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 УЧЕТА ЛЕСНЫХ УЧАСТКОВ

Л109. 25КП01. 005 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки программного обеспечения

Студент	ИСПП-21		08.12.2025	Д.А. Дружинин
	(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 3
Введение 4
1 Анализ и разработка требований 6
1.1 Назначение и область применения 6
1.2 Постановка задачи 7
1.3 Выбор состава программных и технических средств 8
2 Проектирование программного обеспечения 10
2.1 Проектирование интерфейса пользователя 10
2.2 Разработка архитектуры программного обеспечения 13
2.3 Проектирование базы данных 13
3 Разработка и интеграция модулей программного обеспечения 15
3.1 Разработка программных модулей 15
3.2 Реализация интерфейса пользователя 17
3.3 Разграничение прав доступа пользователей 19
3.4 Экспорт и импорт данных 20
4 Тестирование и отладка программного обеспечения 22
4.1 Структурное тестирование 22
4.2 Функциональное тестирование 23
5 Инструкция по эксплуатации программного обеспечения 25
5.1 Установка программного обеспечения 25
5.2 Инструкция по работе 26
Заключение 29
Список использованных источников 30

СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЙ

БД – база данных

ГКУ – государственное казенное учреждение

ОС – операционная система

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

API – Application Programming Interface

DTO – Data Transfer Object

JWT – JSON Web Token

ORM – Object-Relational Mapping

WPF – Windows Presentation Foundation

ВВЕДЕНИЕ

В современном мире информационные системы становятся неотъемлемой частью повседневной деятельности в различных сферах – от здравоохранения и образования до промышленности и природопользования. Особенно остро потребность в автоматизированных решениях ощущается в отраслях, связанных с управлением большими объёмами данных, где требуется точность, оперативность и надёжность. Такие системы позволяют централизовать хранение информации, минимизировать человеческий фактор, ускорить обработку данных и обеспечить своевременный доступ к актуальной информации для всех заинтересованных пользователей. В условиях цифровой трансформации государственного и муниципального управления внедрение специализированных программных решений перестаёт быть опцией и становится необходимостью.

Актуальность разрабатываемого курсового проекта заключается в ускорении процессов учёта лесных участков, решении проблемы фрагментированного и несистематизированного хранения данных, автоматизации документооборота по лесохозяйственным мероприятиям и обеспечении сотрудников лесничеств единым, структурированным и надёжным инструментом для управления информацией.

Целью курсового проектирования является разработка комплексного решения для ведения учета информации о лесных участках и проводимых на них мероприятиях.

Для достижения поставленной цели требуется решить следующие задачи:

- провести сбор требований целевой аудитории,
- проанализировать информационные источники по предметной области,
- спроектировать архитектуру приложения,
- спроектировать диаграмму вариантов использования приложения,

- выбрать состав программных и технических средств для реализации приложения,
- спроектировать БД,
- спроектировать интерфейс оконного приложения,
- создать БД в выбранной СУБД,
- разработать API для некоторых функций приложения,
- реализовать разграничение прав доступа пользователей,
- разработать интерфейс оконного приложения,
- разработать оконное приложение,
- реализовать экспорт данных в виде файлов Excel,
- реализовать работу приложения с БД при помощи API,
- выполнить структурное тестирование ПО,
- выполнить функциональное тестирование ПО,
- разработать программную и эксплуатационную документацию.
- В результате выполнения поставленных задач будет разработано оконное приложение, которое обеспечит эффективное взаимодействие с информацией о лесных участках.

Сбор и анализ требований

1.1 Назначение и область применения

Подсистема разрабатывается для использования сотрудниками лесопромышленного комплекса и предназначена для добавления и обработки информации, как непрерывного, так и периодического лесоустройства, для достижения устойчивого лесоуправления, инновационного и эффективного использования, охраны, защиты и воспроизводства лесов.

Основным назначением внедрения ПО является упрощение и оптимизация процесса учета древесных насаждений в лесных участках и решение задач, связанных с анализом, моделированием и прогнозированием состояния и структуры лесов, проектированием и анализом лесохозяйственной деятельности, с целью снижения временных затрат на поиск, сбор, обработку и анализ имеющихся данных о количественных и качественных характеристиках лесного участка, повышения точности и уменьшения рисков, связанных с человеческими ошибками при актуализации информации по проведенным лесохозяйственным мероприятиям.

Внедрение единой цифровой платформы позволяет привести разрозненные данные к унифицированному формату, что обеспечивает совместимость с государственными информационными системами лесного хозяйства.

ПО разрабатывается в частности для ГКУ Архангельской области «Коношское лесничество», но также предусмотрено его масштабное использование организациями лесопромышленного сектора. В лесничестве основными пользователями ПО являются администратор, который его настраивает и обеспечивает работоспособность, мастера участков, участковые лесничие, инженеры второй категории, которые обеспечивают актуальность данных о лесных участках и лесохозяйственных мероприятиях.

1.2 Постановка задачи

Требуется создать оконное ПО, которое обеспечит возможность использования следующего набора функций:

- создание и удаление информации о пользователях;
- разграничение прав доступа пользователей в ПО;
- просмотр информации о лесных участках;
- фильтрация данных по номеру участка или имени ответственного;
- редактирование информации о лесных участках;
- редактирование информации о лесохозяйственных мероприятиях на лесных участках;
- создание и сохранение отчетов о лесохозяйственных мероприятиях как в целом по лесничеству, так и по отдельным лесным участкам, в формате *.xlsx.

Работа ПО начинается с верификации пользователя. Окно авторизации открывается сразу же после запуска ПО.

Администратор имеет полный доступ ко всем функциям, включая создание и удаление информации о пользователях.

Мастер участка имеет доступ к просмотру списка участков и редактированию информации о лесных участках и о лесохозяйственных мероприятиях по вырубке лесов.

Участковый лесничий имеет доступ к просмотру списка участков и редактированию информации о лесных участках.

Инженер второй категории имеет доступ к просмотру списка участков, редактированию информации о лесных участках, редактированию информации о лесохозяйственных мероприятиях по восстановлению, воспроизводству и защите лесов, и созданию и сохранению отчетов о лесохозяйственных мероприятиях.

На рисунке 1 представлена диаграмма вариантов использования ПО.

