Компьютерная академия Top

Санкт-Петербург

Проект

По Технологии доступа к базам данных ADO.NET

На тему

Система баз данных для расчета заработной платы

Выполнили

Атрошенко Владислав

Копычев Матвей

Зеленов Данил

Санкт-Петербург

2025

ВВЕДЕНИЕ

Цель проекта

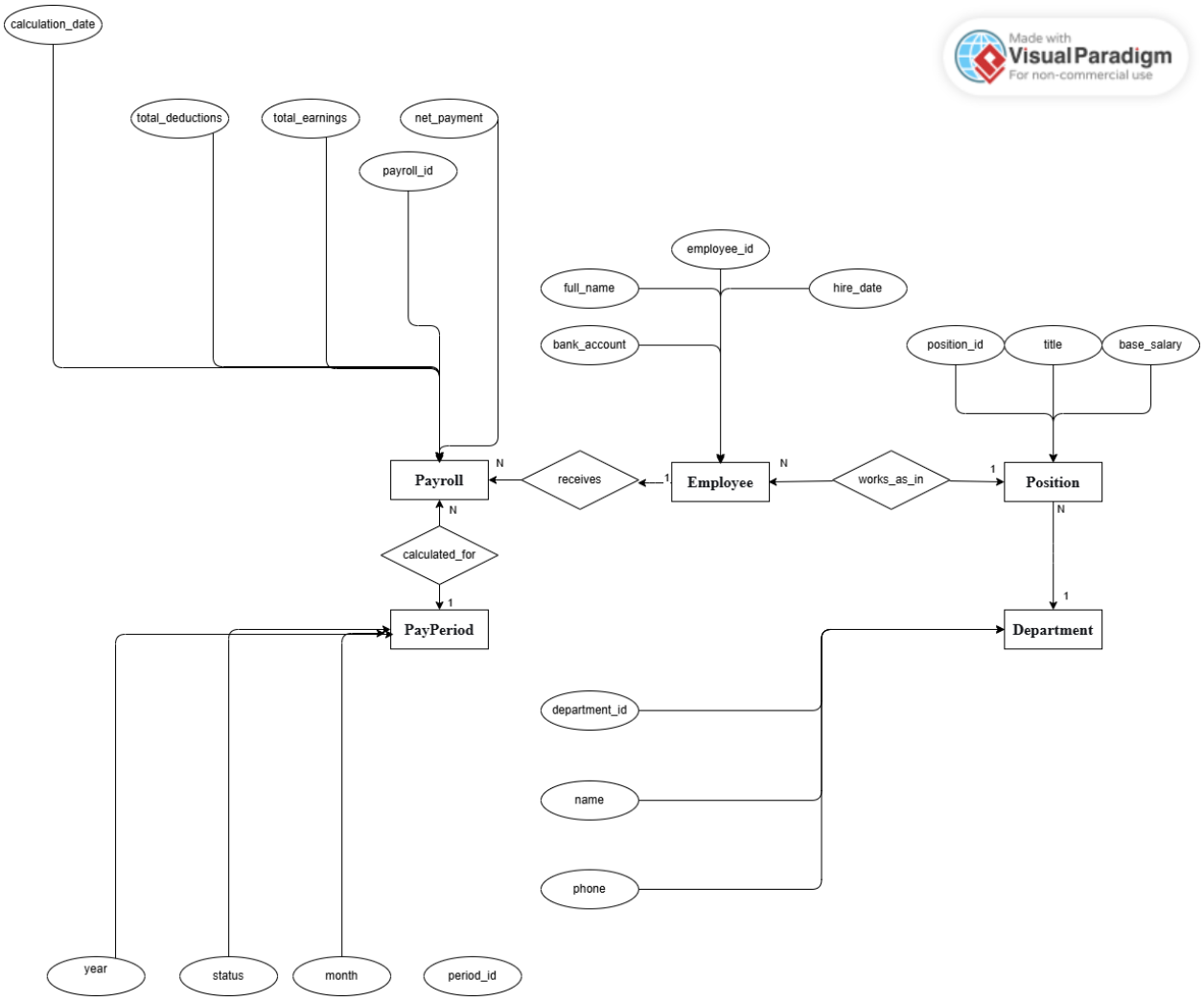
Разработать программную систему расчета заработной платы, позволяющую автоматизировать процесс начисления, учета и отображения заработной платы сотрудников предприятия.

