

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 УЧЁТА

МАТЕРИАЛОВ И СПЕЦТЕХНИКИ ДЛЯ

АВТОДОРОГ

Л109. 24КП01. 016 ПЗ

(Обозначение документа)

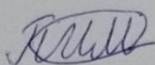
МДК.02.01 Технология разработки

программного обеспечения

Студент

ИСПП-11

(Группа)



(Подпись)

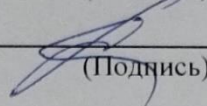
05.12.2024

(Дата)

И.М. Пономарев

(И.О. Фамилия)

Преподаватель



(Подпись)

05.12.2024

(Дата)

Ю.С. Маломан

(И.О. Фамилия)

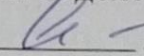
Архангельск 2024

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

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

УТВЕРЖДАЮ

Зав. отделением

 Ю.В. Солодкая
(Подпись) (И.О. Фамилия)

24 октября 2024

Задание

для курсового проектирования по МДК.02.01
Технология разработки программного обеспечения

студенту группы ИСПП-11, 4 курса

Фамилия, имя, отчество Пономареву Ивану Миновичу

1 Тема курсового проекта

Разработка подсистемы учёта материалов и спецтехники для автодорог

2 Исходные данные к проекту

Спроектировать и разработать серверную и клиентскую части многопользовательской информационной системы, автоматизирующей хранение, передачу, обработку и представление информации о требуемых ресурсах на проекты по строительству, ремонту и благоустройству автодорог.

3 Содержание пояснительной записки

Введение

1 Анализ и разработка требований

2 Проектирование программного обеспечения

3 Разработка и интеграция модулей программного обеспечения

4 Тестирование и отладка программного обеспечения

5 Инструкция по эксплуатации программного обеспечения

Заключение

Список использованных источников

4 Перечень графического материала

5 Календарный график работы над проектом на весь период проектирования

25.10-31.10.2024 – анализ поставленной задачи; 01.11-07.11.2024 – проектирование ПО;
08.11-25.11.2024 – разработка и интеграция модулей ПО; 26.11-02.12.2024 – тестирование
и отладка ПО; 25.10-04.12.2024 – написание и проверка программной документации,
оформление пояснительной записки; 05.12.2024 – сдача курсового проекта на проверку;
06.12.2024 - защита курсового проекта

6 Срок сдачи студентом законченного курсового проекта 05 декабря 2024

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	6
1.3 Описание алгоритма функционирования системы	6
1.4 Выбор состава программных и технических средств	8
2 Проектирование программного обеспечения	10
2.1 Разработка архитектуры программного обеспечения.....	10
2.2 Проектирование интерфейса пользователя.....	10
2.3 Проектирование базы данных	11
3 Разработка и интеграция модулей программного обеспечения.....	13
3.1 Разработка программных модулей.....	13
3.2 Реализация интерфейса пользователя.....	14
3.3 Разграничение прав доступа пользователей	17
3.4 Экспорт и импорт данных.....	18
4 Тестирование и отладка программного обеспечения.....	22
4.1 Структурное тестирование.....	22
4.2 Функциональное тестирование	24
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

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В настоящем курсовом проекте применяют следующие сокращения и обозначения:

БД – база данных

ИС – информационная система

ОС – операционная система

ПО – программное обеспечение

СУБД – система управления базами данных

API – программный интерфейс приложения

ASP – активные серверные страницы

HTTP – протокол передачи гипертекста

IDE – интегрированная среда разработки

JSON – обозначение объекта JavaScript

REST – представление состояния передачи

ВВЕДЕНИЕ

Актуальность курсового проекта заключается в том, что применение БД в разработке подсистемы учёта материалов и спецтехники для автодорог позволяет значительно повысить эффективность управления ресурсами и улучшить планирование. В условиях современного строительства и эксплуатации автодорог, где объёмы материалов и техники могут быть значительными, важно иметь систему, которая обеспечит точный учёт, оперативный доступ к информации и возможность анализа данных.

На многих предприятиях по-прежнему используются устаревшие методы учёта, такие как табличные редакторы или общие бухгалтерские программы. Хотя эти инструменты могут предоставлять базовые функции для создания расчётов и ведения учёта, они не способны обеспечить необходимую гибкость и актуальность информации для всех участников процесса. В результате, подрядчики и другие заинтересованные стороны часто сталкиваются с проблемами, связанными с недостаточной прозрачностью данных, задержками в получении информации и трудностями в анализе текущего состояния ресурсов.

Целью курсового проектирования является разработка многопользовательской клиент-серверной системы для учета материалов и спецтехники. Эта система должна обеспечить удобный интерфейс для пользователей, а также надежное хранение и обработку данных.

Для достижения поставленной цели требуется решить следующие задачи:

- выполнить 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 мобильного приложения,
- спроектировать БД,
- создать БД в конкретной СУБД,
- реализовать разграничение прав доступа пользователей,
- реализовать защиту данных,
- разработать интерфейс веб-приложение,
- разработать веб-приложение,
- реализовать экспорт данных в виде файлов .xlsx,
- реализовать работу веб-приложения по средствам контроллеров ASP.NET,
- выполнить структурное тестирование ПО,
- выполнить функциональное тестирование ПО,
- разработать программную и эксплуатационную документацию.

В результате выполнения поставленных задач будет разработано веб-приложение для учёта ресурсов для автодорог.

1 Анализ и разработка требований

1.1 Назначение и область применения

ИС предназначена для автоматизации процессов управления ресурсами в сфере строительства и эксплуатации автодорог. Основное назначение данной ИС заключается в обеспечении точного учёта, мониторинга данных о проектах, что позволяет повысить эффективность работы организации, улучшить планирование и сократить затраты.

Разрабатываемая ИС ориентирована на проектную организацию, занимающуюся разработкой проектной документации для строительства и ремонта автодорог.

1.2 Постановка задачи

Необходимо разработать веб-приложение, которое предоставит доступ к следующей функциональности:

- авторизации пользователей,
- регистрации новых пользователей,
- просмотру и редактированию информации о проектах, их задачах и требуемых ресурсов для выполнения задач,
- просмотру и редактированию информации о имеющихся пользователях в ИС,
- генерации сметы по проекту в файле .xlsx,
- публикации проектов для подрядчиков.

1.3 Описание алгоритма функционирования системы

При запуске приложения отображается страница с опубликованными проектами, в верхней панели приложения находится кнопка аутентификации,

после нажатия на кнопку открывается модальное окно аутентификации с возможностью авторизоваться, как администратор и как проектировщик.

Неавторизованный пользователь может открыть проект, нажав на него, где будут указаны задачи, которые можно аналогично открыть и просмотреть подробную информацию с указанием требуемых ресурсов.

Проектировщик может то

же, что может делать неавторизованный пользователь, а также редактировать информацию о проектах и их составляющих (при указании для задачи ресурсов можно указать существующий или указать новый), генерировать смету по проекту в файл .xlsx.