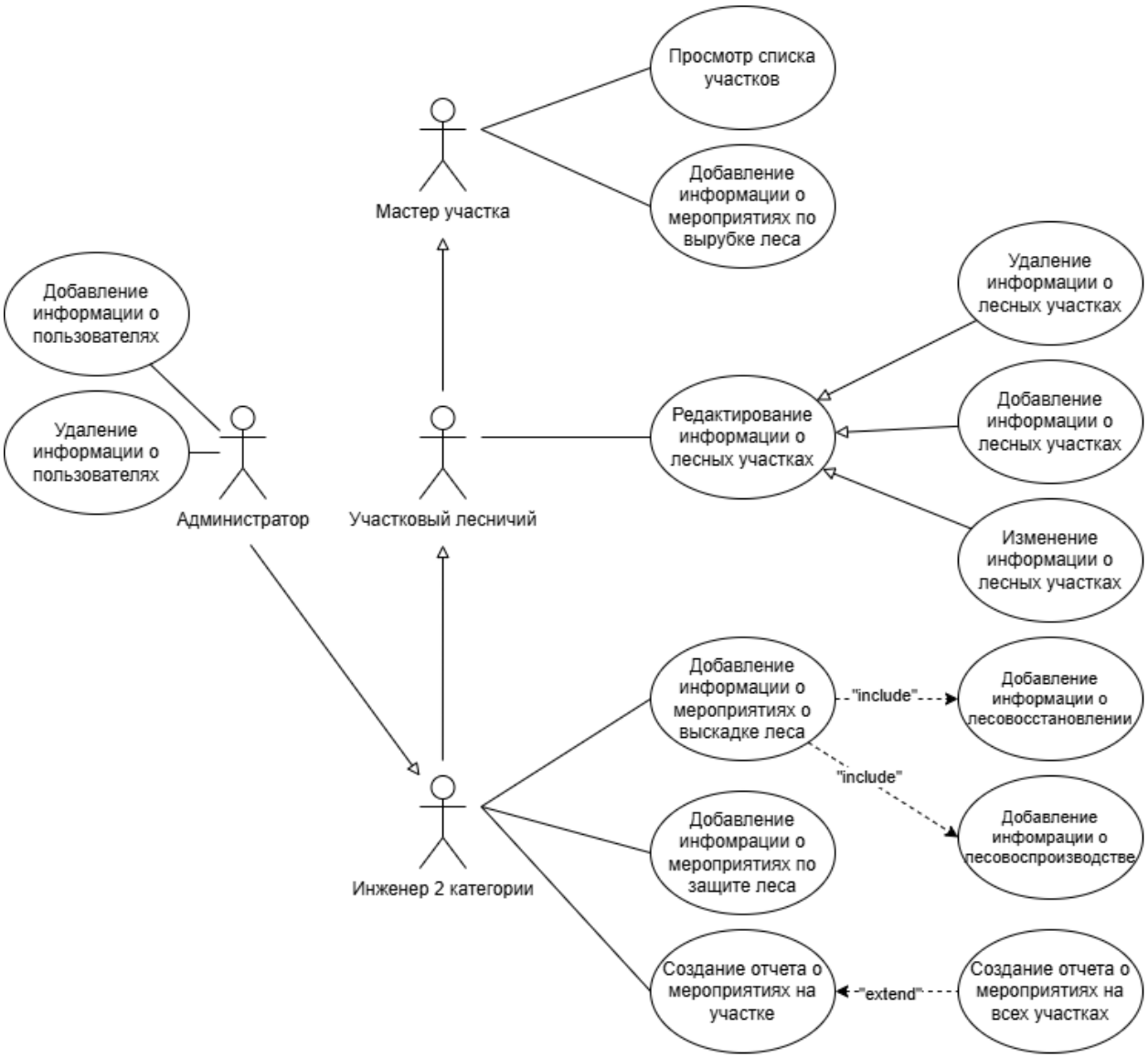

Рисунок 1 – Диаграмма вариантов использования

1.3 Выбор состава программных и технических средств

Согласно цели курсового проектирования необходимо разработать ПО для добавления и обработки информации о лесных участках и лесохозяйственных мероприятиях.

Разрабатываемое ПО будет использоваться на персональных компьютерах и ноутбуках.

Для создания ПО выбран язык программирования C# и графическая подсистема WPF. Выбор сделан в пользу C# в связи с тем, что он имеет ряд

преимуществ перед некоторыми языками программирования за счет его объектно-ориентированности, строгой типизации, кроссплатформенности и развитой экосистеме инструментов и библиотек. WPF выбран благодаря принципиально новому подходу к построению Windows интерфейсов за счет использования традиционных языков программирования, декларативного определения графического интерфейса, независимости от разрешения экрана, аппаратного ускорения графики.

В качестве СУБД выбрана MSSQL – одна из самых удобных и функциональных сред для работы с БД, которая предлагает множество инструментов для разработчиков. Также эта СУБД отличается повышенной безопасностью: она позволяет управлять доступом пользователей к данным с помощью ролей и разрешений, поддерживает как шифрование на уровне БД, так и шифрование отдельных столбцов, помогает отслеживать действия пользователей и изменения данных, тем самым выявляя несанкционированную активность.

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

- ОС Windows 10 (версия 1809) или новее;
- .NET 8 Desktop Runtime;
- процессов с частотой 1 ГГц ;
- оперативная память объемом 2 ГБ;
- место на диске 200 МБ.

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

- ОС Windows 10 (версия 1607) или новее;
- сервер: SQL Server 2022 года;
- .NET Framework 4.7.2 (для SQL Server 2022);
- процессор x64 с частотой 1,4 ГГц
- оперативная память объемом 2 ГБ(рекомендуется 4 и более);
- место на диске минимум 10 ГБ.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В ходе проектирования оконного приложения разработаны wireframe[4] интерфейсов пользователя для ключевых страниц: главной (представлен на рисунке 2), создания лесного участка (представлен на рисунке 3) и подробной информации о лесном участке (представлен на рисунке 4). Наличие визуальных прототипов позволяет более четко определить структуру взаимодействия с системой и способствует согласованности между этапами проектирования и реализации. Это существенно ускоряет разработку визуальной части ПО.