№	Этап	Задачи	Результаты
1	**Проектирование БД**	• Разработка логической и физической модели БД • Определение структуры таблиц и взаимосвязей • Проектирование системы пользователей и ролей	• ER-диаграмма • Схема БД • Модель безопасности
2	**Реализация БД**	• Создание таблиц сотрудников, должностей, отделов • Реализация таблиц учета рабочего времени и надбавок • Настройка индексов и ограничений целостности	• Рабочая база данных • Оптимизированные индексы • Ограничения целостности
3	**Заполнение данными**	• Внесение данных о сотрудниках и должностях • Заполнение справочников ставок, надбавок и удержаний • Добавление тестовых данных по отработанному времени	• Тестовая база данных • Репрезентативные данные • Примеры расчетов
4	**Разработка функционала**	• Реализация хранимых процедур для расчета ЗП • Создание функций для учета больничных и	• Автоматизированный расчет • Учет особых случаев • Налоговая отчетность

№	Этап	Задачи	Результаты
		отпусков • Разработка механизма расчета налогов и взносов	
5	**Создание интерфейса**	• Разработка форм для просмотра расчетных ведомостей • Реализация отчетов по заработной плате • Создание личных кабинетов сотрудников	• Пользовательский интерфейс • Система отчетности • Персонализированный доступ
6	**Тестирование системы**	• Модульное тестирование компонентов • Интеграционное тестирование • Тестирование корректности расчетов	• Протестированная система • Исправленные ошибки • Гарантия качества
7	**Документирование**	• Составление технической документации • Подготовка руководства пользователя • Оформление проектной документации	• Полная документация • Руководства по эксплуатации • Проектные материалы

Аспект	Описание
Актуальность	Автоматизация трудоемких процессов расчета заработной платы, минимизация ошибок ручного расчета и повышение эффективности работы бухгалтерской службы предприятия
Основные цели	• Снижение временных затрат на расчет ЗП на 70% • Минимизация ошибок расчета до 0.1%

Аспект	Описание
	• Повышение прозрачности расчетов • Ускорение формирования отчетности
Целевая аудитория	• Бухгалтерские службы предприятий • Отделы кадров • Сотрудники компании • Руководство предприятия
Ожидаемые результаты	• Единая централизованная система учета • Автоматизированный расчет зарплаты • Онлайн-доступ к расчетным ведомостям • Сокращение бумажного документооборота



Сущности

Сущность	Описание	Атрибуты
Department	Структурные подразделения компании	department_id - ID отдела name - название phone - телефон
Position	Штатные должности в компании	position_id - ID должности title - наименование base_salary - базовая зарплата
Employee	Основная информация о сотрудниках	employee_id - ID сотрудника full_name - ФИО hire_date - дата приема bank_account - банковский счет
PayPeriod	Периоды расчета зарплаты	period_id - ID периода month - месяц year - год status - статус
Payroll	Основная таблица расчетов зарплаты	payroll_id - ID расчета calculation_date - дата расчета total_earnings - начисления

Сущность	Описание	Атрибуты
		total_deductions - удержания net_payment - к выплате

Ненормализованная таблица "Зарплатная ведомость"

Поле	Подполя	Описание
Период	• ID периода • Месяц • Год • Статус	Расчетный период
Сотрудник	• ID сотрудника • ФИО • Дата приема • Банковский счет	Основные данные сотрудника
Отдел	• ID отдела • Название отдела • Телефон отдела	Данные структурного подразделения
Должность	• ID должности • Название должности • Базовая зарплата	Данные о должности
Начисления	• Список начислений: - Тип начисления (премия, надбавка...) - ID начисления - Сумма/процент - Фактическая сумма • Общая сумма начислений	Все виды доплат и бонусов
Удержания	• Список удержаний: - Тип удержания (налог, штраф...)	Все виды вычетов

