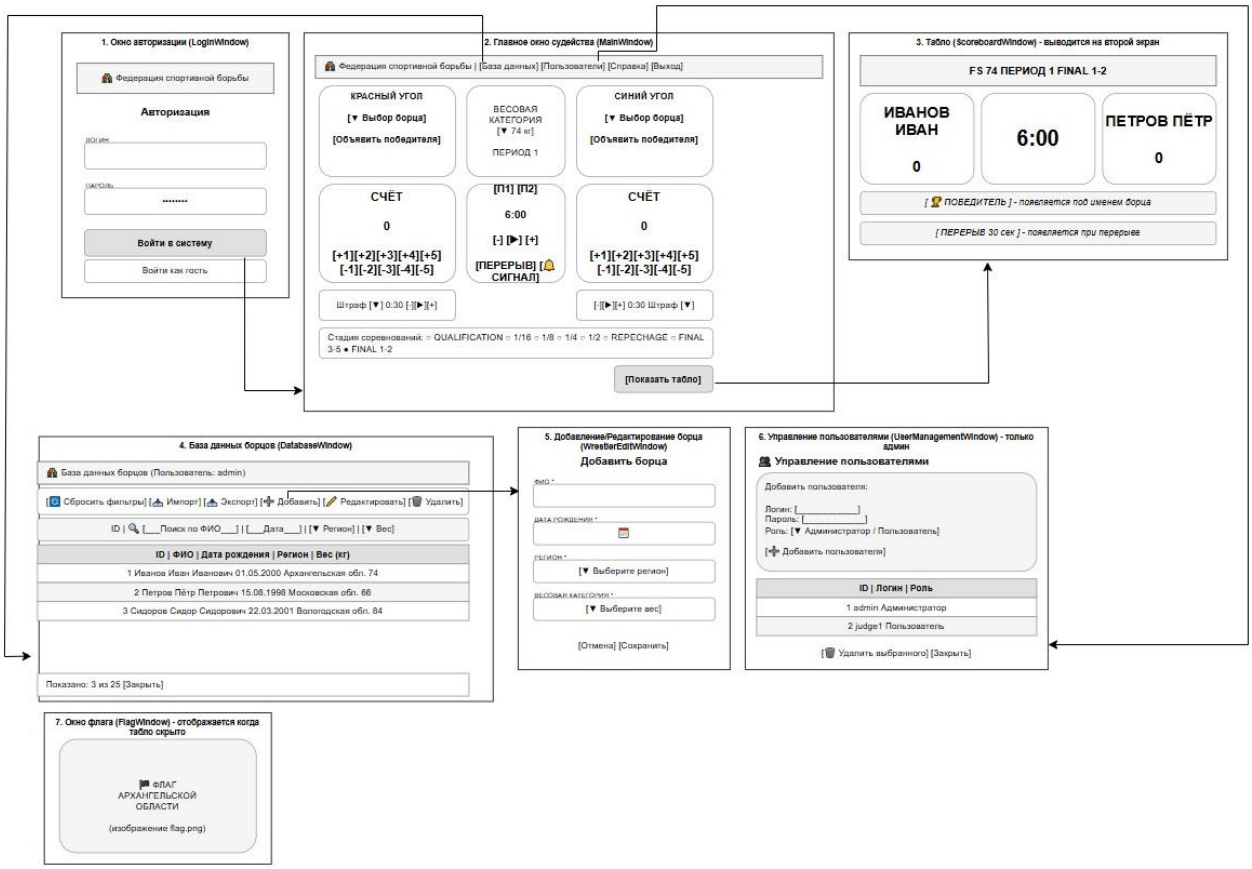

Рисунок 2 – draw.io. Wireframe интерфейса экранов приложения

Интерфейс приложения построен на тёмной теме. Основной фон – угольно-чёрный (#0a0a0a). Функциональные панели выделены тёмно-серым (#1a1a1a). Основной текст выполнен белым (#FFFFFF) для максимальной читаемости.

Ключевые элементы используют смысловые акценты: красный (#DC2626) и синий (#2563EB) идентифицируют борцовские углы. Золотой (#FFD700) выделяет главный таймер и стадию турнира. Зелёный (#22C55E) маркирует кнопки добавления и старта. Ярко-красный (#EF4444) обозначает удаление и выход. Оранжевый (#F59E0B) используется для функций перерыва и редактирования.

2.2 Разработка архитектуры программного обеспечения

Архитектура приложения построена на клиент-серверной модели [5]. Серверная часть реализована с использованием СУБД MSSQL. БД содержит таблицы для хранения информации о пользователях, ролях, борцах, весовых категориях и регионах. Клиентская часть представляет собой оконное приложение на платформе WPF с использованием языка программирования C# и платформы .NET 8.0. Взаимодействие клиентского приложения с БД осуществляется посредством технологии Entity Framework Core. Клиент формирует запросы через контекст БД, который преобразует их в SQL-запросы и выполняет соответствующие операции в БД. Результаты запросов возвращаются в виде объектов модели данных, что обеспечивает безопасный доступ к данным и упрощает разработку.

