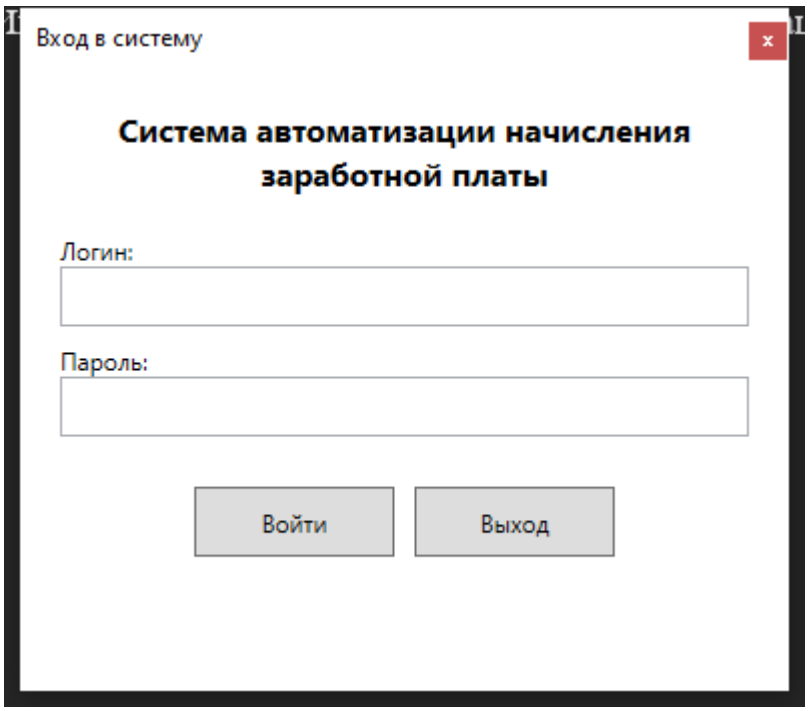
Практическая работа: Реализация модулей информационной системы

Модуль 1: Главное приложение (App.xaml.cs)

Назначение: Инициализация приложения, управление авторизацией

```
using CompanyPayrollApp.Windows;
using System.Windows;

namespace CompanyPayrollApp
{
    public partial class App : Application
    {
        public static string ConnectionString =
            @"Data Source=.\SQLEXPRESS;Initial Catalog=CompanyPayrollDB;Integrated Se-
curity=True;";
        public static string GetEmployeesViewQuery()
        {
            return @"
            SELECT
                e.id as EmployeeID,
                e.last_name + ' ' + e.first_name +
                CASE WHEN e.middle_name IS NOT NULL AND e.middle_name != ''
                     THEN ' ' + e.middle_name ELSE '' END as FullName,
                e.email as Email,
                e.phone as Phone,
                e.birth_date as BirthDate,
                e.hire_date as HireDate,
                e.status as Status,
                d.name as DepartmentName,
                p.title as PositionTitle,
                p.salary as PositionSalary
            FROM Employees e
            LEFT JOIN Departments d ON e.department_id = d.id
            LEFT JOIN Positions p ON e.position_id = p.id";
        }
        protected override void OnStartup(StartupEventArgs e)
        {
            base.OnStartup(e);

            var loginWindow = new LoginWindow();
            var result = loginWindow.ShowDialog();

            if (result == true)
            {
                var mainWindow = new MainWindow();
                MainWindow = mainWindow;
                mainWindow.Show();
            }
            else
            {
                Shutdown();
            }
        }

    }
}
```



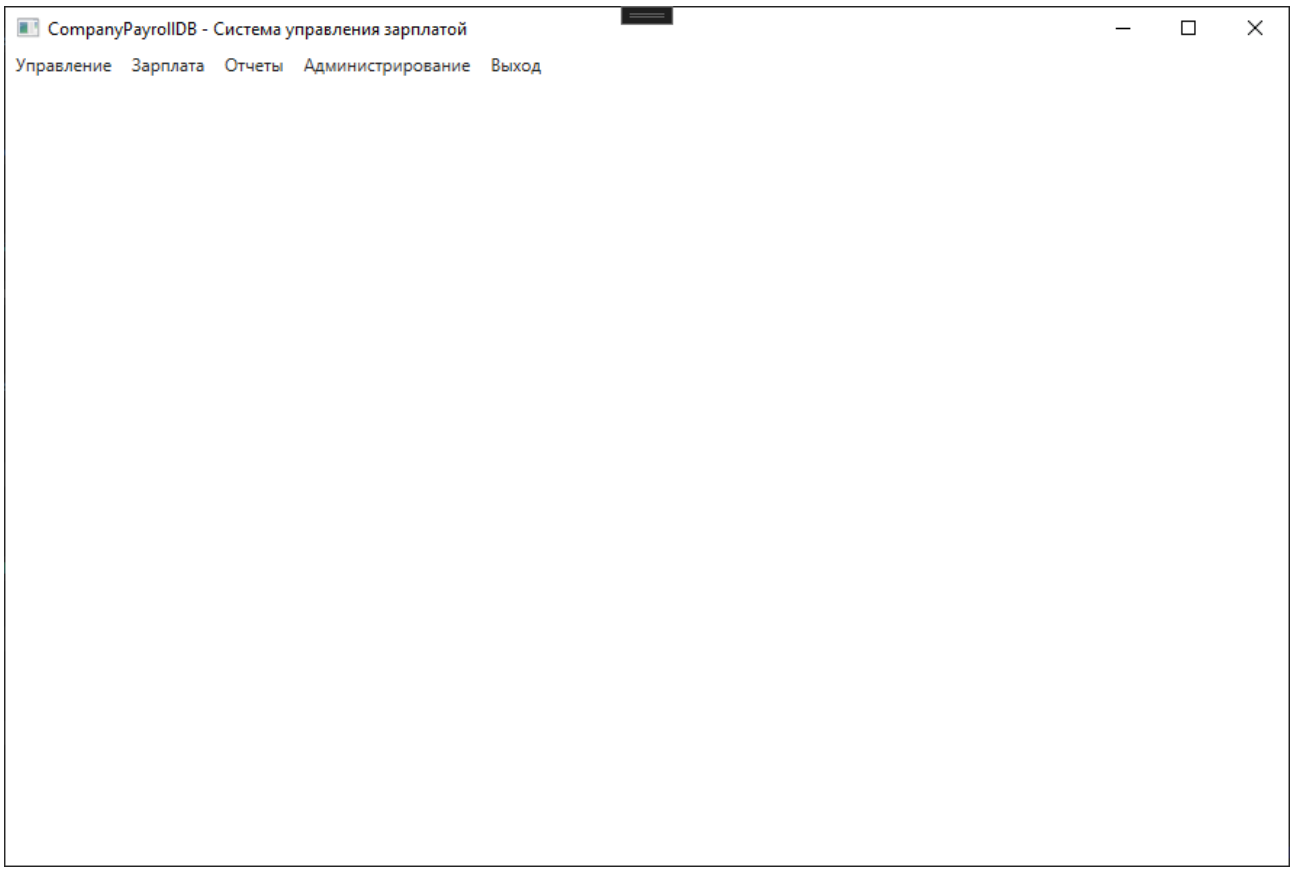
Модуль 2: Главное меню (MainWindow.xaml.cs)

