СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью учебной практики является разработка программного модуля, который позволяет добавлять или удалять, сортировать и фильтровать данные по определённым критериям.

Основная задача – предусмотреть возможность формирования списка сотрудников, сортировать и фильтровать по стажу работу, по окладу, по должности; добавлять новых сотрудников или удалять сотрудников, уже имеющихся в списке.

2 ОПИСАНИЕ ТЕХНОЛОГИЙ ВЫПОЛНЕНИЯ ПРАКТИЧЕСКОГО ЗАДАНИЯ

2.1 КРАТКОЕ ОПИСАНИЕ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Наименование работы: разработка программного модуля программного обеспечения «Сотрудники».

1. Назначение разработки:

Программный модуль «Сотрудники» должен позволять пользователю добавлять, удалять или изменять данные о сотруднике, фильтровать и сортировать данные

2. Требования к программе:

Программное изделие должно выполнять следующие функции:

- добавление новых сотрудников в базу данных;
- удаление сотрудников из базы данных;
- изменение данных о сотруднике;

Входной информацией системы является:

- личная информация о сотруднике (фамилия, имя, отчество, должность, стаж работы, оклад);

Выходной информацией системы является:

- список сотрудников после фильтрации;
- список сотрудников после сортировки;
- личная информация о каждом сотруднике

Разрабатываемая система должна соответствовать требованиям надежности обеспечение устойчивого функционирования:

- Отсутствие ошибок;
- Устойчивость к возможным ошибкам;
- Автосохранение вводимой информации;
- Обеспечивать целостность данных.

2.2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

2.2.1 ЭТАПЫ РАЗРАБОТКИ ПРОГРАММНОГО ПРОДУКТА

2.2.1.1 КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ

Во время разработки программного модуля и проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.). Для построения концептуальной модели необходимо было выполнить следующие этапы:

- выделение сущностей;
- выделение ключевых атрибутов для каждой сущности;
- добавление неключевых атрибутов;
- определение связей между сущностями.

ER-диаграмма(концептуальная модель) представлена на рисунке 1:

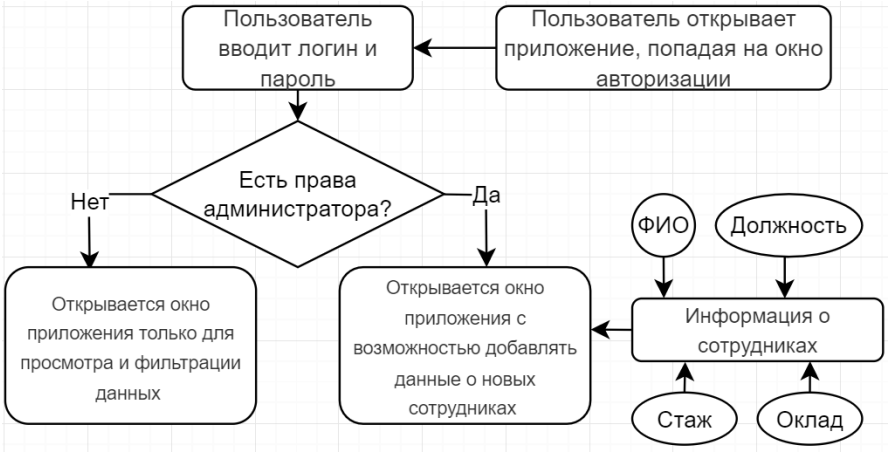


Рис.1. Концептуальная модель «Сотрудники»

2.2.1.2 ЛОГИЧЕСКАЯ МОДЕЛЬ

В процессе разработки программного продукта была разработана БД, которая содержит 2 таблицы. Таблицы, в свою очередь, хранят максимально полную характеристику, информацию и описание объектов.

Оперативными таблицами базы данных являются таблица «Сотрудники», содержащая необходимые данные о сотрудниках и таблица «Авторизация», содержащая данные для входа в приложение – логин и пароль администратора и обычного пользователя.

Таблица 1. «Сотрудники» содержит информацию о сотрудниках. Структура таблицы «Сотрудники»:

Имя поля	**Тип данных**
Фамилия	Текст
Имя	Текст
Отчество	Текст
Должность	Текст
Стаж работы	Число
Оклад	Число

Таблица 2. «Авторизация», содержит всю необходимую информацию для успешной авторизации в приложении. Структура таблицы «Авторизация»:

Имя поля	**Тип данных**
Логин	Текст
Пароль	Текст
Права доступа	Логический

2.2.1.3 ФИЗИЧЕСКАЯ МОДЕЛЬ

При физическом проектировании происходит описание компонентов, сервисов и технологий, используемых для получения решения задачи, которую выполняет программный модуль.