Диаграмма развертывания компонентов представлена на рисунке 3.

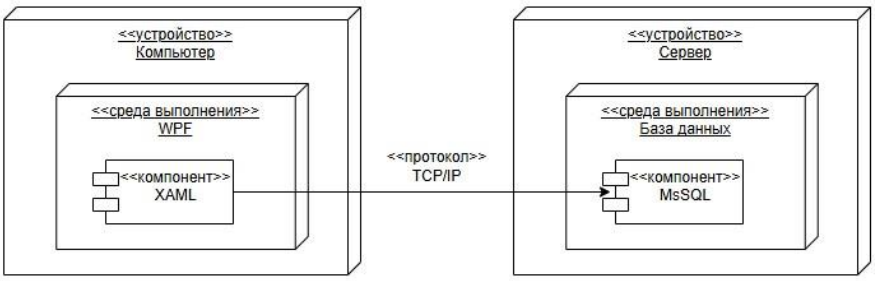

Рисунок 3 – Диаграмма развертывания компонентов

2.3 Проектирование базы данных

В рамках разработки оконного приложения при помощи SSMS спроектирована физическая модель БД, которая отражает структуру хранения данных и взаимосвязи между сущностями [3]. Спроектированная физическая модель представлена на рисунке 4 в виде ERD.

Модель включает 5 таблиц, сгруппированных по модулям приложения, где центральная таблица «Wrestlers» связана с таблицами «Regions» и «WeightCategories» через внешние ключи, а таблица «Users» связана с таблицей «Roles».

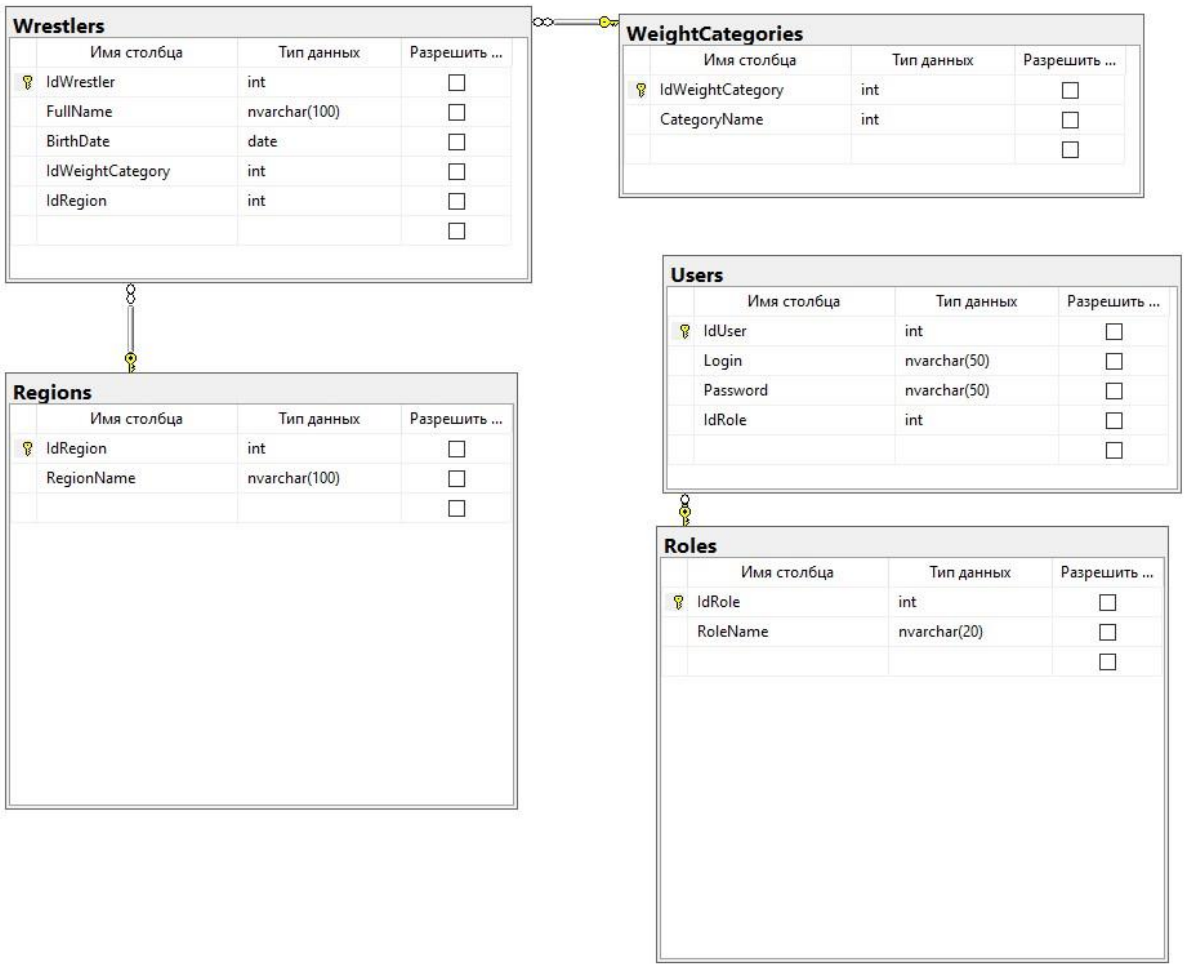

Рисунок 4 – SSMS. Физическая модель БД

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

В рамках курсового проектирования разработано настольное приложение на языке программирования C# с использованием платформы WPF и технологии Entity Framework Core для работы с БД.