Назначение: Навигация между модулями системы

```
using CompanyPayrollApp.Pages;
using CompanyPayrollApp.Windows;
using System;
using System.Windows;
using System.Windows.Controls;

namespace CompanyPayrollApp
{
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            StatusText.Text = "Система управления зарплатой CompanyPayrollDB";
        }

        private void Employees_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var page = new EmployeesPage();
                MainFrame.Navigate(page);
                StatusText.Text = "Модуль: Управление сотрудниками";
            }
            catch (System.Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки модуля сотрудников: {ex.Message}",
                    "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void Departments_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var page = new DepartmentsPage();
```

```
                MainFrame.Navigate(page);
                StatusText.Text = "Модуль: Управление отделами";
            }
            catch (System.Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки модуля отделов: {ex.Message}",
                    "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void CalculatePayroll_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var window = new CalculatePayrollWindow();
                window.Owner = this;
                window.ShowDialog();
                StatusText.Text = "Модуль: Расчет заработной платы";
            }
            catch (System.Exception ex)
            {
                MessageBox.Show($"Ошибка открытия модуля расчета: {ex.Message}",
                    "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }

        private void Exit_Click(object sender, RoutedEventArgs e)
        {
            if (MessageBox.Show("Выйти из системы?", "Подтверждение",
                MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxRe-
sult.Yes)
            {
                Application.Current.Shutdown();
            }
        }
        private void Positions_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var page = new PositionsPage();
                MainFrame.Navigate(page);
                StatusText.Text = "Модуль: Управление должностями";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка");
            }
        }

        private void Payments_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var window = new PaymentsWindow();
                window.Owner = this;
                window.ShowDialog();
                StatusText.Text = "Модуль: Управление выплатами";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка");
            }
        }

        private void ReportEmployees_Click(object sender, RoutedEventArgs e)
```

```
        {
            try
            {
                var page = new ReportEmployeesPage();
                MainFrame.Navigate(page);
                StatusText.Text = "Модуль: Отчет по сотрудникам";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка");
            }
        }

        private void ReportDepartments_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var page = new ReportDepartmentsPage();
                MainFrame.Navigate(page);
                StatusText.Text = "Модуль: Отчет по отделам";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка");
            }
        }
        private void AuditLog_Click(object sender, RoutedEventArgs e)
        {
            try
            {
                var window = new Windows.AuditLogWindow();
                window.Owner = this;
                window.ShowDialog();
                StatusText.Text = "Модуль: Журнал аудита";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка открытия журнала аудита: {ex.Message}",
"Ошибка");
            }
        }
    }
}
```



Модуль 3: Управление сотрудниками (EmployeesPage.xaml.cs)

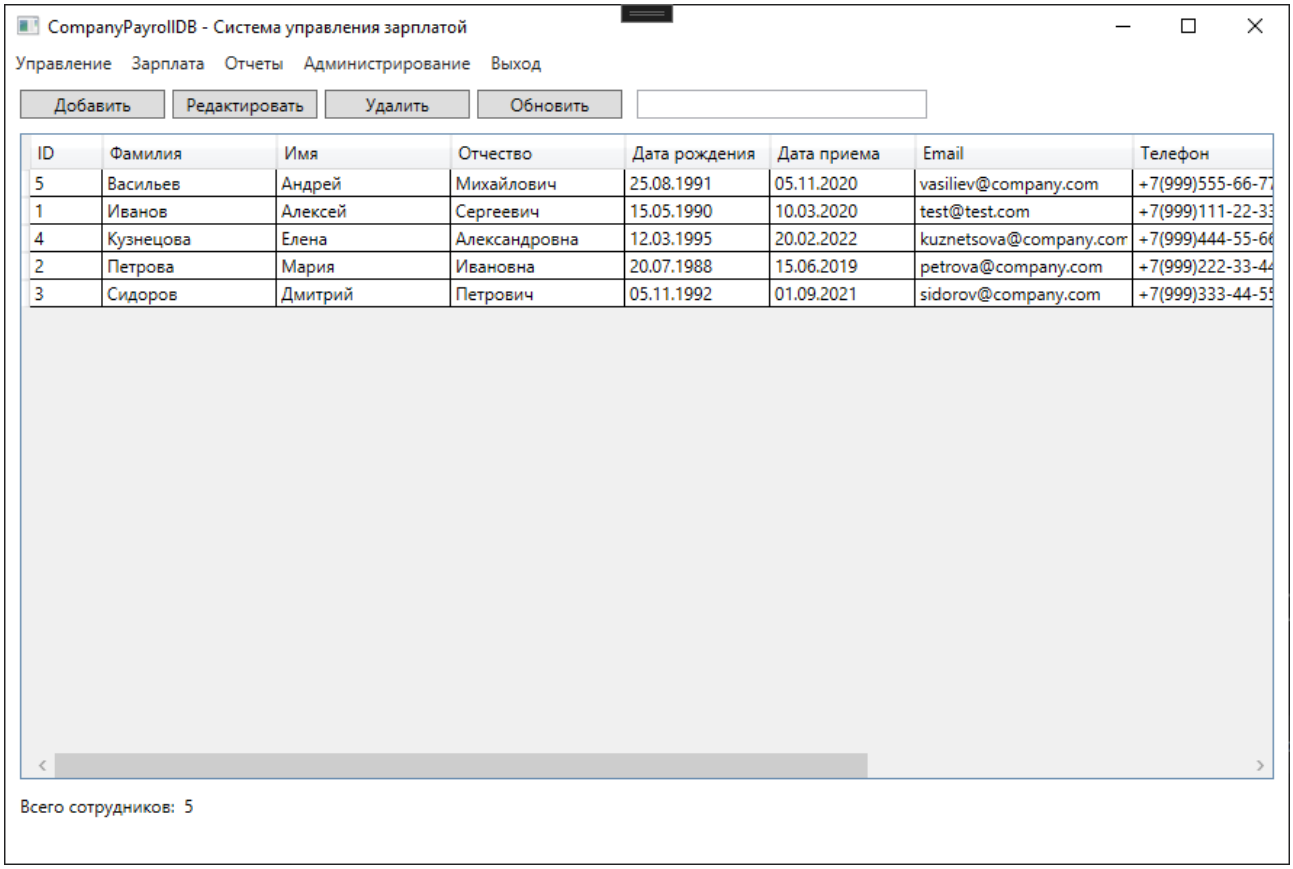
Назначение: CRUD-операции с сотрудниками