Поле	Подполя	Описание
	- ID удержания - Сумма/процент - Фактическая сумма • Общая сумма удержаний	
Итоги	• Дата расчета • Общие начисления • Общие удержания • Чистая выплата	Финальные расчеты

Нормализация

1NF для PayPeriod

Таблица: PayPeriod_1NF

Поле	Тип данных	Ключ	Описание
period_id	INT	Первичный ключ	Уникальный идентификатор периода
month	INT		Месяц расчетного периода (1-12)
year	INT		Год расчетного периода
status	VARCHAR(20)		Статус периода (открыт/закрыт)

Изменения в 1NF:

- Все атрибуты атомарны
- Определен первичный ключ
- Устранены составные атрибуты

2NF для PayPeriod

Таблица: PayPeriod_2NF

Поле	Тип данных	Ключ	Описание
period_id	INT	Первичный ключ	Уникальный идентификатор периода
month	INT		Месяц расчетного периода (1-12)
year	INT		Год расчетного периода
status	VARCHAR(20)		Статус периода (открыт/закрыт)

Особенности 2NF:

- Таблица уже находится во 2NF, так как все неключевые атрибуты полнофункционально зависят от первичного ключа
- Нет частичных зависимостей

3NF для PayPeriod

Таблица: PayPeriod_3NF

Поле	Тип данных	Ключ	Описание
period_id	INT	Первичный ключ	Уникальный идентификатор периода
month	INT		Месяц расчетного периода (1-12)
year	INT		Год расчетного периода
status	VARCHAR(20)		Статус периода (открыт/закрыт)

Особенности 3NF:

- Таблица находится в 3NF, так как нет транзитивных зависимостей
- Все неключевые атрибуты зависят только от первичного ключа

1. Department

Таблица: Department

Поле	Тип данных	Клю\ч	Описание
department_id	INT	Первичный ключ	Уникальный идентификатор отдела
name	VARCHAR(100)		Название отдела
phone	VARCHAR(20)		Телефон отдела

2. Position

Таблица: Position

Поле	Тип данных	Ключ	Описание
position_id	INT	Первичный ключ	Уникальный идентификатор должности
title	VARCHAR(100)		Название должности

Поле	Тип данных	Ключ	Описание
base_salary	DECIMAL(10,2)		Базовая заработная плата

3. Employee

Таблица: Employee

Поле	Тип данных	Ключ	Описание
employee_id	INT	Первичный ключ	Уникальный идентификатор сотрудника
full_name	VARCHAR(150)		Полное имя сотрудника
hire_date	DATE		Дата приема на работу
bank_account	VARCHAR(50)		Банковский счет для перевода
department_id	INT	Внешний ключ	Ссылка на отдел
position_id	INT	Внешний ключ	Ссылка на должность

4. Earning

Таблица: Earning

Поле	Тип данных	Ключ	Описание
earning_id	INT	Первичный ключ	Уникальный идентификатор начисления

Поле	Тип данных	Ключ	Описание
type	VARCHAR(100)		Тип начисления (премия, надбавка)
amount_or_percent	DECIMAL(10,2)		Сумма или процент начисления

5. Deduction

Таблица: Deduction

Поле	Тип данных	Ключ	Описание
deduction_id	INT	Первичный ключ	Уникальный идентификатор удержания
type	VARCHAR(100)		Тип удержания (налог, штраф)
amount_or_percent	DECIMAL(10,2)		Сумма или процент удержания