Серверная часть построена с использованием СУБД Microsoft SQL Server. Взаимодействие с БД реализовано через Entity Framework Core, которое обеспечивает безопасный доступ к данным.

На стороне клиентского приложения реализованы сервисные классы, содержащие методы для выполнения CRUD-операций [2] с сущностями БД. Код метода загрузки данных из БД представлен листингом 1.

Листинг 1 – Код метода загрузки данных из БД

```
private void LoadData()
{
    try
    {
        _isLoading = true;

        using var context = new WrestlingDbContext();

        // Загружаем борцов
        _allWrestlers = context.Wrestlers
            .Include(w => w.WeightCategory)
            .Include(w => w.Region)
            .OrderBy(w => w.FullName)
            .ToList();
        // Загружаем весовые категории
        _weightCategories = context.WeightCategories.OrderBy(wc
=> wc.CategoryName).ToList();
        // Загружаем регионы
        _regions = context.Regions.OrderBy(r =>
r.RegionName).ToList();
        // Заполняем фильтр категорий
        filterCategoryComboBox.Items.Clear();
```

```
        filterCategoryComboBox.Items.Add(new    ComboBoxItem    {
Content = "Все", Tag = 0 });
        foreach (var cat in _weightCategories)
        {
            filterCategoryComboBox.Items.Add(new  ComboBoxItem  {
Content = $"{cat.CategoryName}", Tag = cat.IdWeightCategory });
        }

        filterCategoryComboBox.SelectedIndex = 0;

        // Заполняем фильтр регионов
        filterRegionComboBox.Items.Clear();
        filterRegionComboBox.Items.Add(new ComboBoxItem { Content
= "Все", Tag = 0 });
        foreach (var region in _regions)
        {
            filterRegionComboBox.Items.Add(new   ComboBoxItem   {
Content = region.RegionName, Tag = region.IdRegion });
        }
        filterRegionComboBox.SelectedIndex = 0;

        _isLoading = false;
        ApplyFilters();

        statusTextBlock.Text      =      $"Загружено      борцов:
{_allWrestlers.Count}";
    }
    catch (Exception ex)
    {
        _isLoading = false;
        System.Windows.MessageBox.Show($"Ошибка загрузки данных:
{ex.Message}",           "Ошибка",           MessageBoxButton.OK,
MessageBoxImage.Error);
         }
     }
```

Модель данных включает набор entity-классов, которые осуществляют автоматическое отображение на соответствующие таблицы в БД.

3.2 Реализация интерфейса пользователя

Интерфейс пользователя оконного приложения разработан с использованием технологии WPF и языка разметки XAML для декларативного описания графических интерфейсов.

Для стилизации элементов управления используются ресурсы и шаблоны WPF, что обеспечивает единообразный внешний вид всех компонентов приложения. Интерфейс выполнен в тёмной цветовой схеме с использованием акцентных цветов для функциональных элементов.

В приложении реализована многооконный режим с главным окном судейства, окном табло для вывода на второй экран, окном БД борцов, окном управления пользователями и вспомогательными диалоговыми окнами.

Для отображения данных о борцах разработан используемый компонент таблицы с возможностью фильтрации по столбцам. Код разметки окна с информацией о борцах представлен листингом 2.