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows;
using System.Windows.Controls;

namespace CompanyPayrollApp.Pages
{
    public partial class EmployeesPage : Page
    {
        private SqlConnection connection;

        public EmployeesPage()
        {
            InitializeComponent();
            connection = new SqlConnection(App.ConnectionString);
            LoadEmployees();
        }

        private void LoadEmployees(string search = "")
        {
            try
            {
                if (connection.State != ConnectionState.Open)
                    connection.Open();

                string query = @"
                    SELECT e.*, d.name as department_name, p.title as position_title
                    FROM Employees e
                    LEFT JOIN Departments d ON e.department_id = d.id
                    LEFT JOIN Positions p ON e.position_id = p.id
```

```
                    WHERE (@search = '' OR e.last_name LIKE '%' + @search + '%'
                           OR e.first_name LIKE '%' + @search + '%')
                    ORDER BY e.last_name, e.first_name";

                SqlCommand cmd = new SqlCommand(query, connection);
                cmd.Parameters.AddWithValue("@search", search);

                SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                adapter.Fill(dt);

                EmployeesGrid.ItemsSource = dt.DefaultView;
                TotalCount.Text = dt.Rows.Count.ToString();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки сотрудников: {ex.Message}",
"Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                if (connection.State == ConnectionState.Open)
                    connection.Close();
            }
        }

        private void AddButton_Click(object sender, RoutedEventArgs e)
        {
            var window = new Windows.EmployeeEditWindow();
            window.Closed += (s, args) => LoadEmployees();
            window.ShowDialog();
        }

        private void EditButton_Click(object sender, RoutedEventArgs e)
        {
            if (EmployeesGrid.SelectedItem == null)
            {
                MessageBox.Show("Выберите сотрудника для редактирования", "Внимание",
                    MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            DataRowView row = (DataRowView)EmployeesGrid.SelectedItem;
            int employeeId = Convert.ToInt32(row["id"]);

            var window = new Windows.EmployeeEditWindow(employeeId);
            window.Closed += (s, args) => LoadEmployees();
            window.ShowDialog();
        }

        private void DeleteButton_Click(object sender, RoutedEventArgs e)
        {
            if (EmployeesGrid.SelectedItem == null)
            {
                MessageBox.Show("Выберите сотрудника для удаления", "Внимание",
                    MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            DataRowView row = (DataRowView)EmployeesGrid.SelectedItem;
            string name = $"{row["last_name"]} {row["first_name"]}";
            int employeeId = Convert.ToInt32(row["id"]);

            MessageBoxResult result = MessageBox.Show(
```

```
                $"Вы действительно хотите удалить сотрудника:\n{name}?",
                "Подтверждение удаления",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (result == MessageBoxResult.Yes)
            {
                try
                {
                    if (connection.State != ConnectionState.Open)
                        connection.Open();

                    string checkQuery = "SELECT COUNT(*) FROM Accruals WHERE em-
ployee_id = @id";
                    SqlCommand checkCmd = new SqlCommand(checkQuery, connection);
                    checkCmd.Parameters.AddWithValue("@id", employeeId);
                    int relatedCount = Convert.ToInt32(checkCmd.ExecuteScalar());

                    if (relatedCount > 0)
                    {
                        MessageBox.Show("Нельзя удалить сотрудника, так как есть свя-
занные начисления зарплаты.\nСначала удалите все начисления сотрудника.",
                            "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
                        return;
                    }

                    string deleteQuery = "DELETE FROM Employees WHERE id = @id";
                    SqlCommand deleteCmd = new SqlCommand(deleteQuery, connection);
                    deleteCmd.Parameters.AddWithValue("@id", employeeId);
                    int rowsAffected = deleteCmd.ExecuteNonQuery();

                    if (rowsAffected > 0)
                    {
                        MessageBox.Show("Сотрудник успешно удален", "Успешно",
                            MessageBoxButton.OK, MessageBoxImage.Information);
                        LoadEmployees();
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Ошибка удаления сотрудника: {ex.Message}",
"Ошибка",
                        MessageBoxButton.OK, MessageBoxImage.Error);
                }
                finally
                {
                    if (connection.State == ConnectionState.Open)
                        connection.Close();
                }
            }
        }

        private void RefreshButton_Click(object sender, RoutedEventArgs e)
        {
            SearchBox.Text = "";
            LoadEmployees();
        }

        private void SearchBox_TextChanged(object sender, TextChangedEventArgs e)
        {
            LoadEmployees(SearchBox.Text);
        }

        private void EmployeesGrid_SelectionChanged(object sender, Selection-
ChangedEventArgs e)
```

