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 РАБОТЫ ОТДЕЛА ТЕЛЕМАРКЕТИНГА

Л109. 25КП01. 020 ПЗ

(Обозначение документа)

МДК.02.01 Технология разработки программного обеспечения

Студент	ИСПП-35		08.12.2025	А.Е. Туйкова
	(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

ПАО «МТС» – Публичное акционерное общество «Мобильные ТелеСистемы»

ПК – персональный компьютер

ПО – программное обеспечение

СУБД – система управления базами данных

API – интерфейс прикладного программирования

ASP.NET – Active Server Pages для .NET

ERD – диаграмма «сущность-связь»

IDE – интегрированная среда разработки

ORM – объектно-реляционное отображение

REST – передача репрезентативного состояния

SDK – Software Development Kit

WPF – Windows Presentation Foundation

ВВЕДЕНИЕ

В последнее время информационные системы стали неотъемлемой частью повседневной жизни и бизнеса. Разработка информационных систем охватывает широкий спектр отраслей, включая финансы, образование и торговлю.

В условиях возрастающей конкуренции на рынке телекоммуникаций усиливается необходимость в эффективных решениях для управления клиентскими заявками и обслуживанием. Потребители ожидают более высокого уровня обслуживания и быстрого реагирования на свои запросы, что накладывает дополнительные требования на компании. Внедрение современных информационных систем позволяет не только улучшить качество обслуживания, но и оптимизировать внутренние процессы, снижая затраты и повышая общую продуктивность.

Актуальность разрабатываемого проекта обусловлена необходимостью повышения эффективности процессов оформления заявок на подключение к услугам домашнего интернета и телевидения. Внедрение подсистемы позволит оптимизировать взаимодействие сотрудников отдела телемаркетинга, упростить контроль над статусами заявок, что создаст более прозрачную и быструю систему обработки клиентских заявок.

Основной целью разработки подсистемы является создание оконного приложения, которое интегрирует функции управления данными клиентов и сотрудников, а также инструменты аналитики для оценки эффективности работы операторов. Внедрение нового ПО позволит операторам телемаркетинга более эффективно управлять заявками на подключение услуг, сокращая время их обработки и повышая уровень клиентского обслуживания. Также система обеспечит возможность генерации отчетов, что позволит супервайзерам лучше оценивать производительность сотрудников.

Для достижения поставленной цели требуется решить следующие задачи:

- провести анализ предметной области;
- провести сбор требований к функциональности подсистемы;
- проанализировать информационные источники по предметной области;
- спроектировать диаграмму вариантов использования;
- спроектировать архитектуру подсистемы;
- выбрать состав программных и технических средств для реализации приложения;
- спроектировать БД;
- спроектировать интерфейс оконного приложения;
- создать БД в выбранной СУБД;
- разработать API для взаимодействия оконного приложения с БД;
- разработать оконное приложение;
- реализовать просмотр данных об оформленных заявках оператором;
- реализовать функциональность оформления заявок;
- реализовать фильтрацию и сортировку данных о клиентах;
- реализовать экспорт данных в формате csv;
- реализовать разграничение прав доступа пользователей;
- выполнить структурное и функциональное тестирование ПО;
- разработать программную и эксплуатационную документацию.

1 Сбор и анализ требований

1.1 Назначение и область применения

Основным назначением разработки подсистемы является упрощение процесса оформления заявок на подключение клиентов к услугам домашнего интернета и телевидения в отделе телемаркетинга и улучшение контроля продаж услуг путем отслеживания статуса заявок и анализа успешности их оформления.

Подсистема предназначена для использования в отделе телемаркетинга в сфере фиксированных продаж домашнего интернета и телевидения, например, в ПАО «МТС». Основными пользователями ПО являются сотрудники отдела: супервайзер, операторы и неавторизованный сотрудник.

1.2 Постановка задачи

Требуется разработать оконное приложение, в котором необходимо реализовать следующие функциональные возможности:

- добавление и редактирование информации о клиентах и сотрудниках;
- фильтрация и сортировка данных о клиентах;
- просмотр личной статистики оператора;
- внесение информации о звонках клиенту;
- отслеживание статуса оформленных заявок на подключение услуг;
- формирование отчетов по оформленным заявкам за все время в формате csv.

Подсистема обрабатывает различные входные данные, которые включают персональные сведения о клиентах, информацию об услугах, данные о звонках операторов и учетные записи сотрудников. В результате система формирует выходные данные, такие как заявки на подключение,

отчеты о работе операторов, статистические сведения оператора, а также списки клиентов.

Интерфейс приложения должен быть интуитивно понятным и удобным для пользователей с базовыми навыками работы с компьютером. В целях безопасности предусмотрена обязательная аутентификация пользователей с разграничением прав доступа.

На рисунке 1 представлена диаграмма прецедентов.