Листинг 2 – Код разметки окна с информацией о борцах

```
<Border   Grid.Row="1"   Background="#1a1a1a"   CornerRadius="10"
Padding="15" Margin="0,0,0,15">
    <StackPanel>
        <Grid Margin="0,0,0,15">
            <StackPanel Orientation="Horizontal">
                <Button Content="🔄 Сбросить фильтры"
                        Style="{StaticResource DarkButton}"
                        Click="ResetFilters_Click"
                        Margin="0,0,10,0"/>
            </StackPanel>
            <StackPanel x:Name="crudButtonsPanel"
                        Orientation="Horizontal"
                        HorizontalAlignment="Right">
            <!--Создание  кнопок  для  добавления  редактирования
данных о участниках-->
                <Button Content="📥 Импорт"
                        Style="{StaticResource DarkButton}"
                        Background="#3B82F6"
                        Click="ImportButton_Click"
                        Margin="0,0,5,0"/>
                <Button Content="📤 Экспорт"
                        Style="{StaticResource DarkButton}"
                        Background="#8B5CF6"
                        Click="ExportButton_Click"
                        Margin="0,0,15,0"/>
                <Button Content="➕ Добавить"
                        Style="{StaticResource DarkButton}"
                        Background="#22C55E"
                        Click="AddButton_Click"
```

```
                    Margin="0,0,5,0"/>
                <Button Content="🖉 Редактировать"
                    Style="{StaticResource DarkButton}"
                    Background="#F59E0B"
                    Click="EditButton_Click"
                    Margin="0,0,5,0"/>
                <Button Content="🗑 Удалить"
                    Style="{StaticResource DarkButton}"
                    Background="#EF4444"
                    Click="DeleteButton_Click"/>
            </StackPanel>
        </Grid>

        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="50"/>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="150"/>
                <ColumnDefinition Width="100"/>
            </Grid.ColumnDefinitions>

            <Border Grid.Column="1" Background="#2a2a2a"
                    CornerRadius="6"
                    BorderBrush="#3B82F6" BorderThickness="2">
                <StackPanel Orientation="Horizontal">
                    <TextBlock Text="🔍"
                               Foreground="#3B82F6"
                               FontSize="16"
                               VerticalAlignment="Center"
                               Margin="12,0,8,0"/>
                <!--Создание поля ввода для поиску по имени-->
                    <TextBox x:Name="filterNameTextBox"
                             Background="Transparent"
                             Foreground="White"
                             TextChanged="Filter_Changed"
                             ToolTip="Введите ФИО для поиска"
                             Width="250"/>
                </StackPanel>
            </Border>
            <!--Выпадающий список для выбора региона-->
            <ComboBox x:Name="filterRegionComboBox"
                Grid.Column="3"
                SelectionChanged="Filter_Changed"
                ToolTip="Фильтр по региону"/>
            <ComboBox x:Name="filterCategoryComboBox"
                 Grid.Column="4"
                 SelectionChanged="Filter_Changed"
                 ToolTip="Фильтр по весу"/>
        </Grid>
    </StackPanel>
</Border>
```

3.3 Разграничение прав доступа пользователей

Разграничение прав доступа в приложении построено на основе ролевой модели управления доступом. В системе предусмотрены три роли: Администратор, Пользователь (Судья) и Гость.

Для реализации механизма аутентификации используется класс CurrentUser, который хранит информацию о текущем авторизованном пользователе. Код метода аутентификации пользователя представлен листингом 3.

Листинг 3 – Код метода аутентификации пользователя

```
private void LoginButton_Click(object sender, RoutedEventArgs e)
{
    string login = loginTextBox.Text.Trim();
    string password = passwordPasswordBox.Password;
    if               (string.IsNullOrEmpty(login)               ||
string.IsNullOrEmpty(password))
    {
        ShowError("Введите логин и пароль");
        return;
    }
    try
    {
        using var context = new WrestlingDbContext();
        // Поиск пользователя в БД
        var user = context.Users
            .Include(u => u.Role)
            .FirstOrDefault(u => u.Login == login && u.Password ==
password);
        if (user != null)
        {
            // Сохранение данных текущего пользователя
            CurrentUser.User = user;
            CurrentUser.IsGuest = false;

            // Открытие главного окна
            MainWindow mainWindow = new MainWindow();
            mainWindow.Show();
            this.Close();
        }
        else
        {
```

```
                ShowError("Неверный логин или пароль");
                passwordPasswordBox.Clear();
            }
        }
        catch (Exception ex)
        {
            ShowError($"Ошибка: {ex.Message}");
        }
}
```

Для обеспечения разграничения доступа реализован статический класс CurrentUser, хранящий информацию о текущей сессии. Код класса представлен листингом 4.

Листинг 4 – Код класса хранения данных текущего пользователя

```
public static class CurrentUser
{
    public static User? User { get; set; }
    public static bool IsGuest { get; set; } = false;
    public static bool IsAdmin => User?.Role?.RoleName ==
"Администратор";
    public static bool CanEdit => !IsGuest && User != null;

    public static void Logout()
    {
        User = null;
        IsGuest = false;
    }
}
```

Проверка прав доступа выполняется при инициализации каждого окна. Код метода настройки прав доступа представлен листингом 5.

Листинг 5 – Код метода настройки прав доступа

```
private void SetupUserPermissions(){
// Показываем кнопку управления пользователями только для
администов
    if (CurrentUser.IsAdmin)
    {usersButton.Visibility = Visibility.Visible;}
    // Гости могут только смотреть БД, судейство недоступно
    if (CurrentUser.IsGuest) {
```

```
        judgingRow1.Visibility = Visibility.Collapsed;
        judgingRow2.Visibility = Visibility.Collapsed;
        judgingRow3.Visibility = Visibility.Collapsed;
        judgingRow4.Visibility = Visibility.Collapsed;
        judgingRow5.Visibility = Visibility.Collapsed;

        MessageBox.Show(
            "Вы вошли как гость.\nДоступен только просмотр БД
борцов.",
            "Режим гостя",
            MessageBoxButton.OK,
            MessageBoxImage.Information);
    }
}
```

3.4 Экспорт и импорт данных

В рамках курсового проектирования реализована функция импорта и экспорта данных о борцах в формате Excel с использованием библиотеки ClosedXML.

