**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КОМИ**

**Государственное профессиональное образовательное учреждение**

**«Воркутинский арктический горно-политехнический колледж»**

**КУРСОВОЙ ПРОЕКТ**

По дисциплине МДК.05.02 Разработка кода информационных систем

**Регистрация командировочных удостоверений**

Выполнил студент гр. ИСП-22 /\_\_\_\_\_\_\_\_\_\_\_\_\_/Цибко Виктор Васильевич/

(подпись) (Ф.И.О.)

**ОЦЕНКА:\_\_\_\_\_\_\_\_\_\_\_\_\_**

Дата:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПРОВЕРИЛ**

Научный руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_/ Егоров Данил Павлович/

(подпись) (Ф.И.О.)

Воркута

2025

**Оглавление**

[ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ 4](#_Toc196819601)

[1.1 Платформа .NET 4](#_Toc196819602)

[1.2 Язык программирования С# 4](#_Toc196819603)

[1.3 Windows Presentation Foundation (WPF) 5](#_Toc196819604)

[1.4 СУБД SQL server 5](#_Toc196819605)

[1.5 Microsoft SQL Server Management Studio 6](#_Toc196819606)

[1.6 Entity Framework 6](#_Toc196819607)

[ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ 7](#_Toc196819608)

[2.1 Разработка диаграммы ERD 7](#_Toc196819609)

[2.2 Разработка базы данных 8](#_Toc196819610)

[ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 12](#_Toc196819611)

[3.1 Разработка прототипа информационной системы 12](#_Toc196819612)

[3.2 Программирование информационной системы 14](#_Toc196819613)

[3.2.1 Разработка модуля «Авторизация» 14](#_Toc196819614)

[3.2.2 Разработка модуля «Окно главное» 16](#_Toc196819615)

[3.2.3 Разработка модуля «Создать командировку» 19](#_Toc196819616)

[3.2.4 Разработка модуля «Добавление пользователей» 22](#_Toc196819617)

[3.2.5. Разработка модуля «Данные о пользователях» 24](#_Toc196819618)

[ЗАКЛЮЧЕНИЕ 26](#_Toc196819619)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 27](#_Toc196819620)

**ВВЕДЕНИЕ**

Современные организации активно используют командировки для решения рабочих задач, деловых встреч, участия в конференциях и обмена опытом. Однако процесс оформления командировочных удостоверений, учета затрат и контроля за перемещением сотрудников часто связан с бумажным документооборотом, что приводит к потере времени, возможным ошибкам и сложности в управлении данными.

Актуальность данной работы обусловлена необходимостью автоматизации процессов регистрации командировочных удостоверений, что позволит значительно сократить время на оформление документов, повысить точность данных. Разработка информационной системы позволит минимизировать ошибки, возникающие при ручном вводе информации, а также обеспечит удобный доступ к данным для всех заинтересованных лиц.

**Объект:** информационная система «Регистрация командировочных удостоверений».

**Предмет:** анализ бизнес-процессов «Регистрация командировочных удостоверений».

**Цель работы:** разработать информационную систему «Регистрация командировочных удостоверений».

**Задачи:**

• выбрать инструментарий;

• спроектировать базу данных;

• разработать информационную систему.

# ГЛАВА 1. ВЫБОР ИНСТРУМЕНТАРИЯ

## Платформа .NET

Платформа .NET (произносится как «дот нет») представляет собой программную среду, разработанную компанией Microsoft для создания, развертывания и выполнения различных приложений. Она включает в себя виртуальную машину Common Language Runtime (CLR), библиотеку классов Framework Class Library (FCL) и различные инструменты разработки.

Основные особенности .NET:

* Поддержка множества языков программирования (C#, VB.NET, F# и др.).
* Кроссплатформенность (благодаря .NET Core и .NET 5+ приложения могут работать на Windows, Linux и macOS).
* Автоматическое управление памятью (сборка мусора, управление ресурсами).
* Безопасность и надежность благодаря встроенным механизмам аутентификации и авторизации.
* Поддержка современных технологий, таких как облачные вычисления, машинное обучение и интернет вещей (IoT).

## Язык программирования С#

C# (си шарп) – это объектно-ориентированный язык программирования, разработанный компанией Microsoft в начале 2000-х годов. Он является основным языком для разработки приложений на платформе .NET.

Основные особенности C#:

* Простота и удобочитаемость кода.
* Автоматическое управление памятью.
* Поддержка объектно-ориентированного программирования (ООП) и современных языковых конструкций, таких как LINQ, асинхронное программирование (async/await) и паттерны проектирования.
* Высокая производительность и оптимизация за счет компиляции в промежуточный код (IL) с последующей Just-In-Time (JIT) компиляцией.
* Кроссплатформенность благодаря .NET.

## Windows Presentation Foundation (WPF)

Windows Presentation Foundation (WPF) – это графическая подсистема для создания пользовательских интерфейсов Windows-приложений. Она была представлена Microsoft как часть .NET Framework 3.0.

Основные возможности WPF:

* Использование языка разметки XAML для описания интерфейсов.
* Поддержка векторной графики и масштабируемости UI.
* Гибкость и мощные возможности стилизации и темизации.
* Поддержка мультимедиа (аудио, видео, анимация).
* Связывание данных (Data Binding) и шаблоны данных (Data Templates), что упрощает разработку сложных пользовательских интерфейсов.

## СУБД SQL server

Microsoft SQL Server – это реляционная система управления базами данных (СУБД), разработанная Microsoft. Она используется для хранения, обработки и управления данными.

Ключевые характеристики SQL Server:

* Высокая производительность и масштабируемость.
* Поддержка сложных запросов SQL.
* Встроенные механизмы безопасности и управления пользователями.
* Репликация и резервное копирование для обеспечения отказоустойчивости.
* Интеграция с другими продуктами Microsoft, такими как Azure, Power BI и .NET.

## Microsoft SQL Server Management Studio

SQL Server Management Studio (SSMS) – это инструмент для управления Microsoft SQL Server. Он предоставляет графический интерфейс для работы с базами данных.

Функциональность SSMS:

* Создание, редактирование и выполнение SQL-запросов.
* Управление объектами базы данных (таблицы, представления, хранимые процедуры).
* Мониторинг производительности и безопасности базы данных.
* Визуализация схем данных.
* Возможность работы с SQL Server Analysis Services (SSAS) и SQL Server Reporting Services (SSRS).