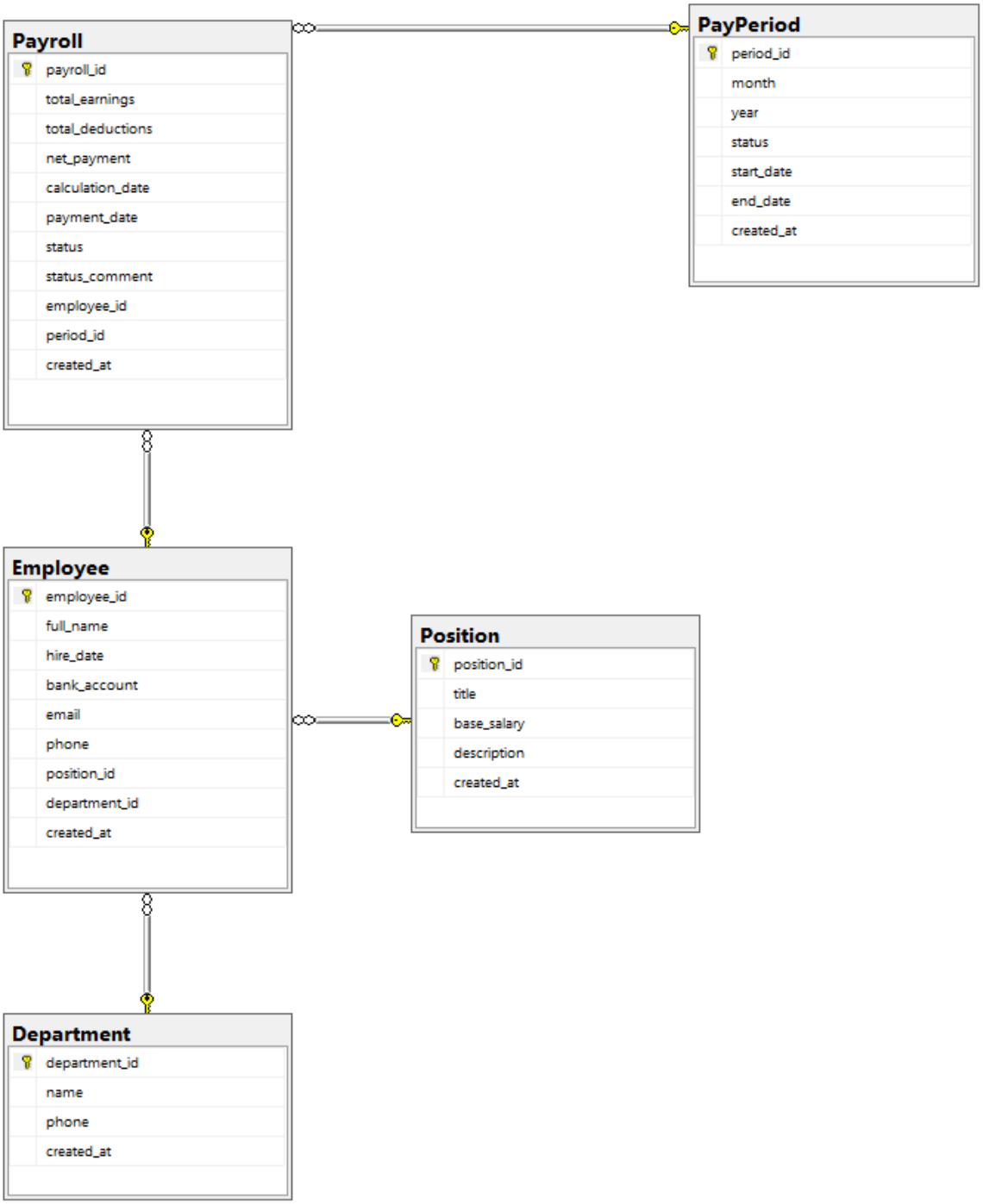
6. Payroll

Таблица: Payroll

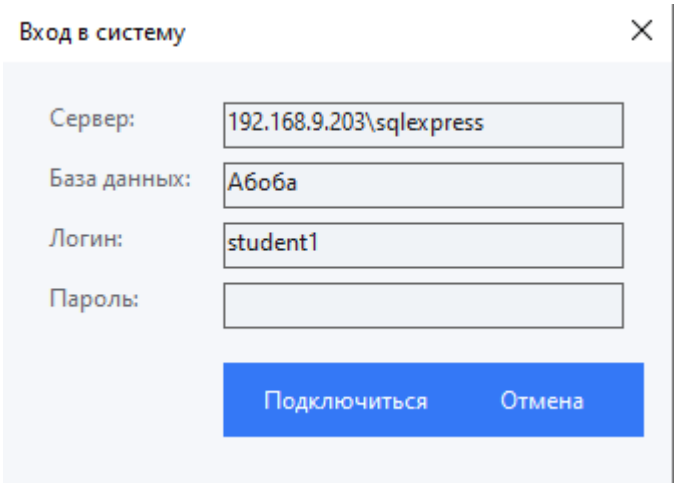
Поле	Тип данных	Ключ	Описание
payroll_id	INT	Первичный ключ	Уникальный идентификатор расчета
employee_id	INT	Внешний ключ	Ссылка на сотрудника