Физическая модель разрабатываемого программного модуля представлена на рисунке 2:

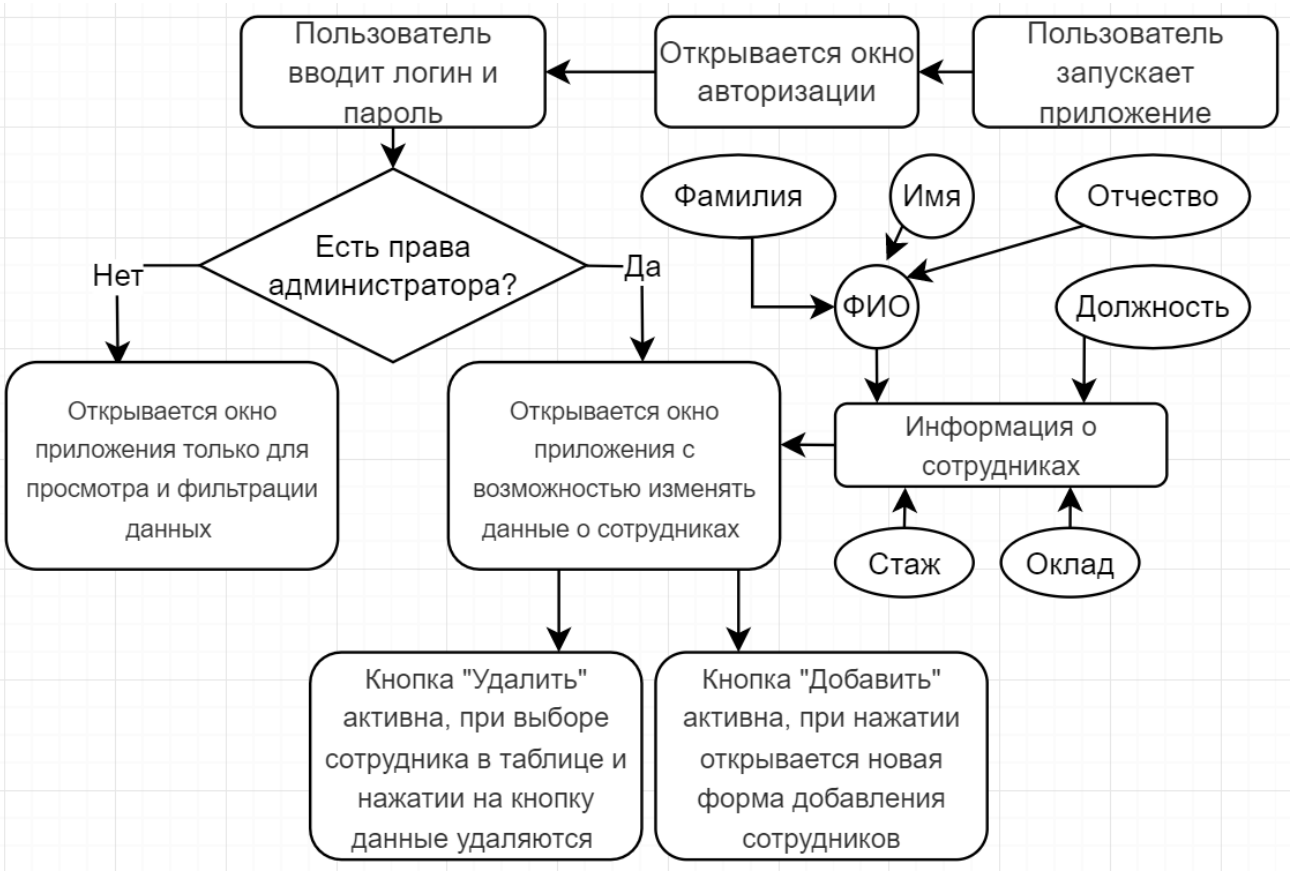


Рис.2. Физическая модель «Сотрудники»

2.2.2 РАЗРАБОТКА ПРОГРАММНОЙ ОБОЛОЧКИ С ПОМОЩЬЮ ОБЪЕКТНО-ОРИЕНТИРОВАННОГО ЯЗЫКА ПРОГРАММИРОВАНИЯ

При разработке программного модуля были использованы следующие компоненты:

- DataGridView – предоставляет настраиваемую таблицу для отображения данных для вывода информации и экспорта в Excel;
- Button – компонент для выполнения различных функций программы;
- GroupBox для формирования отсека информации о сотруднике, поиске и сортировки.
- TextBox – компонент для ввода и вывода информации.
- RadioButton – компонент для выбора критерия при поиске и сортировки.

