МИНИСТЕРСТВО ОБРАЗОВАНИЯ КИРОВСКОЙ ОБЛАСТИ

Кировское областное государственное профессиональное образовательное

бюджетное учреждение

«Слободской колледж педагогики и социальных отношений»

**ОТЧЕТ**

**по производственной практике**

**ПМ.11. Разработка, администрирование и защита баз данных**

**Тема: «Разработка базы данных для учёта коммунальных платежей»**

Студент

Горовенко Кирилл Алексеевич

Группа 22П-1

Специальность 09.02.07 Информационные системы и программирование

Руководитель практики от колледжа:

*Махнев Александр Анатольевич*

Руководитель практики от организации:

*\_\_\_\_\_\_\_\_\_\_ Павленко Сергей Михайлович*

подпись

УТВЕРЖДАЮ:

Директор

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
 Наименование организации

\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись расшифровка

М. П.

2024-2025 уч. год

**Оглавление**

1. Введение3

2. Анализ предметной области3

3. Проектирование базы данных4

4. Реализация базы данных6

5. Разработка программного приложения7

6. Уровни доступа для пользователей7

7. Создание запросов и отчетов8

8. Резервное копирование и восстановление8

9. Результаты8

10. Заключение8

11. Приложения 9

**1. Введение**

**Цель практики:**   
Разработка информационной системы для учета коммунальных платежей за газ с использованием СУБД Microsoft SQL Server и создание клиентского приложения с функциями CRUD.

**Задачи:**

* Анализ предметной области
* Проектирование нормализованной структуры БД
* Реализация базы данных в MS SQL Server
* Создание приложения на C# WPF для работы с данными

**2. Анализ предметной области**

Система должна обеспечивать:

* Учет клиентов и привязанных к ним квартир
* Фиксацию показаний газовых счетчиков
* Формирование счетов по тарифам
* Обработку платежей
* Контроль доступа пользователей

Проведен анализ требований к системе учета коммунальных платежей. Определены основные сущности и их атрибуты:

* Клиенты
* Квартиры
* Здания
* Счетчики газа
* Показания счетчиков
* Счета
* Платежи
* Тарифы
* Пользователи системы

**3. Проектирование базы данных**

Концептуальная модель (ER-диаграмма) (Рис. 1):

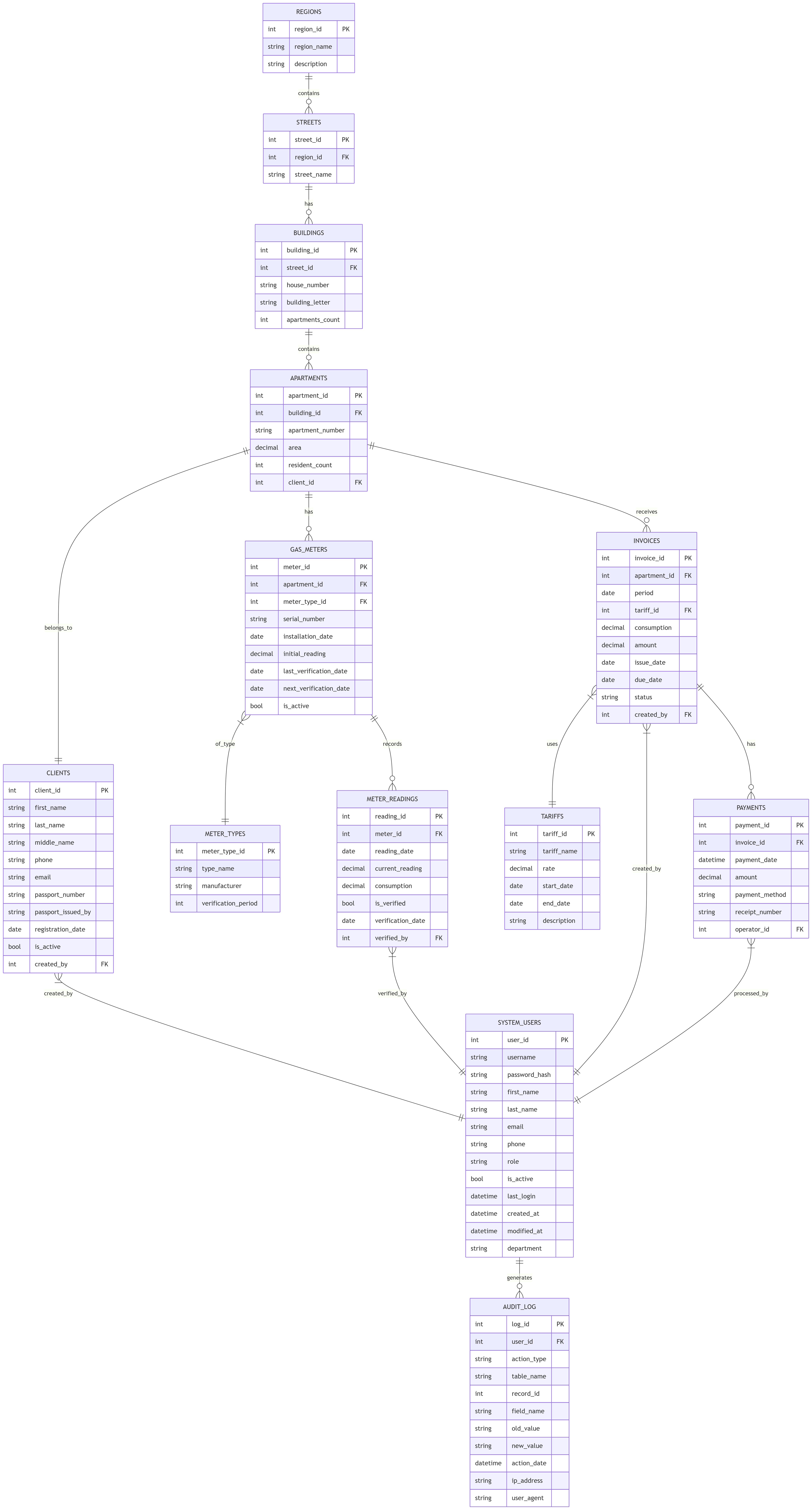


Рисунок 1 – ER-диаграмма

Связи между таблицами

1. Regions → Streets → Buildings → Apartments
   * Регионы содержат улицы (1 ко многим)
   * Улицы содержат здания (1 ко многим)
   * Здания содержат квартиры (1 ко многим)