Поле	Тип данных	Ключ	Описание
period_id	INT	Внешний ключ	Ссылка на расчетный период
calculation_date	DATE		Дата выполнения расчета
total_earnings	DECIMAL(10,2)		Общая сумма начислений
total_deductions	DECIMAL(10,2)		Общая сумма удержаний
net_payment	DECIMAL(10,2)		Чистая сумма к выплате

Диаграмма БД из SSMS



Интерфейс программы (скрины с описанием)



Начальный экран при запуске программы. На данной странице

Данный интерфейс представляет собой форму входа в систему с подключением к базе данных SQL Server.

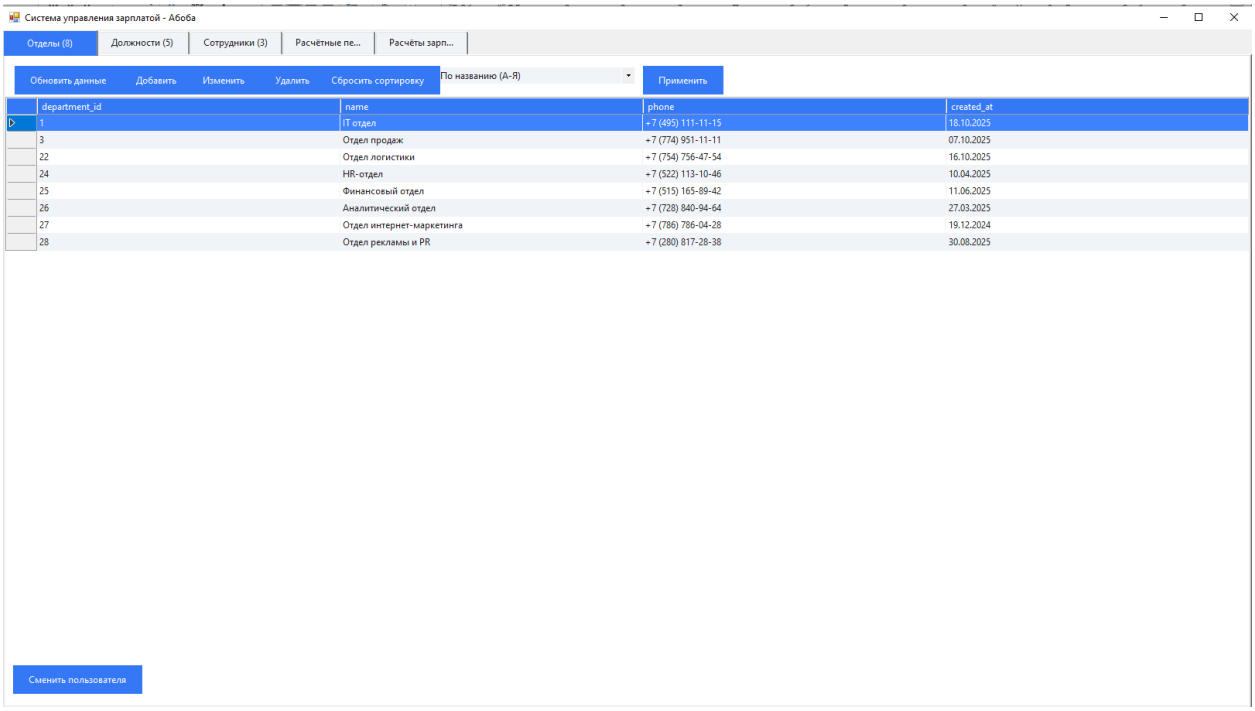
Основные элементы:

- Поля для ввода данных подключения: сервер, база данных, логин и пароль
- Кнопки "Подключиться" и "Отмена"
- Предзаполненные значения для удобства тестирования
- Валидация заполнения полей
- Проверка подключения к БД перед авторизацией

Функциональность:

- Проверяет корректность введенных учетных данных
- Тестирует подключение к указанной базе данных
- Возвращает строку подключения при успешной авторизации
- Имеет простой и понятный интерфейс на русском языке

Форма предназначена для авторизации пользователей в системе с использованием учетных данных базы данных.