Функция экспорта позволяет сохранить список борцов в файл формата *.xlsx с автоматическим форматированием заголовков и подбором ширины столбцов. Код функции экспорта представлен листингом 6.

Листинг 6 – Код функции экспорта данных в Excel

```
private void ExportButton_Click(object sender, RoutedEventArgs e)
{
    try
    {
        var saveDialog = new Microsoft.Win32.SaveFileDialog
        {
            Filter = "Excel файлы (*.xlsx)|*.xlsx",
            DefaultExt = "xlsx",
            FileName = $"Борцы_{DateTime.Now:yyyy-MM-dd}"
        };

        if (saveDialog.ShowDialog() == true)
        {
            using var workbook = new XLWorkbook();
            var worksheet = workbook.Worksheets.Add("Борцы");
            // Формирование заголовков таблицы
```

```
            worksheet.Cell(1, 1).Value = "ФИО";
            worksheet.Cell(1, 2).Value = "Дата рождения";
            worksheet.Cell(1, 3).Value = "Регион";
            worksheet.Cell(1, 4).Value = "Весовая категория (кг)";
            // Стилизация заголовков
            var headerRange = worksheet.Range(1, 1, 1, 4);
            headerRange.Style.Font.Bold = true;
            headerRange.Style.Fill.BackgroundColor              =
XLColor.DarkBlue;
            headerRange.Style.Font.FontColor = XLColor.White;
            // Заполнение данными
            var dataToExport = wrestlersDataGrid.ItemsSource as
List<Wrestler>
                               ?? _allWrestlers;
            int row = 2;
            foreach (var wrestler in dataToExport)
            {
                worksheet.Cell(row, 1).Value = wrestler.FullName;
                worksheet.Cell(row,          2).Value          =
wrestler.BirthDate.ToString("dd.MM.yyyy");
                worksheet.Cell(row,          3).Value          =
wrestler.Region?.RegionName ?? "";
                worksheet.Cell(row,          4).Value          =
wrestler.WeightCategory?.CategoryName ?? 0;
                row++;
            }
            // Автоподбор ширины столбцов
            worksheet.Columns().AdjustToContents();
            workbook.SaveAs(saveDialog.FileName);
            MessageBox.Show(
                $"Экспортировано {dataToExport.Count} борцов в
файл:\n{saveDialog.FileName}",
                "Экспорт завершён",
                MessageBoxButton.OK,
                MessageBoxImage.Information);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка     экспорта:     {ex.Message}",
"Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}
```

Функция импорта включает предварительное отображение инструкции с описанием требуемого формата файла, парсинг данных из Excel с валидацией и сохранение записей в БД. Код функции импорта представлен листингом 7.

Листинг 7 – Код функции импорта данных из Excel

```
private void ImportButton_Click(object sender, RoutedEventArgs
e)
{
    // Отображение инструкции по формату файла
    string formatInfo =
        "ФОРМАТ EXCEL-ФАЙЛА ДЛЯ ИМПОРТА:\n\n" +
        "Файл должен содержать следующие столбцы (в
порядке):\n\n" +
        "1. ФИО — полное имя борца (текст)\n" +
        "2. Дата рождения — в формате ДД.ММ.ГГГГ\n" +
        "3. Регион — название региона (должен существовать в
БД)\n" +
        "4. Весовая категория — вес в кг (должен существовать в
БД)\n\n" +
        "Доступные регионы: " + string.Join(", ",
_regions.Select(r => r.RegionName)) +
        "\n\nДоступные весовые категории (кг): " +
        string.Join(", ", _weightCategories.Select(c =>
c.CategoryName)) +
        "\n\nПродолжить импорт?";
    var result = MessageBox.Show(formatInfo, "Формат файла для
импорта",
                                 MessageBoxButton.YesNo,
MessageBoxImage.Information);
    if (result != MessageBoxResult.Yes) return;
    var openDialog = new Microsoft.Win32.OpenFileDialog
    {
        Filter = "Excel файлы (*.xlsx)|*.xlsx|Все файлы
(*.*)|*.*",
        DefaultExt = "xlsx"
    };
    if (openDialog.ShowDialog() == true)
    {
        try
        {
            int imported = 0;
            int skipped = 0;
            var errors = new List<string>();
            using var workbook = new
XLWorkbook(openDialog.FileName);
            var worksheet = workbook.Worksheet(1);
            var rows = worksheet.RowsUsed().Skip(1); // Пропуск
заголовка
            using var context = new WrestlingDbContext();
            foreach (var row in rows)
            {
                try
                {
```

```
                    string fullName =
row.Cell(1).GetString().Trim();
                    string birthDateStr =
row.Cell(2).GetString().Trim();
                    string regionName =
row.Cell(3).GetString().Trim();
                    string weightStr =
row.Cell(4).GetString().Trim();
                    if (string.IsNullOrEmpty(fullName)) {
skipped++; continue; }
                    // Валидация даты рождения
                    if (!DateTime.TryParse(birthDateStr, out
DateTime birthDate))
                    {
                        errors.Add($"Строка {row.RowNumber()}:
неверный формат даты");
                        skipped++; continue;
                    }
                    // Поиск региона в справочнике
                    var region = _regions.FirstOrDefault(r =>
                        r.RegionName.Equals(regionName,
StringComparison.OrdinalIgnoreCase));
                    if (region == null)
                    {
                        errors.Add($"Строка {row.RowNumber()}:
регион не найден");
                        skipped++; continue;
                    }
                    // Поиск весовой категории
                    if (!int.TryParse(weightStr, out int
weight)) { skipped++; continue; }
                    var weightCategory =
_weightCategories.FirstOrDefault(c =>
                        c.CategoryName == weight);
                    if (weightCategory == null) { skipped++;
continue; }
                    // Создание и сохранение записи борца
                    var wrestler = new Wrestler
                    {
                        FullName = fullName,
                        BirthDate = birthDate,
                        IdRegion = region.IdRegion,
                        IdWeightCategory =
weightCategory.IdWeightCategory
                    };
                    context.Wrestlers.Add(wrestler);
                    imported++;
                }
                catch { skipped++; }
            }
            context.SaveChanges();
            LoadData();
```

```
            MessageBox.Show($"Импортировано:
{imported}\nПропущено: {skipped}",
                            "Результат импорта",
MessageBoxButton.OK, MessageBoxImage.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка импорта: {ex.Message}",
"Ошибка",
                            MessageBoxButton.OK,
MessageBoxImage.Error);
        }
    }
}}
```