```
        {
        }

        private void Page_Unloaded(object sender, RoutedEventArgs e)
        {
            connection?.Dispose();
        }
    }
}
```



Модуль 4: Расчет заработной платы (CalculatePayrollWindow.xaml.cs)

Назначение: Расчет зарплаты с учетом отработанных дней и премий

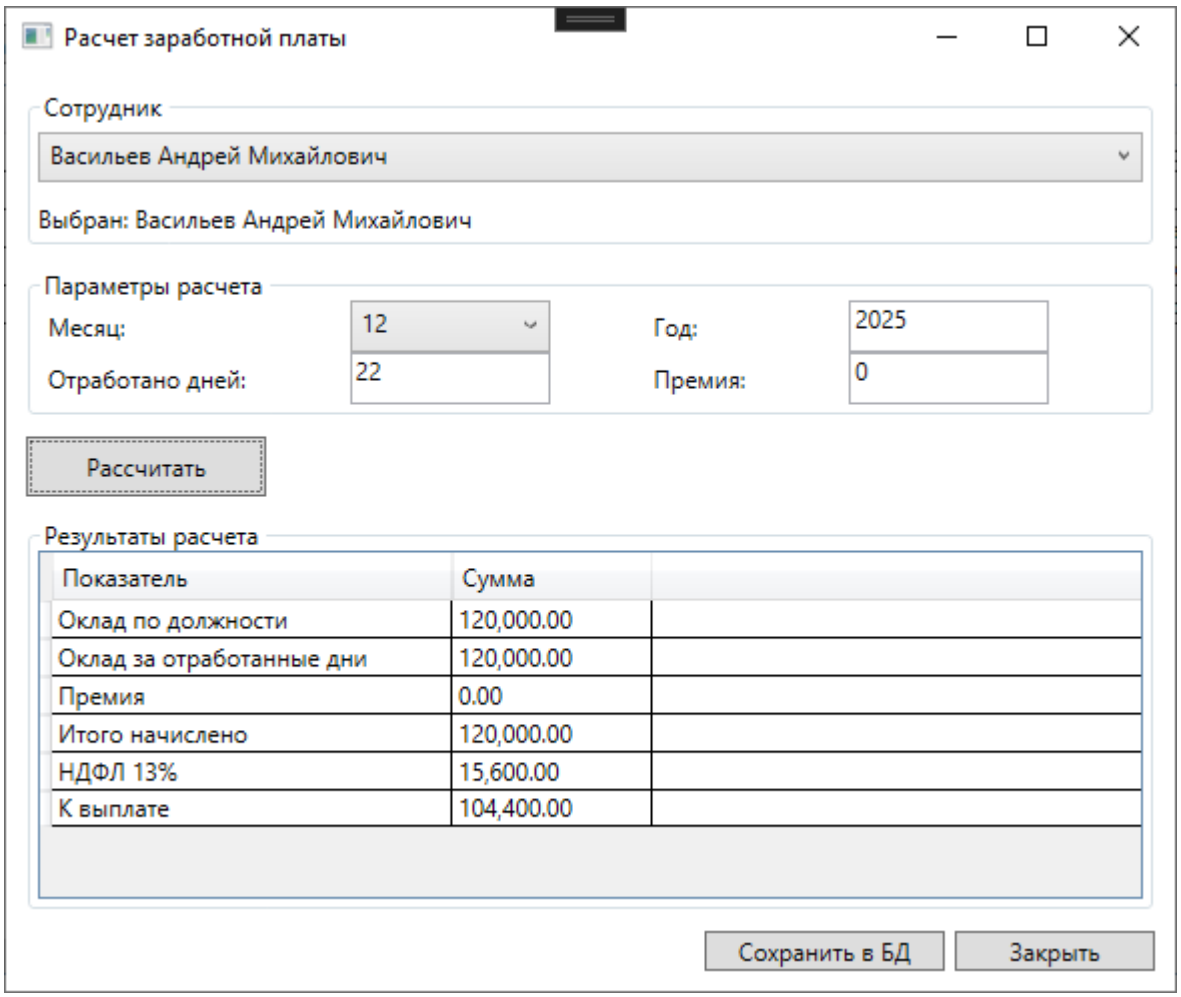
```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Windows;
using System.Windows.Controls;

namespace CompanyPayrollApp.Windows
{
    public partial class CalculatePayrollWindow : Window
    {
        private SqlConnection connection;
        private dynamic calculationResult;

        public class CalculationResult
        {
            public string Item { get; set; }
            public decimal Amount { get; set; }
            public string Note { get; set; }
        }

        public CalculatePayrollWindow()
```

```
        {
            InitializeComponent();
            connection = new SqlConnection(App.ConnectionString);

            InitializeComboBoxes();
            LoadEmployees();
        }

        private void InitializeComboBoxes()
        {
            for (int i = 1; i <= 12; i++)
            {
                MonthCombo.Items.Add(i.ToString());
            }
            MonthCombo.SelectedIndex = DateTime.Now.Month - 1;

            YearBox.Text = DateTime.Now.Year.ToString();
        }

        private void LoadEmployees()
        {
            try
            {
                connection.Open();
                string query = @"
            SELECT e.id,
                   e.last_name + ' ' + e.first_name +
                   CASE WHEN e.middle_name IS NOT NULL AND e.middle_name != ''
                        THEN ' ' + e.middle_name ELSE '' END as full_name,
                   p.salary as position_salary
            FROM Employees e
            LEFT JOIN Positions p ON e.position_id = p.id
            WHERE e.status = 'Активен'
            ORDER BY e.last_name";

                SqlCommand cmd = new SqlCommand(query, connection);
                var reader = cmd.ExecuteReader();

                EmployeeCombo.Items.Clear();
                while (reader.Read())
                {
                    EmployeeCombo.Items.Add(new
                    {
                        id = reader["id"],
                        full_name = reader["full_name"].ToString(),
                        position_salary = reader["position_salary"] != DBNull.Value ?
                            Convert.ToDecimal(reader["position_salary"]) : 0
                    });
                }

                if (EmployeeCombo.Items.Count > 0)
                {
                    EmployeeCombo.SelectedIndex = 0;
                    dynamic selected = EmployeeCombo.SelectedItem;
                    EmployeeInfo.Text = $"Выбран: {selected.full_name}";
                }
                else
                {
                    EmployeeInfo.Text = "Нет активных сотрудников";
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки сотрудников: {ex.Message}",
"Ошибка");
```