Администратор может то же, что может делать проектировщик, а также редактировать информацию о ресурсах и пользователях.

На рисунке 1 изображена диаграмма вариантов использования приложения различными категориями пользователей.

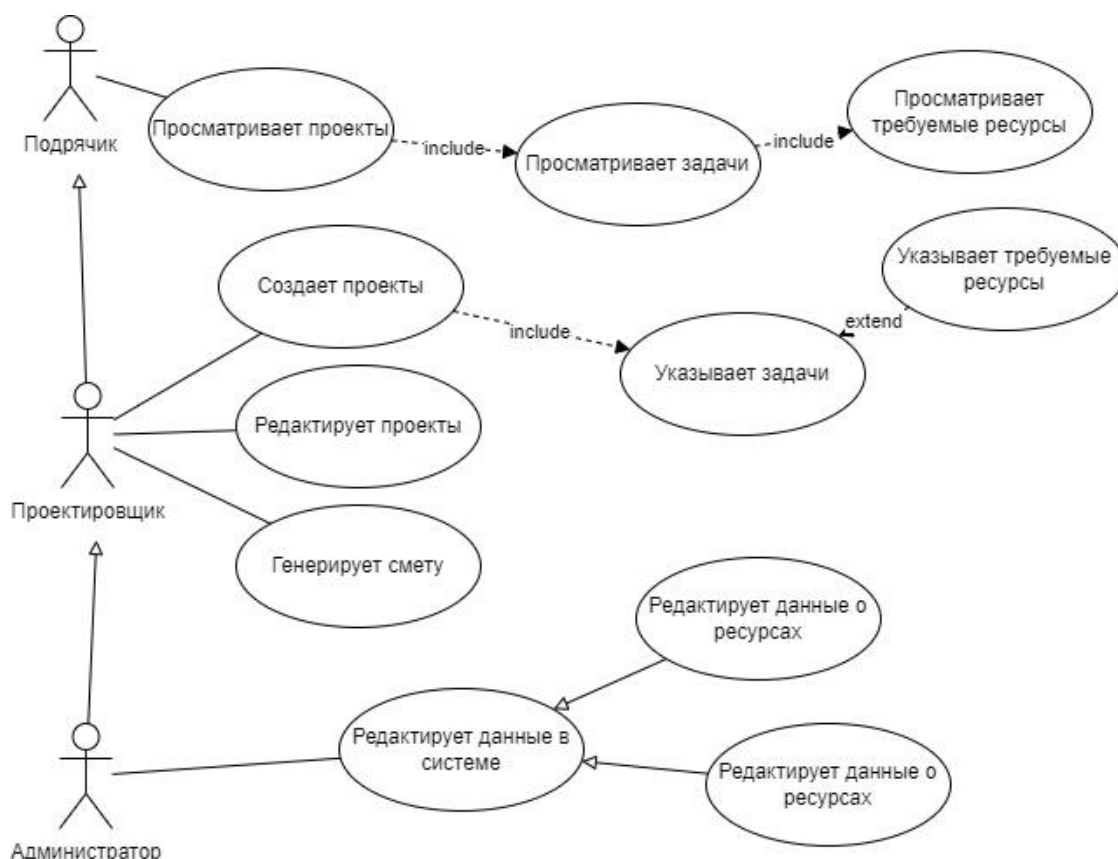


Рисунок 1 – Диаграмма вариантов использования

1.4 Выбор состава программных и технических средств

Согласно цели проекта требуется создать веб-приложение для учета материалов и спецтехники.

Работа с ИС будет осуществляться на мобильных устройствах и персональных компьютерах, которые имеют браузер.

В качестве СУБД выбрана MySQL версии не ниже 8.0. MySQL была выбрана благодаря своей высокой производительности, надежности и широкому распространению. Она поддерживает многопользовательский доступ, что является важным для нашей системы, а также предоставляет мощные инструменты для работы с данными, включая возможности для создания сложных запросов и оптимизации производительности. Кроме того, MySQL имеет активное сообщество и обширную документацию, что упрощает процесс разработки и устранения возможных проблем. и индексации, поддерживает различные типы данных, включая JSON.

Приложение будет написано на языке программирования C#, который был выбран благодаря своей сильной типизации, поддержке объектно-ориентированного программирования и богатой стандартной библиотеке. C# обеспечивает высокую надежность и стабильность за счет обнаружения ошибок на этапе компиляции, а также позволяет создавать модульные и легко поддерживаемые приложения.

Для разработки приложения будет использоваться интегрированная среда разработки Visual Studio 2022. Эта IDE была выбрана благодаря своему мощному функционалу, удобному интерфейсу и отличной поддержке языка C#. Visual Studio 2022 предлагает инструменты для отладки, тестирования и профилирования, а также интеграцию с системами контроля версий и доступ к множеству расширений, что делает процесс разработки более эффективным и продуктивным.

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

- дистрибутив Linux (например, Ubuntu 20.04 LTS),
- сервер БД: MySQL не ниже 6.0,
- процессор минимум 2 ядра (рекомендуется 4 и более),
- свободная оперативная память объемом 1 ГБ,
- ПО для конфигурирования, управления и администрирования сервера БД: MySQL Workbench версия 8.0,
- места на диске минимум 20 ГБ.

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

- любая ОС, поддерживающая современные веб-браузеры,
- процессор с частотой 1,6 ГГц,
- оперативная память в объеме 1 ГБ.

2 Проектирование программного обеспечения

2.1 Разработка архитектуры программного обеспечения

Архитектура приложения построена на основе клиент-серверной модели и включает в себя серверную часть, веб-приложение и БД. Все компоненты будут развернуты в контейнерах Docker.

Для серверной части будет создан API, позволяющий клиенту взаимодействовать с БД. Она будет использовать СУБД MySQL [5], которая будет хранить информацию о пользователях, проектах, задачах, материалах, спецтехнике и рабочих.

Веб-приложение будет написано на фреймворке ASP.NET. Взаимодействие с сервером будет происходить при помощи HTTP-запросов к RESTful API, ответы будут приходить в формате JSON. Для отправки запросов будет использоваться веб-клиент ASP.NET [3]. Архитектура системы отображена на диаграмме развертывания на рисунке 2

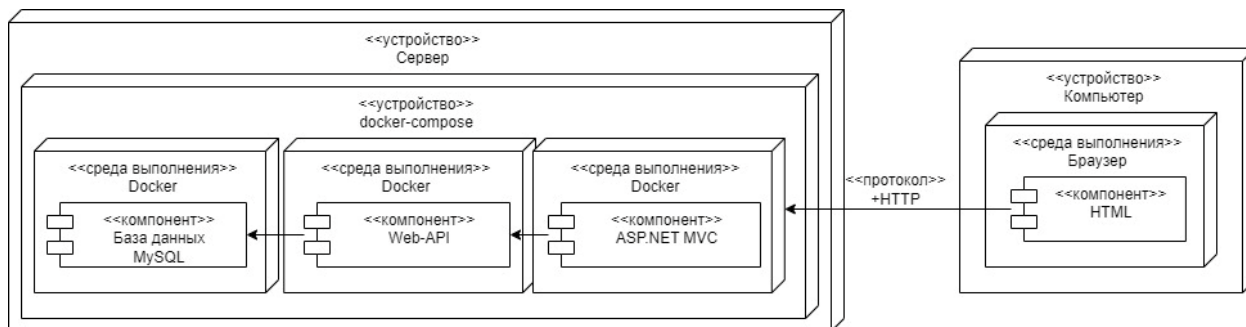


Рисунок 3 – Диаграмма развертывания

2.2 Проектирование базы данных

Требуется разработать БД для системы учета ресурсов. Система будет использоваться проектировщиками, администраторами, подрядчиками [3].