Клиенты и квартиры

1. Clients → Apartments
   * Клиенты владеют квартирами (1 ко многим)

Учет газа

1. Apartments → GasMeters → MeterReadings
   * В квартирах установлены газовые счетчики (1 ко многим)
   * По счетчикам снимаются показания (1 ко многим)
2. MeterTypes → GasMeters
   * Типы счетчиков связаны с конкретными счетчиками (1 ко многим)

Финансовые операции

1. Apartments → Invoices → Payments
   * Квартирам выставляются счета (1 ко многим)
   * По счетам производятся платежи (1 ко многим)
2. Tariffs → Invoices
   * Тарифы применяются к счетам (1 ко многим)

Системные связи

1. SystemUsers → AuditLog
   * Пользователи системы оставляют записи в журнале аудита (1 ко многим)
2. SystemUsers → Clients (created\_by)
   * Пользователи создают записи клиентов (1 ко многим)
3. SystemUsers → Invoices (created\_by)
   * Пользователи создают счета (1 ко многим)
4. SystemUsers → MeterReadings (verified\_by)
   * Пользователи подтверждают показания счетчиков (1 ко многим)
5. SystemUsers → Payments (operator\_id)
   * Пользователи регистрируют платежи (1 ко многим)

**4. Реализация базы данных**

Физическая модель (Рис. 2):

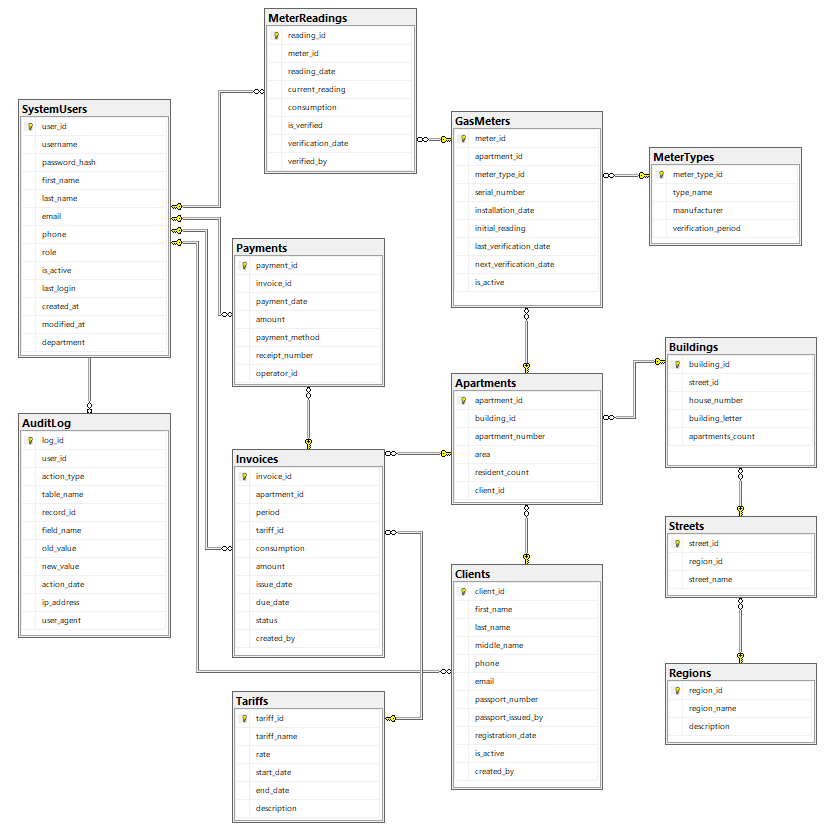


Рисунок 2 – Физическая модель базы данных

**5. Разработка программного приложения**

Создано приложение на C# с использованием WPF для взаимодействия с базой данных. Основные функции:

* Авторизация пользователей с разными ролями (администратор, менеджер, техник, бухгалтер и оператор) (Приложение 1 - 6).
* Управление клиентами, квартирами, зданиями, счетчиками, показаниями, счетами, платежами и пользователями (Приложение 7 - 15).
* Формирование отчетов (Приложение 16).

**6. Уровни доступа для пользователей**

Реализованы роли:

* **Администратор**: полный доступ ко всем функциям.
* **Менеджер**: управление клиентами, квартирами, зданиями.
* **Техник**: работа со счетчиками и показаниями.
* **Бухгалтер**: управление счетами и платежами.
* **Оператор**: ограниченный доступ к данным клиентов и счетов.

**7. Создание запросов и отчетов**

Реализованы отчеты:

* Потребление газа по месяцам.
* Платежи по клиентам.
* Задолженности по квартирам.
* Показания счетчиков.

Метод для печати отчета (Приложение 17).

**8. Резервное копирование и восстановление**

Реализована функция резервного копирования базы данных доступная пользователю «Администратор» (Приложение 18).

**9. Результаты**

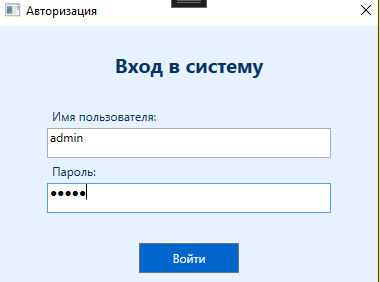
1. Разработана база данных для учета коммунальных платежей.
2. Создано программное приложение с графическим интерфейсом.
3. Реализованы все требуемые функции, включая CRUD-операции, отчеты и резервное копирование.