```
            }
            finally
            {
                connection.Close();
            }
        }

        private void Calculate_Click(object sender, RoutedEventArgs e)
        {
            if (EmployeeCombo.SelectedItem == null)
            {
                MessageBox.Show("Выберите сотрудника", "Ошибка");
                return;
            }

            dynamic selectedEmployee = EmployeeCombo.SelectedItem;
            int employeeId = selectedEmployee.id;
            decimal positionSalary = selectedEmployee.position_salary;
            string employeeName = selectedEmployee.full_name;

            if (!int.TryParse(MonthCombo.Text, out int month) ||
                !int.TryParse(YearBox.Text, out int year) ||
                !int.TryParse(WorkDaysBox.Text, out int workDays) ||
                !decimal.TryParse(BonusBox.Text, out decimal bonus))
            {
                MessageBox.Show("Проверьте правильность ввода данных", "Ошибка");
                return;
            }

            if (workDays <= 0 || workDays > 31)
            {
                MessageBox.Show("Количество рабочих дней должно быть от 1 до 31",
"Ошибка");
                return;
            }

            try
            {
                decimal baseSalary = (positionSalary / 22) * workDays;
                decimal totalGross = baseSalary + bonus;
                decimal tax = totalGross * 0.13m;
                decimal netSalary = totalGross - tax;

                var results = new List<CalculationResult>
                {
                    new CalculationResult { Item = "Оклад по должности", Amount = po-
sitionSalary },
                    new CalculationResult { Item = "Оклад за отработанные дни", Amount
= baseSalary },
                    new CalculationResult { Item = "Премия", Amount = bonus },
                    new CalculationResult { Item = "Итого начислено", Amount = total-
Gross },
                    new CalculationResult { Item = "НДФЛ 13%", Amount = tax },
                    new CalculationResult { Item = "К выплате", Amount = netSalary }
                };

                CalculationGrid.ItemsSource = results;

                calculationResult = new
                {
                    EmployeeId = employeeId,
                    EmployeeName = employeeName,
                    Month = month,
                    Year = year,
                    WorkDays = workDays,
```

```
                    BaseSalary = baseSalary,
                    Bonus = bonus,
                    TotalGross = totalGross,
                    Tax = tax,
                    NetSalary = netSalary
                };

                var saveButton = FindName("SaveButton") as Button;
                if (saveButton != null)
                    saveButton.IsEnabled = true;

                MessageBox.Show("Расчет выполнен успешно!", "Результат",
                    MessageBoxButton.OK, MessageBoxImage.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка расчета: {ex.Message}", "Ошибка");
            }
        }

        private void Save_Click(object sender, RoutedEventArgs e)
        {
            if (calculationResult == null)
            {
                MessageBox.Show("Сначала выполните расчет", "Ошибка");
                return;
            }

            try
            {
                connection.Open();

                string accrualQuery = @"
                    INSERT INTO Accruals
                    (employee_id, month, year, work_days, base_salary, bonus, total)
                    VALUES (@employeeId, @month, @year, @workDays, @baseSalary, @bo-
nus, @total);
                    SELECT SCOPE_IDENTITY();";

                SqlCommand accrualCmd = new SqlCommand(accrualQuery, connection);
                accrualCmd.Parameters.AddWithValue("@employeeId", calculationRe-
sult.EmployeeId);
                accrualCmd.Parameters.AddWithValue("@month", calculationResult.Month);
                accrualCmd.Parameters.AddWithValue("@year", calculationResult.Year);
                accrualCmd.Parameters.AddWithValue("@workDays", calculationRe-
sult.WorkDays);
                accrualCmd.Parameters.AddWithValue("@baseSalary", calculationRe-
sult.BaseSalary);
                accrualCmd.Parameters.AddWithValue("@bonus", calculationResult.Bonus);
                accrualCmd.Parameters.AddWithValue("@total", calculationResult.Total-
Gross);

                int accrualId = Convert.ToInt32(accrualCmd.ExecuteScalar());

                string deductionQuery = @"
                    INSERT INTO Deductions
                    (accrual_id, type, amount, description)
                    VALUES (@accrualId, 'НДФЛ 13%', @taxAmount, 'Подоходный налог')";

                SqlCommand deductionCmd = new SqlCommand(deductionQuery, connection);
                deductionCmd.Parameters.AddWithValue("@accrualId", accrualId);
                deductionCmd.Parameters.AddWithValue("@taxAmount", calculationRe-
sult.Tax);
                deductionCmd.ExecuteNonQuery();
```

```
                MessageBox.Show($"Начисление сохранено!\n\n" +
                               $"ID начисления: {accrualId}\n" +
                               $"Сотрудник: {calculationResult.EmployeeName}\n" +
                               $"Период: {calculationResult.Month}.{calculationRe-
sult.Year}\n" +
                               $"Сумма к выплате: {calculationResult.NetSalary:N2}
руб.",
                    "Сохранено", MessageBoxButton.OK, MessageBoxImage.Information);

                var saveButton = FindName("SaveButton") as Button;
                if (saveButton != null)
                    saveButton.IsEnabled = false;

                calculationResult = null;

                CalculationGrid.ItemsSource = null;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка");
            }
            finally
            {
                connection.Close();
            }
        }