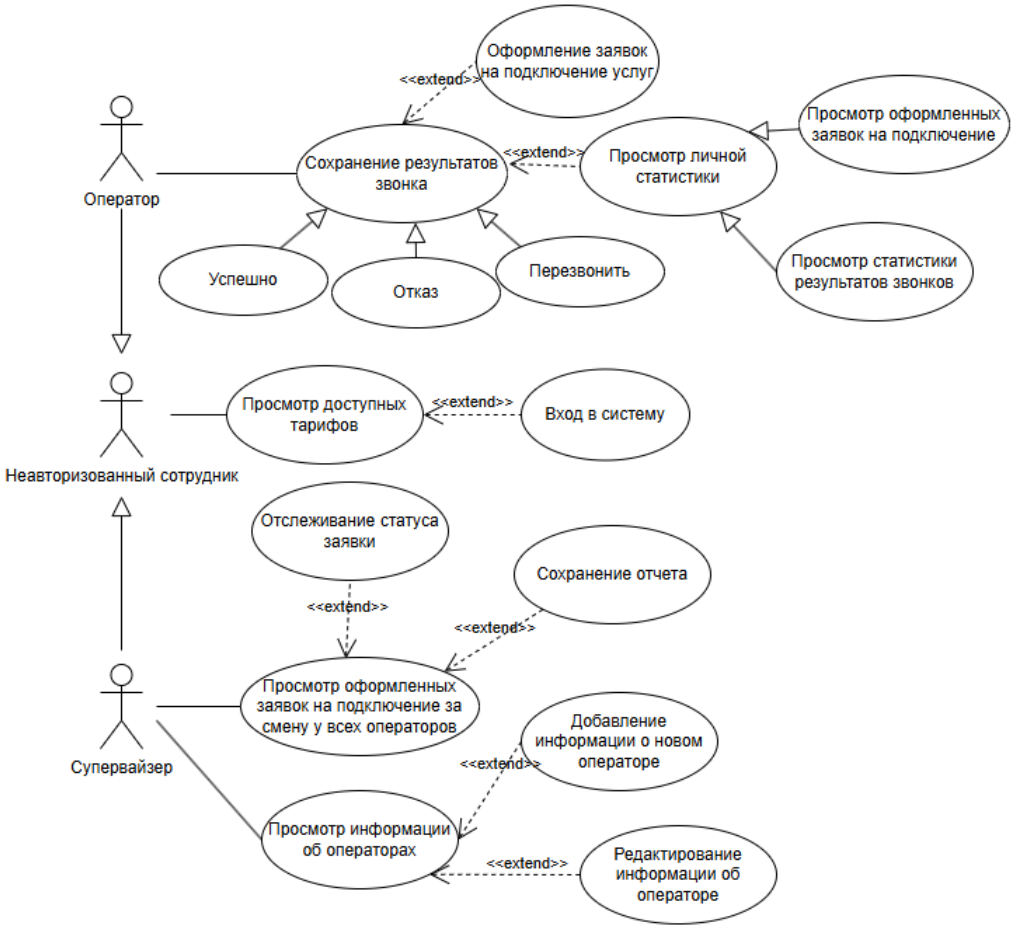

Рисунок 1 – Диаграмма прецедентов

При запуске оконного приложения неавторизованным пользователем отображается окно с тарифами и услугами, а также доступна возможность авторизоваться в системе.

После авторизации супервайзеру предоставляется возможность просмотра, добавления и редактирования информации об операторах отдела. Супервайзер может просматривать список заявок на подключение услуг домашнего интернета и телевидения, оформленных операторами, а также отслеживать статус заявок. На основе оформленных заявок предоставлена возможность сформировать и сохранить отчет за все время в формате csv.

Для оператора доступно оформление заявок на подключение услуг домашнего интернета и телевидения, также есть возможность добавления, редактирования, фильтрации и сортировки информации о клиентах по ФИО и номеру телефона, внесение и сохранение статуса звонка клиенту. В личной статистике оператора отображается информация о количестве совершенных им звонков, а также оформленных заявок на подключение услуг.

1.3 Выбор состава программных и технических средств

Согласно цели проекта требуется разработать оконное приложение. Работа с приложением будет доступна на ПК и ноутбуках с ОС Windows.

В качестве СУБД выбрана MySQL, так как она предлагает ряд значительных преимуществ для разработки и обращения с данными, например, высокая производительность, обеспечивающая быструю обработку запросов.

Для разработки клиентской части выбраны платформа WPF и язык программирования C#, так как они позволяют создавать современные и удобные пользовательские интерфейсы с различной функциональностью.

Для серверной части будет использован ASP.NET, так как это многофункциональный и гибкий фреймворк для разработки API. ASP.NET обеспечивает высокую производительность и безопасность, а также предлагает встроенные возможности для создания RESTful-сервисов.

Для разработки приложения будет использоваться IDE Visual Studio 2022, так как она имеет обширный каталог расширений, которые позволяют

добавлять новые функции, такие как поддержка фреймворков, инструментов для тестирования, имеет встроенную поддержку Git, что упрощает управление версиями кода.

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

- ОС Ubuntu Server 22.04 LTS или аналогичная;
- сервер БД: MySQL не ниже 8.0;
- процессор с 2 ядрами по 2 ГГц;
- оперативная память объемом 2 ГБ;
- ПО для работы API: .NET SDK версией не ниже 8.0;
- не менее 30 ГБ свободного дискового пространства.

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

- ОС Windows 10 и выше (64-бит);
- .NET не ниже 8.0 версии;
- процессор с частотой 2 ГГц;
- оперативная память: минимум 4 ГБ;
- постоянное подключение по локальной сети.

2 Проектирование программного обеспечения

2.1 Проектирование интерфейса пользователя

В рамках разработки оконного приложения спроектирован интерфейс пользователя в виде wireframe. Эти визуальные представления наглядно показывают структуру приложения и его основные элементы [4].

На рисунке 2 в виде wireframe представлены следующие страницы ПО: список оформленных заявок, список клиентов и личный кабинет оператора.