Модели БД созданы при помощи MySQL Workbench. На рисунке 3 в виде ERD показана часть физической модели предметной области, связанной с поиском экспертов для созданной задачи.

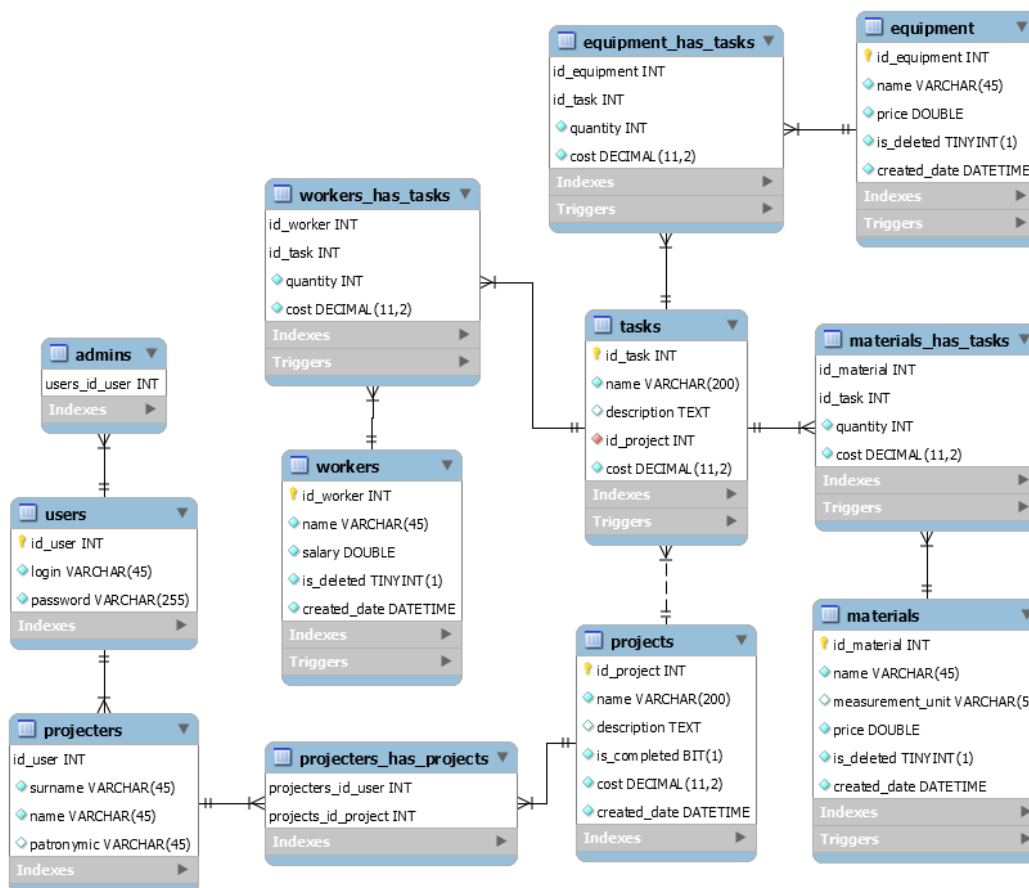


Рисунок 3 – MySQL Workbench. Физическая модель БД

2.3 Проектирование интерфейса пользователя

В рамках разработки веб-приложения создан интерфейс пользователя в виде мокапов при помощи инструмента Figma. Эти визуальные представления позволяют наглядно увидеть структуру приложения, его основные элементы и функциональность.

Мокапы главной страницы с проектами, страницы с материалами, страницы одного из проектов интерфейса для роли Администратора веб-приложения представлены на рисунке 4.