## Entity Framework

Entity Framework (EF) – это объектно-реляционный сопоставитель (ORM) для .NET, который упрощает работу с базами данных. Он позволяет разработчикам работать с данными в виде объектов, а не таблиц SQL.

Преимущества Entity Framework:

* Автоматическое создание и управление базами данных на основе моделей C#.
* Поддержка различных стратегий доступа к данным (Code First, Database First, Model First).
* Гибкость запросов с использованием LINQ.
* Автоматическая обработка транзакций и изменений данных.
* Улучшенная производительность благодаря кэшированию и отслеживанию изменений.

Entity Framework является стандартным инструментом для работы с базами данных в .NET-приложениях, что делает его мощным решением для разработки сложных систем.

# ГЛАВА 2. ПРОЕТИРОВАНИЕ БАЗЫ ДАННЫХ

## 2.1 Разработка диаграммы ERD

ER-диаграмма – это метод визуализации и описания логической структуры базы данных, основанный на представлении сущностей (entities) вместе с их атрибутами и связями между сущностями. ER-диаграммы являются графическими моделями, которые помогают разработчикам баз данных понять и описать основные компоненты концептуальной модели, а также их отношения.

На ER-диаграммах сущности представляют собой объекты или концепты, которые существуют независимо друг от друга и могут быть чем-то важным для системы базы данных, например: сотрудники, менеджеры, командировки и т.д. Атрибуты определяют характеристики каждой сущности и помогают описать ее свойства, такие как имя, должность, контактная информация и т.д.

Связи между сущностями на ER-диаграммах описывают, как сущности взаимодействуют или связаны друг с другом. Бинарные связи представляют отношения между двумя сущностями, например, «каждый курс имеет одного преподавателя». Множественные связи могут быть использованы для описания отношений «один ко многим» или «многие ко многим».

**Назначение**:

Диаграмма взаимосвязи объектов (ER-диаграмма, или просто ERD) показывает, как взаимодействуют объекты (например, люди, предметы или понятия). Такие концептуальные модели данных помогают разработчикам и проектировщикам баз данных визуализировать взаимосвязи между ключевыми элементами программного обеспечения.

Эта диаграмма является разновидностью блок-схемы. Это логическая модель, демонстрирующая, как данные перемещаются от одного объекта к другому. Благодаря такому наглядному формату разработчики программного обеспечения и проектировщики баз данных могут легко визуализировать структуру системы.

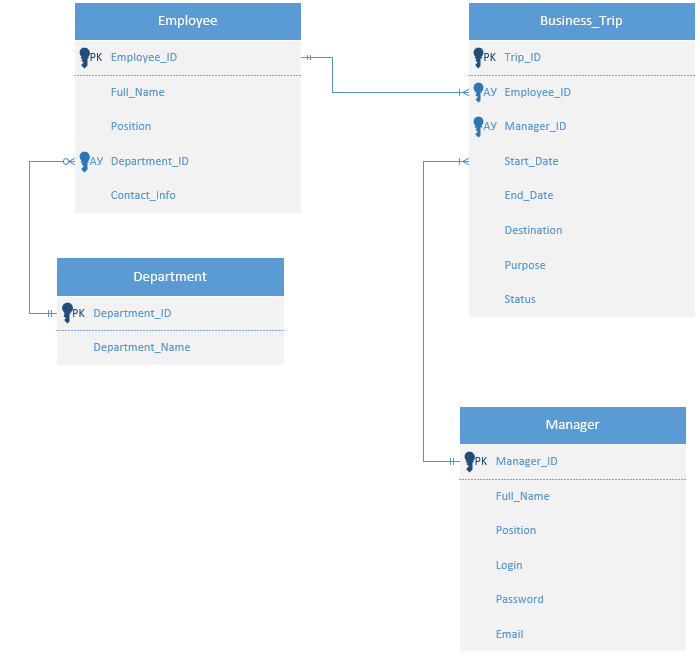


Рис 2.1 ER-диаграмма (ER-diagram)

На рис 2.1 показана ER-диаграмма информационной системы «Регистрация командировочных удостоверений». Она показывает сущности и как они связаны между собой внутри. Сущностями являются: Сотрудник, департамент, командировка, менеджер. Эти сущности обладают своими атрибутами.

## 2.2 Разработка базы данных

Разработка базы данных будет происходить в программе SSMS (SQL Server Management Studio).

SQL Server Management Studio (SSMS) — это интегрированная среда для управления любой инфраструктурой SQL. Используйте SSMS для доступа, настройки, управления, администрирования и разработки всех компонентов SQL Server, Базы данных SQL Azure, Управляемого экземпляра SQL Azure, SQL Server на виртуальной машине Azure и Azure Synapse Analytics. SSMS предоставляет единую комплексную служебную программу, которая сочетает в себе широкую группу графических инструментов с множеством многофункциональных редакторов сценариев для предоставления доступа к SQL Server разработчикам и администраторам баз данных любого уровня квалификации.

Для того чтобы создать в программе SSMS базу данных необходимо в «Object Explorer» нажать правой кнопкой мышки по пункту «Databases» и нажать на пункт «New Database» как показано на рисунке 2.2.

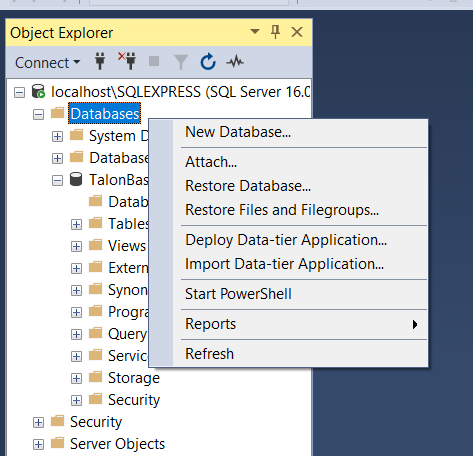


Рис 2.2 Создание БД

Далее нам выводится окно, в котором мы даем название для базы данных и нажимаем кнопку «ОК». В «Object Explorer» появится созданная база даных, развернем ее и нажнем правой кнопкой мыши о папке «Database Diagrams» и нажмем на «New Database Diagram» как показано на рисунке 2.3.