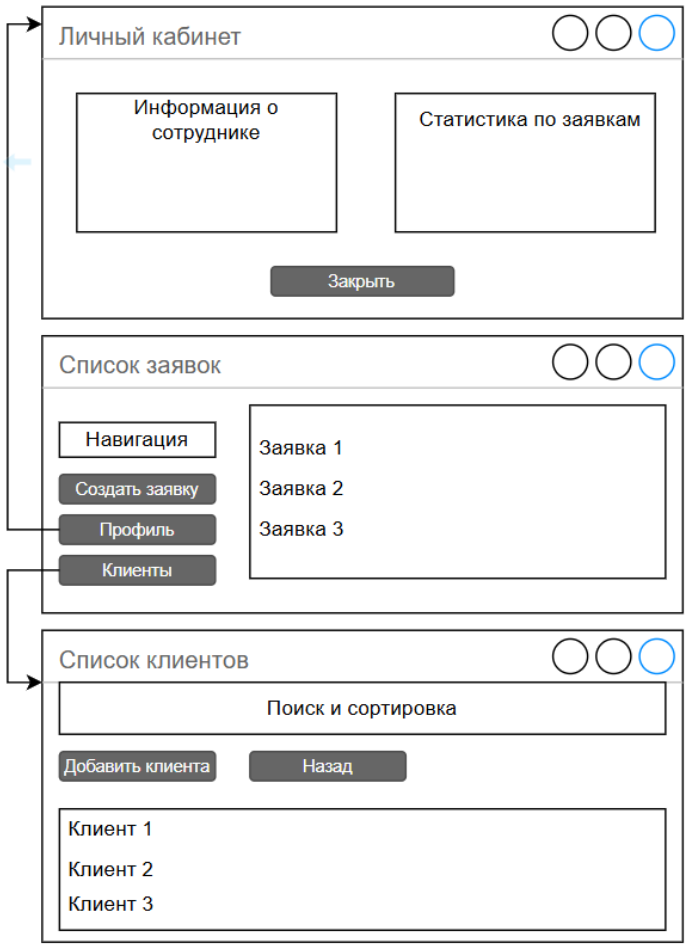

Рисунок 2 – Wireframe основных страниц

Для интерфейса приложения используется следующая цветовая палитра:

- основной цвет текста #333333;

- вторичный цвет текста #e4002b;
- цвет фона #ffffff.

Для текста используется семейство шрифтов Segoe.

2.2 Разработка архитектуры программного обеспечения

Подсистема предназначена для работников отдела телемаркетинга. Она построена на клиент-серверной модели и включает три основных элемента: клиент, сервер и БД. Клиентская часть, реализованная на C# с помощью WPF, взаимодействует с API. Серверная часть, основанная на ASP.NET Web API, обрабатывает входящие запросы и взаимодействует с БД MySQL для хранения информации. Диаграмма развертывания представлена на рисунке 3.

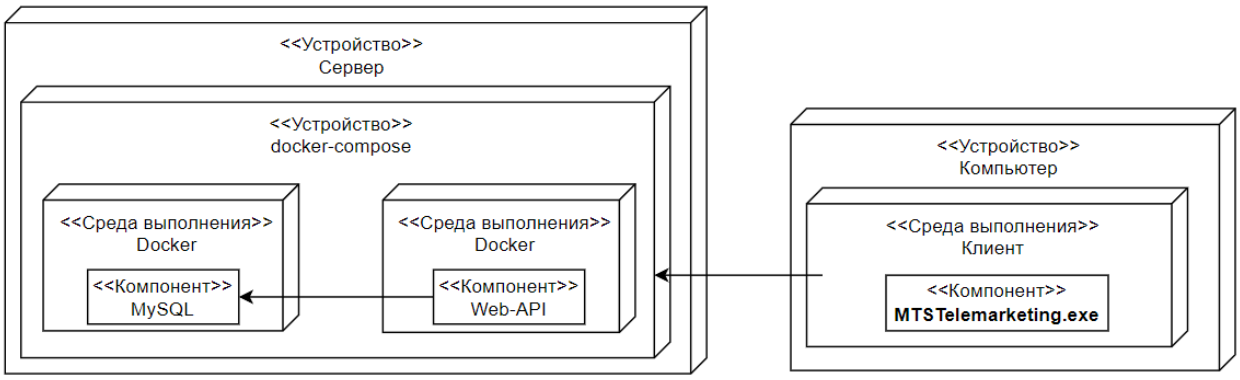

Рисунок 3 – Диаграмма развертывания

2.3 Проектирование БД

Требуется разработать БД для отдела телемаркетинга. Модели БД созданы при помощи draw.io. На рисунке 4 в виде ERD показана физическая модель предметной области [3].

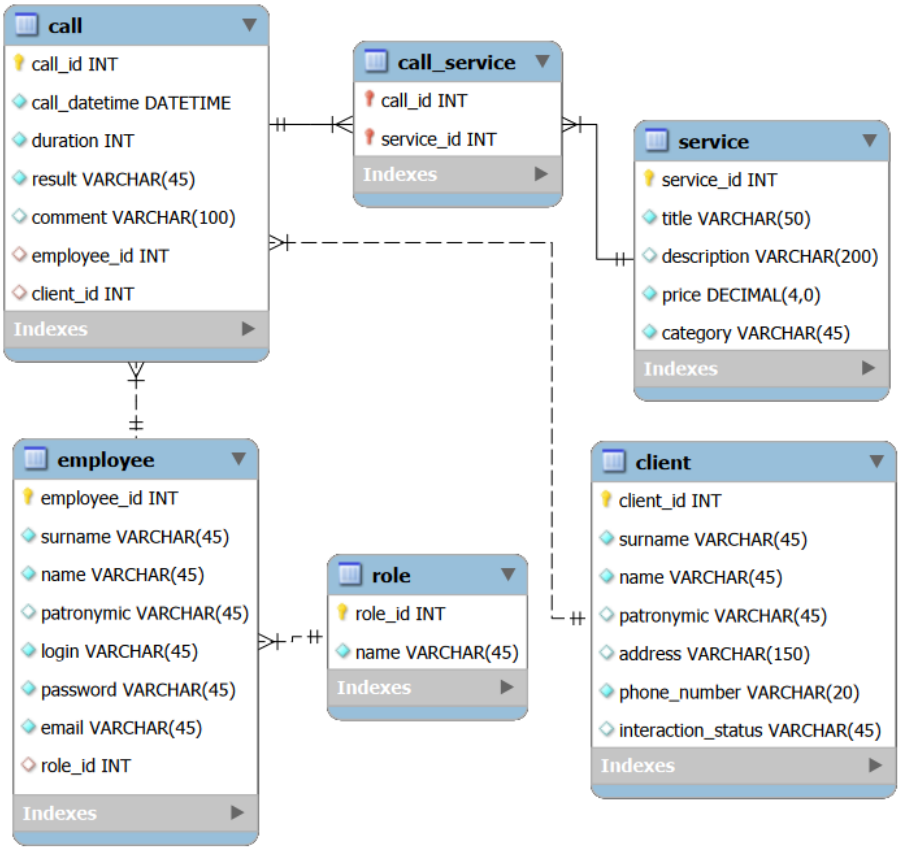

Рисунок 4 – Физическая модель БД

3 Разработка и интеграция модулей программного обеспечения

3.1 Разработка программных модулей

Серверная часть реализована в виде Web-API на языке C# с использованием ASP.NET Core. В качестве ORM используется Entity Framework Core, который обеспечивает взаимодействие с БД и автоматическое соответствие сущностей таблицам БД.