**10. Заключение**

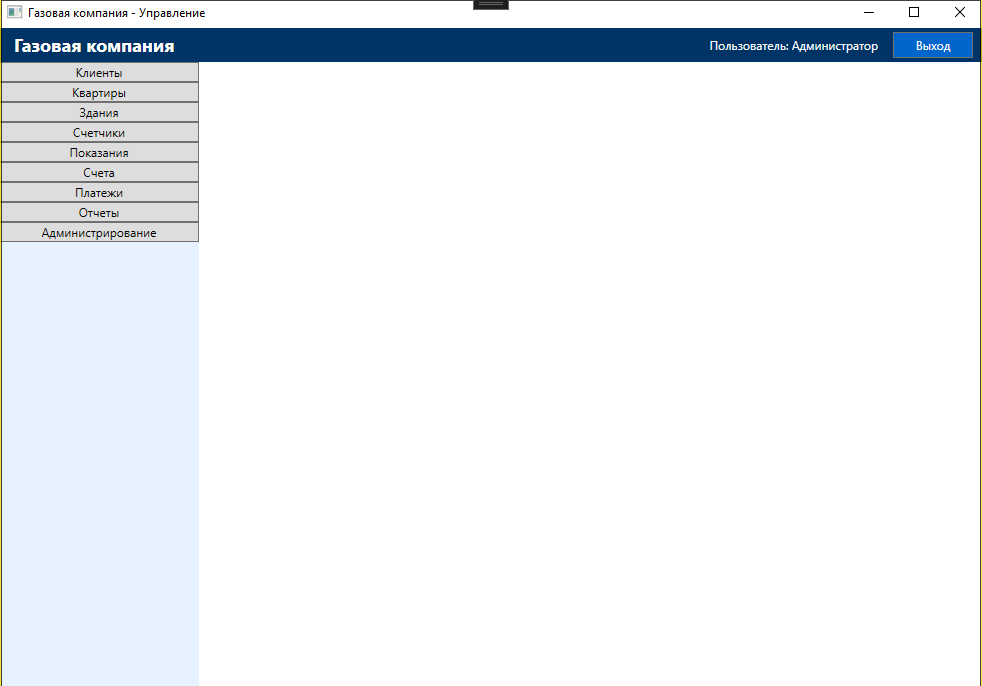
В ходе практики были получены навыки проектирования баз данных, работы с Microsoft SQL Server и разработки приложений на C#. Все поставленные задачи выполнены в срок. Приложение готово к использованию в реальных условиях.

**11.** **Приложения**

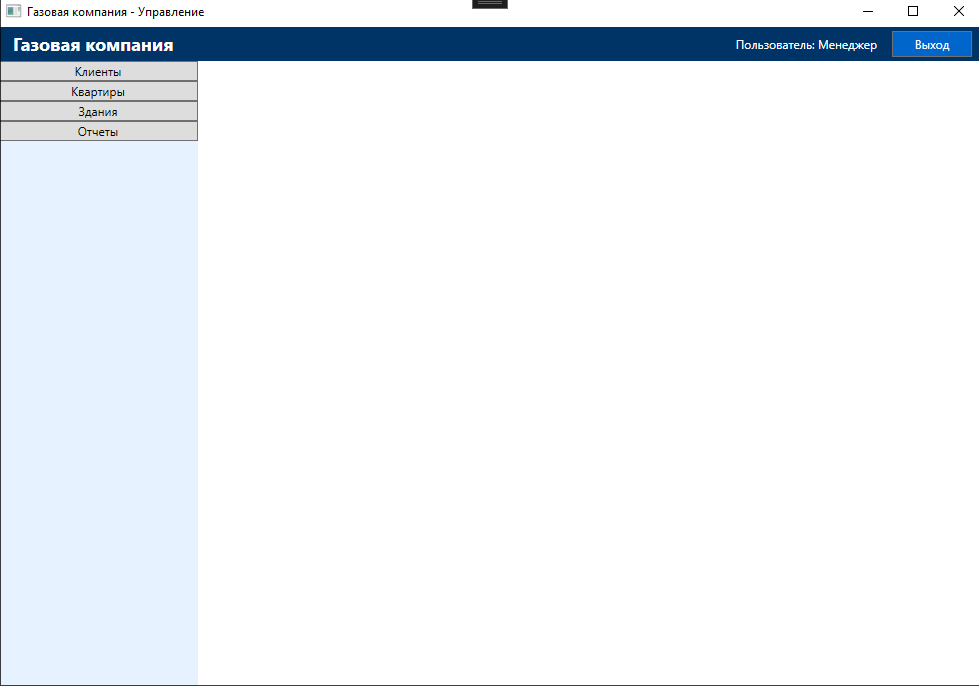
Скриншоты интерфейса:



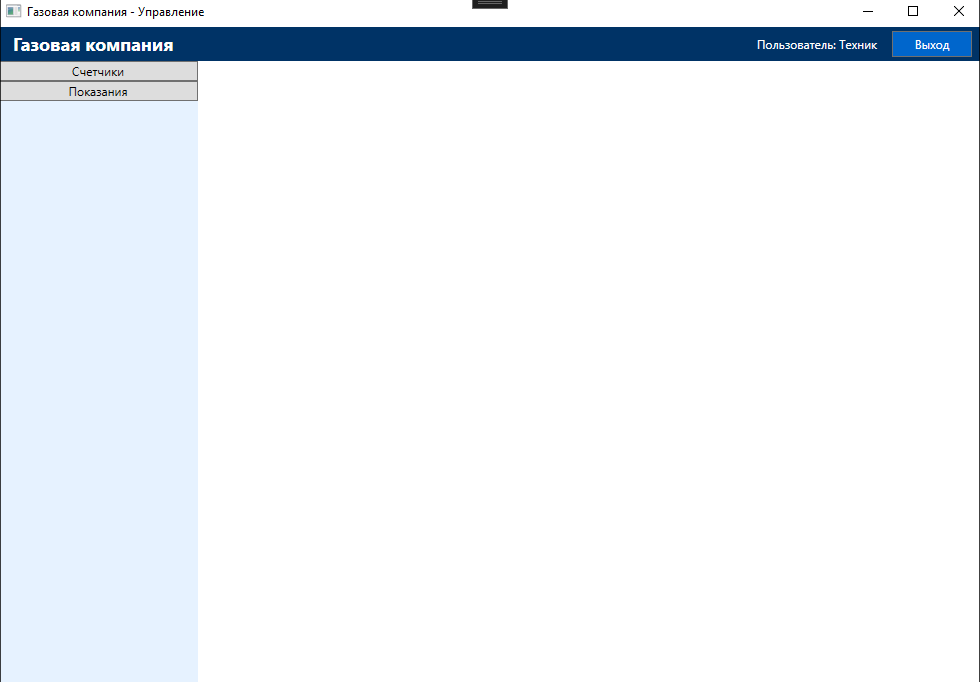
Приложение 1 – Авторизация



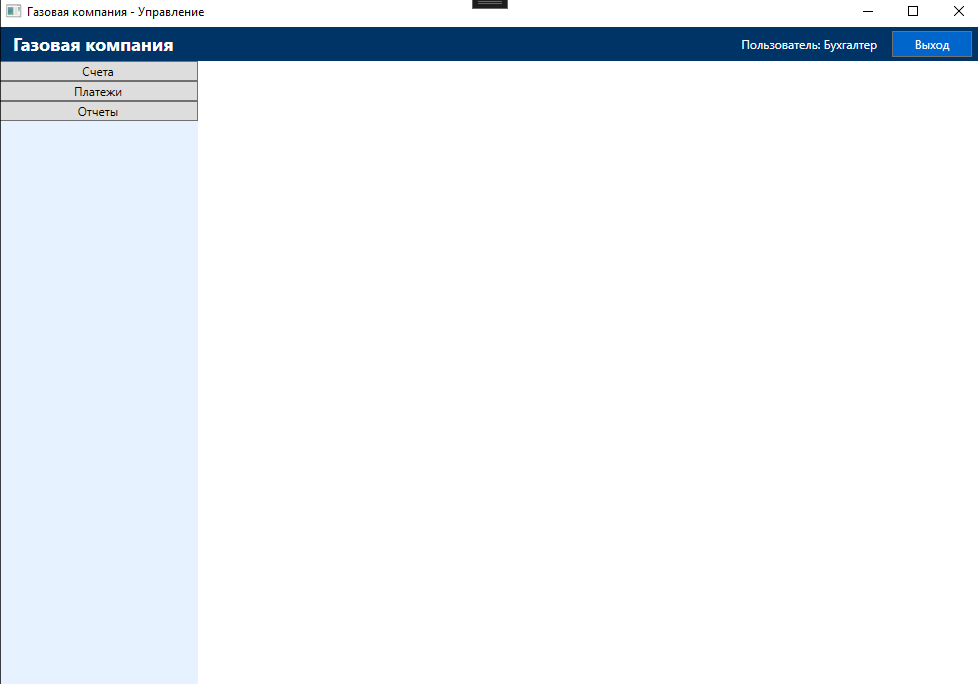
Приложение 2 – Главное окно с интерфейсом администратора



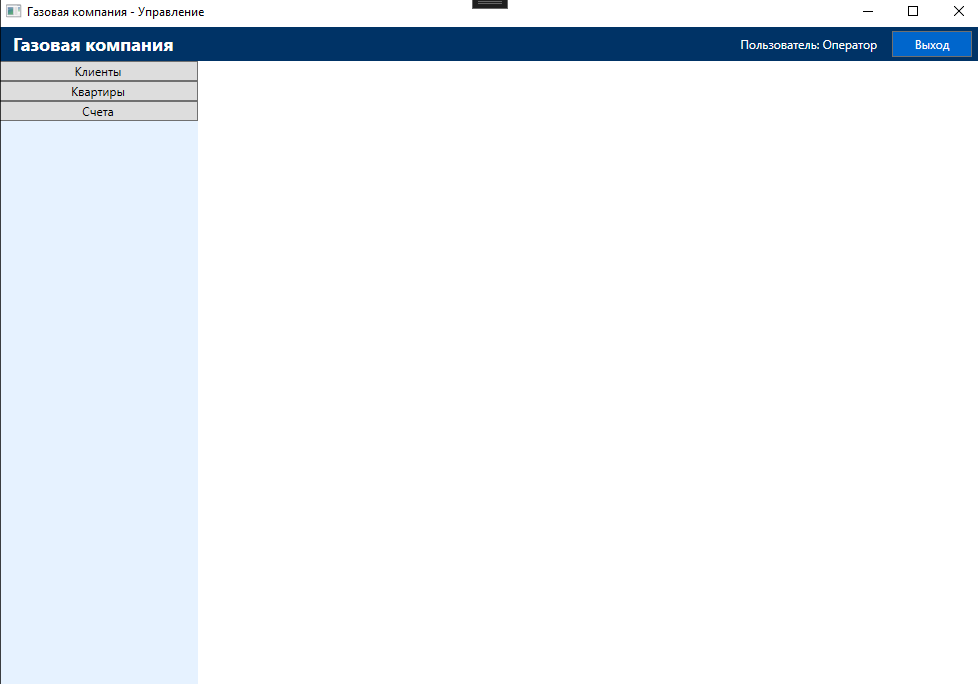
Приложение 3 – Главное окно с интерфейсом менеджера



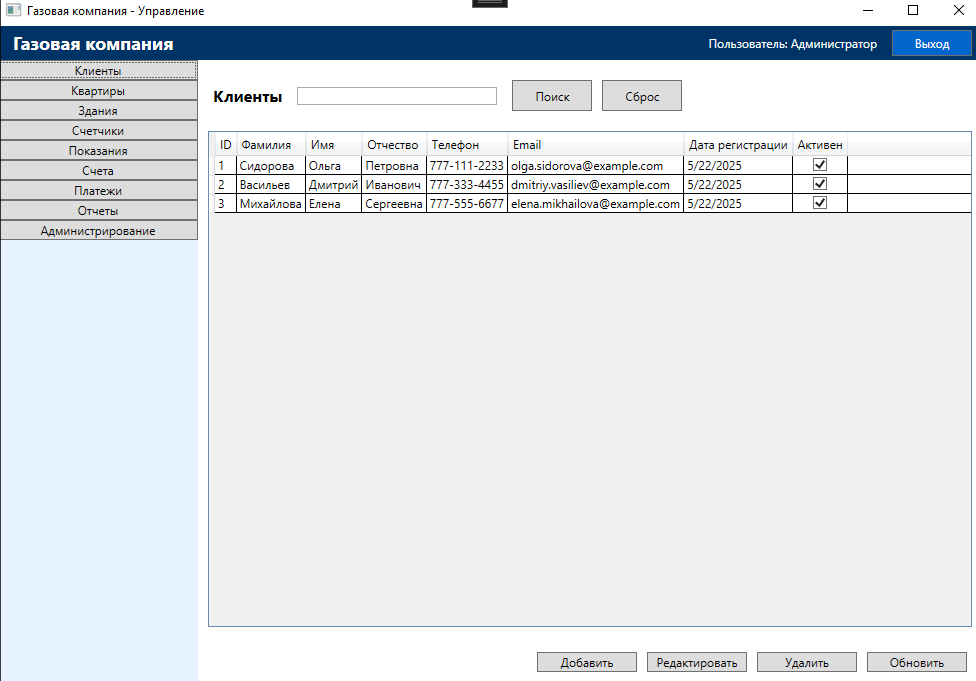
Приложение 4 – Главное окно с интерфейсом техника