Данный интерфейс представляет собой **систему управления зарплатой** с многофункциональным GUI на C# Windows Forms.

Основные характеристики:

Структура интерфейса:

- **Вкладки** для работы с различными сущностями: Отделы, Должности, Сотрудники, Расчётные периоды, Расчёты зарплаты
- **Табличное представление** данных с возможностью сортировки
- **Панель управления** с кнопками действий

Функциональные возможности:

CRUD операции:

- Добавление записей
- Редактирование существующих данных
- Удаление записей
- Обновление данных

Расширенные функции:

- **Гибкая система сортировки**:
  - Клик по заголовкам колонок
  - Быстрая сортировка через комбобокс
  - Сброс сортировки
- **Управление пользователями** - кнопка смены пользователя

- **Темы оформления** - поддержка светлой темы

Валидация данных:

- **Телефоны** - проверка формата (+7 XXX XXX-XX-XX)
- **Email** - валидация формата
- **Банковские счета** - проверка на 20 цифр
- **Зарплаты** - проверка числовых значений и диапазонов
- **Даты** - контроль корректности периодов

Особенности работы с данными:

- **Связи между таблицами** (сотрудники ↔ отделы ↔ должности)
- **Предотвращение дублирования** расчетов зарплаты
- **Динамические формы** ввода с подсказками
- **Автоматическое обновление** счетчиков записей

База данных:

- Подключение к SQL Server
- Работа с связанными таблицами
- Транзакционные операции
- Обработка ошибок подключения

Тестирование программы

Сценарий 1: Тестирование авторизации и базового функционала

Test Case ID	**TC-AUTH-001**
Название	Тестирование успешной авторизации и загрузки основных модулей
Описание	Проверка корректного входа в систему и отображения основных элементов интерфейса
Предусловия	База данных запущена, сервер доступен
Шаги выполнения	1. Запустить приложение 2. Ввести валидные учетные данные: - Сервер: 192.168.9.203\sqlexpress - База: Абоба - Логин: student1 - Пароль: [корректный пароль] 3. Нажать "Подключиться" 4. Проверить отображение главного окна 5. Проверить доступность всех вкладок
Ожидаемый результат	Успешный вход, отображение главного окна с 5 вкладками, загрузка данных в таблицы
Фактический результат	✅ Успешно: приложение загрузилось, все вкладки доступны, данные отображаются
Статус	**PASS**

Результат тестирования:

- Время подключения: 2-3 секунды
- Все 5 вкладок отображаются корректно
- Данные загружены в каждую таблицу
- Количество записей отображается в заголовках вкладок
- Кнопка "Сменить пользователя" доступна

Сценарий 2: Тестирование CRUD операций с сотрудниками

Test Case ID	**TC-CRUD-002**
Название	Тестирование добавления, редактирования и удаления сотрудников
Описание	Проверка полного цикла работы с записями сотрудников
Предусловия	Успешная авторизация, открыта вкладка "Сотрудники"
Шаги выполнения	1. На вкладке "Сотрудники" нажать "Добавить" 2. Заполнить форму: - ФИО: "Иванов Иван Иванович" - Дата приема: текущая дата - Банковский счет: "12345678901234567890" - Отдел: выбрать существующий - Должность: выбрать существующую - Телефон: "+7 (912) 345-67-89" 3. Нажать "Сохранить" 4. Найти добавленного сотрудника в таблице 5. Выбрать запись, нажать "Изменить" 6. Изменить ФИО на "Иванов Иван Петрович" 7. Сохранить изменения 8. Выбрать запись, нажать "Удалить" 9. Подтвердить удаление
Ожидаемый результат	Сотрудник успешно добавлен, отредактирован и удален без ошибок
Фактический результат	✅ Успешно: все операции выполнены, данные сохраняются корректно