Для получения данных об оформленных заявках у оператора создан метод GetCallsByEmployee в контроллере CallsController. Код метода представлен листингом 1.

Листинг 1 – Код метода GetCallsByEmployee

```
// Метод получения данных об офоромленных заявках оператором
[HttpGet("employee/{employeeId}")]
public async Task<ActionResult<IEnumerable<CallDTO>>>
GetCallsByEmployee(int employeeId)
{
    var calls = await _context.Calls
        .Include(c => c.Client)
        .Include(c => c.Services)
        .Where(c => c.EmployeeId == employeeId)
        .OrderByDescending(c => c.CallDatetime)
        .ToListAsync();

    var callDtos = calls.Select(c => new CallDTO
    {
        CallId = c.CallId,
        CallDatetime = c.CallDatetime,
        Duration = c.Duration,
        Result = c.Result,
        Comment = c.Comment,
        EmployeeId = c.EmployeeId,
        ClientId = c.ClientId,
        Client = c.Client != null ? new ClientDTO
        {
            ClientId = c.Client.ClientId,
            Surname = c.Client.Surname,
            Name = c.Client.Name,
```

```
            Patronymic = c.Client.Patronymic
        } : null,
        Services = c.Services.Select(s => new ServiceDTO
        {
            ServiceId = s.ServiceId,
            Title = s.Title,
            Description = s.Description,
            Price = s.Price,
            Category = s.Category
        }).ToList()
    }).ToList();

    return Ok(callDtos);
}
```

3.2 Реализация интерфейса пользователя

Для реализации интерфейса приложения используется платформа WPF с языком программирования C#. WPF предоставляет инструменты для создания интуитивно понятных пользовательских интерфейсов, а также позволяет эффективно разделять логику приложения и визуальные компоненты.

Для отображения списка клиентов создан элемент управления DataGrid, который предназначен для записи и представления информации с определенными параметрами. Разметка элемента управления представлена листингом 2.

Листинг 2 – Разметка элемента отображения списка клиентов

```
<!-- Разметка элемента отображения списка клиентов -->
<Border Grid.Row="3"
        Style="{StaticResource CardStyle}"
        Margin="20,0,20,10"
        Padding="0">
    <DataGrid x:Name="ClientsDataGrid"
              Style="{StaticResource MtsDataGrid}"
              ColumnHeaderStyle="{StaticResource
MtsDataGridHeader}"
              CellStyle="{StaticResource MtsDataGridCell}"
              Sorting="ClientsDataGrid_Sorting">
```

```
        <DataGrid.Resources>
            <Style TargetType="DataGridRow">
                <Setter Property="BorderBrush"
Value="{StaticResource MtsGray}"/>
                <Setter Property="BorderThickness"
Value="0,0,0,1"/>
                <Setter Property="MinHeight" Value="40"/>
                 <!-- Триггеры для изменения внешнего вида строк -->
                <Style.Triggers>
                    <Trigger Property="IsMouseOver"
Value="True">
                        <Setter Property="Background"
Value="{StaticResource MtsLightGray}"/>
                    </Trigger>
                    <Trigger Property="IsSelected" Value="True">
                        <Setter Property="Background"
Value="{StaticResource MtsLightRed}"/>
                        <Setter Property="Foreground"
Value="{StaticResource MtsWhite}"/>
                    </Trigger>
                </Style.Triggers>
            </Style>
        </DataGrid.Resources>
<!--столбцы для отображения данных -->
        <DataGrid.Columns>
            <DataGridTextColumn Header="ID"
                                Binding="{Binding ClientId}"
                                Width="70"
                                CanUserSort="True"/>
            <DataGridTextColumn Header="Фамилия"
                                Binding="{Binding Surname}"
                                Width="150"
                                CanUserSort="True"/>
            <DataGridTextColumn Header="Имя"
                                Binding="{Binding Name}"
                                Width="150"
                                CanUserSort="True"/>
            <DataGridTextColumn Header="Отчество"
                                Binding="{Binding Patronymic}"
                                Width="150"/>
            <DataGridTextColumn Header="Телефон"
                                Binding="{Binding PhoneNumber}"
                                Width="180"/>
            <DataGridTextColumn Header="Адрес"
                                Binding="{Binding Address}"
                                Width="*"/>
            <DataGridTextColumn Header="Статус"
                                Binding="{Binding
InteractionStatus}"
                                Width="120"
                                CanUserSort="True">
               <!-- Стиль для текстового блока внутри ячейки -->
                <DataGridTextColumn.ElementStyle>
```

```
                    <Style TargetType="TextBlock">
                        <Setter Property="HorizontalAlignment"
Value="Center"/>
                        <Setter Property="VerticalAlignment"
Value="Center"/>
                        <Setter Property="Padding" Value="5"/>
                        <Setter Property="FontWeight"
Value="SemiBold"/>
                        <Setter Property="Foreground"
Value="{StaticResource MtsWhite}"/>
                        <Style.Triggers>
                            <DataTrigger Binding="{Binding
InteractionStatus}" Value="Новый">
                                <Setter Property="Background"
Value="{StaticResource MtsGreen}"/>
                            </DataTrigger>
                            <DataTrigger Binding="{Binding
InteractionStatus}" Value="Потенциальный">
                                <Setter Property="Background"
Value="{StaticResource MtsBlue}"/>
                            </DataTrigger>
                            <DataTrigger Binding="{Binding
InteractionStatus}" Value="Активный">
                                <Setter Property="Background"
Value="{StaticResource MtsPurple}"/>
                            </DataTrigger>
                            <DataTrigger Binding="{Binding
InteractionStatus}" Value="Неактивный">
                                <Setter Property="Background"
Value="{StaticResource MtsOrange}"/>
                            </DataTrigger>
                        </Style.Triggers>
                    </Style>
                </DataGridTextColumn.ElementStyle>
                <DataGridTextColumn.CellStyle>
                    <Style TargetType="DataGridCell"
BasedOn="{StaticResource MtsDataGridCell}">
                        <Setter Property="HorizontalAlignment"
Value="Stretch"/>
                        <Setter Property="VerticalAlignment"
Value="Stretch"/>
                        <Setter Property="Padding" Value="0"/>
                        <Setter Property="BorderThickness"
Value="0"/>
                    </Style>
                </DataGridTextColumn.CellStyle>
            </DataGridTextColumn>
        </DataGrid.Columns>
    </DataGrid>
</Border>
```