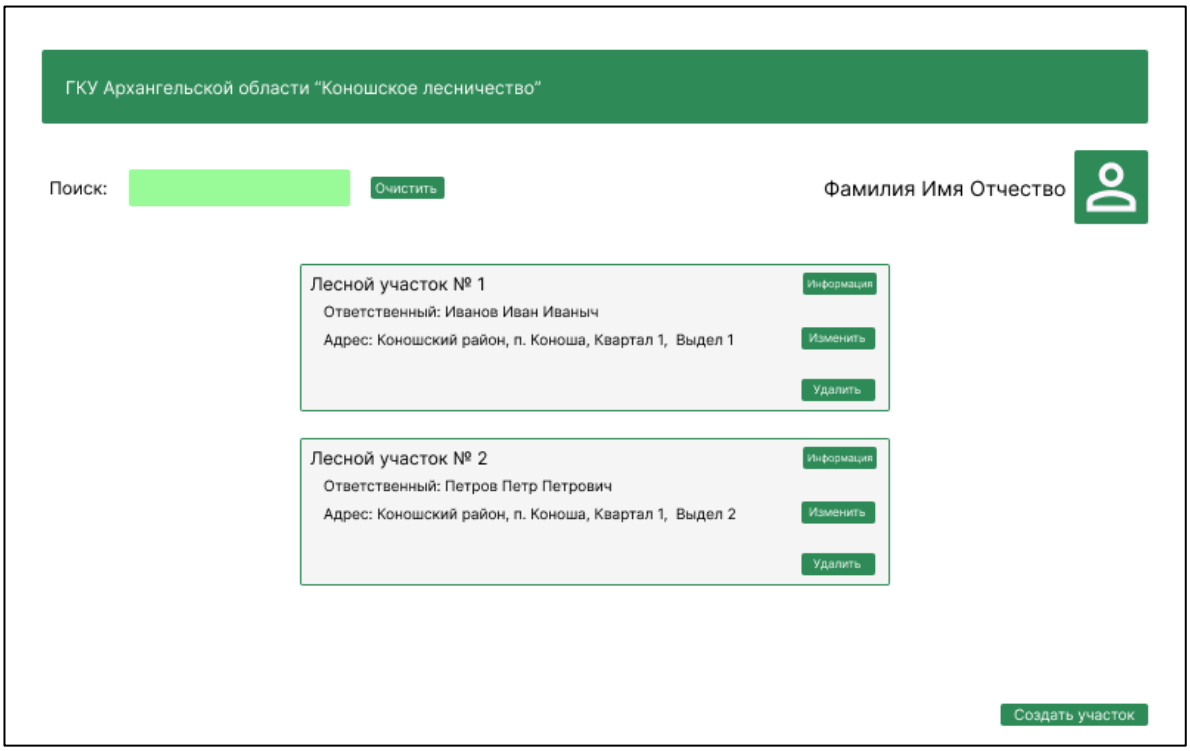

Рисунок 2 – Wireframe для главной страницы

Рисунок 3 – Wireframe для страницы создания лесного участка

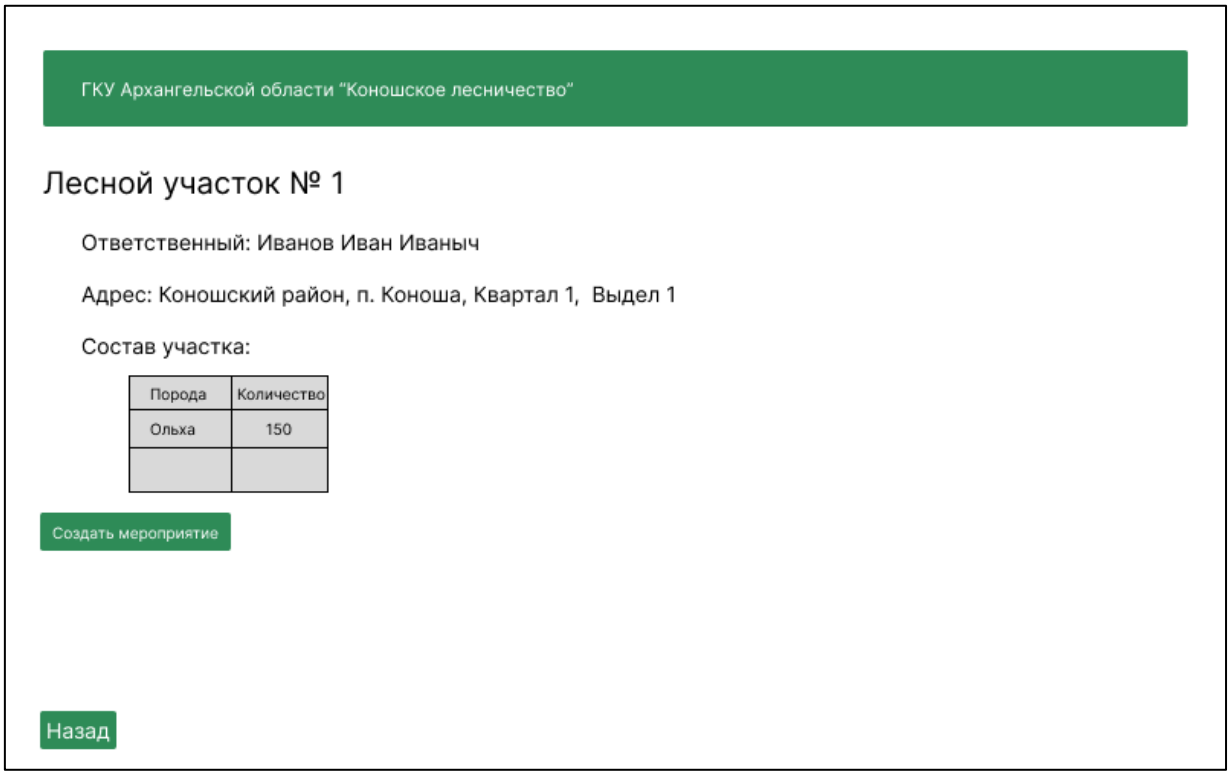

Рисунок 4 – Wireframe для страницы подробной информации о лесном участке

В процессе проектирования выбрана следующая цветовая палитра: #2E8B57 в качестве основного цвета для заголовка, границы и основных элементов интерфейса; #98FB98 в качестве вторичного цвета, который будет немного освежать интерфейс, он используется для полей ввода информации; #FFFFFF в качестве цвета подписей на элементах.

В приложении был использован логотип и пиктограмма, взятые из библиотеки Google Icons, которая предоставляет бесплатные и стандартизированные значки для веб- и оконных приложений. Использование элементов из этой библиотеки обеспечивает единый визуальный стиль и соответствие современным стандартам по дизайну интерфейсов.

Выбран логотип, который отображает основную тематику ПО и будет интуитивно понятен для пользователя (представлен на рисунке 5).

Рисунок 5 – Логотип приложения

На рисунке 6 предоставлена пиктограмма для обозначения профиля пользователя.

Рисунок 6 – Пиктограмма профиля пользователя

2.2 Разработка архитектуры программного обеспечения

Подсистема реализована по клиент-серверной архитектуре[5] и предназначена для использования в корпоративной среде. Компоненты распределены по двум физическим узлам: клиентским рабочим станциям и выделенному серверу БД. Серверная часть будет реализована с использованием ORM EntityFrameworkCore. Клиентское приложение будет работать на ПК и взаимодействовать с серверной частью через API. Приложение отправляет HTTP-запросы к API, который обрабатывает их и выполняет соответствующие операции в БД. Диаграмма развертывания подсистемы, представленная на рисунке 7.

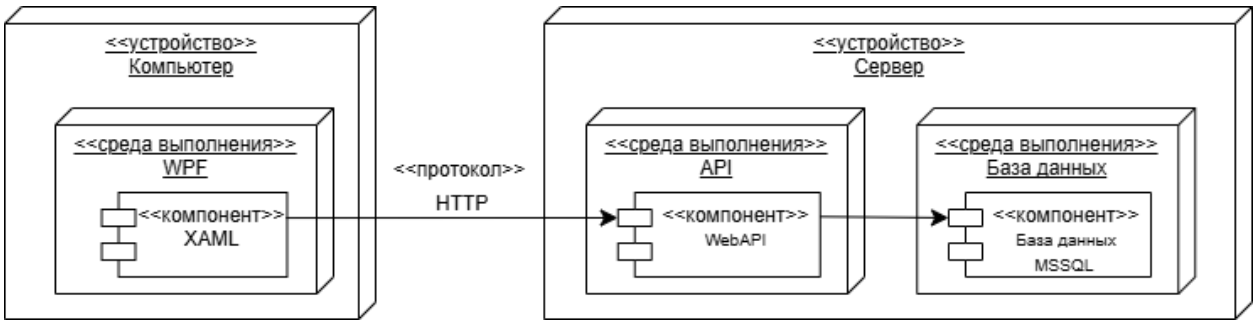

Рисунок 7 – Диаграмма развертывания подсистемы

2.3 Проектирование базы данных

В разрабатываемой подсистеме будет храниться информация о лесных участках, лесохозяйственных мероприятиях на участках, а также информация о работниках лесничества.

В БД требуется хранить информацию о работниках, лесных участках, количестве деревьев, породе деревьев, лесохозяйственных мероприятиях, типах мероприятий.

На рисунке 8 представлена физическая модель БД[3], разработанная для СУБД Microsoft SQL Server 2022.