4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

В рамках курсового проектирования проведено структурное тестирование приложения [1]. Для этого использовался фреймворк xUnit, предоставляющий набор инструментов для написания unit-тестов на платформе .NET, а также набор атрибутов и утверждений для проверки корректности работы программных модулей. Код unit-теста переключения время периода с последующим изменением времени представлен листингом 8.

Листинг 8 – Код unit-теста переключения время периода

```
[Fact]
public void SetPeriod_ChangesMatchTime()
{
    // Arrange - подготовка к тесту: создаем экземпляр сервиса

    var service = new ScoreboardService();

    // Act - установка периода 1

    service.SetPeriod(1);
    Assert.Equal(360, service.TimeSeconds); // 6 минут

    // Act - установка периода 2

    service.SetPeriod(2);
    Assert.Equal(180, service.TimeSeconds); // 3 минуты
}
```

4.2 Функциональное тестирование

В рамках курсового проектирования проведено функциональное тестирование приложения для оценки удобства использования. В таблице 1 представлен набор тестов разработанного приложения и их результаты.

Таблица 1 – Набор тестов приложения

Действие	Ожидаемый результат	Полученный результат
В окне судейства нажать кнопку «+1» для красного угла	Счёт красного угла увеличивается на 1, обновляется на главном табло	Соответствует ожидаемому
В окне судейства запустить основной таймер, нажать «Старт»	Таймер начинает обратный отсчёт от 6:00 (первый период)	Соответствует ожидаемому
В окне судейства при работающем таймере нажать кнопку «Перерыв»	Таймер останавливается, активируется 30-секундный таймер перерыва, на табло отображается статус «ПЕРЕРЫВ»	Соответствует ожидаемому
В окне информации о борцах нажать кнопку «Добавить борца», в поля «ФИО», «Дата рождения», «Регион», «Весовая категория» ввести данные «Сергеев Сильвестр Андреевич», «09.04.1974», «Россия», «97» и нажать «Сохранить»	Появляется новая запись о борце с введёнными данными	Соответствует ожидаемому
В окне информации о борцах выбрать существующую запись, нажать «Редактировать», изменить «ФИО» на «Васильков Василий Васильевич» и сохранить	«ФИО» выбранной записи измениться на «Васильков Василий Васильевич»	Соответствует ожидаемому
В окне информации о борцах выбрать запись, нажать «Удалить» и подтвердить действие	Запись удаляется из таблицы и БД	Соответствует ожидаемому
В окне информации о борцах ввести «Б» в поле поиска по ФИО	В таблице отображаются только записи, содержащие введённый текст	Соответствует ожидаемому
В окне судейства нажать кнопку «Показать табло»	На месте окна флага на втором мониторе открывается полноэкранное окно табло с текущим счётом и временем на таймере	Соответствует ожидаемому
В окне судейства выбрать весовую категорию и разных борцов из выпадающих списков для красного и синего углов	На табло и в интерфейсе обновляются весовая категория и имена выбранных спортсменов	Соответствует ожидаемому
В окне судейства по окончании матча нажать кнопку «Победа красного»	На табло и в интерфейсе отображается индикатор победителя, счёт фиксируется	Соответствует ожидаемому