3.3 Разграничение прав доступа пользователей

Для разграничения прав доступа создана таблица Employee, в которой хранится информация о сотрудниках, включая их логин и пароль. Код модели сотрудника представлен листингом 3.

Листинг 3 – Код модели сотрудника

```
//класс модели Employee
public partial class Employee
{
    public int EmployeeId { get; set; }

    public string Surname { get; set; } = null!;

    public string Name { get; set; } = null!;

    public string? Patronymic { get; set; }

    public string Login { get; set; } = null!;

    public string Password { get; set; } = null!;

    public string? Email { get; set; }

    public ulong IsActive { get; set; }

    public int? RoleId { get; set; }

    public virtual ICollection<Call> Calls { get; set; } = new
List<Call>();

    public virtual Role? Role { get; set; }
}
```

Для авторизации пользователя в контроллере AuthController создан метод Login. Код метода авторизации представлен листингом 4.

Листинг 4 – Код метода Login

```
//Метод авторизации пользователя
[HttpPost("login")]
```

```
public async Task<ActionResult<Employee>> Login([FromBody]
LoginRequest request)
{
    if (request == null || string.IsNullOrEmpty(request.Login)
|| string.IsNullOrEmpty(request.Password))
    {
        return BadRequest(new { message = "Логин и пароль
обязательны" });
    }

    var employee = await _context.Employees
        .Include(e => e.Role)
        .FirstOrDefaultAsync(e => e.Login == request.Login &&
e.Password == request.Password);

    if (employee == null)
    {
        return Unauthorized(new { message = "Неверный логин или
пароль" });
    }

    var response = new
    {
        employee.EmployeeId,
        employee.Surname,
        employee.Name,
        employee.Patronymic,
        employee.Login,
        employee.Email,
        employee.RoleId
    };

    return Ok(response);
}
```

В окне отображения списка заявок для пользователя с ролью «Супервайзер» доступны кнопки «Сотрудники» и «Экспорт». Код реализации этой логики представлен листингом 5.

Листинг 5 – Код отображения кнопок «Сотрудники» и «Экспорт»

```
//Если пользователь супервайзер, то отображаем кнопки
if (_isSupervisor)
{
    RoleText.Text = "Супервайзер";
    ExportButton.Visibility = Visibility.Visible;
    EmployeesButton.Visibility = Visibility.Visible;
}
```

```
else
{
    RoleText.Text = "Оператор";
    ModeText.Text = "📞 Мои заявки";
}
```

3.4 Экспорт и импорт данных

В приложении реализован экспорт информации по оформленным заявкам в формате csv. Код для формирования csv файла представлен в листинге 6.

Листинг 6 – Код метода ExportToCsv

```
//Код создания отчета в формате csv
private async Task ExportToCsv(string filePath)
{
    try
    {
        // Получаем все заявки для экспорта
        var allCalls = await GetAllCallsForExport();

        if (allCalls == null || !allCalls.Any())
        {
            MessageBox.Show("Нет данных для экспорта",
"Информация",
                MessageBoxButton.OK,
MessageBoxImage.Information);
            StatusText.Text = "Нет данных для экспорта";
            return;
        }

        // Создаем CSV файл
        using (var writer = new StreamWriter(filePath, false,
System.Text.Encoding.UTF8))
        {
            // Заголовки
            writer.WriteLine("Дата и время;Длительность
(мин);Результат;Комментарий;Оператор;Клиент;Услуги");

            // Данные
            foreach (var call in allCalls)
            {
                string comment = call.Comment?.Replace("\"",
"\"\"") ?? "";
```

```
                var line = $"{call.CallDatetime:dd.MM.yyyy
HH:mm};" +
                           $"{call.Duration};" +
                           $"{call.Result};" +
                           $"\"{comment}\";" +
                           $"{call.OperatorName};" +
                           $"{call.ClientName};" +
                           $"{call.ServicesList}";

                writer.WriteLine(line);
            }
        }

        StatusText.Text = $"Экспорт завершен: {allCalls.Count}
записей";
        MessageBox.Show($"Данные успешно экспортированы в
файл:\n{filePath}\n\nВсего записей: {allCalls.Count}",
            "Успех", MessageBoxButton.OK,
MessageBoxImage.Information);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка при экспорте: {ex.Message}",
"Ошибка",
            MessageBoxButton.OK, MessageBoxImage.Error);
        StatusText.Text = "Ошибка экспорта";
    }
}
```

4 Тестирование и отладка ПО

4.1 Структурное тестирование

Во время выполнения курсового проекта проведено структурное тестирование контроллера ClientsController. Код unit-теста для проверки обработки пустого значения поля фамилии в запросе создания клиента представлен листингом 7.