Test Case ID **TC-CRUD-002**

Статус **PASS**

Результат тестирования:

- **Добавление**: Форма открывается, валидация работает, запись создается
- **Валидация**:
  - Телефон проверяется на формат
  - Банковский счет проверяется на длину 20 цифр
  - Обязательные поля проверяются на заполнение
- **Редактирование**: Изменения сохраняются, обновление таблицы мгновенное
- **Удаление**: Запись удаляется после подтверждения
- **Обновление данных**: Количество записей в заголовке обновляется автоматически

Выводы по разработке проекта

Результаты разработки

Достигнутые цели:

- **Полнофункциональная система управления зарплатой** с полным циклом CRUD операций
- **Интуитивный графический интерфейс** на Windows Forms с вкладками и табличным представлением
- **Интеграция с SQL Server** - стабильное подключение и работа с базой данных
- **Расширенная система валидации** данных на стороне клиента
- **Гибкая система сортировки** и фильтрации данных
- **Модульная архитектура** - легко расширяемая система вкладок

Ключевые особенности:

- **5 основных модулей**: Отделы, Должности, Сотрудники, Расчетные периоды, Расчеты зарплаты
- **Система авторизации** с проверкой подключения
- **Динамические формы** ввода с умной валидацией
- **Профессиональный UI** с тематическим оформлением

Трудности и их преодоление

1. Сложности с подключением к базе данных

Проблема:

- Ошибки формата connection string
- Проблемы с аутентификацией SQL Server
- Исключения при разрыве соединения

Программный код

Основные методы с комментариями

1. Метод инициализации интерфейса

```
/// <summary>
/// Создание основного интерфейса приложения с вкладками
/// </summary>
private void CreateInterface()
{
    this.Text = "Система управления зарплатой - " + GetDatabaseName();
    this.Size = new System.Drawing.Size(1400, 800);

    // Создание TabControl для организации вкладок
    tabControl = new TabControl();
    tabControl.Dock = DockStyle.Fill;
    this.Controls.Add(tabControl);

    // Добавление вкладок для различных сущностей системы
    AddTab("Отделы", "Department", "SELECT * FROM Department");
    AddTab("Должности", "Position", "SELECT * FROM Position");
    AddTab("Сотрудники", "Employee",
        @"SELECT e.employee_id, e.full_name, e.hire_date, e.bank_account,
        d.name as department_name, p.title as position_title,
        p.base_salary, e.phone
        FROM Employee e
        JOIN Department d ON e.department_id = d.department_id
        JOIN Position p ON e.position_id = p.position_id");
        AddTab("Расчётные периоды", "PayPeriod", "SELECT * FROM PayPeriod");
    AddTab("Расчёты зарплаты", "Payroll",
        @"SELECT py.payroll_id, e.full_name, pp.month, pp.year,
        py.calculation_date, py.total_earnings,
        py.total_deductions, py.net_payment, py.status
        FROM Payroll py
        JOIN Employee e ON py.employee_id = e.employee_id
        JOIN PayPeriod pp ON py.period_id = pp.period_id");

        AddReconnectButton();
}
```

2. Метод создания вкладки с DataGridView

```
/// <summary>
/// Создание динамической вкладки с таблицей данных и элементами управления
/// </summary>
/// <param name="tabName">Название вкладки</param>
/// <param name="tableName">Имя таблицы в БД</param>
/// <param name="query">SQL запрос для получения данных</param>
private void AddTab(string tabName, string tableName, string query)
{
    // Создание новой вкладки
    TabPage tabPage = new TabPage(tabName);
    tabControl.Controls.Add(tabPage);

    // Создание DataGridView для отображения данных
    DataGridView dataGridView = new DataGridView();
    dataGridView.Dock = DockStyle.Fill;
    dataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    dataGridView.ReadOnly = true;
    dataGridView.AllowUserToAddRows = false;
    dataGridView.AllowUserToDeleteRows = false;
    dataGridView.SelectionMode = DataGridViewSelectionMode.FullRowSelect;

    // Подписка на событие сортировки по клику на заголовок
    dataGridView.ColumnHeaderMouseClick += DataGridView_ColumnHeaderMouseClick;

    // Создание панели с кнопками управления
    Panel buttonPanel = CreateButtonPanel(tableName, dataGridView, query, tabPage);

    // Добавление элементов на вкладку
    tabPage.Controls.Add(dataGridView);
    tabPage.Controls.Add(buttonPanel);

    // Инициализация настроек сортировки
    currentSortSettings[tabName] = new SortSettings
    {
        SortOrder = CustomSortOrder.None,
        SortType = "simple"
    };

    // Первоначальная загрузка данных
    LoadDataToGrid(dataGridView, query, tabPage, tableName);
}
```