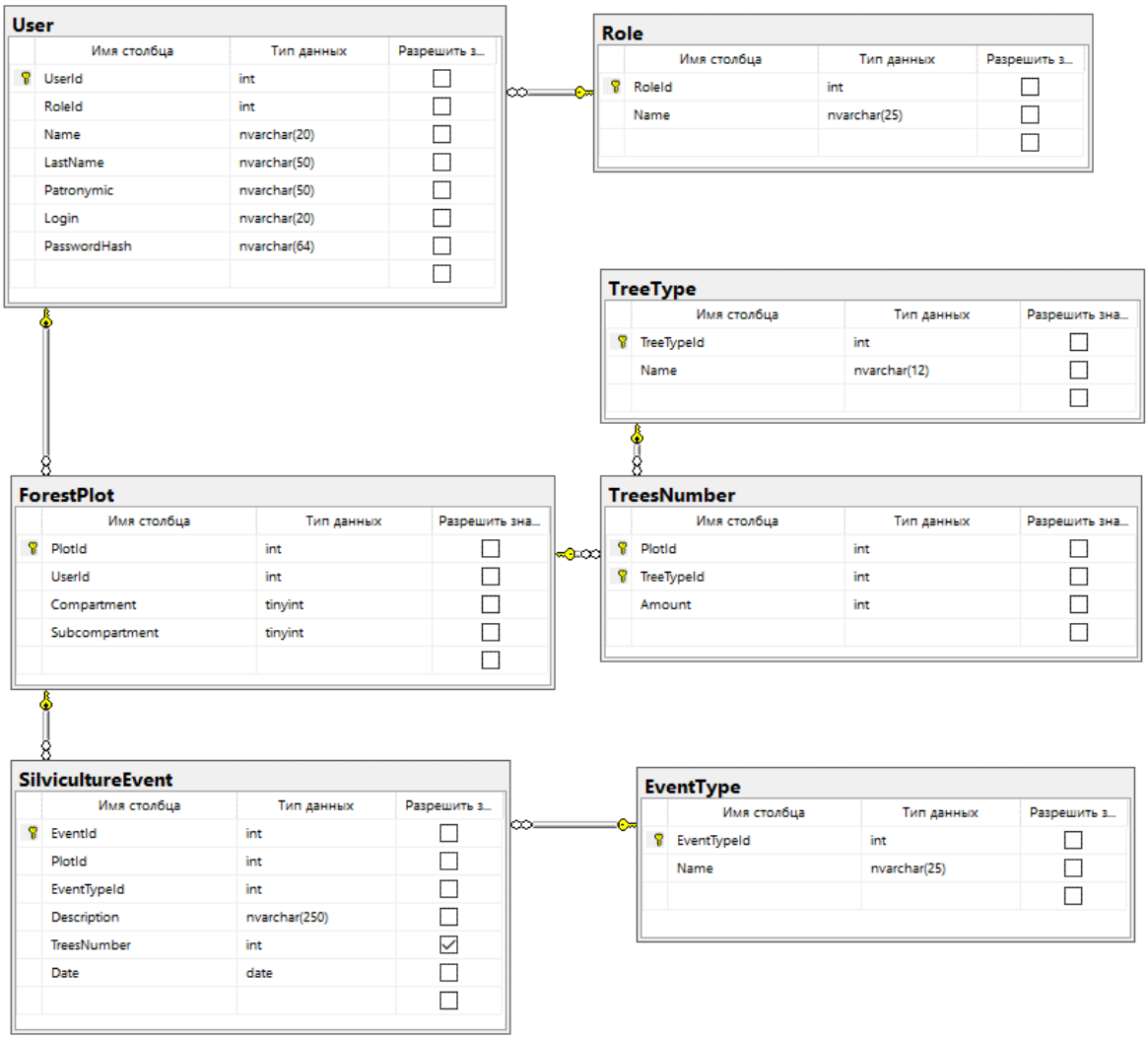

Рисунок 8 – Физическая модель БД

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

В рамках курсового проекта разработано оконное приложение на языке программирования C# с использованием архитектурного паттерна code-behind[2].

Серверная часть построена с использованием ORM EntityFrameworkCore. Взаимодействие с сервером реализовано через API с использованием библиотеки HttpClient.

Для реализации получения информации об участках с сервера использовано API, которое отправляет и получает информацию в формате DTO. DTO представляют собой простые объекты для передачи информации. Код метода получения информации об участках представлен листингом 1.

Листинг 1 – Код метода для получения списка участков в API

```
/// <summary>
/// Возвращает список лесных участков.
/// </summary>
/// <returns>List<ForestPlotDto> информаии об
участках.</returns>
 [HttpGet]
public async Task<ActionResult<IEnumerable<ForestPlotDto>>>
GetForestPlots()
{
    //Получаем информацию об участка из сервиса
    var forestPlots = await _service.GetForestPlotsAsync();
    //Проверяем на null, если null возвращае NotFound()
    if (forestPlots == null)
        return NotFound("Не удалось получить список участков!");
    //Иначе возвращаем список участков
    return forestPlots;
}
```

Для реализации получения информации из БД, разработан сервис, который получает информацию и приводит их к формату DTO, затем отправляет в API. Код метода сервиса для получения списка участков из БД представлен листингом 2.

Листинг 2 – Код метода для получения списка участков из БД

```
/// <summary>
/// Возвращает список лесных участков.
/// </summary>
/// <returns>List<ForestPlotDto> информаии об
участках.</returns>
public async Task<List<ForestPlotDto>> GetForestPlotsAsync()
{
    try
    {
       // Получаем всю нужную информацию о лесных участках из БД
       var forestPlots = await _context.ForestPlots
       .Include(fp => fp.User)
       .Include(fp => fp.TreesNumbers)
       .ThenInclude(tn => tn.TreeType)
       .ToListAsync();
       //Проверяем не вернулся ли null
       if (forestPlots == null)
           return null;
        //Приводим информацию, полученную из БД к типу
ForestPlotDto и возвращаем список лесных участков
        return forestPlots.Select(forestPlot => new
ForestPlotDto
        {
            PlotId = forestPlot.PlotId,
            Responsible = $"{forestPlot.User.LastName}
{forestPlot.User.Name} {forestPlot.User.Patronymic}".Trim(),
            Compartment = forestPlot.Compartment,
            Subcompartment = forestPlot.Subcompartment,
            TreesComposition =
forestPlot.TreesNumbers.Select(treesNumber => new TreesNumberDto
            {
                TreeType = treesNumber.TreeType.Name,
                Amount = treesNumber.Amount,
            }).ToList()
        }).ToList();
    }
    catch (DbException)
    {
        //При исключении выбрасываем заглушку
        throw;
    }}
```

Для реализации получения информации об участках с сервера использован глобально созданный экземпляр HttpClient. Код метода для получения информации об участках представлен листингом 3.