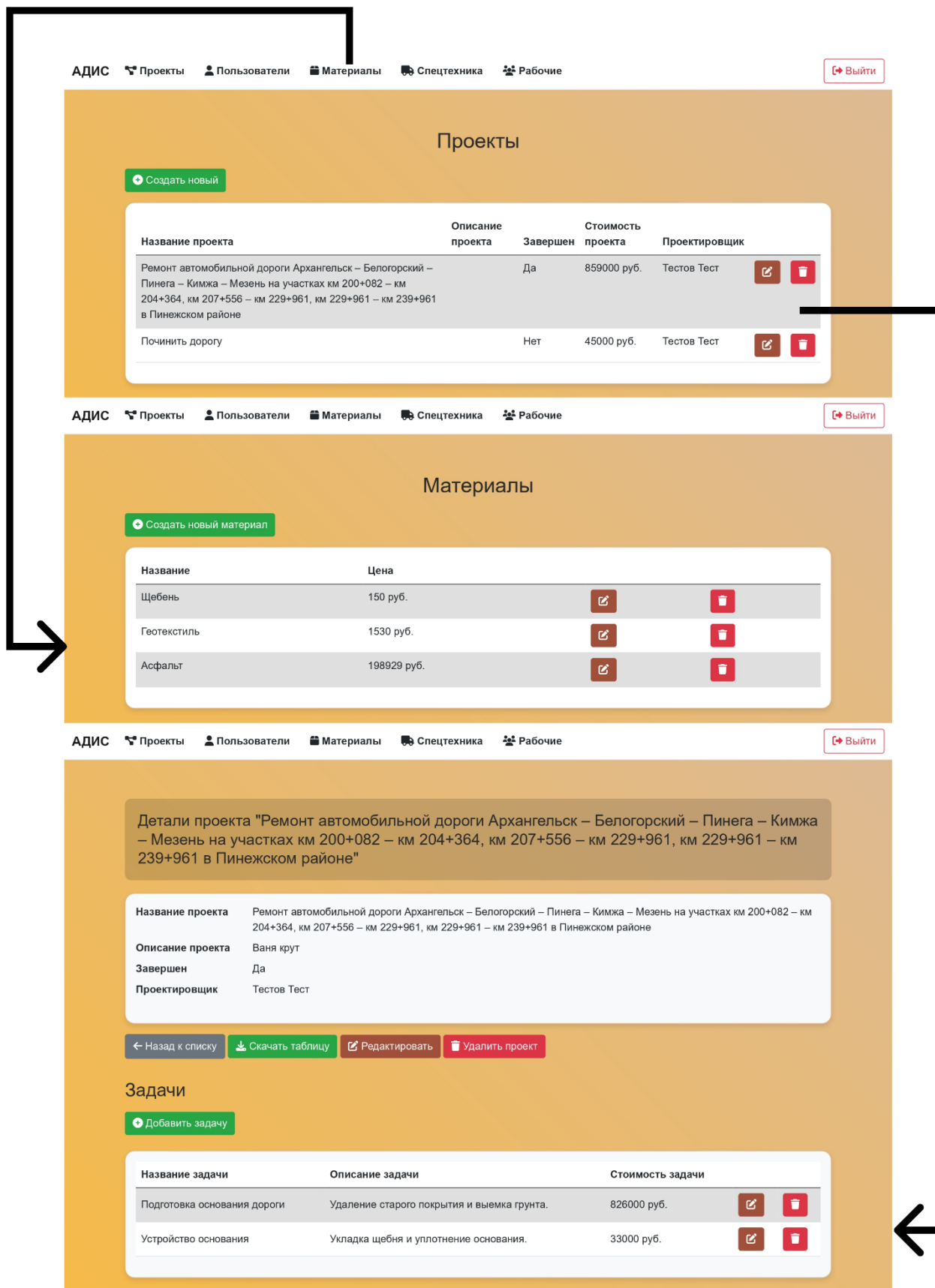


Рисунок 4 – Figma. Мокапы интерфейса пользователя веб-приложения

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

Для курсового проекта разработано веб-приложение на C# с использованием фреймворка ASP.NET в Visual Studio [2].

Взаимодействие мобильного приложения с сервером будет происходить при помощи HTTP-запросов к API, а ответы будут возвращаться в формате JSON. Для реализации HTTP-запросов используется сетевой клиент, который позволяет отправлять запросы и получать ответы от сервера. Код метода для получения списка проектов путем отправки GET-запроса на сервер представлен листингом 1.

Листинг 1 – Код метода для отправки GET-запроса на сервер

```
// Получение страницы с проектами
// GET: Projects
public async Task<IActionResult> Index()
{
    // Проверяем, аутентифицирован ли пользователь
    if (User.Identity.IsAuthenticated)
    {
        // Если пользователь имеет роль "Admin"
        if (User.IsInRole("Admin"))
        {
            // Получаем список всех проектов и возвращаем
            // представление с этими данными
            return View(await
                _httpClientService.GetHttpClient().GetFromJsonAsync<List<Project>>("Projects"));
        }

        // Если пользователь не администратор, получаем
        // проекты, связанные с конкретным пользователем
        return View(await
            _httpClientService.GetHttpClient().GetFromJsonAsync<List<Project>>($"Projects?idProjecter={Int32.Parse(User.FindFirst(ClaimTypes.NameIdentifier).Value)}"));
    }
    else
    {

```

```

        // Если пользователь не аутентифицирован, получаем
        только завершённые проекты
        return View(await
        _httpClientService.GetHttpClient().GetFromJsonAsync<List<Project>>("Projects?isCompleted=true"));
    }
}

```

Код представления в API серверной части для поиска материалов по имени представлен листингом 2.

Листинг 2 – Код представления для поиска материалов по имени в API

```

// GET: api/Materials
[HttpGet]
public async Task<ActionResult<IEnumerable<Material>>>
GetMaterials([FromQuery] string? name, [FromQuery] bool?
isDeleted)
{
    var query = _context.Materials.AsQueryable();

    // Фильтрация по имени, если оно указано
    if (!string.IsNullOrEmpty(name))
    {
        query = query.Where(m => m.Name == name);
    }

    // Фильтрация по isDeleted, если оно указано
    if (isDeleted.HasValue)
    {
        query = query.Where(m => m.IsDeleted ==
isDeleted.Value);
    }

    // Получение списка материалов
    return await query.ToListAsync();
}

```

3.2 Реализация интерфейса пользователя

Интерфейс разработан с использованием постраничной навигации и различных элементов управления, что упрощает взаимодействие пользователя

с приложением. В данном разделе представлено представление, которое позволяет пользователю вводить информацию о новой спецтехнике.

На странице реализована форма, которая включает поля для ввода названия, цены и количества спецтехники. Для улучшения пользовательского опыта добавлено автозаполнение для поля "Название", которое срабатывает при вводе текста. Также предусмотрены механизмы валидации, которые отображают ошибки, если введенные данные не соответствуют требованиям.

Код представления для добавления спецтехники представлен листингом 3.

Листинг 3 – Код представления для добавления спецтехники

```
@model AutodeskInfoSystem.Models.Equipment

@{
    ViewData["Title"] = "Добавление спецтехники"; //
    Устанавливаем заголовок страницы
}

<h1 class="mb-4">Добавление спецтехники</h1> <!-- Заголовок
страницы -->

<div class="card p-4"> <!-- Контейнер для формы с отступами -->
    <div class="row"> <!-- Строка для размещения элементов
формы -->
        <div class="col"> <!-- Колонка для формы -->
            <form asp-action="Create"> <!-- Указываем действие
контроллера для обработки формы -->
                <div asp-validation-summary="ModelOnly"
class="text-danger"></div> <!-- Отображение ошибок валидации
модели -->
                    <div class="form-group"> <!-- Группа для поля
"Название" -->
                        <label asp-for="Name" class="control-
label">Название</label> <!-- Метка для поля "Название" -->
                            <input asp-for="Name" class="form-control"
id="equipmentName" onkeyup="fetchSimilarNames()" /> <!-- Поле
ввода для названия с автозаполнением -->
                                <span asp-validation-for="Name"
class="text-danger"></span> <!-- Отображение ошибок валидации
для поля "Название" -->
                                    <ul id="suggestions" class="list-group mt-
2" style="display:none;"></ul> <!-- Список для автозаполнения,
скрыт по умолчанию -->
```



```

</div>

<div class="form-group"> <!-- Группа для поля
"Цена" -->
    <label asp-for="Price" class="control-
label">Цена</label> <!-- Метка для поля "Цена" -->
    <input asp-for="Price" class="form-control"
id="equipmentPrice" /> <!-- Поле ввода для цены -->
    <span asp-validation-for="Price"
class="text-danger"></span> <!-- Отображение ошибок валидации
для поля "Цена" -->
</div>

<div class="form-group"> <!-- Группа для поля
"Количество" -->
    <label asp-for="Quantity" class="control-
label">Количество</label> <!-- Метка для поля "Количество" -->
    <input asp-for="Quantity" class="form-
control" /> <!-- Поле ввода для количества -->
    <span asp-validation-for="Quantity"
class="text-danger"></span> <!-- Отображение ошибок валидации
для поля "Количество" -->
</div>

<input type="hidden" name="IdTask"
value="@ViewBag.IdTask" /> <!-- Скрытое поле для передачи
идентификатора задачи -->

<div class="form-group"> <!-- Группа для кнопки
отправки формы -->
    <button class="btn btn-primary"
type="submit"> <!-- Кнопка для отправки формы -->
        <i class="fas fa-check"></i> <!--
Иконка для кнопки -->
        Создать
    </button>
</div>

<div class="mt-3"> <!-- Контейнер для кнопки
"Назад" -->
    <a asp-controller="Tasks" asp-
action="Details" asp-route-id="@ViewBag.IdTask" class="btn btn-
outline-primary"> <!-- Ссылка для возврата на страницу деталей
задачи -->
        <i class="fas fa-arrow-left"></i> <!--
Иконка для ссылки -->
        Назад
    </a>
</div>
</form>
</div>
</div>
</div>