3. Метод загрузки данных в таблицу

```
/// <summary>
/// Загрузка данных из базы данных в DataGridView
/// </summary>
/// <param name="dataGridView">Целевой DataGridView</param>
/// <param name="query">SQL запрос</param>
/// <param name="tabPage">Родительская вкладка</param>
/// <param name="tableName">Имя таблицы</param>
private void LoadDataToGrid(DataGridView dataGridView, string query, TabPage
tabPage, string tableName)
{
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            // Создание адаптера данных и заполнение DataTable
            SqlDataAdapter adapter = new SqlDataAdapter(query, connection);
            DataTable table = new DataTable();
            adapter.Fill(table);
            dataGridView.DataSource = table;

            // Обновление заголовка вкладки с количеством записей
            string originalName = tabPage.Text.Replace($" ({table.Rows.Count})",
"").Split('(')[0].Trim();
            tabPage.Text = $"{originalName} ({table.Rows.Count})";

            // Обновление индикаторов сортировки
            UpdateSortIndicators(dataGridView);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки данных: {ex.Message}", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
                                        }
```

5. Метод создания параметров для SQL команды

```
/// <summary>
/// Создание SQL параметра на основе значения контрола и целевого типа данных
/// </summary>
/// <param name="columnName">Имя колонки</param>
/// <param name="inputControl">Контрол ввода</param>
/// <param name="targetType">Целевой тип данных</param>
/// <returns>SqlParameter или null если значение пустое</returns>
private SqlParameter CreateParameter(string columnName, Control inputControl, Type
targetType)
{
    // Обработка ComboBox для foreign keys
    if (inputControl is ComboBox combo)
    {
        if (targetType == typeof(int))
        {
            if (combo.SelectedValue == null) return new SqlParameter("@" +
columnName, DBNull.Value);
            return new SqlParameter("@" + columnName,
Convert.ToInt32(combo.SelectedValue));
        }
        if (combo.SelectedItem == null) return new SqlParameter("@" + columnName,
DBNull.Value);
        return new SqlParameter("@" + columnName, combo.SelectedItem.ToString());
    }

    // Обработка TextBox с различными типами данных
    if (inputControl is TextBox tb)
    {
        if (string.IsNullOrEmpty(tb.Text)) return null;
        if (targetType == typeof(decimal))
        {
            return new SqlParameter("@" + columnName,
                decimal.Parse(tb.Text.Replace(',', '.'),
                    System.Globalization.CultureInfo.InvariantCulture));
        }
        if (targetType == typeof(int))
        {
            return new SqlParameter("@" + columnName, int.Parse(tb.Text));
        }
        return new SqlParameter("@" + columnName, tb.Text);
    }

    // Обработка DateTimePicker
    if (inputControl is DateTimePicker dtp)
    {
        return new SqlParameter("@" + columnName, dtp.Value);
    }

    // Обработка NumericUpDown
    if (inputControl is NumericUpDown nud)
    {
        return new SqlParameter("@" + columnName, (int)nud.Value);
    }

    // Обработка CheckBox
    if (inputControl is CheckBox chk)
    {
```

```
    return new SqlParameter("@" + columnName, chk.Checked);
}

// Обработка MaskedTextBox (для телефонов)
if (inputControl is MaskedTextBox mtb)
{
    return new SqlParameter("@" + columnName, mtb.Text);
}

return null;
                                        }
```