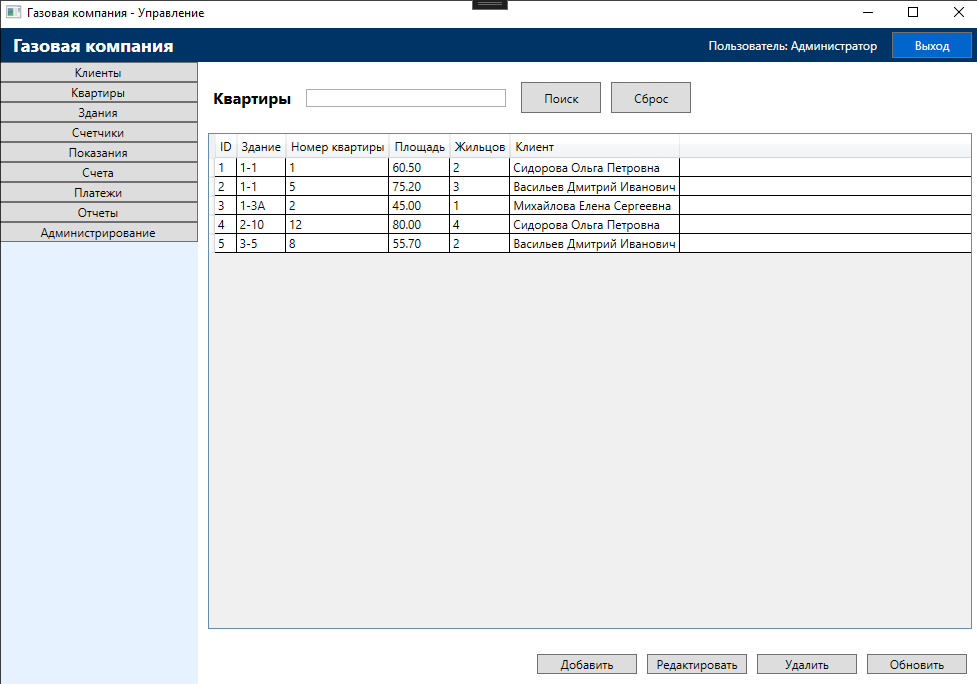
Приложение 5 – Главное окно с интерфейсом бухгалтера



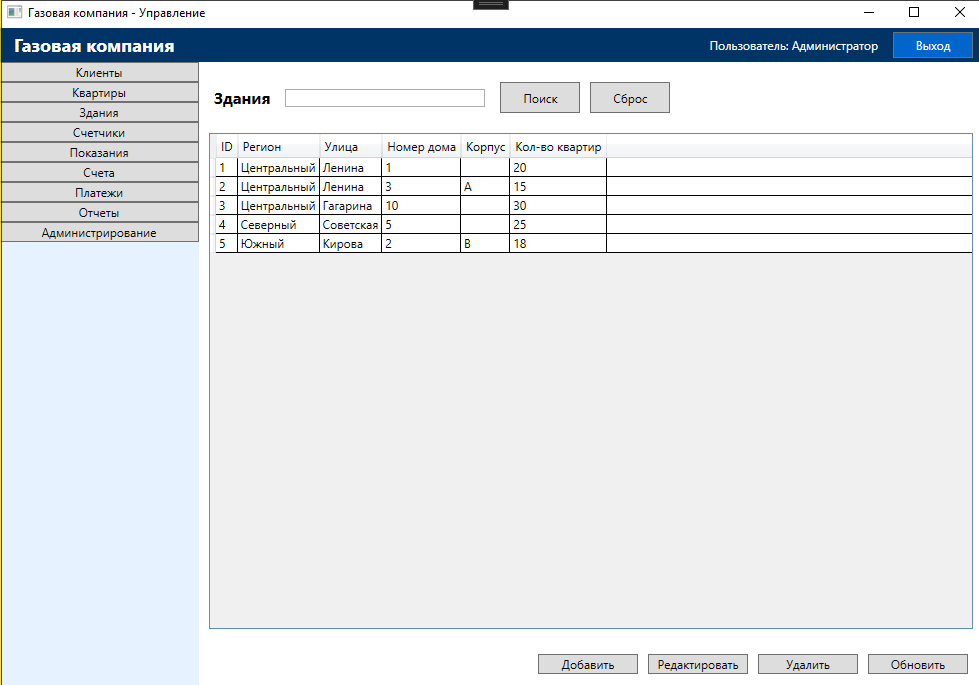
Приложение 6 – Главное окно с интерфейсом техника



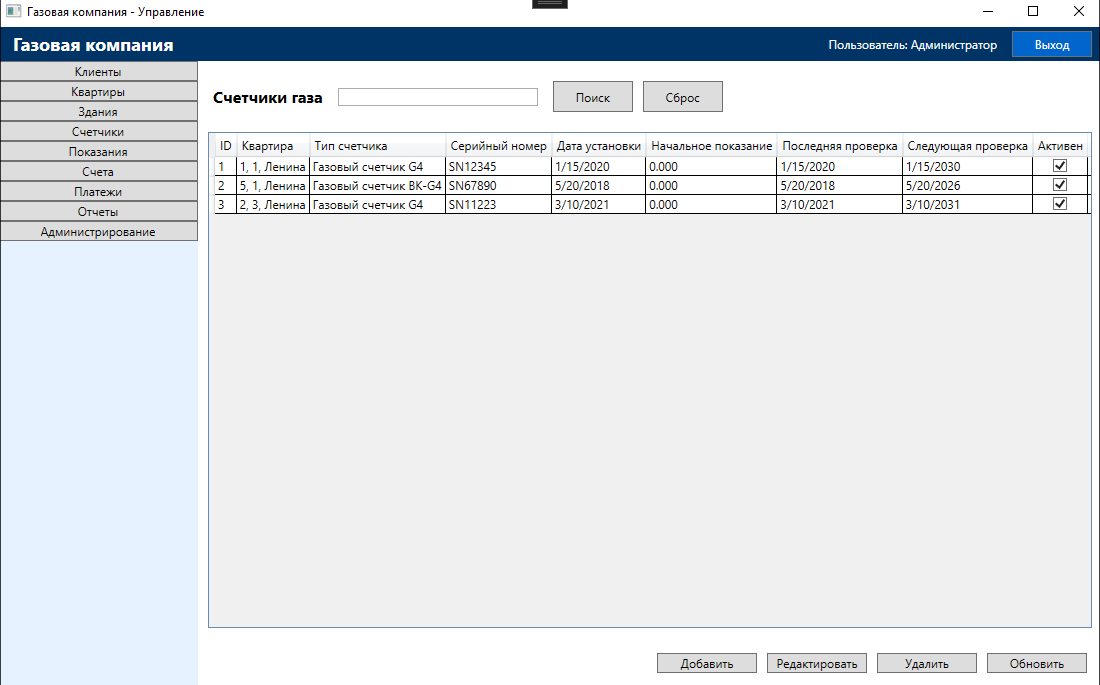
Приложение 7 – Интерфейс управления клиентами