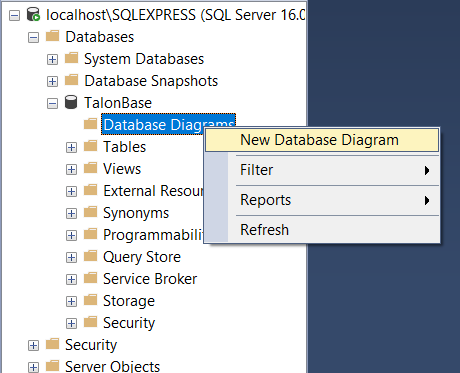


Рис. 2.3 Создание диаграммы

После чего нам выведется поле для создания диаграммы баз данных, на котором мы можем создавать таблицы с названием столбцов и их типов данных.

Для создания таблицы данных необходимо нажать правой кнопкой мыши в любом месте диаграммы, выбираем «New Table…» и даем название для нашей таблицы. Для информационной системы «Регистрация командировочных удостоверений» необходимо создать пять таблиц: Eployee (Сотрудник), Manager (Менеджер), Bussiness\_Trip (Командировка), Department (Департамент). После чего заполним таблицы необходимыми полями и типами данных из ERD.

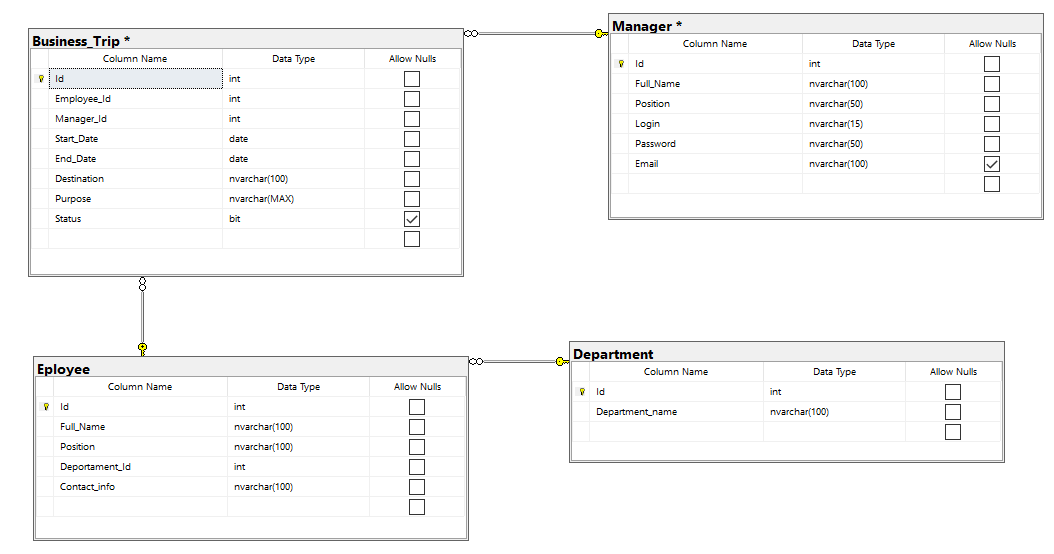


Рис. 2.4 Все таблицы

Далее создаем первичный ключ и удерживая ключ создаем связи как в ERD сущностях. После сохраняем таблицы сочетанием клавиш «Ctrl + S» и в высветившимся окне пишем название диаграммы.

Созданные таблицы можно увидеть в «Object Explorer» кликнув на название базы данных и перейдя в папку «Tables».

# ГЛАВА 3. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 Разработка прототипа информационной системы

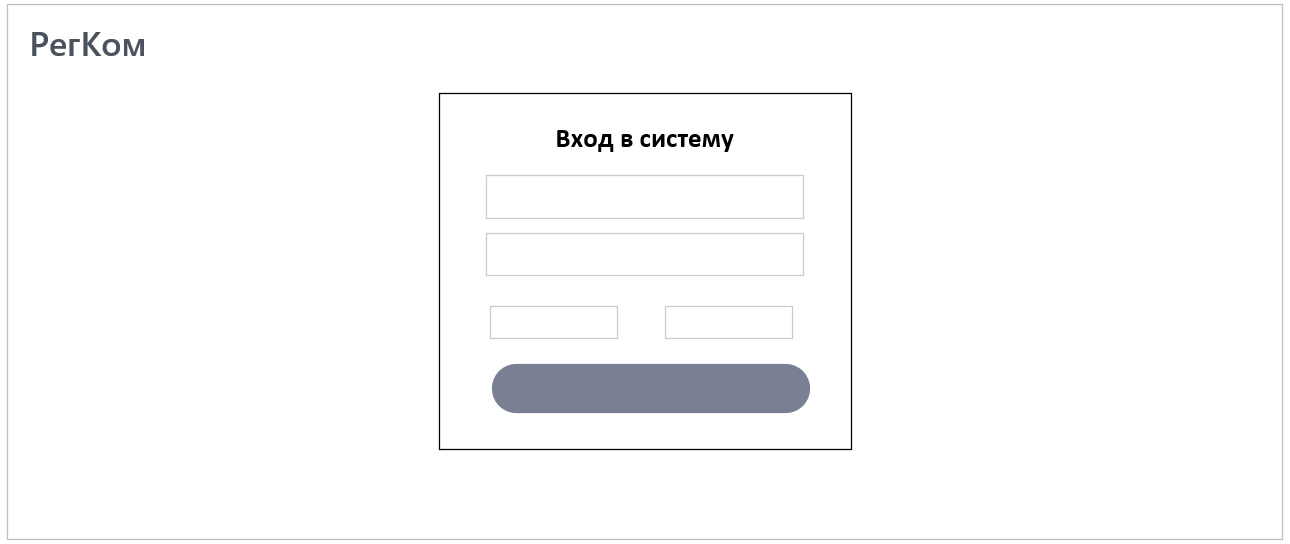


Рис. 3.1 Окно авторизации

На рисунке 3.1 представлено окно «Авторизация», на котором показаны поля ввода пароля, логина, капчи и кнопка авторизации.