Продолжение таблицы 1

Действие	Ожидаемый результат	Полученный результат
В окне судейства переключить период с первого на второй	Основной таймер автоматически устанавливается на 3:00, интерфейс обновляется	Соответствует ожидаемому
В окне судейства нажать кнопку «Сигнал»	Воспроизводится звуковой сигнал	Соответствует ожидаемому
Войти под ролью «Гость»	Открывается главное окно с ограниченным функционалом	Соответствует ожидаемому
Войти под ролью «Администратор», открыть окно управления пользователями	Отображается список всех пользователей, доступны кнопки управления ролями и удаления	Соответствует ожидаемому
В окне информации о борцах нажать кнопку «Экспорт в Excel» с применённым фильтром по весовой категории	Создаётся файл .xlsx, содержащий только отфильтрованные записи	Соответствует ожидаемому
В окне информации о борцах нажать кнопку «Импорт из Excel», выбрать файл *.xlsx с данными о борцах (заполненные столбцы «ФИО», «Дата рождения», «Регион», «Весовая категория»)	В таблице отображаются новые записи	Соответствует ожидаемому
В окне судейства нажать кнопку «Скрыть табло»	На втором экране отображается полноэкранное изображение флага	Соответствует ожидаемому

5 Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

Для функционирования системы на сторонне сервера достаточны следующие программные и технические средства:

- ОС Windows 10/11 (64-bit);
- 2-ядерный процессор частотой не ниже 2 ГГц;
- оперативная память объемом 8 ГБ;
- SSD объемом 25 ГБ;
- дополнительное ПО: SSMS.

Для развёртывания БД нужно установить MSSQL 2022 на сервер, подключиться к серверу БД с помощью SQL Server Management Studio, вставить и выполнить SQL-скрипт из репозитория проекта.

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

- ОС Windows 10/11 (64-bit);
- .NET Desktop Runtime 8.0 или новее;
- 2-ядерный процессор частотой не ниже 2 ГГц;
- оперативная память в объеме 4 ГБ;
- свободное место в хранилище 1 ГБ;
- постоянное интернет-подключение.

Для установки оконного приложения требуется распаковать архив KPWrestlingScoreboard.zip из репозитория, разархивировать его в любую удобную папку и открыть файл KPWrestlingScoreboard.exe.

В приложении используются следующие учётные данные:

- логин: admin,
- пароль: admin.

5.2 Инструкция по работе

При запуске приложения пользователя встречает экран авторизации, в котором необходимо ввести логин и пароль. При успешной авторизации пользователя открывается окно судейства и окно флага. Окно судейства представлено на рисунке 5.

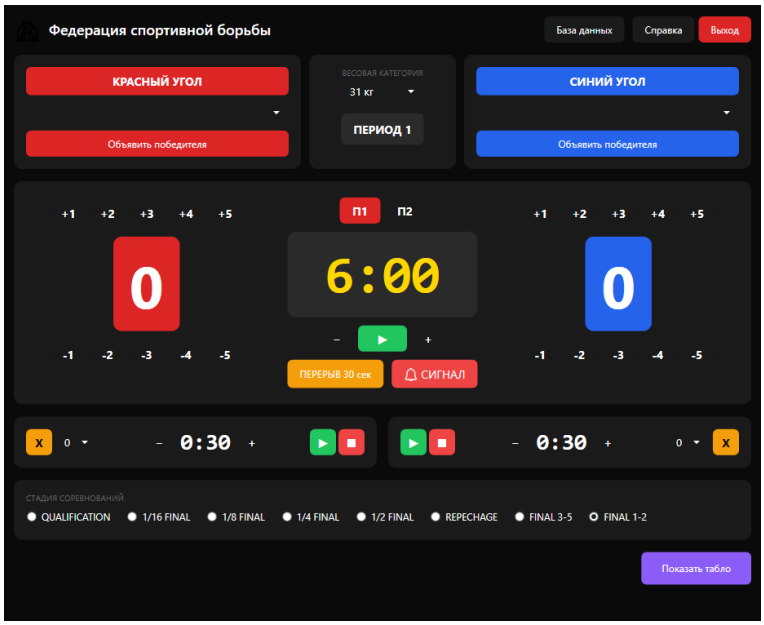

Рисунок 5 – KPWrestlingScoreboard. Вид окна судейства

При переходе на окно судейства пользователю представлены таймер, выпадающие списки для выбора весовой категории и борцов из выбранной категории, счётчик очков, кнопки для начисления и снятия баллов, кнопки для переключения периода, запуска перерыва и подачи звукового сигнала, а также переключатель для выбора стадии соревнований.

Чтобы отобразилось табло, пользователю нужно нажать на кнопку «Показать табло» в нижней части окна судейства. После этого табло откроется поверх окна флага. Табло представлено на рисунке 6.