Листинг 3 – Код метода для отправки GET-запроса на сервер

```
/// <summary>
/// Получает список лесных участков.
/// </summary>
/// <returns>List<ForestPlotDto> информаии об
участках.</returns>
public async Task<List<ForestPlotDto>> GetForestPlotsAsync()
{
    //Выполняем GET-запрос к эндпоинту "ForestPlots", использую
уже глобально созданный экземпляр HttpClient
    var response = await App.HttpClient.GetAsync("ForestPlots");
    //Проверяем успешное выполнение запроса
    if (response.IsSuccessStatusCode)
    {
        //Читаем информацию из JSON-ответа
        var forestPlots = await
response.Content.ReadFromJsonAsync<List<ForestPlotDto>>();
        //Возвращаем список ForestPlotDto
        return forestPlots;
    }
    //Иначе возвращаем null
    return null;
}
```

3.2 Реализация интерфейса пользователя

Интерфейс разработан с использованием фреймворка WPF, при использовании которого реализована постраничная навигация, что позволяет пользователю легко перемещаться меду различными разделами приложения. В приложении разработаны различные элементы управления, стили, которые упрощают работу и делают интерфейс более интуитивно понятным. Навигация в приложении реализована с помощью элемента Frame, который управляет элементами Page, представляющими страницы в приложении.

Для отображения кнопок создания пользователя и кнопки перехода на страницу экспорта разработан PopUp, код верстки которой представлен листингом 4.

Листинг 4 – Код верстки PopUp

```
<!--Помещаем все в контейнер StackPanel-->
<StackPanel Orientation = "Horizontal"
HorizontalAlignment="Right">
    <!--Label для отображения ФИО пользователя-->
    <Label Content = "{Binding Source={x:Static
Application.Current}, Path= UserFullName}"
VerticalAlignment="Center"/>
    <!--Кнопка для вызова PopUp-->
    <ToggleButton x:Name="UserProfilButton" Background="#2E8B57"
Margin="5 0 17 0" Height="50" Width="50">
        <!--Иконка на кнопке-->
        <Image Source = "/Icons/person_icon.png" />
    </ ToggleButton >
    <!--PopUp с настройками открытия и расположения-->
    < Popup IsOpen="{Binding IsChecked,
ElementName=UserProfilButton}"
    PlacementTarget="{Binding ElementName=UserProfilButton}"
    Placement="Bottom"
    StaysOpen="False"
    HorizontalOffset="-82"
    VerticalOffset="1"
    x:Name="UserPopUp">
        <!--Граница для контейнера с кнопками-->
        <Border Background = "#F5F5F5" BorderBrush="Black"
BorderThickness="1" CornerRadius="2">
            <!--Помещаем кнопки в контейнер StackPanel-->
            <StackPanel>
                <Button x:Name="ReportButton" Content="Экспорт"
Width="120" Visibility="Collapsed" Click="ReportButton_Click"/>
                <Button x:Name="UserButton"
Content="Пользователи" Width="120" Visibility="Collapsed"
Click="UserButton_Click"/>
                <Separator x:Name="PopUpSeparator"
Visibility="Collapsed" Margin="0 5"/>
                <Button Content = "Выйти" Width="120"/>
            </StackPanel>
        </Border>
    </Popup>
</StackPanel>
```

3.3 Разграничение прав доступа пользователей

В приложении реализовано разграничение прав доступа пользователей на основе ролей пользователей, хранящихся в БД. Для этого в приложении реализована авторизация, с использованием JWT.

Код метода авторизации с использованием JWT представлен листингом 5.

Листинг 5 – Код метода авторизации пользователей

```
/// <summary>
/// Авторизует пользователя в приложении.
/// </summary>
/// <param name="login">Логин пользователя</param>
/// <param name="password">Пароль пользователя</param>
/// <returns>HttpStatusCode</returns>
public async Task<HttpStatusCode> Login(string login, string
password)
{
    //Формируем объект передачи и отправляем POST-запрос
    var response = await
App.HttpClient.PostAsJsonAsync("Account",
        new LoginDto
        {
            Login = login,
            PasswordHash =
Convert.ToBase64String(SHA256.HashData(Encoding.UTF8.GetBytes(pa
ssword))),
        });
    //Проверяем успешность запроса
    if (response.IsSuccessStatusCode)
    {
        //Возвращаем объект LoginResponseDto
        var loginResponse = await
response.Content.ReadFromJsonAsync<LoginResponseDto>();
        //Устанавливаем ФИО пользователя
        App.UserFullName = loginResponse.UserFullName;
        //Устанавливаем роль ползователя
        App.UserRole = loginResponse.UserRole;
        //Устанавливаем заголовок для HttpClienta
        App.HttpClient.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Bearer", loginResponse.Token);
    }
    //Взвращаем HttpStatusCode
    return response.StatusCode;}
```

3.4 Экспорт данных

В приложении реализован экспорт данных в формат *.xlsx о мероприятиях на лесном участке или во всем лесничестве при помощи библиотек ClosedXML. Код метода экспорта данных представлен листингом 6.

Листинг 6 – Код метода экспорта данных

```
private async void ExportButton_Click(object sender,
RoutedEventArgs e)
{
    // Проверяем заполнен ли лист мероприятий
    if (_eventList == null || _eventList.Count == 0)
    {
        string msg = selectedPlotId.HasValue
           ? "Нет мероприятий для выбранного участка."
           : "Нет данных для экспорта.";
        MessageBox.Show(msg, "Информация", MessageBoxButton.OK,
MessageBoxImage.Information);
        return;
    }
    // Создаем новый экземпляр Excel-книги
    using var workbook = new XLWorkbook();
    var ws = workbook.Worksheets.Add("Мероприятия");

    // Устанавливаем заголовки стобцов
    ws.Cell(1, 1).Value = "Номер мероприятия";
    ws.Cell(1, 2).Value = "Тип мероприятия";
    ws.Cell(1, 3).Value = "Участок проведения";
    ws.Cell(1, 4).Value = "Описание";
    ws.Cell(1, 5).Value = "Дата";
    ws.Cell(1, 6).Value = "Тип древесины";
    ws.Cell(1, 7).Value = "Количество древесины";
    // Устанавливаем стиль для текста и фона ячеек
    var headerRange = ws.Range(1, 1, 1, 7);
    headerRange.Style.Font.Bold = true;
    headerRange.Style.Fill.BackgroundColor = XLColor.LightGray;
    // Загружаем данные в строки
    for (int i = 0; i < _eventList.Count; i++)
    {
        var row = i + 2;
        var ev = _eventList[i];

        ws.Cell(row, 1).Value = ev.EventId;
        ws.Cell(row, 2).Value = ev.EventType;
        ws.Cell(row, 3).Value = $"Лесной участок №{ev.PlotId}";
        ws.Cell(row, 4).Value = ev.Description;
```

```
        ws.Cell(row, 5).Value = ev.Date.ToString("dd.MM.yyyy",
CultureInfo.InvariantCulture);
        ws.Cell(row, 6).Value = ev.TreeType ?? "";
        ws.Cell(row, 7).Value = ev.TreesNumber?.ToString() ??
"";
    };
    // +1 потому что первая строка — заголовок
    int lastRow = _eventList.Count + 1
    // Количество столбцов
    int lastColumn = 7;
    var dataRange = ws.Range(1, 1, lastRow, lastColumn);
    // Центрируем текст по горизонтали и вертикали
    dataRange.Style.Alignment.Horizontal =
XLAlignmentHorizontalValues.Center;
    dataRange.Style.Alignment.Vertical =
XLAlignmentVerticalValues.Center;
    // Добавление границ для наглядности
    dataRange.Style.Border.OutsideBorder =
XLBorderStyleValues.Thin;
    dataRange.Style.Border.InsideBorder =
XLBorderStyleValues.Thin;
    ws.Columns().AdjustToContents();
    // Открываем окно для сохранения
    var dialog = new SaveFileDialog
    {
        FileName = selectedPlotId.HasValue
            ? $"Отчёт_участок_{selectedPlotId.Value}.xlsx"
            : "Отчёт_все_мероприятия.xlsx",
        DefaultExt = ".xlsx",
        Filter = "Файлы Excel (*.xlsx)|*.xlsx"
    };
    // Проверяем открылось ли окно сохранения
    if (dialog.ShowDialog() != true) return;
    // Записываем и сохраняем файл
    using var fs = new FileStream(dialog.FileName,
FileMode.Create);
    workbook.SaveAs(fs);
```

4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

В ходе курсового проектирования проведено структурное тестирование[1] метода CreateForestPlotAsync с помощью библиотеки xUnit. Код метода теста представлен листингом 7.