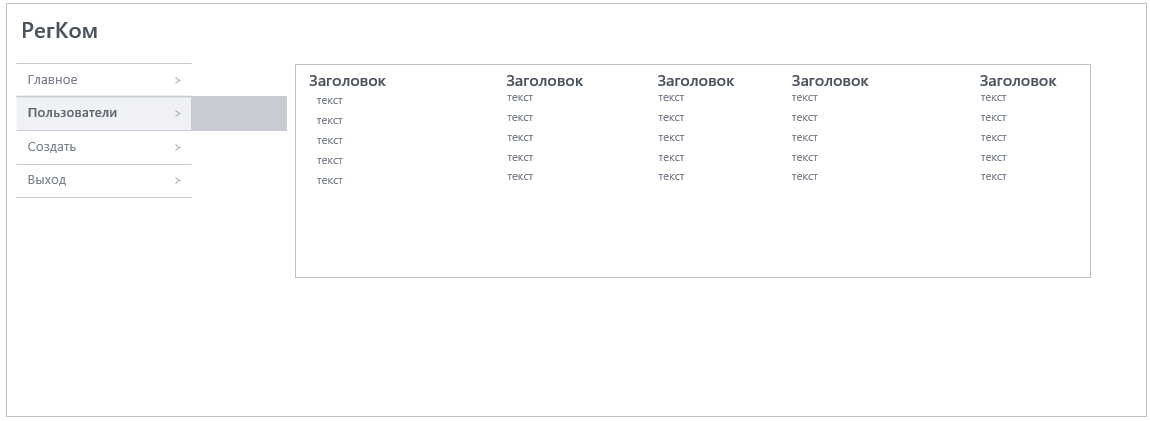


Рис. 3.2 Главное окно

На рисунке представлено окно «Главное меню», на котором показан все командировки, зарегистрированные в базе данных.

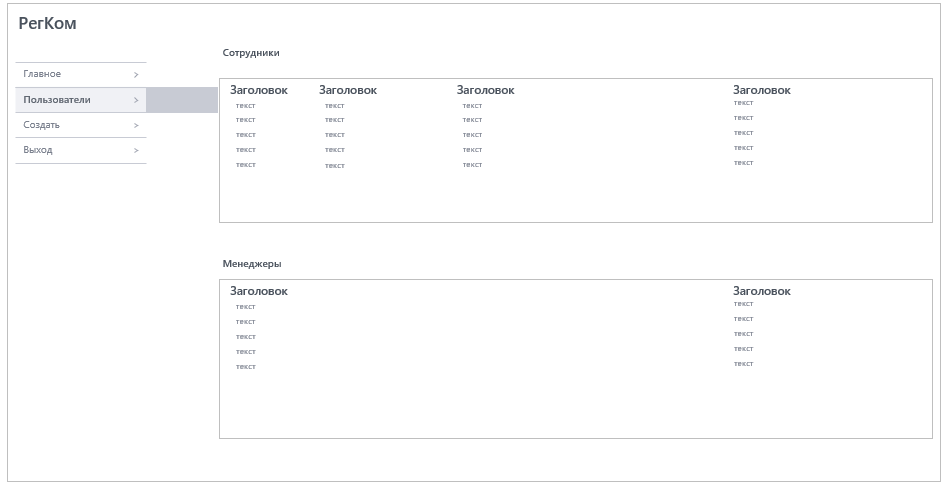


Рис 3.3 Окно пользователи

На рисунке представлено окно «Пользователи», на котором показаны все сотрудники и администраторы.

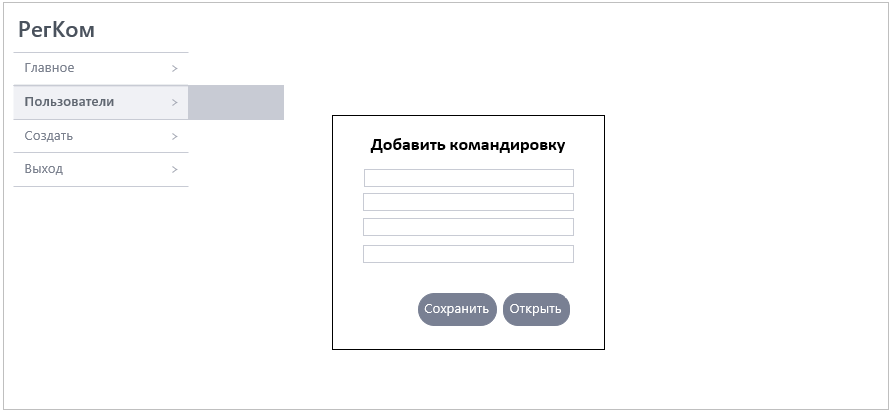
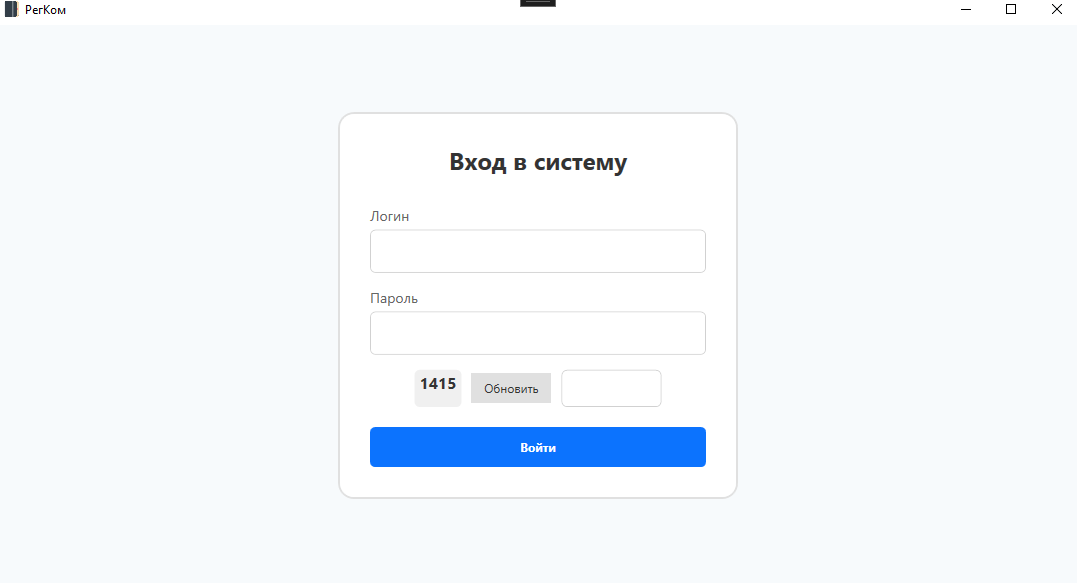


Рис. 3.4 Окно создания командировки

На рисунке представлено окно «создания командировки», на котором показаны поле ввода данных о командировки, а также кнопки сохранить и открыть.