        private void Close_Click(object sender, RoutedEventArgs e)
        {
            this.Close();
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            connection?.Dispose();
        }
        private void EmployeeCombo_SelectionChanged(object sender, Selection-
ChangedEventArgs e)
        {
            if (EmployeeCombo.SelectedItem != null)
            {
                dynamic selected = EmployeeCombo.SelectedItem;
                EmployeeInfo.Text = $"Выбран: {selected.full_name}";
            }
        }
    }
}
```



Модуль 5: Управление выплатами (PaymentsWindow.xaml.cs)

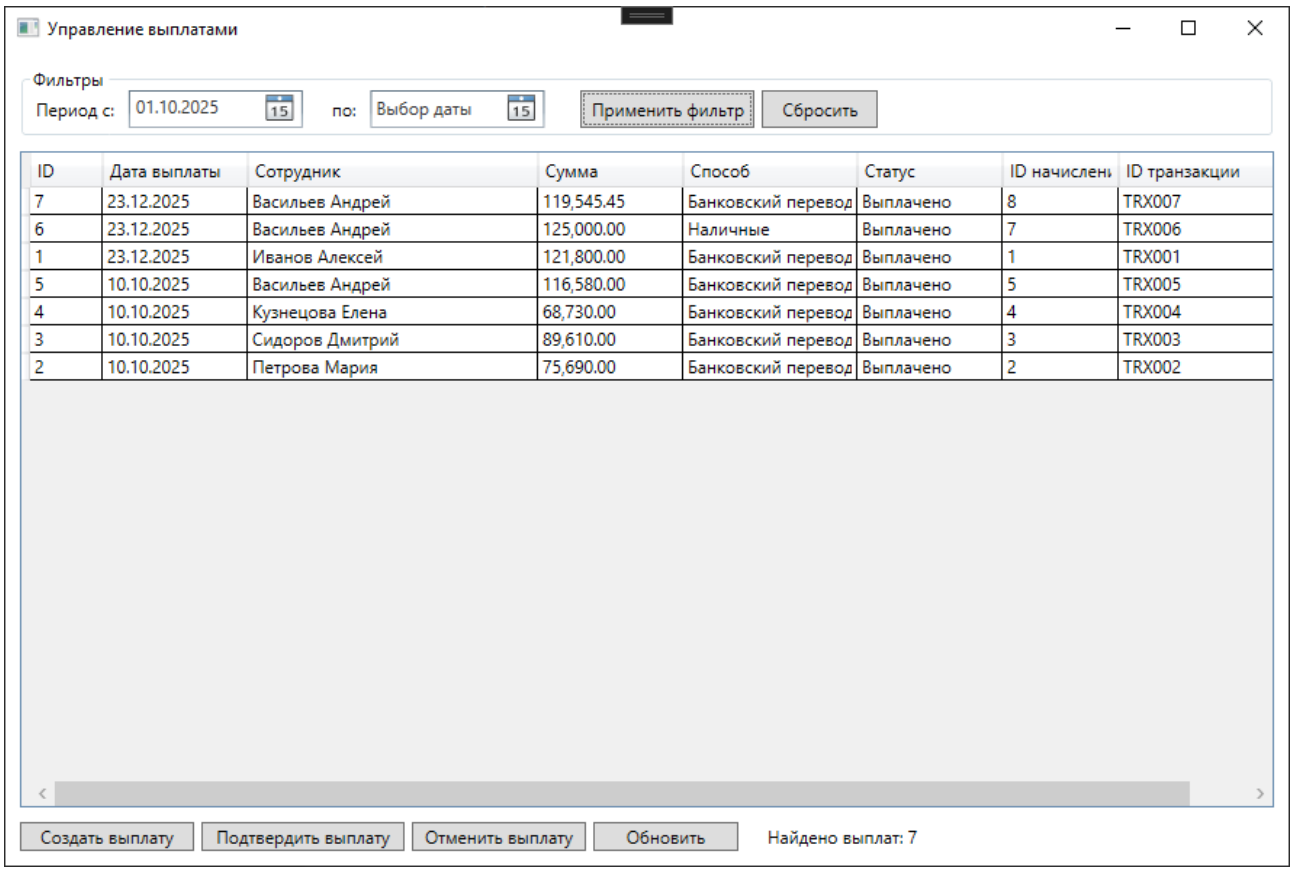
Назначение: Создание, подтверждение и отмена выплат зарплаты

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows;

namespace CompanyPayrollApp.Windows
{
    public partial class PaymentsWindow : Window
    {
        public PaymentsWindow()
        {
            InitializeComponent();
            LoadPayments();
        }

        private void LoadPayments()
        {
            using (SqlConnection connection = new SqlConnection(App.ConnectionString))
            {
                try
                {
                    connection.Open();

                    string query = @"
                        SELECT p.*,
                               e.last_name + ' ' + e.first_name as employee_name
```

```
                        FROM Payments p
                        LEFT JOIN Accruals a ON p.accrual_id = a.id
                        LEFT JOIN Employees e ON a.employee_id = e.id
                        WHERE (@dateFrom IS NULL OR p.payment_date >= @dateFrom)
                          AND (@dateTo IS NULL OR p.payment_date <= @dateTo)
                        ORDER BY p.payment_date DESC, p.id DESC";

                    SqlCommand cmd = new SqlCommand(query, connection);

                    if (DateFromPicker.SelectedDate.HasValue)
                        cmd.Parameters.AddWithValue("@dateFrom", DateFromPicker.Se-
lectedDate.Value);
                    else
                        cmd.Parameters.AddWithValue("@dateFrom", DBNull.Value);

                    if (DateToPicker.SelectedDate.HasValue)
                        cmd.Parameters.AddWithValue("@dateTo", DateToPicker.Selected-
Date.Value);
                    else
                        cmd.Parameters.AddWithValue("@dateTo", DBNull.Value);

                    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                    DataTable dt = new DataTable();
                    adapter.Fill(dt);

                    PaymentsGrid.ItemsSource = dt.DefaultView;
                    StatusText.Text = $"Найдено выплат: {dt.Rows.Count}";
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Ошибка загрузки выплат: {ex.Message}", "Ошибка",
                        MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
        }

        private void CreatePaymentButton_Click(object sender, RoutedEventArgs e)
        {
            var window = new CreatePaymentWindow();
            window.Closed += (s, args) => LoadPayments();
            window.ShowDialog();
        }

        private void ConfirmPaymentButton_Click(object sender, RoutedEventArgs e)
        {
            if (PaymentsGrid.SelectedItem == null)
            {
                MessageBox.Show("Выберите выплату для подтверждения", "Внимание",
                    MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            DataRowView row = (DataRowView)PaymentsGrid.SelectedItem;
            int paymentId = Convert.ToInt32(row["id"]);
            string currentStatus = row["status"].ToString();

            if (currentStatus == "Выплачено")
            {
                MessageBox.Show("Выплата уже подтверждена", "Информация",
                    MessageBoxButton.OK, MessageBoxImage.Information);
                return;
            }

            MessageBoxResult result = MessageBox.Show(
                "Подтвердить выплату?\nСтатус будет изменен на 'Выплачено'.",
```

```
                "Подтверждение выплаты",
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (result == MessageBoxResult.Yes)
            {
                using (SqlConnection connection = new SqlConnection(App.Connection-
String))
                {
                    try
                    {
                        connection.Open();

                        string query = @"
                            UPDATE Payments
                            SET status = 'Выплачено',
                                payment_date = @paymentDate
                            WHERE id = @id";

                        SqlCommand cmd = new SqlCommand(query, connection);
                        cmd.Parameters.AddWithValue("@id", paymentId);
                        cmd.Parameters.AddWithValue("@paymentDate", DateTime.Now);

                        int rowsAffected = cmd.ExecuteNonQuery();

                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("Выплата подтверждена", "Успешно",
                                MessageBoxButton.OK, MessageBoxImage.Information);
                            LoadPayments();
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show($"Ошибка подтверждения: {ex.Message}",
"Ошибка",
                            MessageBoxButton.OK, MessageBoxImage.Error);
                    }
                }
            }
        }

        private void CancelPaymentButton_Click(object sender, RoutedEventArgs e)
        {
            if (PaymentsGrid.SelectedItem == null)
            {
                MessageBox.Show("Выберите выплату для отмены", "Внимание",
                    MessageBoxButton.OK, MessageBoxImage.Warning);
                return;
            }

            DataRowView row = (DataRowView)PaymentsGrid.SelectedItem;
            int paymentId = Convert.ToInt32(row["id"]);
            string currentStatus = row["status"].ToString();

            if (currentStatus == "Отменена")
            {
                MessageBox.Show("Выплата уже отменена", "Информация",
                    MessageBoxButton.OK, MessageBoxImage.Information);
                return;
            }

            MessageBoxResult result = MessageBox.Show(
                "Отменить выплату?\nСтатус будет изменен на 'Отменена'.",
                "Отмена выплаты",
```

```
                MessageBoxButton.YesNo,
                MessageBoxImage.Question);

            if (result == MessageBoxResult.Yes)
            {
                using (SqlConnection connection = new SqlConnection(App.Connection-
String))
                {
                    try
                    {
                        connection.Open();

                        string query = "UPDATE Payments SET status = 'Отменена' WHERE
id = @id";
                        SqlCommand cmd = new SqlCommand(query, connection);
                        cmd.Parameters.AddWithValue("@id", paymentId);

                        int rowsAffected = cmd.ExecuteNonQuery();

                        if (rowsAffected > 0)
                        {
                            MessageBox.Show("Выплата отменена", "Успешно",
                                MessageBoxButton.OK, MessageBoxImage.Information);
                            LoadPayments();
                        }
                    }
                    catch (Exception ex)
                    {
                        MessageBox.Show($"Ошибка отмены: {ex.Message}", "Ошибка",
                            MessageBoxButton.OK, MessageBoxImage.Error);
                    }
                }
            }
        }

        private void ApplyFilter_Click(object sender, RoutedEventArgs e)
        {
            LoadPayments();
        }

        private void ResetFilter_Click(object sender, RoutedEventArgs e)
        {
            DateFromPicker.SelectedDate = null;
            DateToPicker.SelectedDate = null;
            LoadPayments();
        }

        private void RefreshButton_Click(object sender, RoutedEventArgs e)
        {
            LoadPayments();
        }

        private void PaymentsGrid_SelectionChanged(object sender, System.Windows.Con-
trols.SelectionChangedEventArgs e)
        {
        }
    }
}
```