```

```
@section Scripts { <!-- Раздел для подключения скриптов -->
    <script src="~/js/create-and-edit-equipment.js"></script>
    <!-- Подключение скрипта для автозаполнения и других функций -->
}
```

3.3 Разграничение прав доступа пользователей

В веб-приложении разработано разграничение прав доступа пользователей. Для этого в приложении реализованы авторизация и регистрация. Неавторизованный пользователь может просматривать завершённые проекты, с их задачами и требуемые ресурсы. Пользователь с ролью Проектировщик имеет возможность к тому, что может неавторизованный пользователь, редактировать информацию о проектах, задачах и требуемых ресурсов. Страница списка проекта для роли Проектировщика на рисунке 5, а для неавторизованного пользователя на рисунке 6.

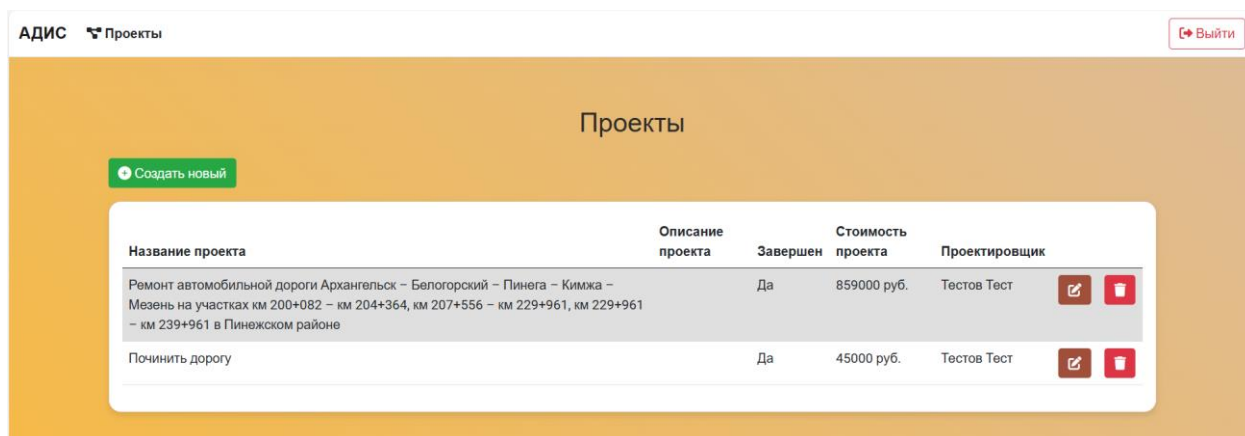


Рисунок 5 – АДИС. Вид страницы списка проекта для роли Проектировщик

АДИС

Проекты

Войти

Проекты

Название проекта	Описание проекта	Завершен	Стоимость проекта
Ремонт автомобильной дороги Архангельск – Белогорский – Пинега – Кимжа – Мезень на участках км 200+082 – км 204+364, км 207+556 – км 229+961, км 229+961 – км 239+961 в Пинежском районе		Да	859000 руб.
Починить дорогу		Да	45000 руб.

Рисунок 6 – АДИС. Вид страницы списка проекта для неавторизованного пользователя

3.4 Экспорт данных

В веб-приложении реализован экспорт данных в файл .xlsx при помощи библиотеки ClosedXML [4].

Пример экспорта данных о проекте, его задачах и требуемых ресурсов можно увидеть на листинге 4.

Листинг 4 – Код экспорта данных

```
//Генерируется лист Excel на основе данных о проекте
private void GenerateProjectWorksheet(IXLWorksheet worksheet,
Project project)
{
    // Устанавливаем заголовок "Смета" в ячейку A1
    worksheet.Cell(1, 1).Value = "Смета";
    worksheet.Cell(1, 1).Style.Alignment.Horizontal =
    XLAAlignmentHorizontalValues.Center; // Центрируем текст
    worksheet.Cell(1, 1).Style.Font.Bold = true; // Делаем
    текст жирным
    worksheet.Range(worksheet.Cell(1, 1), worksheet.Cell(1,
    6)).Merge(); // Объединяем ячейки A1:F1

    // Устанавливаем подзаголовок с названием проекта в ячейку
    A2
    worksheet.Cell(2, 1).Value = $"по проекту
    \"{project.Name}\"";
    worksheet.Cell(2, 1).Style.Alignment.Horizontal =
    XLAAlignmentHorizontalValues.Center; // Центрируем текст
```

```

        worksheet.Cell(2, 1).Style.Alignment.WrapText = true; //
Переносим текст, если он длинный
        worksheet.Cell(2, 1).Style.Font.Bold = true; // Делаем
текст жирным
        worksheet.Range(worksheet.Cell(2, 1), worksheet.Cell(2,
6)).Merge(); // Объединяем ячейки A2:F2

        // Устанавливаем заголовки столбцов
        worksheet.Cell(3, 1).Value = "Наименование задачи";
        worksheet.Cell(3, 2).Value = "Наименование ресурса";
        worksheet.Cell(3, 3).Value = "Количество";
        worksheet.Cell(3, 4).Value = "Единица измерения";
        worksheet.Cell(3, 5).Value = "Цена, зарплата, руб.";
        worksheet.Cell(3, 6).Value = "Стоимость, руб.";
        worksheet.Range(worksheet.Cell(3, 1), worksheet.Cell(3,
6)).Style.Font.Bold = true; // Делаем заголовки жирными

        int row = 3; // Начинаем с третьей строки для заполнения
данными
        foreach (var task in project.Tasks) // Проходим по всем
задачам проекта
        {
            row++; // Переходим на следующую строку
            worksheet.Cell(row, 1).Value = task.Name; // Записываем
название задачи
            worksheet.Range(worksheet.Cell(row, 1),
worksheet.Cell(row, 6)).Merge(); // Объединяем ячейки для
названия задачи

            // Если у задачи есть материалы
            if (task.MaterialsHasTasks.Count > 0)
            {
                row++; // Переходим на следующую строку
                worksheet.Cell(row, 2).Value = "Материалы"; //
Записываем заголовок "Материалы"
                worksheet.Cell(row, 2).Style.Alignment.Horizontal =
XlAlignmentHorizontalValues.Center; // Центрируем текст
                worksheet.Range(worksheet.Cell(row, 2),
worksheet.Cell(row, 6)).Merge(); // Объединяем ячейки
                worksheet.Cell(row, 2).Style.Font.Bold = true; //
Делаем текст жирным

                // Проходим по всем материалам задачи
                foreach (var material in task.MaterialsHasTasks)
                {
                    row++; // Переходим на следующую строку
                    worksheet.Cell(row, 2).Value =
material.IdMaterialNavigation.Name; // Записываем название
материала
                    worksheet.Cell(row, 3).Value =
material.Quantity; // Записываем количество