Результат тестирования приведен на рисунке 5.

Листинг 7 – Код unit-теста для метода CreateClient

```
//Unit-тест для проверки пустого значения поля фамилии в запросе
[Fact]
public async Task
CreateClient_ReturnsBadRequest_WhenSurnameIsEmpty()
{
    // Arrange
    var request = new CreateClientRequest
    {
        Surname = "",
        Name = "abc",
        PhoneNumber = "1234567890"
    };

    // Act
    var result = await _controller.CreateClient(request);

    // Assert
    var badRequestResult =
Assert.IsType<ActionResult<Client>>(result);
    var response = badRequestResult.Result as
BadRequestObjectResult;

    Assert.NotNull(response);
    var message =
response?.Value?.GetType().GetProperty("message")?.GetValue(resp
onse?.Value, null);

    Assert.Equal("Фамилия, имя и телефон обязательны", message);
}
```

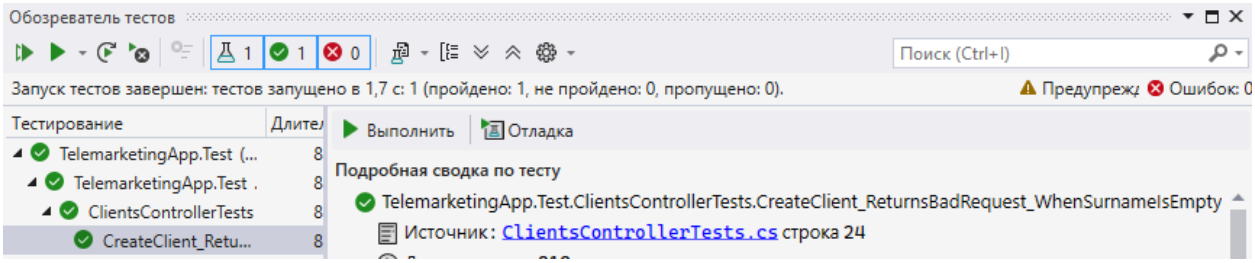

Рисунок 5 – Visual Studio 2022. Вид вкладки с результатами тестирования

4.2 Функциональное тестирование

Во время курсового проектирования проведено функциональное тестирование методом «черного ящика», результаты тестирования приведены в таблице 1.

Таблица 1 – Набор тестов

Действие	Ожидаемый результат	Фактический результат
Открыть окно со списком заявок, нажать на кнопку «Создать заявку» и в открывшемся окне ввести соответствующие данные: адрес подключения: « г. Москва, ул. Ленина, д. 15, кв. 42 », услуга: «Домашний интернет 100 Мбит/с», оборудование: «Роутер Wi-Fi 5», ФИО клиента: «Смирнов Иван Александрович», телефон клиента: «+79161234567», email: «smirnov@email.ru», дата подключения: 15.12.2025, продолжительность: 150, результат: «успешно»	Отображается окно с сообщением «Заявка успешно создана!»	Совпадает с ожидаемым

Продолжение таблицы 1

Действие	Ожидаемый результат	Фактический результат
Открыть окно со списком заявок, нажать на кнопку «Создать заявку» и в открывшемся окне ввести соответствующие данные: адрес подключения: « г. Москва, ул. Ленина, д. 15, кв. 42 », оборудование: «Роутер Wi-Fi 5», ФИО клиента: «Смирнов Иван Александрович», телефон клиента: «+79161234567», email: «smirnov@email.ru», дата подключения: 15.12.2025, продолжительность: 150, результат: «успешно»	Отображается окно с сообщением «Ошибка. Выберите услугу»	Совпадает с ожидаемым
Авторизоваться под логином и паролем супервайзера. Открыть окно со списком заявок, нажать на кнопку «Экспорт»	Открывается окно с выбором пути для сохранения	Совпадает с ожидаемым
Авторизоваться под логином и паролем супервайзера. Открыть окно со списком заявок, нажать на кнопку «Сотрудники». Выбрать сотрудника «Иванов Алексей Иванович» и нажать на кнопку с пиктограммой карандаша. В открывшемся окне изменить следующие данные: логин: «ivanov2», новый пароль: «pass123»	Отображается окно с сообщением «Данные сотрудника обновлены»	Совпадает с ожидаемым
Авторизоваться под логином и паролем супервайзера. Открыть окно со списком заявок, нажать на кнопку «Сотрудники». Выбрать сотрудника «Новиков Андрей» и нажать на кнопку с пиктограммой двери. В открывшемся окне подтвердить увольнение оператора	Отображается окно с сообщением «Сотрудник успешно уволен»	Совпадает с ожидаемым

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

5 Инструкция по эксплуатации ПО

5.1 Установка программного обеспечения

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

- ОС Ubuntu Server 22.04 LTS или аналогичная;
- сервер БД: MySQL версией не ниже 8.0;
- процессор с 2 ядрами по 2 ГГц;
- оперативная память объемом 2 ГБ;
- ПО для работы API: .NET SDK версией не ниже 8.0;
- не менее 30 ГБ свободного дискового пространства.

Для развертывания серверной части необходимо:

- установить .NET 8 SDK и MySQL;
- склонировать репозиторий с исходным кодом API;
- настроить строку подключения к базе данных в файле appsettings.json;
- выполнить миграции базы данных;
- в терминале использовать команду, представленную листингом 8.

Листинг 8 – Код запуска API средствами docker-compose

```
#Код запуска API средствами docker-compose
docker-compose up -d –build
```

5.2 Инструкция по работе

При запуске приложения отображаются тарифы с информацией о наполнении каждого. Интерфейс окна с тарифами представлен на рисунке 6.