Результат тестирования представлен в рисунке 9.

Листинг 7 – Код теста метода создания участка

```
[Fact]
public async Task CreateForestPlotAsync_ValidDto_ReturnsTrue()
{
    // Arrange
    // Создаём in-memory контекст базы данных
    var options = new DbContextOptionsBuilder<ForestryContext>()
        .UseInMemoryDatabase(databaseName:
Guid.NewGuid().ToString())
        .Options;
    // Вызываем контекст БД и передаем в него параметры
    using var context = new ForestryContext(options);
    // Создаем экземпляр сервиса и передаем в него контекст БД
    var service = new ForestPlotService(context);
    // Создаем объекты DTO для создания записи в БД
    var createDto = new CreateForestPlotDto
    {
        PlotId = 1,
        UserId = 1,
        Compartment = 10,
        Subcompartment = 2,
        TreeComposition = new List<CreateTreesNumberDto>
    {
        new CreateTreesNumberDto { TreeTypeId = 1, Amount = 100
},
        new CreateTreesNumberDto { TreeTypeId = 2, Amount = 50 }
    }
    };
    // Act
    // Выполняем запрос на создание записи
    var result = await service.CreateForestPlotAsync(createDto);
    // Assert
    Assert.True(result);
    // Проверяем добавился ли участок
```

```
    var savedPlot = await context.ForestPlots
        .Include(fp => fp.TreesNumbers)
        .FirstOrDefaultAsync(fp => fp.PlotId ==
createDto.PlotId);
    // Проверяем данные на корректность
    Assert.NotNull(savedPlot);
    Assert.Equal(createDto.UserId, savedPlot.UserId);
    Assert.Equal((byte)createDto.Compartment,
savedPlot.Compartment);
    Assert.Equal((byte)createDto.Subcompartment,
savedPlot.Subcompartment);
    Assert.Equal(2, savedPlot.TreesNumbers.Count);
    Assert.Contains(savedPlot.TreesNumbers, tn => tn.TreeTypeId
== 1 && tn.Amount == 100);
    Assert.Contains(savedPlot.TreesNumbers, tn => tn.TreeTypeId
== 2 && tn.Amount == 50);
}
```

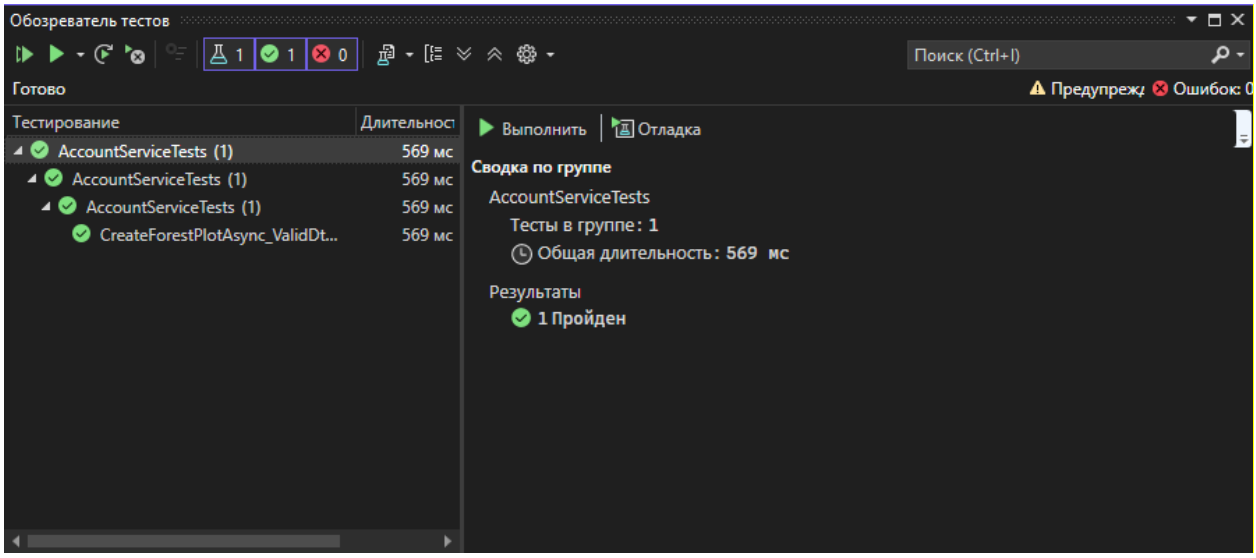

Рисунок 9 – Visual Studio. Вид окна «Обозреватель тестов»

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование окна авторизации методом «чёрного ящика», результаты тестирования представлены в таблице 1.

Таблица 1 – Набор тестов

Действие	Ожидаемый результат	Фактический результат
Открыть приложение	Открывается страница авторизации	Совпадает с ожидаемым
Ввести логин admin и пароль admin, нажать кнопку «Войти»	Открывается диалоговое окно с сообщением «Авторизация прошла успешно!» и откроется главная страница	Совпадает с ожидаемым
Нажать на кнопку «Создать лесной участок»	Открывается страница создания участка	Совпадает с ожидаемым
Выбрать породу «Сосна» и ввести количество 100, и нажать кнопку «+»	Справа от полей вода отобразиться список со строкой «Сосна – 100»	Совпадает с ожидаемым
Ввести номер участка 1, выбрать ответственного «Максимов Максим Максимович», ввести квартал 1 и выдел 1, выбрать тип «Сосна» и ввести количество 100, затем нажать кнопку «+», после этого нажать на кнопку «Создать»	Открывается диалоговое окно с сообщением «Участок успешно создан!» и откроется главная страница	Совпадает с ожидаемым
Нажать на кнопку «Назад»	Открывается главная страница	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

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

- ОС Windows 10 (версия 1607) или новее;
- сервер: SQL Server 2022 года;
- .NET Framework 4.7.2
- процессор x64 с частотой 1,4 ГГц
- оперативная память объемом 2 ГБ (рекомендуется 4 и более);
- место на диске минимум 10 ГБ.

Для развёртывания БД нужно подключиться к серверу MSSQL, с помощью SQL Server Manager Studio, вставить и запустить DBscript.sql из репозитория.

Для установки серверной части требуется перейти в терминале в желаемою папку для API, предварительно скачав архив Forestry.Api.zip из репозитория, разархивировать его в эту папку и изменить файл appsettings.json, изменив имя сервера на имя сервера, на котором будет расположена БД и открыть файл WebAPI.exe.