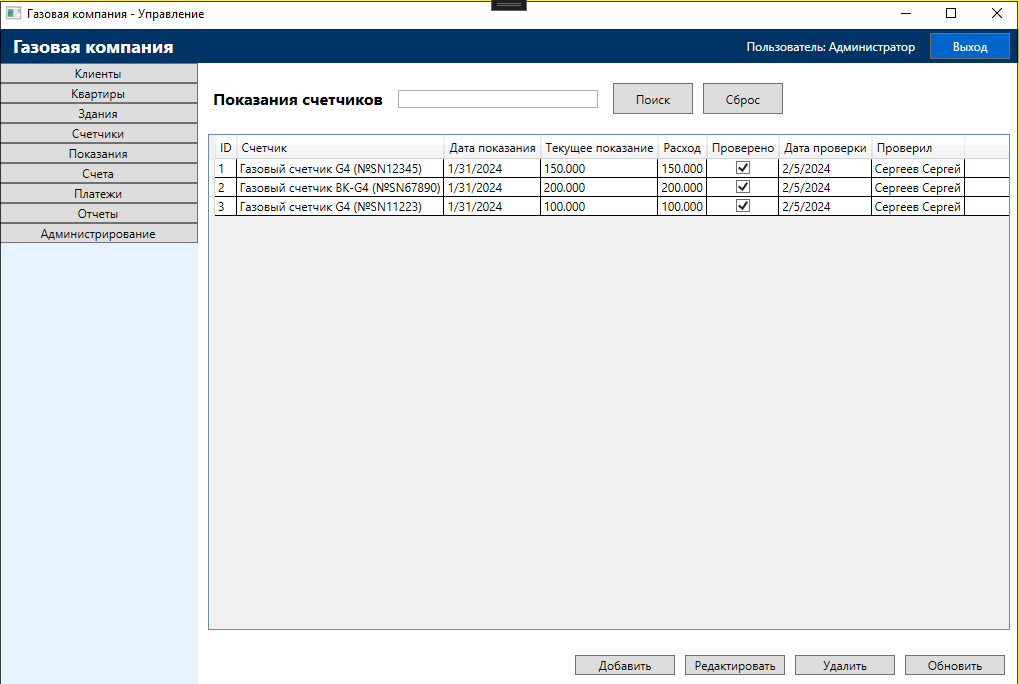
Приложение 8 – Интерфейс управления квартирами



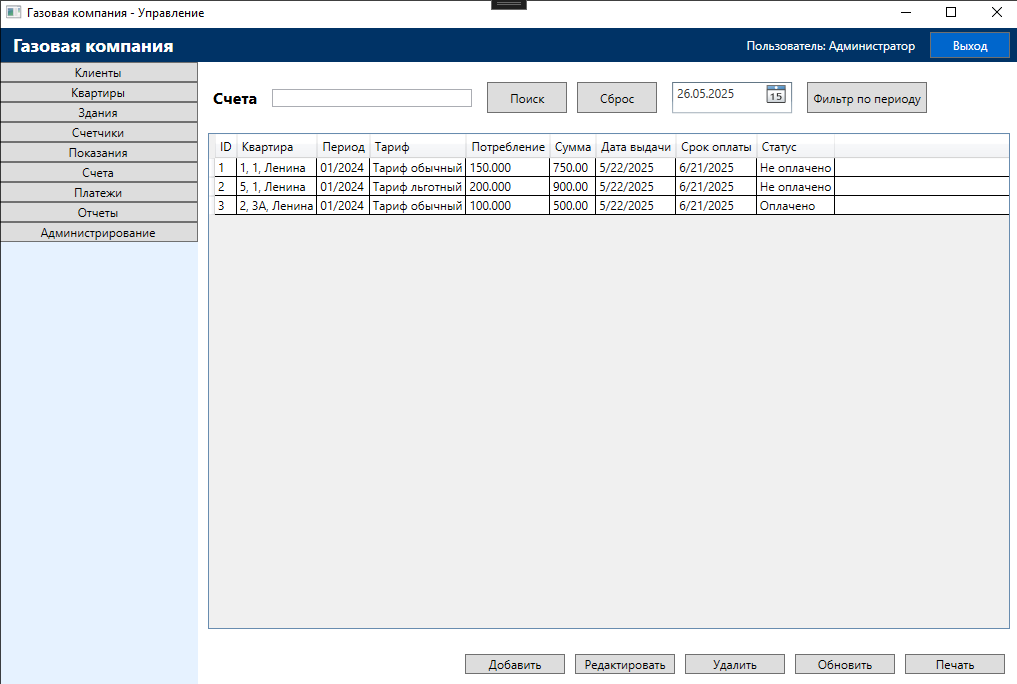
Приложение 9 – Интерфейс управления зданиями



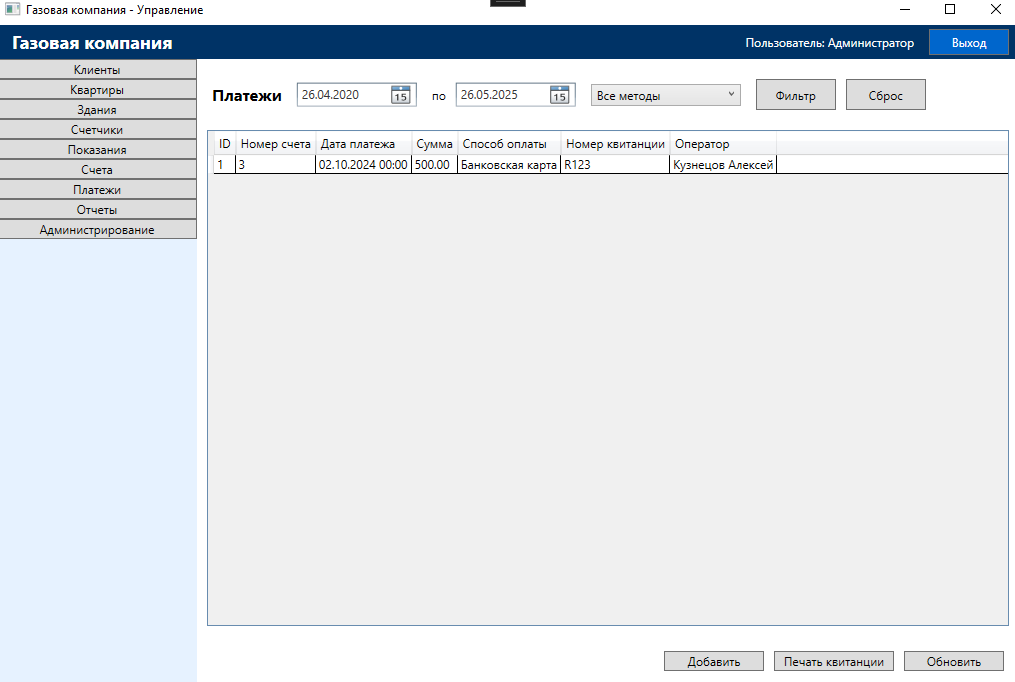
Приложение 10 – Интерфейс управления счетчиками