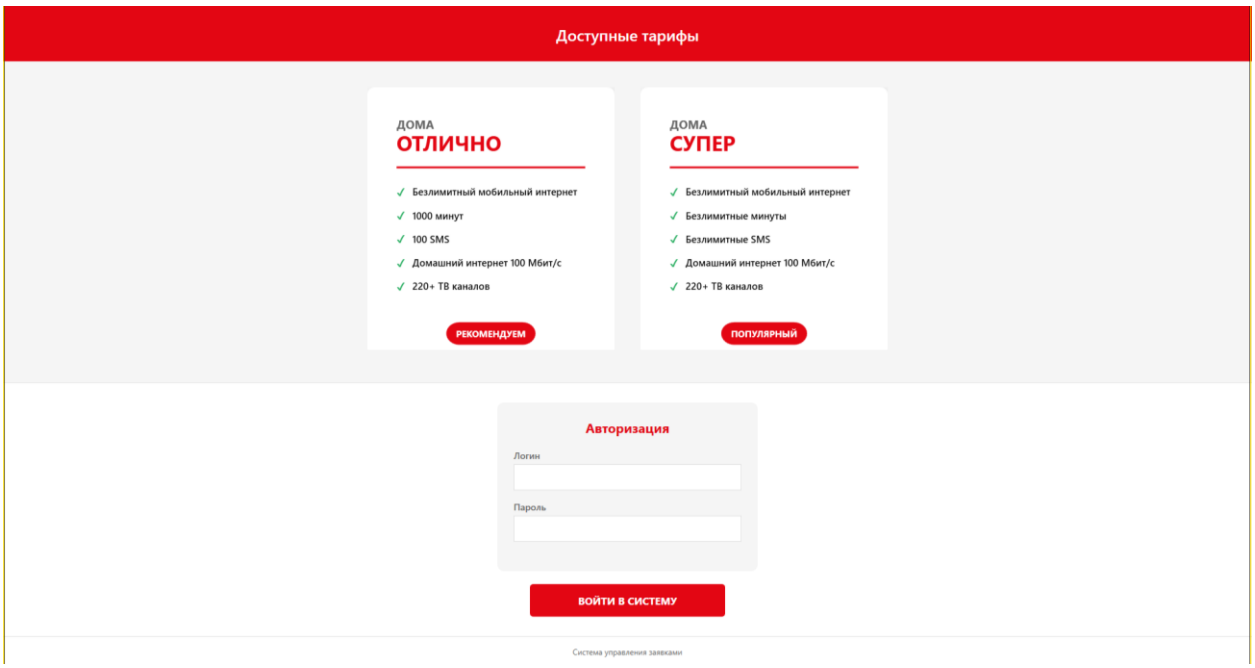

Рисунок 6 – TelemarketingApp. Вид стартового окна

После успешной авторизации осуществляется переход к окну со списком заявок оператора. Пользователю с ролью «Супервайзер» доступен просмотр заявок всех операторов, а также кнопка «Экспорт» для генерации отчета за смену в формате csv. Вид окна со списком заявок операторов представлен на рисунке 7.

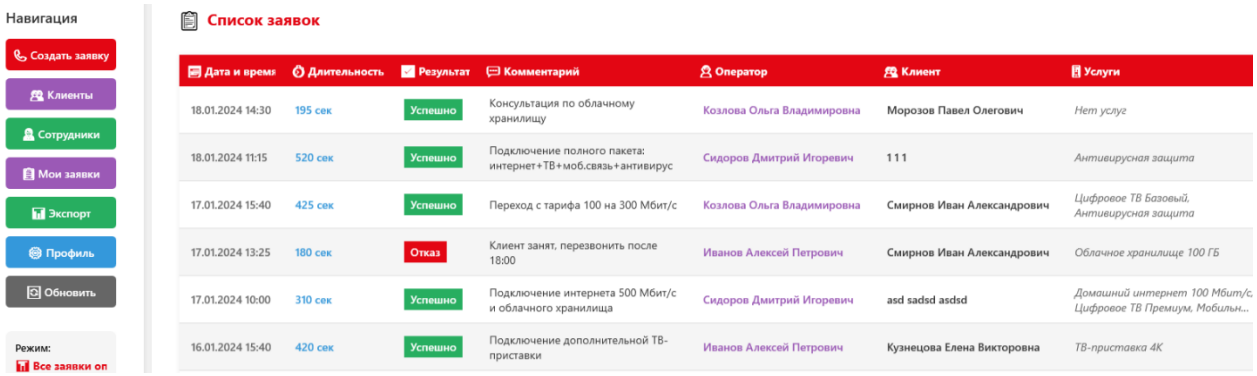

Рисунок 7 – TelemarketingApp. Вид окна списка заявок

Для создания заявки в левой части окна необходимо нажать кнопку «Создать заявку», ввести требуемые данные и нажать кнопку «Создать». Вид окна создания заявки представлен на рисунке 8.