## 3.2 Программирование информационной системы

## 3.2.1 Разработка модуля «Авторизация»



*Рисунок 3.2.1.1 Окно «Авторизация»*

Методы проверки Логина, пароля и капчи

private void GenerateCaptcha()

{

Random random = new Random();

\_captchaCode = random.Next(1000, 9999).ToString();

CaptchaText.Text = \_captchaCode;

}

private void RefreshCaptcha\_Click(object sender, RoutedEventArgs e)

{

GenerateCaptcha();

CaptchaInput.Text = "";

}

private void LoginBox\_GotFocus(object sender, RoutedEventArgs e)

{

TextBox loginBox = sender as TextBox;

if (loginBox != null && loginBox.Text == "Логин")

{

loginBox.Text = "";

loginBox.Foreground = Brushes.Black;

}

}

private void LoginBox\_LostFocus(object sender, RoutedEventArgs e)

{

TextBox loginBox = sender as TextBox;

if (loginBox != null && string.IsNullOrWhiteSpace(loginBox.Text))

{

loginBox.Text = "Логин";

loginBox.Foreground = Brushes.Gray;

}

}

private void Login\_Click(object sender, RoutedEventArgs e)

{

ErrorText.Visibility = Visibility.Collapsed;

string login = LoginBox.Text;

string password = \_isPasswordVisible ? \_passwordTextBox.Text : PasswordBox.Password;

// Валидация полей

if (string.IsNullOrWhiteSpace(login) || login == "Логин")

{

ShowError("Введите логин.");

return;

}

if (string.IsNullOrWhiteSpace(password))

{

ShowError("Введите пароль.");

return;

}

// Проверка капчи

if (CaptchaInput.Text != \_captchaCode)

{

ShowError("Неверный код капчи.");

GenerateCaptcha();

CaptchaInput.Text = "";

return;

}

// Проверка логина и пароля

var user = \_context.Manager.FirstOrDefault(m => m.Login == login && m.Password == password);

if (user == null)

{

ShowError("Неверный логин или пароль.");

GenerateCaptcha();

CaptchaInput.Text = "";

return;

}

\_mainWindow.SetUser(user);

}

private void ShowError(string message)

{

ErrorText.Text = message;

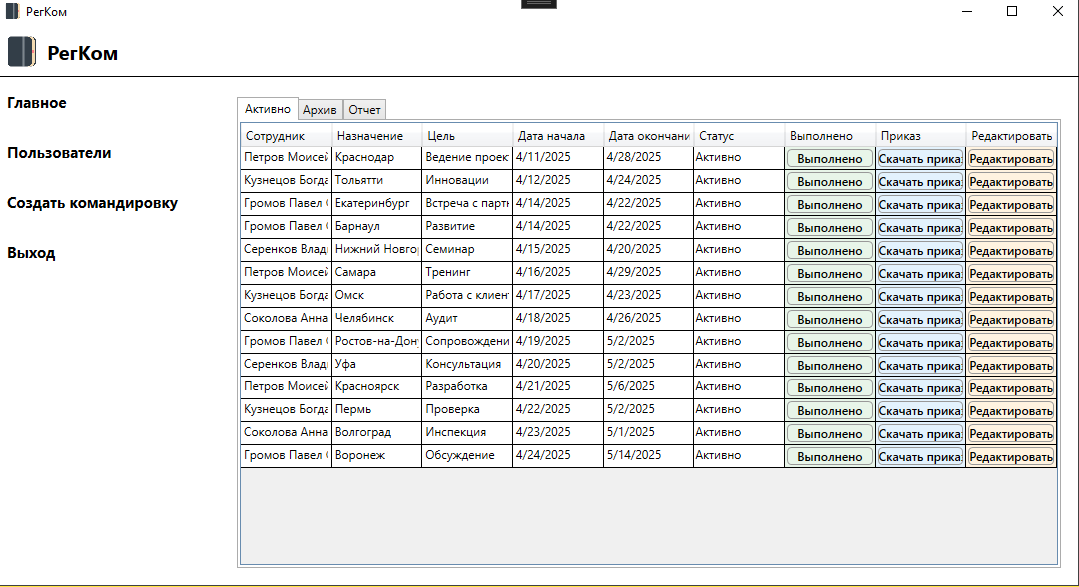
ErrorText.Visibility = Visibility.Visible;

}

}

В данном модуле было разработано окно авторизации и капча с проверкой логина и пароля пользователя.

## 3.2.2 Разработка модуля «Окно главное»



*Рисунок 3.2.2 Окно «Главное меню»*

Методы загрузки командировок с разделением на активные и архив, редактирования и скачивания:

private void LoadTrips()

{

var trips = context.Business\_Trip

.Include("Eployee") // Загрузка связанных данных о сотруднике

.Include("Manager") // Загрузка данных о менеджере

.Include("Eployee.Department") // Загрузка данных об отделе сотрудника

.ToList();

ActiveTripsDataGrid.ItemsSource = trips.Where(t => t.Status != true).ToList(); // Заполнение таблицы активными командировками

ArchivedTripsDataGrid.ItemsSource = trips.Where(t => t.Status == true).ToList();

private void DownloadOrder\_Click(object sender, RoutedEventArgs e)

{

try

{

Button button = sender as Button;

if (button == null || !(button.Tag is int tripId))

{

MessageBox.Show("Не удалось определить командировку.");

return;

}

var trip = context.Business\_Trip

.Include("Eployee")

.Include("Manager")

.Include("Eployee.Department")

.FirstOrDefault(t => t.Id == tripId);

if (trip == null)

{

MessageBox.Show("Командировка не найдена.");

return;

}

string projectRoot = Directory.GetParent(AppDomain.CurrentDomain.BaseDirectory).Parent.Parent.FullName;

string templatePath = Path.Combine(projectRoot, "Приказ.doc");

if (!File.Exists(templatePath))

{

MessageBox.Show($"Шаблон не найден по пути: {templatePath}");

return;

}

string ordersFolder = Path.Combine(projectRoot, "Приказы");

Directory.CreateDirectory(ordersFolder);

string fileName = $"Приказ\_о\_командировке\_{trip.Id}.doc";

string outputPath = Path.Combine(ordersFolder, fileName);

if (File.Exists(outputPath))