```

```

        worksheet.Cell(row, 4).Value =
material.IdMaterialNavigation.MeasurementUnit; // Записываем
единицу измерения
        worksheet.Cell(row, 5).Value =
material.IdMaterialNavigation.Price; // Записываем цену
        worksheet.Cell(row, 6).Value = material.Cost;
// Записываем стоимость
    }
}

// Если у задачи есть спецтехника
if (task.EquipmentHasTasks.Count > 0)
{
    row++; // Переходим на следующую строку
    worksheet.Cell(row, 2).Value = "Спецтехника"; //
Записываем заголовок "Спецтехника"
    worksheet.Cell(row, 2).Style.Alignment.Horizontal =
XlAlignmentHorizontalValues.Center; // Центрируем текст
    worksheet.Range(worksheet.Cell(row, 2),
worksheet.Cell(row, 6)).Merge(); // Объединяем ячейки
    worksheet.Cell(row, 2).Style.Font.Bold = true; //
Делаем текст жирным

    // Проходим по всем единицам спецтехники
    foreach (var equipment in task.EquipmentHasTasks)
    {
        row++; // Переходим на следующую строку
        worksheet.Cell(row, 2).Value =
equipment.IdEquipmentNavigation.Name; // Записываем название
спецтехники
        worksheet.Cell(row, 3).Value =
equipment.Quantity; // Записываем количество
        worksheet.Cell(row, 4).Value = "шт"; //
Указываем единицу измерения (штуки)
        worksheet.Cell(row, 5).Value =
equipment.IdEquipmentNavigation.Price; // Записываем цену
        worksheet.Cell(row, 6).Value = equipment.Cost;
// Записываем стоимость
    }
}

// Если у задачи есть рабочие
if (task.WorkersHasTasks.Count > 0)
{
    row++; // Переходим на следующую строку
    worksheet.Cell(row, 2).Value = "Рабочие"; //
Записываем заголовок "Рабочие"
    worksheet.Cell(row, 2).Style.Alignment.Horizontal =
XlAlignmentHorizontalValues.Center; // Центрируем текст
    worksheet.Range(worksheet.Cell(row, 2),
worksheet.Cell(row, 6)).Merge(); // Объединяем ячейки
    worksheet.Cell(row, 2).Style.Font.Bold = true; //
Делаем текст жирным

```

```

        // Проходим по всем рабочим
        foreach (var worker in task.WorkersHasTasks)
        {
            row++; // Переходим на следующую строку
            worksheet.Cell(row, 2).Value =
worker.IdWorkerNavigation.Name; // Записываем имя рабочего
            worksheet.Cell(row, 3).Value = worker.Quantity;
// Записываем количество
            worksheet.Cell(row, 5).Value =
worker.IdWorkerNavigation.Salary; // Записываем зарплату
рабочего
            worksheet.Cell(row, 6).Value = worker.Cost; //
Записываем стоимость
        }
    }

    // Переходим на строку для итогов
    row++;
    worksheet.Cell(row, 1).Value = "Всего:"; // Записываем
заголовок "Всего"
    worksheet.Cell(row, 1).Style.Alignment.Horizontal =
XlAlignmentHorizontalValues.Right; // Выравниваем текст по
правому краю
    worksheet.Range(worksheet.Cell(row, 1), worksheet.Cell(row,
5)).Merge(); // Объединяем ячейки для заголовка
    worksheet.Cell(row, 1).Style.Font.Bold = true; // Делаем
текст жирным
    worksheet.Cell(row, 6).Value = project.Cost; // Записываем
общую стоимость проекта
    worksheet.Cell(row, 6).Style.Font.Bold = true; // Делаем
текст жирным

    // Автоматически подгоняем ширину столбцов под содержимое
    worksheet.Columns().AdjustToContents();
    worksheet.Rows().AdjustToContents();

    // Устанавливаем границы для всего диапазона данных
    worksheet.Range(worksheet.Cell(1, 1), worksheet.Cell(row,
6)).Style.Border.SetBottomBorder(XlBorderStyleValues.Medium);
    worksheet.Range(worksheet.Cell(1, 1), worksheet.Cell(row,
6)).Style.Border.SetLeftBorder(XlBorderStyleValues.Medium);
    worksheet.Range(worksheet.Cell(1, 1), worksheet.Cell(row,
6)).Style.Border.SetRightBorder(XlBorderStyleValues.Medium);
    worksheet.Range(worksheet.Cell(1, 1), worksheet.Cell(row,
6)).Style.Border.SetTopBorder(XlBorderStyleValues.Medium);
}

```


4 Тестирование и отладка программного обеспечения

4.1 Структурное тестирование

Во время курсового проектирования проведено структурное тестирование для сервиса токенов `TokenService` [1]. Для этого использовалась библиотека `Moq` для создания имитации поведения зависимостей конфигурации и библиотека `xUnit` для написания юнит-тестов. Код юнит-теста для создания токенов представлен листингом 5.

Листинг 5 – Код unit-теста для `TokenService`

```
public class TokenServiceTests
{
    private readonly TokenService _tokenService; //
    Экземпляр сервиса токенов
    private readonly Mock<IConfiguration> _configMock; //
    Мок для конфигурации

    public TokenServiceTests()
    {
        // Настройка мока конфигурации
        _configMock = new Mock<IConfiguration>();
        // Установка значения для ключа JWT
        _configMock.Setup(config =>
config["JWT:Key"]).Returns("SuperSecretKey_JSM_The_Best_8374129
04361249126341789268041263489127346784263490"); // Обновленный
ключ
        // Установка значения для издателя JWT
        _configMock.Setup(config =>
config["JWT:Issuer"]).Returns("my_issuer");
        // Установка значения для аудитории JWT
        _configMock.Setup(config =>
config["JWT:Audience"]).Returns("my_audience");

        // Инициализация TokenService с моком конфигурации
        _tokenService = new
TokenService(_configMock.Object);
    }

    [Fact]
    public void
CreateToken_ReturnsValidToken_WhenUserIsAdmin()
```

```

{
    // Arrange
    // Создание пользователя с ролью администратора
    var user = new User
    {
        Login = "adminUser",
        IdUser = 1,
        Admin = new Admin() // Указываем, что это
администратор
    };

    // Act
    // Создание токена для пользователя
    var token = _tokenService.CreateToken(user);

    // Assert
    var tokenHandler = new JwtSecurityTokenHandler();
    // Создание обработчика токенов
    var tokenValidationParameters = new
TokenValidationParameters
    {
        ValidateIssuerSigningKey = true, // Включаем
проверку ключа подписи
        IssuerSigningKey = new
SymmetricSecurityKey(System.Text.Encoding.UTF8.GetBytes("SuperS
ecretKey_JSM_The_Best_83741290436124912634178926804126348912734
6784263490")), // Указываем ключ для проверки
        ValidateIssuer = false, // Отключаем проверку
издателя
        ValidateAudience = false // Отключаем проверку
аудитории
    };

    // Проверяем, что токен можно валидировать
    var principal = tokenHandler.ValidateToken(token,
tokenValidationParameters, out var validatedToken);
    Assert.NotNull(principal); // Убедитесь, что токен
не равен null
    Assert.Equal("adminUser",
principal.FindFirst(ClaimTypes.Name)?.Value); // Проверяем, что
имя пользователя соответствует ожидаемому
    Assert.Equal("1",
principal.FindFirst(ClaimTypes.NameIdentifier)?.Value); //
Проверяем, что идентификатор пользователя соответствует
ожидаемому
    Assert.Equal("Admin",
principal.FindFirst(ClaimTypes.Role)?.Value); // Проверяем, что
роль соответствует ожидаемой
    }
}