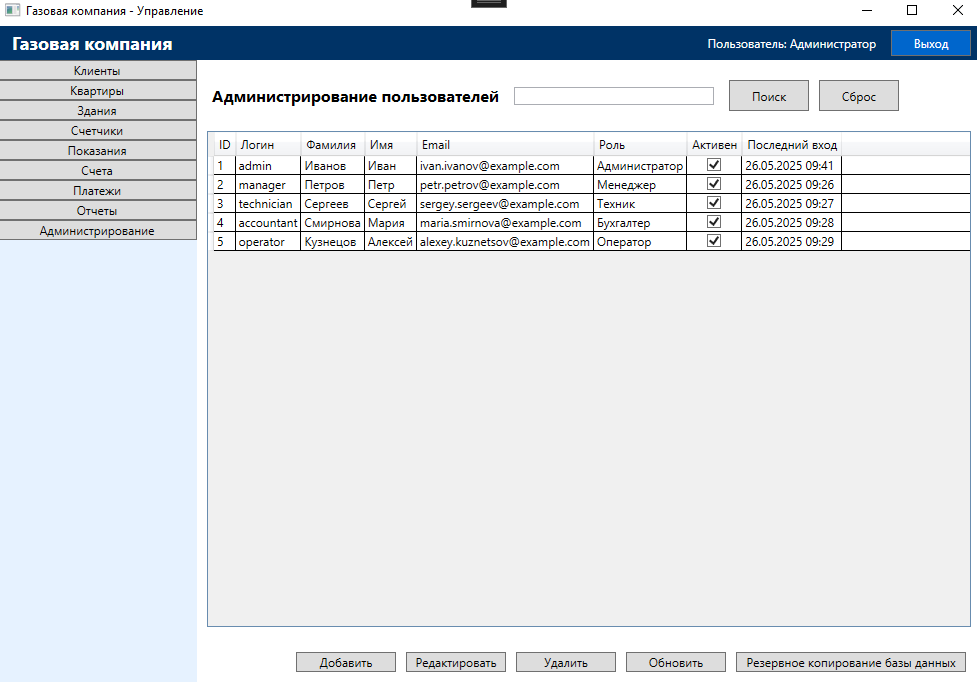
Приложение 11 – Интерфейс управления показаниями счетчиков



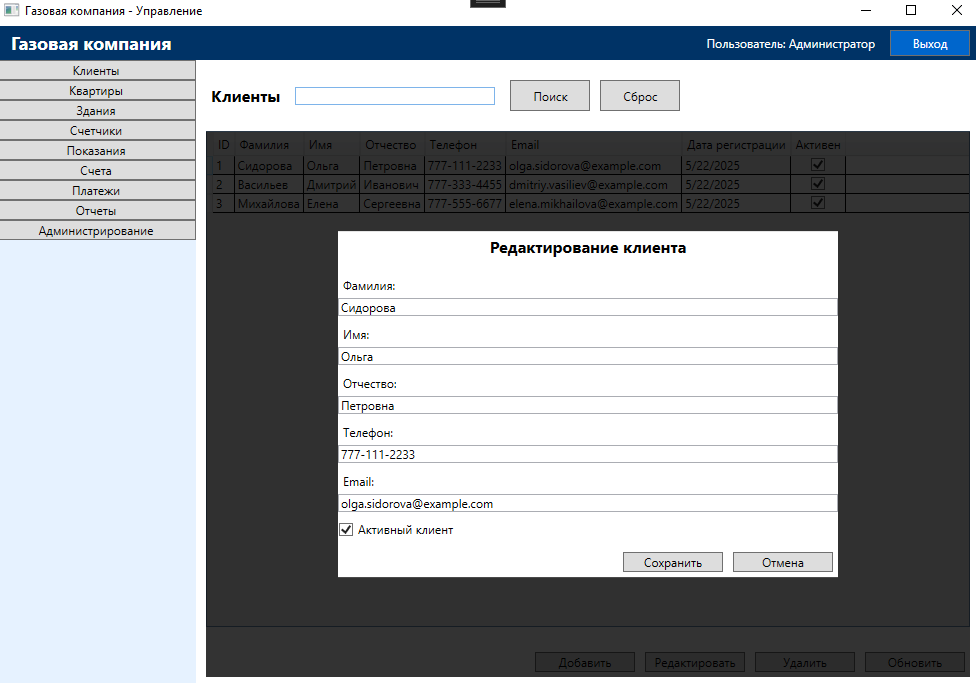
Приложение 12 – Интерфейс управления счетами



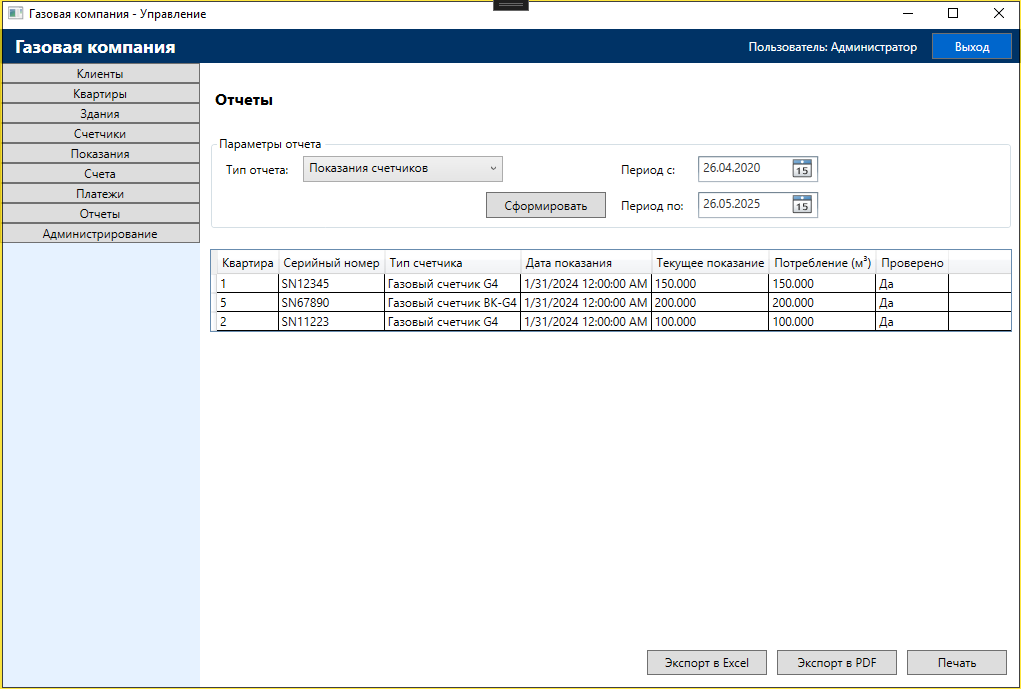
Приложение 13 – Интерфейс управления платежами