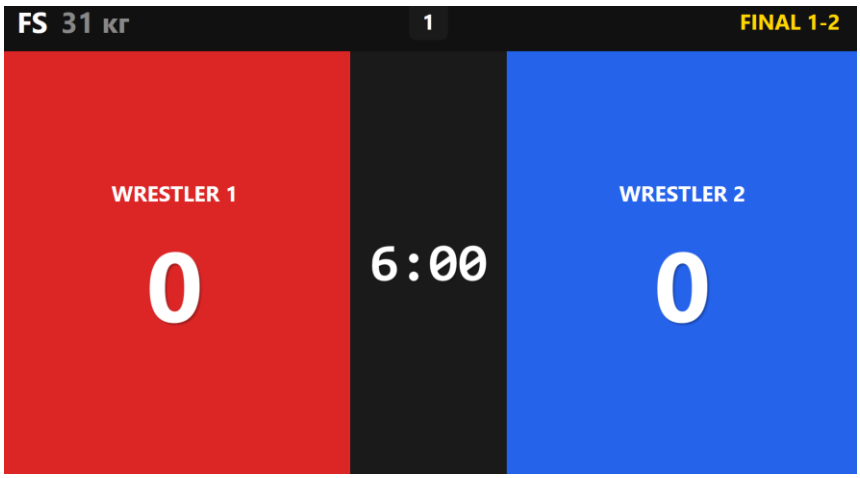

Рисунок 6 – KPWrestlingScoreboard. Вид табло

Также из окна судейства можно перейти на окно с информацией о борцах, нажав кнопку «База данных». Там пользователь может просмотреть список участников, редактировать его, импортировать и экспортировать данные. Окно с информацией о борцах представлено на рисунке 7.

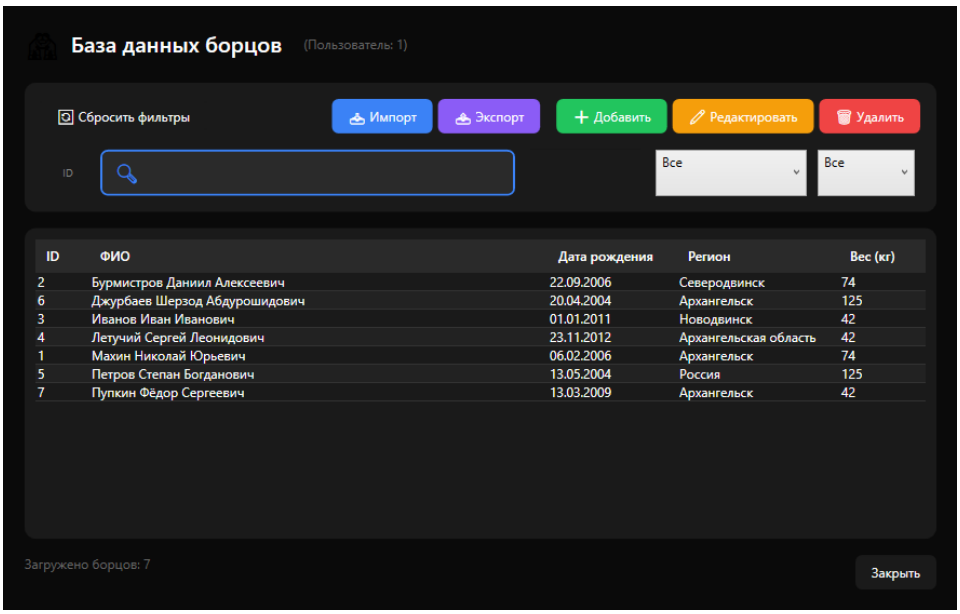

Рисунок 7 – KPWrestlingScoreboard. Вид окна с информацией о борцах

ЗАКЛЮЧЕНИЕ

Целью курсового проектирования являлась разработка оконного приложения для судейства проведения соревнований по греко-римской и вольной борьбе.

В ходе проектирования решены ключевые задачи, направленные на создание функционального и удобного приложения. Разработанное ПО отвечает современным требованиям, предоставляя инструментарий для судейства соревнований, а именно полную автоматизацию процесса ведения матча

Для достижения поставленной цели решены следующие задачи:

- изучены правила, регламенты и особенности судейства соревнований по греко-римской и вольной борьбе;
- спроектирована диаграмма вариантов использования;
- спроектирована физическая модель предметной области;
- спроектирована диаграмма развертывания компонентов;
- разработана БД для хранения информации о участниках соревнований и пользователях подсистемы;
- реализованы импорт и экспорт данных;
- реализовано разграничение прав доступа пользователей;
- реализована фильтрация, поиск и сортировка данных;
- реализована защита данных;
- выполнено тестирование и отладка ПО и проанализированы результаты тестирования;
- разработана программная и эксплуатационная документация

В результате приложение служит не только инструментом для проведения соревнований, но и способствует повышению объективности и скорости судейства.

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: 17.11.2025). – 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: 19.11.2025). – 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. – URL: https://znanium.ru/catalog/product/2096940 (дата обращения: 05.11.2025). – 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 (дата обращения: 01.11.2025). – 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 (дата обращения: 18.11.2025). – Режим доступа: по подписке. – Текст: электронный.