{

int counter = 1;

string newFileName;

do

{

newFileName = $"Приказ\_о\_командировке\_{trip.Id}\_{counter}.doc";

outputPath = Path.Combine(ordersFolder, newFileName);

counter++;

} while (File.Exists(outputPath));

fileName = newFileName;

}

string fullNameEmployee = trip.Eployee?.Full\_Name ?? "";

string positionEmployee = trip.Eployee?.Position ?? "";

string departmentName = trip.Eployee?.Department?.Department\_name ?? "—";

string fullNameManager = trip.Manager?.Full\_Name ?? "";

var wordApp = new Word.Application();

var document = wordApp.Documents.Open(templatePath);

void Replace(string placeholder, string value)

{

var range = document.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: placeholder, ReplaceWith: value ?? "", Replace: Word.WdReplace.wdReplaceAll);

}

Replace("{Id документа}", Guid.NewGuid().ToString().Substring(0, 8));

Replace("{Дата на момент создания}", DateTime.Now.ToShortDateString());

Replace("{Full\_Name employee}", fullNameEmployee);

Replace("{Position employee}", positionEmployee);

Replace("{Id departament}", departmentName);

Replace("{Destination}", trip.Destination);

Replace("{Start\_date}", trip.Start\_Date.ToShortDateString());

Replace("{End\_date}", trip.End\_Date.ToShortDateString());

Replace("{purpose}", trip.Purpose);

Replace("{Full\_Name Manager}", fullNameManager);

document.SaveAs2(outputPath);

document.Close();

wordApp.Quit();

MessageBox.Show($"Приказ успешно сгенерирован в папке Приказы: {fileName}");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при генерации приказа: " + ex.Message);

}

}

private void LoadComboBoxes()

{

EmployeeComboBox.ItemsSource = context.Eployee.ToList();

ManagerComboBox.ItemsSource = context.Manager.ToList();

}

private void LoadTrip(int tripId)

{

\_trip = context.Business\_Trip

.Include("Eployee")

.Include("Manager")

.FirstOrDefault(t => t.Id == tripId);

if (\_trip == null)

{

MessageBox.Show("Командировка не найдена.");

Close();

return;

}

EmployeeComboBox.SelectedValue = \_trip.Employee\_Id;

ManagerComboBox.SelectedValue = \_trip.Manager\_Id;

DestinationBox.Text = \_trip.Destination;

PurposeBox.Text = \_trip.Purpose;

StartDatePicker.SelectedDate = \_trip.Start\_Date;

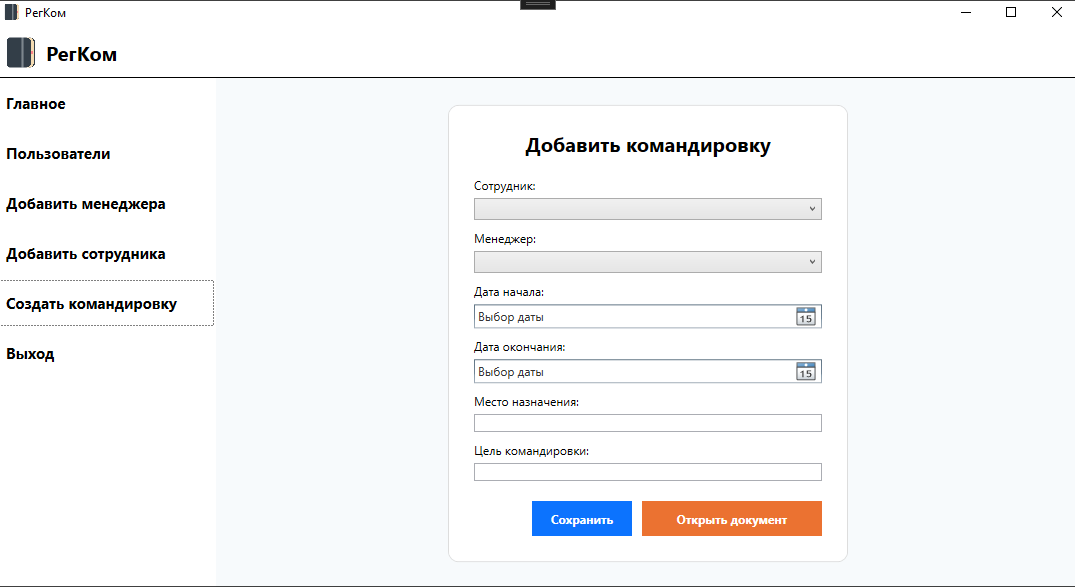
EndDatePicker.SelectedDate = \_trip.End\_Date;

StatusCheckBox.IsChecked = \_trip.Status;

}

В данном модуле было разработано главное окно с данными о командировках, так же было разработано отдельное окно с редактированием командировок. Пользователь может скачать приказ.

## 3.2.3 Разработка модуля «Создать командировку»



*Рисунок 3.2.4 «Создание командировок»*

Методы создания командировки:

private void LoadComboBoxes()

{

using (var db = new TalonBaseEntities())

{

EmployeeComboBox.ItemsSource = db.Eployee.ToList();

ManagerComboBox.ItemsSource = db.Manager.ToList();

}

}

private void Save\_Click(object sender, RoutedEventArgs e)

{

try

{

using (var db = new TalonBaseEntities())

{

Business\_Trip trip = new Business\_Trip

{

Employee\_Id = (int)EmployeeComboBox.SelectedValue,

Manager\_Id = (int)ManagerComboBox.SelectedValue,

Start\_Date = StartDatePicker.SelectedDate ?? DateTime.Now,

End\_Date = EndDatePicker.SelectedDate ?? DateTime.Now,

Destination = DestinationBox.Text,

Purpose = PurposeBox.Text,

Status = false

};

db.Business\_Trip.Add(trip);

db.SaveChanges();

\_lastTripId = trip.Id;

MessageBox.Show("Командировка успешно сохранена.");

}

}

catch (Exception ex)

{

string errorMessage = ex.Message;

if (ex.InnerException != null)

errorMessage += "\n\n" + ex.InnerException.Message;

if (ex.InnerException?.InnerException != null)

errorMessage += "\n\n" + ex.InnerException.InnerException.Message;

MessageBox.Show("Ошибка сохранения:\n" + errorMessage);

}

}

private void GenerateFile\_Click(object sender, RoutedEventArgs e)

{

try

{

if (!\_lastTripId.HasValue)

{

MessageBox.Show("Сначала сохраните командировку.");

return;