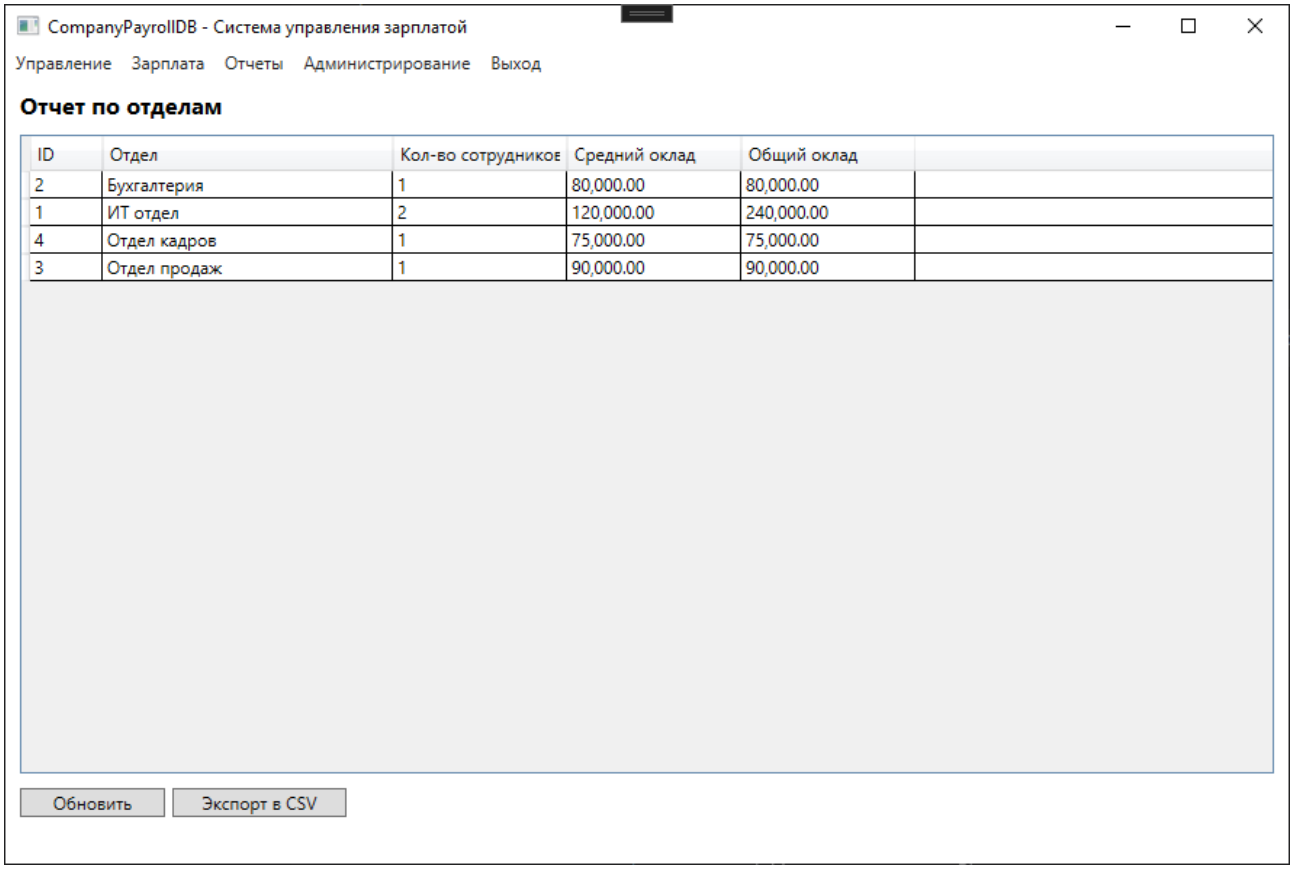
Модуль 6: Отчет по отделам (ReportDepartmentsPage.xaml.cs)