```

На рисунке 7 изображена консоль с результатами тестирования, где отображается информация о перехваченных исключениях и успешном тестировании.

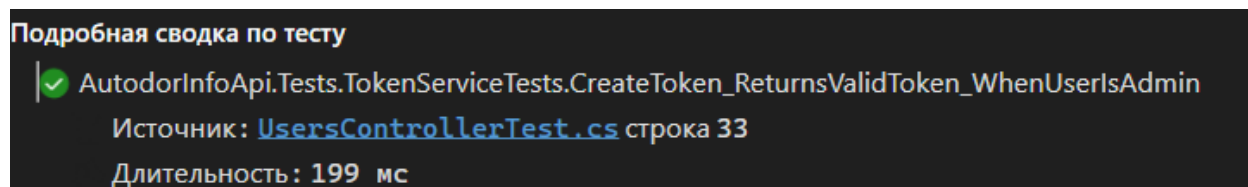


Рисунок 7 – Visual Studio. Результаты unit-тестирования

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование веб-приложения методом «черного ящика», результаты тестирования представлены в таблице 1.

Таблица 1 – Набор тестов веб-приложения

Действие	Ожидаемый результат	Фактический результат
Нажать на кнопку войти	Открытие модального окна для авторизации	Совпадает с ожидаемым
Нажать на кнопку войти, ввести логин «admin» и пароль «admin» в соответствующие поля и нажать кнопку «Войти»	Открытие страницы списков проектов для пользователя с ролью Администратор	Совпадает с ожидаемым
Нажать кнопку «Создать проект», ввести в поле название проекта «Ремонт дороги» и нажать кнопку «Создать»	Открытие страницы списков проектов, где отобразиться созданный проект	Совпадает с ожидаемым
Нажатие на строчку с проектом	Переход на страницу нажатого проекта	Совпадает с ожидаемым
Нажать кнопку «Скачать таблицу» на странице проекта	Таблица скачивается	Совпадает с ожидаемым

5 Инструкция по эксплуатации программного обеспечения

5.1 Установка программного обеспечения

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

- дистрибутив Linux (например, Ubuntu 20.04 LTS)
- сервер БД: MySQL не ниже 6.0,
- процессор минимум 2 ядра (рекомендуется 4 и более),
- свободная оперативная память объемом 1 ГБ,
- ПО для конфигурирования, управления и администрирования сервера БД: MySQL Workbench версия 8.0,
- места на диске минимум 20 ГБ.

Для установки серверной части требуется установить Docker с docker-compose, в папке environment настроить систему, перейти в корневую папку решения и в терминале использовать команду `docker compose up -d`.

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

- любая ОС, поддерживающая современные веб-браузеры,
- процессор с частотой 1,6 ГГц,
- оперативная память в объеме 1 ГБ.

Веб-приложение можно будет найти в браузере на порту 5000, если не изменен файл `docker-compose.yaml`.

В веб-приложении используются данные указывающиеся в файле `environment/default.env`.

5.2 Инструкция по работе

При запуске приложения пользователи попадают на страницу, где представлен список проектов. В зависимости от их статуса авторизации и роли, отображаемая информация будет различаться.

Неавторизованный пользователь увидит только завершённые проекты, что позволяет ему ознакомиться с уже выполненными работами, но не даёт доступа к более детальной информации.

Пользователь с ролью Проектировщик получит доступ к проектам, которые он сам создал, что позволяет ему управлять своими задачами и отслеживать прогресс.

Администратор, обладая полными правами, сможет просмотреть все проекты, что даёт ему возможность контролировать и управлять всеми аспектами работы.

Страница проектов для роли Администратор представлена на рисунке 8, где можно увидеть все доступные функции и информацию.

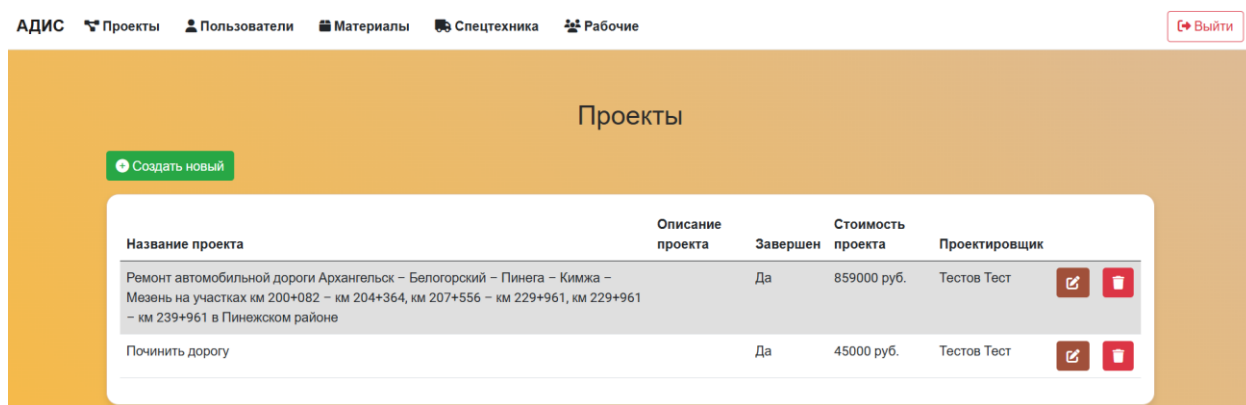


Рисунок 8 – АДИС. Вид страницы списка проекта для роли Администратор

После этого все пользователи могут открыть информацию о проекте, нажав на его строку. На странице отобразятся задачи для выполнения проекта, а также будет возможность скачать таблицу с данными о проекте. Страница с

подробностями проекта для авторизованного пользователя представлена на рисунке 9.