Приложение 14 – Интерфейс администрирования



Приложение 15 – Интерфейс редактирования данных



Приложение 16 – Интерфейс управления отчетами

Код программы:

Приложение 17 – Код метода печати отчетов:

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

{

if (reportGrid.ItemsSource == null)

{

MessageBox.Show("Нет данных для печати", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

DataView dataView = reportGrid.ItemsSource as DataView;

if (dataView == null) return;

string reportType = (cmbReportType.SelectedItem as ComboBoxItem)?.Content.ToString() ?? "Отчет";

// Создаем документ для печати

FlowDocument document = new FlowDocument();

document.PagePadding = new Thickness(50);

document.ColumnGap = 0;

document.ColumnWidth = 700;

// Заголовок документа

Paragraph header = new Paragraph(new Run("ГАЗОВАЯ КОМПАНИЯ"));

header.FontSize = 18;

header.FontWeight = FontWeights.Bold;

header.TextAlignment = TextAlignment.Center;

header.Margin = new Thickness(0, 0, 0, 20);

document.Blocks.Add(header);

// Название отчета

Paragraph reportHeader = new Paragraph(new Run(reportType));

reportHeader.FontSize = 16;

reportHeader.FontWeight = FontWeights.Bold;

reportHeader.TextAlignment = TextAlignment.Center;

reportHeader.Margin = new Thickness(0, 0, 0, 10);

document.Blocks.Add(reportHeader);

// Период отчета

if (dpStartDate.SelectedDate != null && dpEndDate.SelectedDate != null)

{

Paragraph period = new Paragraph(new Run($"Период: с {dpStartDate.SelectedDate.Value:dd.MM.yyyy} по {dpEndDate.SelectedDate.Value:dd.MM.yyyy}"));

period.FontSize = 12;

period.TextAlignment = TextAlignment.Center;

period.Margin = new Thickness(0, 0, 0, 20);

document.Blocks.Add(period);

}

// Дата формирования

Paragraph date = new Paragraph(new Run($"Дата формирования: {DateTime.Now:dd.MM.yyyy HH:mm}"));

date.FontSize = 12;

date.TextAlignment = TextAlignment.Center;

date.Margin = new Thickness(0, 0, 0, 30);

document.Blocks.Add(date);

// Таблица с данными

Table table = new Table();

table.CellSpacing = 0;

table.Margin = new Thickness(0, 0, 0, 30);

// Добавляем колонки

foreach (DataColumn column in dataView.Table.Columns)

{

table.Columns.Add(new TableColumn() { Width = GridLength.Auto });

}

// Строки таблицы

table.RowGroups.Add(new TableRowGroup());

// Заголовки колонок

TableRow headerRow = new TableRow();

foreach (DataColumn column in dataView.Table.Columns)

{

headerRow.Cells.Add(new TableCell(new Paragraph(new Run(column.ColumnName))

{

FontWeight = FontWeights.Bold,

Margin = new Thickness(5)

}));

}

table.RowGroups[0].Rows.Add(headerRow);

// Данные

foreach (DataRowView rowView in dataView)

{

TableRow row = new TableRow();

foreach (DataColumn column in dataView.Table.Columns)

{

string value = rowView[column.ColumnName].ToString();

if (column.DataType == typeof(DateTime))

{

DateTime dateValue = (DateTime)rowView[column.ColumnName];

value = dateValue.ToString("dd.MM.yyyy");

}

else if (column.DataType == typeof(decimal))

{

decimal decimalValue = (decimal)rowView[column.ColumnName];

value = decimalValue.ToString("N2");

}

row.Cells.Add(new TableCell(new Paragraph(new Run(value))

{

Margin = new Thickness(5)

}));

}

table.RowGroups[0].Rows.Add(row);

}

document.Blocks.Add(table);

// Подпись

Paragraph sign = new Paragraph(new Run("Ответственный: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"));

sign.FontSize = 12;

sign.Margin = new Thickness(0, 30, 0, 0);

document.Blocks.Add(sign);

// Печать документа

PrintDialog printDialog = new PrintDialog();

if (printDialog.ShowDialog() == true)

{

printDialog.PrintDocument(((IDocumentPaginatorSource)document).DocumentPaginator, reportType);

LogUserAction(\_userId, "PRINT", "Reports", 0);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при печати отчета: {ex.Message}", "Ошибка",

MessageBoxButton.OK, MessageBoxImage.Error);

}

}

Приложение 17 – Код метода резервного копирования базы данных:

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

{

try

{

// Диалог для выбора места сохранения резервной копии

var saveFileDialog = new SaveFileDialog

{

Filter = "Backup files (\*.bak)|\*.bak",

Title = "Создание резервной копии базы данных",

FileName = $"GasCompanyDB\_Backup\_{DateTime.Now:yyyyMMdd\_HHmmss}.bak"

};

if (saveFileDialog.ShowDialog() == true)

{

string backupPath = saveFileDialog.FileName;

// Выполнение резервного копирования

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

string backupQuery = $"BACKUP DATABASE [GasCompanyDB] TO DISK = '{backupPath}' WITH FORMAT, MEDIANAME = 'GasCompanyBackup', NAME = 'Полная резервная копия GasCompanyDB';";

using (SqlCommand command = new SqlCommand(backupQuery, connection))

{

command.ExecuteNonQuery();

}

}

MessageBox.Show($"Резервная копия успешно создана: {backupPath}", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

// Логирование действия

LogUserAction(\_userId, "BACKUP", "Database", 0);

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании резервной копии: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}