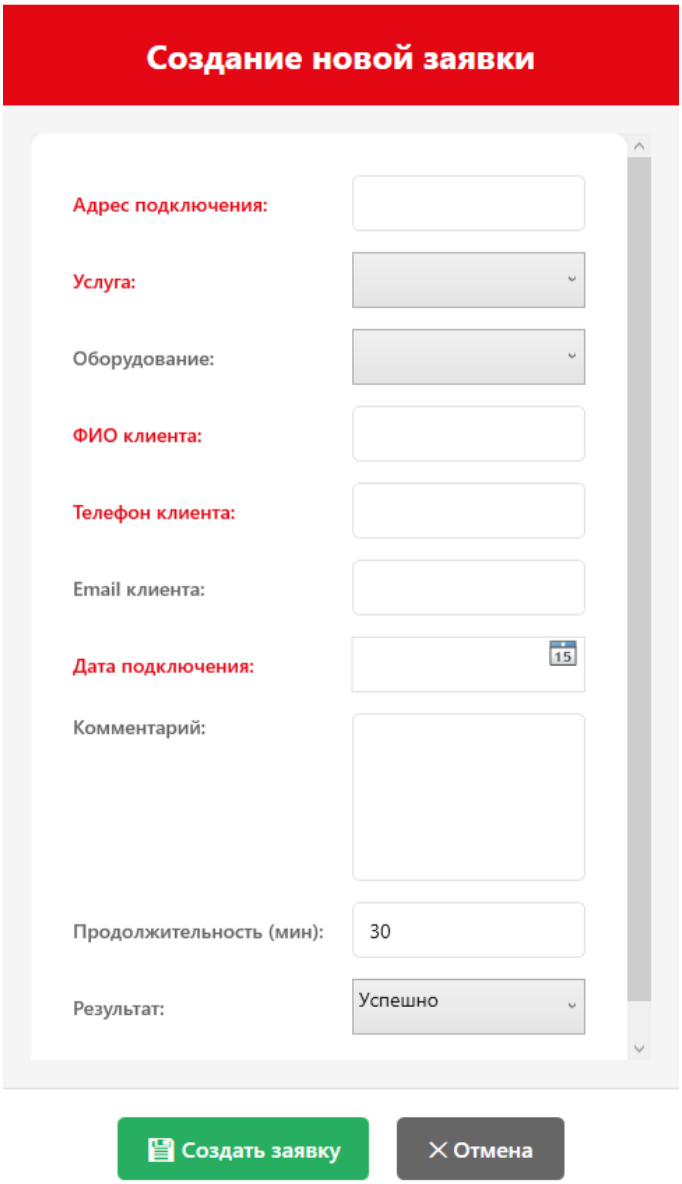

Рисунок 8 – TelemarketingApp. Вид окна создания заявки

Для просмотра статистики оператора в окне списка заявок необходимо нажать на кнопку профиль. Вид окна профиля оператора представлен на рисунке 9.

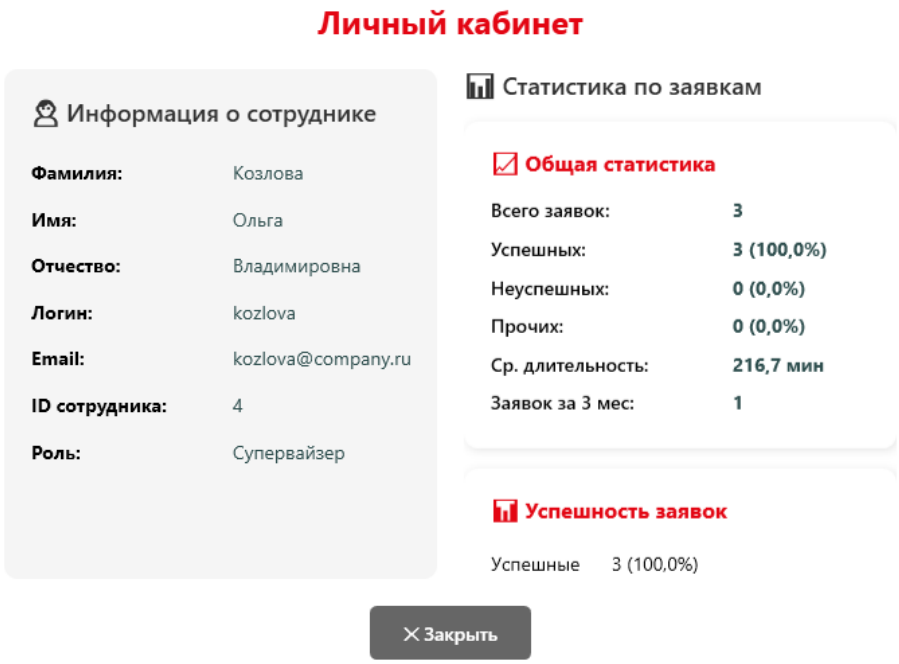

Рисунок 9 – TelemarketingApp. Вид окна профиля оператора

ЗАКЛЮЧЕНИЕ

Целью курсового проектирования являлась разработка подсистемы для автоматизации процессов оформления заявок на подключение к услугам домашнего интернета и телевидения.

В ходе выполнения курсового проекта поставленная цель достигнута, разработано оконное приложение для управления заявками на подключение к услугам домашнего интернета и телевидения.

В процессе достижения цели решены следующие задачи:

- проведен анализ предметной области;
- выполнен сбор требований к функциональности подсистемы;
- проанализированы информационные источники по предметной области;
- спроектирована диаграмма вариантов использования;
- спроектирована архитектура подсистемы;
- выбран состав программных и технических средств для реализации приложения;
- спроектирована БД;
- спроектирован интерфейс оконного приложения;
- создана БД в выбранной СУБД;
- разработана API для взаимодействия оконного приложения с БД;
- разработано оконное приложение;
- реализован просмотр данных об оформленных заявках оператором;
- реализована функциональность оформления заявок;
- реализована фильтрация и сортировка данных о клиентах;
- реализован экспорт данных в формате csv;
- внедрено разграничение прав доступа пользователей;
- выполнено структурное и функциональное тестирование ПО;
- разработаны программная и эксплуатационная документация.

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: 05.11.2024). – 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: 11.11.2024). – 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, 2023. — 368 с. - URL: https://znanium.com/catalog/product/1912454 (дата обращения: 13.11.2024). – Режим доступа: по подписке. — Текст : электронный.

4. Тидвелл, Д. Разработка интерфейсов. Паттерны проектирования. 3-е изд. – Санкт-Петербург : Питер, 2022. – 560 с. – Текст : электронный. – URL: https://ibooks.ru/bookshelf/386796/reading (дата обращения: 17.11.2025). – Режим доступа: для зарегистрир. пользователей.

5. Федорова, Г. Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности : учебное пособие. — Москва : КУРС : ИНФРА-М, 2024. — 336 с. – URL: https://znanium.ru/catalog/product/2083407 (дата обращения: 20.11.2025). – Режим доступа: по подписке. – Текст : электронный.