}

string projectRoot = Directory.GetParent(AppDomain.CurrentDomain.BaseDirectory).Parent.Parent.FullName;

string templatePath = Path.Combine(projectRoot, "Приказ.doc");

string ordersFolder = Path.Combine(projectRoot, "Приказы");

Directory.CreateDirectory(ordersFolder);

string fileName = $"Приказ\_о\_командировке\_{\_lastTripId}.doc";

string outputPath = Path.Combine(ordersFolder, fileName);

int employeeId = (int)EmployeeComboBox.SelectedValue;

int managerId = (int)ManagerComboBox.SelectedValue; // Fixed: Changed from EmployeeComboBox to ManagerComboBox

string fullNameEmployee = "", positionEmployee = "", departmentName = "", fullNameManager = "";

using (var db = new TalonBaseEntities())

{

var employee = db.Eployee.Find(employeeId);

var manager = db.Manager.Find(managerId);

var department = db.Department.Find(employee?.Deportament\_Id);

if (employee != null)

{

fullNameEmployee = employee.Full\_Name;

positionEmployee = employee.Position;

departmentName = department?.Department\_name ?? "—";

}

if (manager != null)

{

fullNameManager = manager.Full\_Name;

}

}

var wordApp = new Word.Application();

var document = wordApp.Documents.Open(templatePath);

void Replace(string placeholder, string value)

{

var range = document.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: placeholder, ReplaceWith: value ?? "", Replace: Word.WdReplace.wdReplaceAll);

}

Replace("{Id документа}", Guid.NewGuid().ToString().Substring(0, 8));

Replace("{Дата}", DateTime.Now.ToShortDateString());

Replace("{Full\_Name employee}", fullNameEmployee);

Replace("{Position employee}", positionEmployee);

Replace("{Id departament}", departmentName);

Replace("{Destination}", DestinationBox.Text);

Replace("{Start\_date}", StartDatePicker.SelectedDate?.ToShortDateString());

Replace("{End\_date}", EndDatePicker.SelectedDate?.ToShortDateString());

Replace("{purpose}", PurposeBox.Text);

Replace("{Full\_Name Manager}", fullNameManager);

document.SaveAs2(outputPath);

document.Close();

wordApp.Quit();

Process.Start(new ProcessStartInfo(outputPath) { UseShellExecute = true });

MessageBox.Show($"Документ успешно сгенерирован в папке Приказы: {fileName}");

}

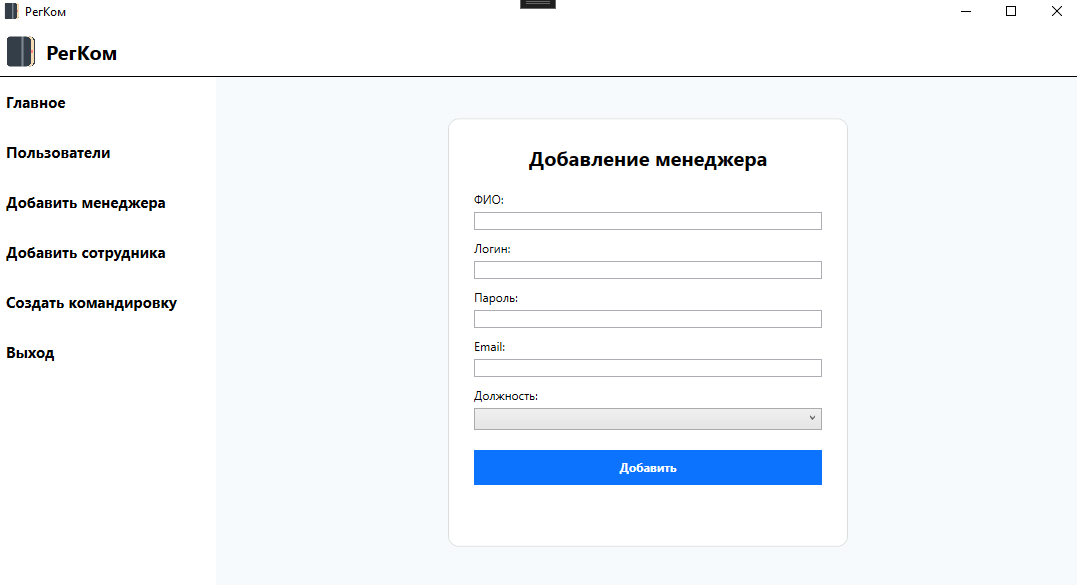
catch (Exception ex)

{

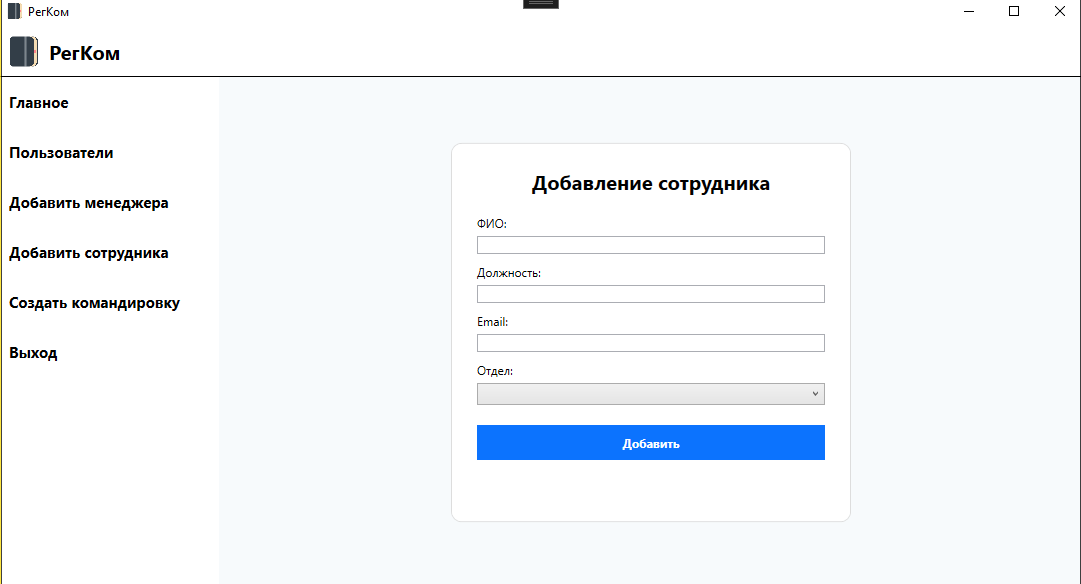
MessageBox.Show("Ошибка при генерации документа: " + ex.Message);

В данном методе было разработано создание командировочных талонов. Все введенные данные сохраняются в базу данных. Так же пользователь может открыть документ с данными о командировках.