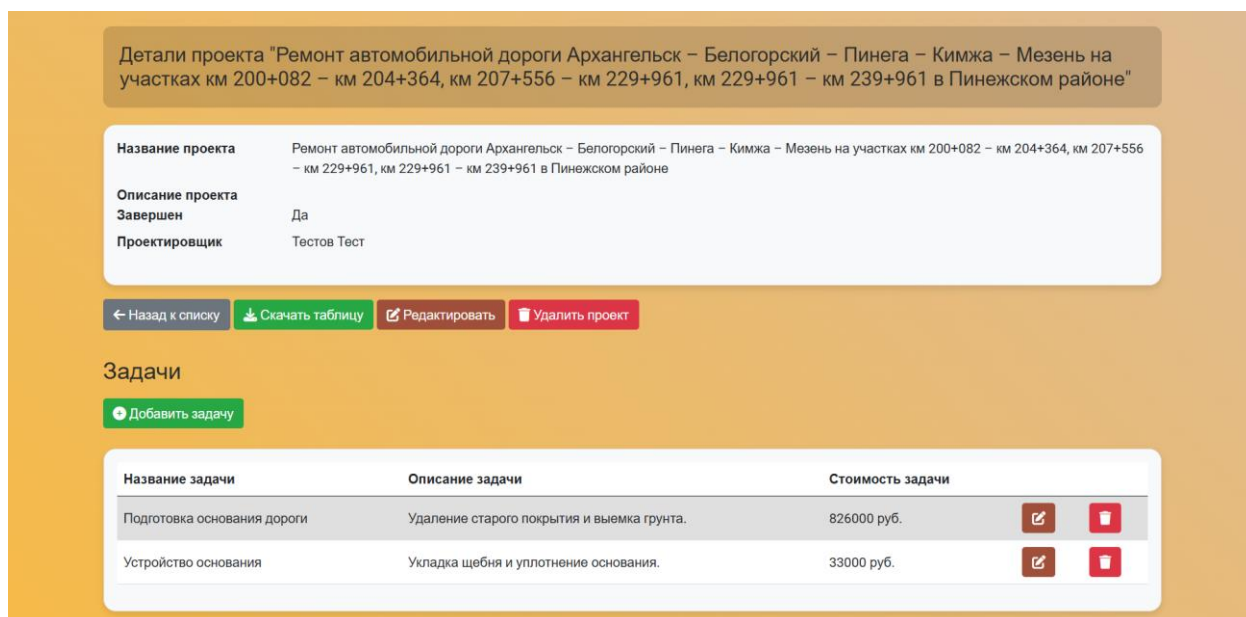


Рисунок 9 – АДИС. Вид страницы подробностей проекта для авторизованного пользователя

После можно нажать на задачу и откроется страница с требуемыми ресурсами на эту задачу.

Если пользователь будет авторизован, то на каждой странице сопровождаются кнопками с созданием, редактированием, удалением проекта, задачи, требуемого ресурса.

При создании или редактировании любого ресурса под полем названия появляются подсказки, при введении названия, нажав на одну из подсказок, автозаполнятся поля названия и цены.

Пользователю с ролью Администратор на верхней панели имеет также кнопки: «Пользователи», «Материалы», «Спецтехника», «Рабочие».

Кнопка «Пользователи» открывает страницу со списком пользователей. Вверху списка есть кнопка «Зарегистрировать пользователя», при нажатии можно будет ввести данные нового пользователя и создать его. Также у строки с пользователем есть кнопки «Редактировать» со значком карандашика,

которая открывает страницу с формой редактирования, «Удалить» со значком мусорной корзины, которая уведомит пользователя о удалении пользователя. Страница списка пользователей представлена рисунком 10.

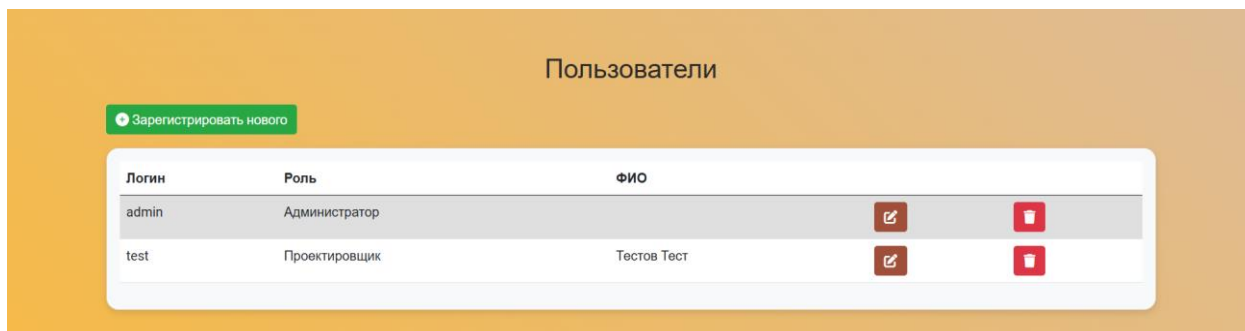


Рисунок 10 – АДИС. Вид страницы списка пользователей

Страницы «Материалы», «Спецтехника», «Рабочие» разработаны соответственно.

ЗАКЛЮЧЕНИЕ

В ходе курсового проектирования достигнута поставленная цель: разработана многопользовательская клиент-серверная система для учета материалов и спецтехники. Эта система должна обеспечить удобный интерфейс для пользователей, а также надежное хранение и обработку данных. Кроме того, решены все поставленные задачи:

- выполнен 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 мобильного приложения,
- спроектирована БД,
- создана БД в конкретной СУБД,
- реализовано разграничение прав доступа пользователей,
- реализована защита данных,
- разработан интерфейс веб-приложение,
- разработано веб-приложение,
- реализован экспорт данных в виде файлов .xlsx,
- реализована работа веб-приложения по средствам контроллеров ASP.NET,
- выполнено структурное тестирование ПО,
- выполнено функциональное тестирование ПО,
- разработана программная и эксплуатационная документацию.

В результате выполнения поставленных задач разработано веб-приложение для учёта ресурсов для автодорог.

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: 25.11.2024). – 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, 2023. – 400 с. – URL: <https://znanium.com/catalog/product/1895679> (дата обращения: 12.11.2024). – Режим доступа: по подписке. – Текст : электронный.

3. Гивакс, Д. Д. Паттерны проектирования API. – Санкт-Петербург : Питер, 2023. – 512 с. – URL: <https://ibooks.ru/bookshelf/390212/reading> (дата обращения: 24.11.2024). – Режим доступа: для зарегистрир. пользователей. – Текст: электронный.

4. Дадян, Э. Г. Данные: хранение и обработка : учебник. – Москва : ИНФРА-М, 2020. – 205 с. – URL: <https://znanium.com/catalog/product/1045133> (дата обращения: 17.11.2024). – Режим доступа: по подписке. – Текст : электронный.

5. 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, 2023. — 368 с. - URL: <https://znanium.com/catalog/product/1912454> (дата обращения: 03.11.2024). – Режим доступа: по подписке. — Текст : электронный.