Назначение: Формирование отчета по отделам с использованием представления vDepartmentReport

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows;
using System.Windows.Controls;
using Microsoft.Win32;
using System.IO;

namespace CompanyPayrollApp.Pages
{
    public partial class ReportDepartmentsPage : Page
    {
        private SqlConnection connection;

        public ReportDepartmentsPage()
        {
            InitializeComponent();
            connection = new SqlConnection(App.ConnectionString);
            LoadReport();
        }

        private void LoadReport()
        {
            try
            {
                connection.Open();

                string query = "SELECT * FROM vDepartmentReport ORDER BY depart-
ment_name";
```

```
                SqlCommand cmd = new SqlCommand(query, connection);
                SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                adapter.Fill(dt);

                ReportGrid.ItemsSource = dt.DefaultView;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки отчета: {ex.Message}", "Ошибка",
                    MessageBoxButton.OK, MessageBoxImage.Error);
            }
            finally
            {
                connection.Close();
            }
        }

        private void RefreshButton_Click(object sender, RoutedEventArgs e)
        {
            LoadReport();
        }

        private void ExportButton_Click(object sender, RoutedEventArgs e)
        {
            SaveFileDialog saveDialog = new SaveFileDialog
            {
                Filter = "CSV файлы (*.csv)|*.csv|Все файлы (*.*)|*.*",
                FileName = $"Отчет_отделы_{DateTime.Now:yyyyMMdd}.csv"
            };

            if (saveDialog.ShowDialog() == true)
            {
                try
                {
                    using (StreamWriter writer = new StreamWriter(saveDialog.FileName,
false, System.Text.Encoding.UTF8))
                    {
                        writer.WriteLine("ID;Отдел;Кол-во сотрудников;Средний
оклад;Общий оклад");

                        connection.Open();
                        string query = "SELECT * FROM vDepartmentReport ORDER BY de-
partment_name";
                        SqlCommand cmd = new SqlCommand(query, connection);

                        using (SqlDataReader reader = cmd.ExecuteReader())
                        {
                            while (reader.Read())
                            {
                                writer.WriteLine(
                                    $"{reader["id"]};" +
                                    $"{reader["department_name"]};" +
                                    $"{reader["employee_count"]};" +
                                    $"{reader["avg_salary"]:N2};" +
                                    $"{reader["total_salary"]:N2}");
                            }
                        }
                    }

                    MessageBox.Show($"Отчет сохранен: {saveDialog.FileName}",
"Успешно",
                        MessageBoxButton.OK, MessageBoxImage.Information);
                }
```

```
                catch (Exception ex)
                {
                    MessageBox.Show($"Ошибка экспорта: {ex.Message}", "Ошибка",
                        MessageBoxButton.OK, MessageBoxImage.Error);
                }
                finally
                {
                    connection.Close();
                }
            }
        }

        private void Page_Unloaded(object sender, RoutedEventArgs e)
        {
            connection?.Dispose();
        }
    }
}
```



Модуль 7: Журнал аудита (AuditLogWindow.xaml.cs)

Назначение: Просмотр журнала изменений сотрудников

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Windows;
using System.Windows.Controls;

namespace CompanyPayrollApp.Windows
{
    public partial class AuditLogWindow : Window
    {
        private SqlConnection connection;
```

```
        public AuditLogWindow()
        {
            InitializeComponent();
            connection = new SqlConnection(App.ConnectionString);
            LoadAuditLog();
        }

        private void LoadAuditLog()
        {
            try
            {
                if (connection.State != ConnectionState.Open)
                    connection.Open();

                string query = @"
            SELECT * FROM AuditLog
            WHERE (@action = '' OR action = @action)
              AND (@dateFrom IS NULL OR change_date >= @dateFrom)
              AND (@dateTo IS NULL OR change_date <= @dateTo)";

                SqlCommand cmd = new SqlCommand(query, connection);

                ComboBoxItem actionItem = ActionFilterCombo.SelectedItem as ComboBox-
Item;
                string action = (actionItem != null && actionItem.Con-
tent.ToString() != "Все") ?
                    actionItem.Content.ToString() : "";
                cmd.Parameters.AddWithValue("@action", action);

                if (DateFromPicker.SelectedDate.HasValue)
                    cmd.Parameters.AddWithValue("@dateFrom", DateFromPicker.Selected-
Date.Value);
                else
                    cmd.Parameters.AddWithValue("@dateFrom", DBNull.Value);

                if (DateToPicker.SelectedDate.HasValue)
                    cmd.Parameters.AddWithValue("@dateTo", DateToPicker.Selected-
Date.Value.AddDays(1));
                else
                    cmd.Parameters.AddWithValue("@dateTo", DBNull.Value);

                SqlDataAdapter adapter = new SqlDataAdapter(cmd);
                DataTable dt = new DataTable();
                adapter.Fill(dt);

                dt.DefaultView.Sort = "change_date DESC";

                AuditGrid.ItemsSource = dt.DefaultView;
                StatusText.Text = $"Загружено записей: {dt.Rows.Count}";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки журнала: {ex.Message}", "Ошибка");
            }
            finally
            {
                if (connection.State == ConnectionState.Open)
                    connection.Close();
            }
        }

        private void ApplyFilter_Click(object sender, RoutedEventArgs e)
        {
            LoadAuditLog();
        }
```

```
        private void ResetFilter_Click(object sender, RoutedEventArgs e)
        {
            ActionFilterCombo.SelectedIndex = 0;
            DateFromPicker.SelectedDate = null;
            DateToPicker.SelectedDate = null;
            LoadAuditLog();
        }

        private void Window_Closed(object sender, EventArgs e)
        {
            connection?.Dispose();
        }
    }
}
```