## 3.2.4 Разработка модуля «Добавление пользователей»



*Рисунок 3.2.3.1 «Добавление менеджера»*



*Рисунок 3.2.3.2 «Добавление сотрудника»*

Методы создания пользователей:

private void AddEmployee\_Click(object sender, RoutedEventArgs e)

{

// Валидация

if (string.IsNullOrWhiteSpace(FullNameBox.Text) ||

string.IsNullOrWhiteSpace(PositionBox.Text) ||

string.IsNullOrWhiteSpace(EmailBox.Text) ||

DepartmentBox.SelectedItem == null)

{

StatusText.Text = "Заполните все поля.";

return;

}

// Создание сотрудника

var employee = new Eployee

{

Full\_Name = FullNameBox.Text,

Position = PositionBox.Text,

Contact\_info = EmailBox.Text,

Deportament\_Id = (int)DepartmentBox.SelectedValue // Получаем Id выбранного отдела

// НЕ ЗАДАЁМ employee.Id, так как он автоинкрементный

};

try

{

context.Eployee.Add(employee);

context.SaveChanges();

StatusText.Text = "Сотрудник успешно добавлен!";

ClearFields();

}

catch (System.Exception ex)

{

StatusText.Text = $"Ошибка: {ex.Message}";

}

}

private void ClearFields()

{

FullNameBox.Text = "";

PositionBox.Text = "";

EmailBox.Text = "";

DepartmentBox.SelectedIndex = -1;

}

}

private void CreateUser\_Click(object sender, RoutedEventArgs e)

{

// Валидация

if (string.IsNullOrWhiteSpace(FullNameBox.Text) ||

string.IsNullOrWhiteSpace(LoginBox.Text) ||

string.IsNullOrWhiteSpace(PasswordBox.Password) ||

PositionBox.SelectedItem == null ||

string.IsNullOrWhiteSpace(EmailBox.Text))

{

StatusText.Text = "Заполните все поля.";

return;

}

// Проверка на уникальность логина

if (context.Manager.Any(m => m.Login == LoginBox.Text))

{

StatusText.Text = "Такой логин уже существует.";

return;

}

// Создание менеджера

var manager = new Manager

{

Full\_Name = FullNameBox.Text,

Login = LoginBox.Text,

Password = PasswordBox.Password,

Position = (PositionBox.SelectedItem as ComboBoxItem)?.Content.ToString(),

Email = EmailBox.Text

// НЕ ЗАДАЁМ manager.Id!

};

try

{

context.Manager.Add(manager);

context.SaveChanges();

StatusText.Text = "Пользователь успешно создан!";

ClearFields();

}

catch (System.Exception ex)

{

StatusText.Text = $"Ошибка: {ex.Message}";

}

}

private void ClearFields()

{

FullNameBox.Text = "";

LoginBox.Text = "";

PasswordBox.Password = "";

PositionBox.SelectedIndex = -1;

EmailBox.Text = "";

}

}

В данном модуле было разработаны страницы добавления пользователей.

## 3.2.5. Разработка модуля «Данные о пользователях»



*Рисунок 3.2.5 «Данные о пользователях»*

Методы отображение пользователей и разграничение прав на отображение:

private void LoadUserData()

{

var mainWindow = Application.Current.MainWindow as MainWindow;

if (mainWindow != null)

{

currentUser = (mainWindow.DataContext as Manager) ?? throw new Exception("Пользователь не найден.");

}

switch (currentUser.Position)

{

case "Администратор":

case "Руководитель":

// Для администратора и руководителя показываем всё

EmployeeActionsColumn.Visibility = Visibility.Visible;

ManagersHeader.Visibility = Visibility.Visible;

ManagersGrid.Visibility = Visibility.Visible;

break;

case "Сотрудник":

// Для сотрудника скрываем управление менеджерами и действия

EmployeeActionsColumn.Visibility = Visibility.Collapsed;

ManagersHeader.Visibility = Visibility.Collapsed;

ManagersGrid.Visibility = Visibility.Collapsed;

break;

default:

MessageBox.Show("Неизвестная роль пользователя.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

break;

}

}

В данном методе было сделано отображение всех пользователей, а так же разграничение прав на показ. То есть сотрудник не видит менеджеров, в отличии от руководителя и администратора

# ЗАКЛЮЧЕНИЕ

В рамках курсового проекта была разработана автоматизированная информационная система "Регистрация командировочных удостоверений", предназначенная для учета и управления служебными поездками сотрудников организации. Система реализует весь необходимый функционал согласно техническому заданию: ввод и редактирование информации о командировках, ведение регистрационного журнала, формирование отчетов, разграничение прав доступа и обеспечение информационной безопасности.

Разработка выполнена с учетом современных требований к надежности и масштабируемости. Программное обеспечение обладает модульной структурой, включает авторизацию с капчей, ролевую систему доступа (администратор, кадровый сотрудник, руководитель), и позволяет экспортировать отчеты в формат MS Word. База данных построена по реляционной модели, что обеспечивает структурированное хранение данных и их целостность.

Использование системы значительно сокращает время на выполнение рутинных операций. Регистрация одной командировки теперь занимает несколько секунд вместо нескольких минут, а формирование отчетов — 3–4 минуты вместо 6–12 часов. Это позволяет повысить производительность труда сотрудников, снизить количество ошибок, связанных с человеческим фактором, и обеспечить более строгий контроль над исполнением командировок.

Система готова к внедрению в организацию и может быть расширена в будущем за счет добавления новых модулей и интеграции с другими корпоративными системами. Все этапы разработки — от проектирования до реализации и тестирования — выполнены в полном объеме, что подтверждает достижение целей и успешное завершение курсового проекта.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Руководство по WPF [Электронный ресурс] — режим доступа:

<https://metanit.com/sharp/wpf/1.php>

1. Документация по C# [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/overview>

1. Документация по Microsoft SQL [Электронный ресурс] — режим доступа:

<https://learn.microsoft.com/ru-ru/ssms/sql-server-management-studio-ssms>