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

- ОС Windows 10 (версия 1809) или новее;
- .NET 8 Desktop Runtime;
- процессов с частотой 1 ГГц ;
- оперативная память объемом 2 ГБ;
- место на диске 200 МБ.

Для запуска оконного приложения требуется распаковать скачать архив Forestry.WPF.zip из репозитория и разархивировать его в любую удобную папку.

В приложении используются следующие учётные данные:

- логин: admin,
- пароль: admin.

5.2 Инструкция по работе

При запуске приложения, пользователя встречает окно авторизации. Для авторизации требуется ввести учетные данные в поля ввода логина и пароля и нажать кнопку «Войти». Окно авторизации представлено на рисунке 10.

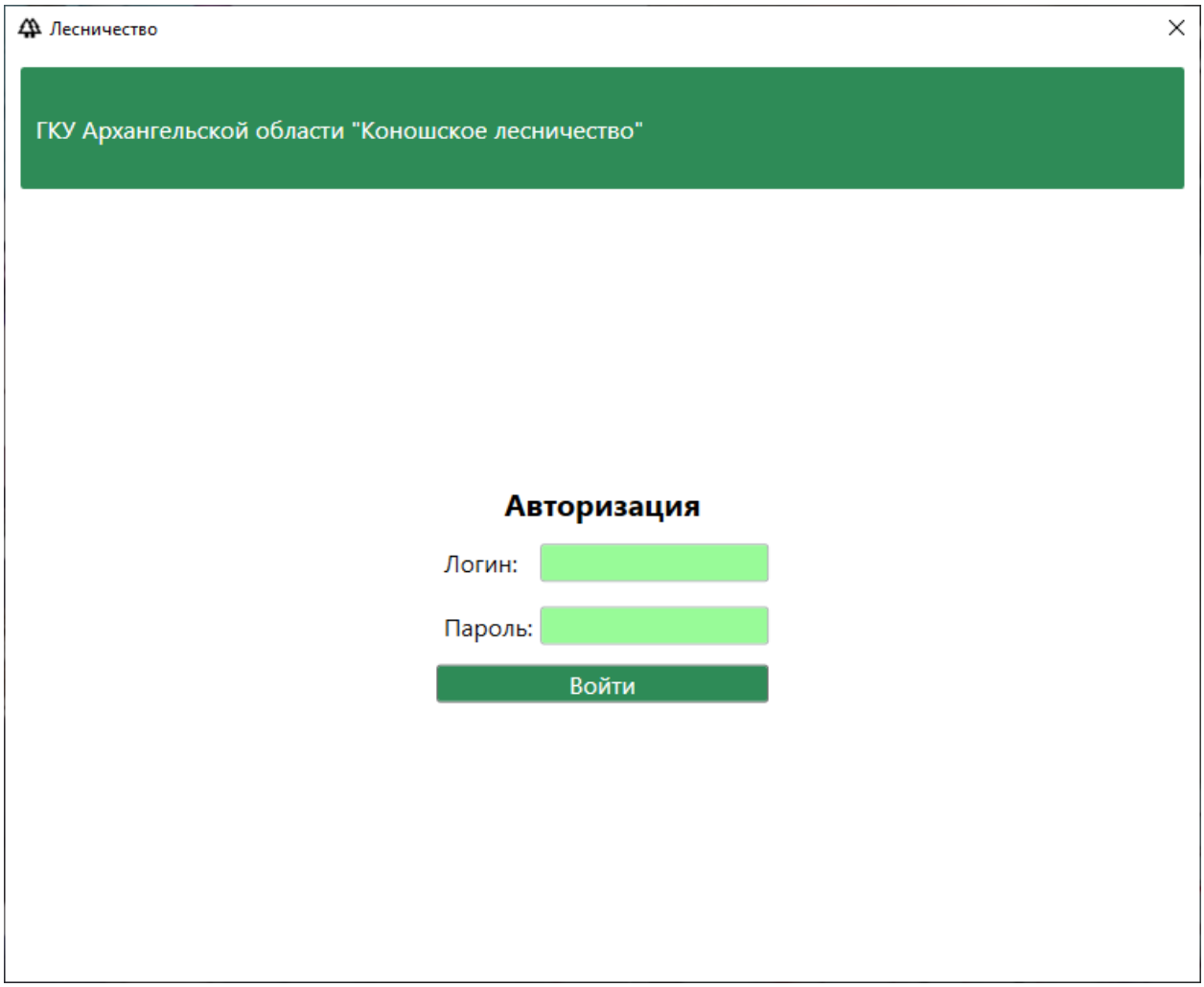

Рисунок 10 – Лесничество. Страница авторизации

После авторизации пользователь перенаправляется на главную страницу, где ему доступен просмотр карточек лесных участков, поле поиска по карточкам, кнопка профиля пользователя с доступом к страницам «Пользователи» и «Экспорт», и кнопке «Выйти», также внизу страницы расположена кнопка «Создать лесной участок», которая перенаправляет пользователя на страницу «Создания участка». Главное окно представлено на рисунке 11.

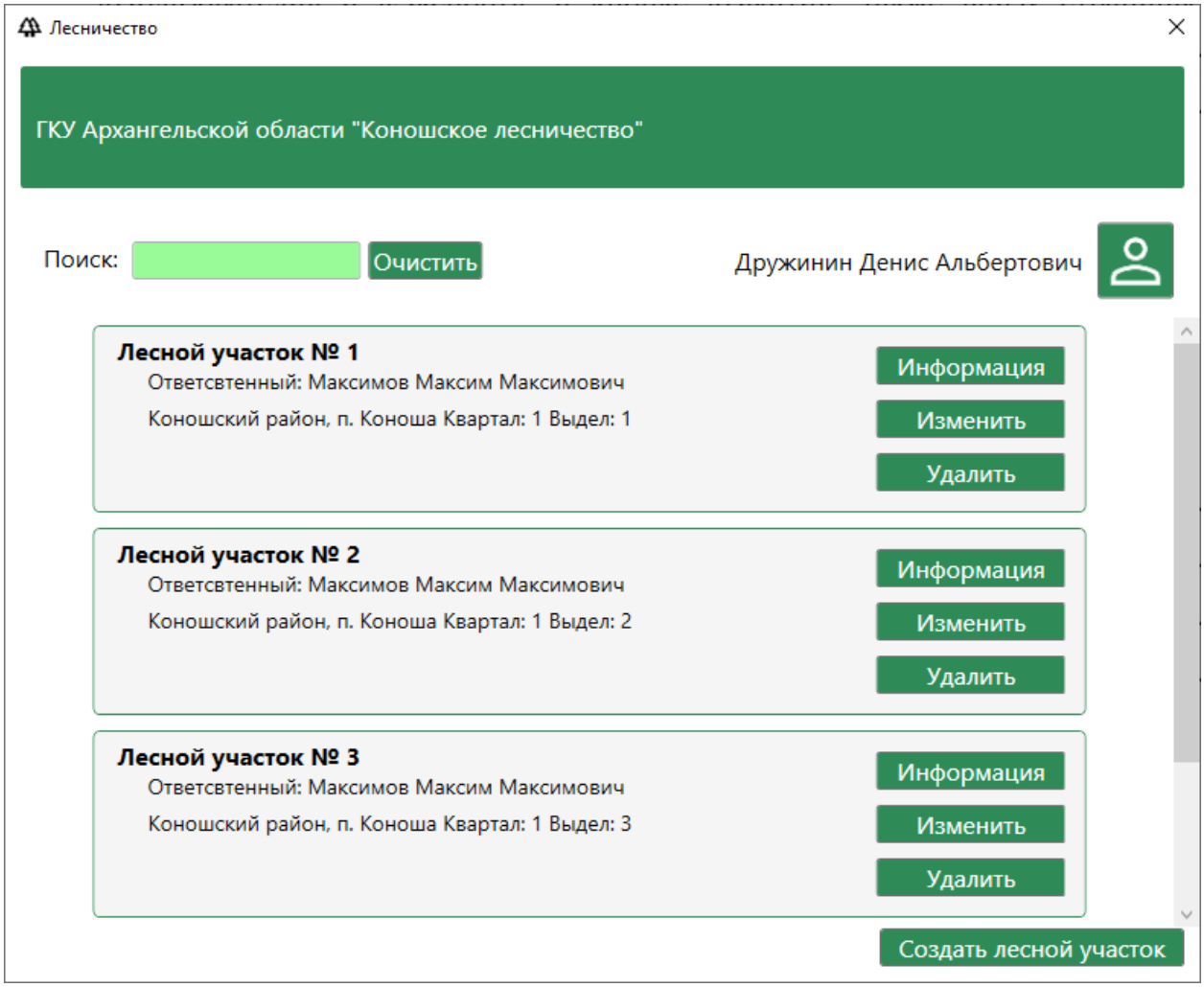

Рисунок 11 – Лесничество. Главная страница

На карточках, отображающих лесные участки расположены кнопки «Информация» и «Изменить, которые перенаправляют пользователя на соответствующие страницы и кнопка «Удалить», которая удаляет участок, на

котором она была нажата. Страница «Подробной информации» представлена рисунком 12.

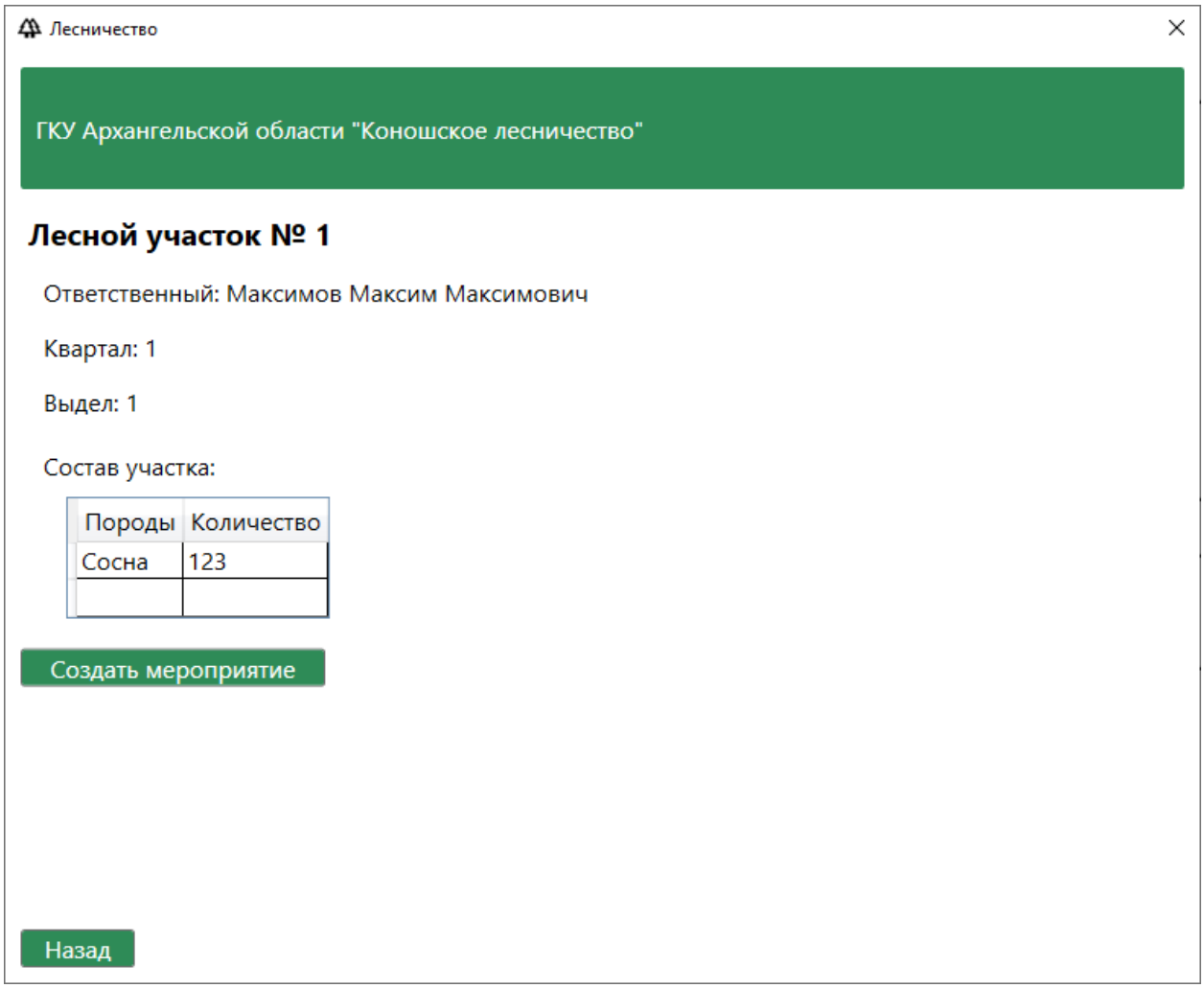

Рисунок 12 – Лесничество. Страница подробной информации

После перехода пользователя на страницу «Подробной информации» ему будет доступна кнопка «Создать мероприятие», которая перенаправляет пользователя на страницу «Создание лесохозяйственных мероприятий».

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования достигнута поставленная цель: разработана подсистема учета лесных участков, которая поможет обеспечить учет информации о лесных участках и лесохозяйственных мероприятия, проходящих на них. Разработанное ПО отвечает современным требованиям, предоставляя инструментарий для ведения учета информации о лесных участках и мероприятиях, проводимых на них.

Кроме того, решены все поставленные задачи:

- проведён сбор требований целевой аудитории,
- проанализированы информационные источники по предметной области,
- спроектирована архитектура приложения,
- спроектирована диаграмма вариантов использования приложения,
- выбран состав программных и технических средств для реализации приложения,
- спроектирована БД,
- спроектирован интерфейс оконного приложения,
- создана БД в выбранной СУБД,
- разработано API для некоторых функций приложения,
- реализовано разграничение прав доступа пользователей,
- разработан интерфейс оконного приложения,
- разработано оконное приложение,
- реализован экспорт данных в виде файлов Excel,
- реализована работа приложения с БД при помощи API,
- выполнено структурное тестирование ПО,
- выполнено функциональное тестирование ПО,
- разработана программная и эксплуатационная документация.

В результате приложение представляем собой работоспособное средства учета в лесничествах, способное повысить эффективность работы.

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: 30.11.2025). – 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: 13.11.2025). – 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: 6.11.2025). – 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: 4.11.2025). – 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: 16.11.2025). – Режим доступа: по подписке. – Текст : электронный.