При запуске программы Сотрудники открывается форма авторизации, представленная на рисунке 3.

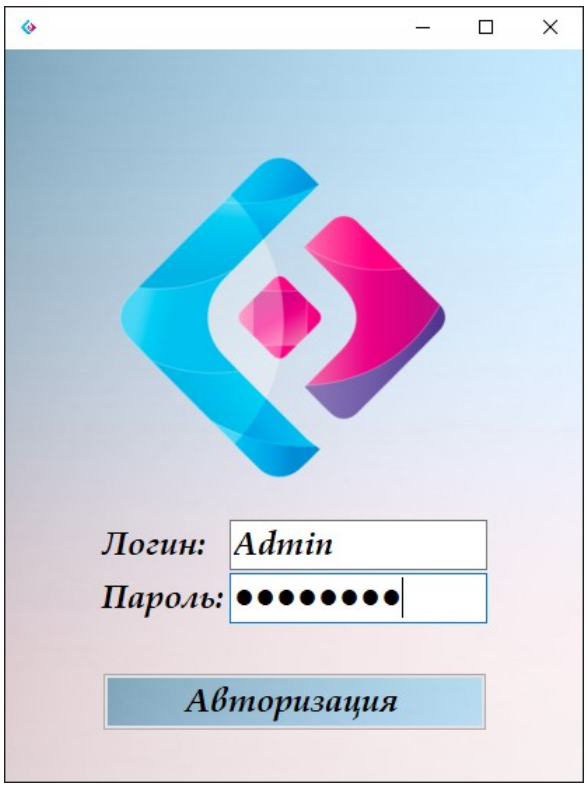


Рис.3. Форма авторизации

Листинг программного модуля окна авторизации представлен ниже:

```
//объявление функции с переменными – логин и пароль
bool authorization(string login, string password)       {
        SqlConnection conn = new SqlConnection();
        conn.ConnectionString = connStr;
        conn.Open();
        SqlCommand cmd = conn.CreateCommand();
//создание запроса для проверки логина
        cmd.CommandText = "SELECT * FROM authorize WHERE Login = @login";
        cmd.Parameters.Add("@login", SqlDbType.NVarChar, 20);
        cmd.Parameters["@login"].Value = login;
        SqlDataReader myDATA = cmd.ExecuteReader();
        if (myDATA.Read() == false)
        {
           return false;
        }
        string password_BD = myDATA["Password"].ToString();
        password = MD5Hash(password);
        if (password_BD != password)
        {
//диалоговое окно ошибки авторизации
           MessageBox.Show("Неверный логин или пароль! Повторите попытку");
           return false;
        }
        myDATA.Close();
        conn.Close();
        return true;
    }
//кнопка «Авторизация»
    private void authorizeBtn_Click(object sender, EventArgs e)
    {
        string login = loginTB.Text;
        string password = passwordTB.Text;
        bool success = authorization(login, password);
        if (success != false)
        {
//открытие главной формы при успешной авторизации
           Staffers st = new Staffers();
           st.Show();
           this.Hide();
        }
    }
```

Пользователь вводит логин и пароль в поля ввода. При неудачной авторизации появляется диалоговое окно, представленное на рисунке 4:

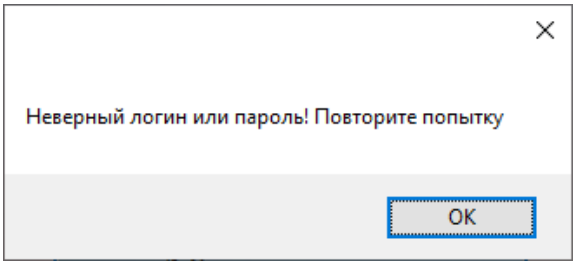


Рис.4. Диалоговое окно с сообщением об ошибке авторизации.

При успешной авторизации открывается главная форма приложения, представленная на рисунке 5:

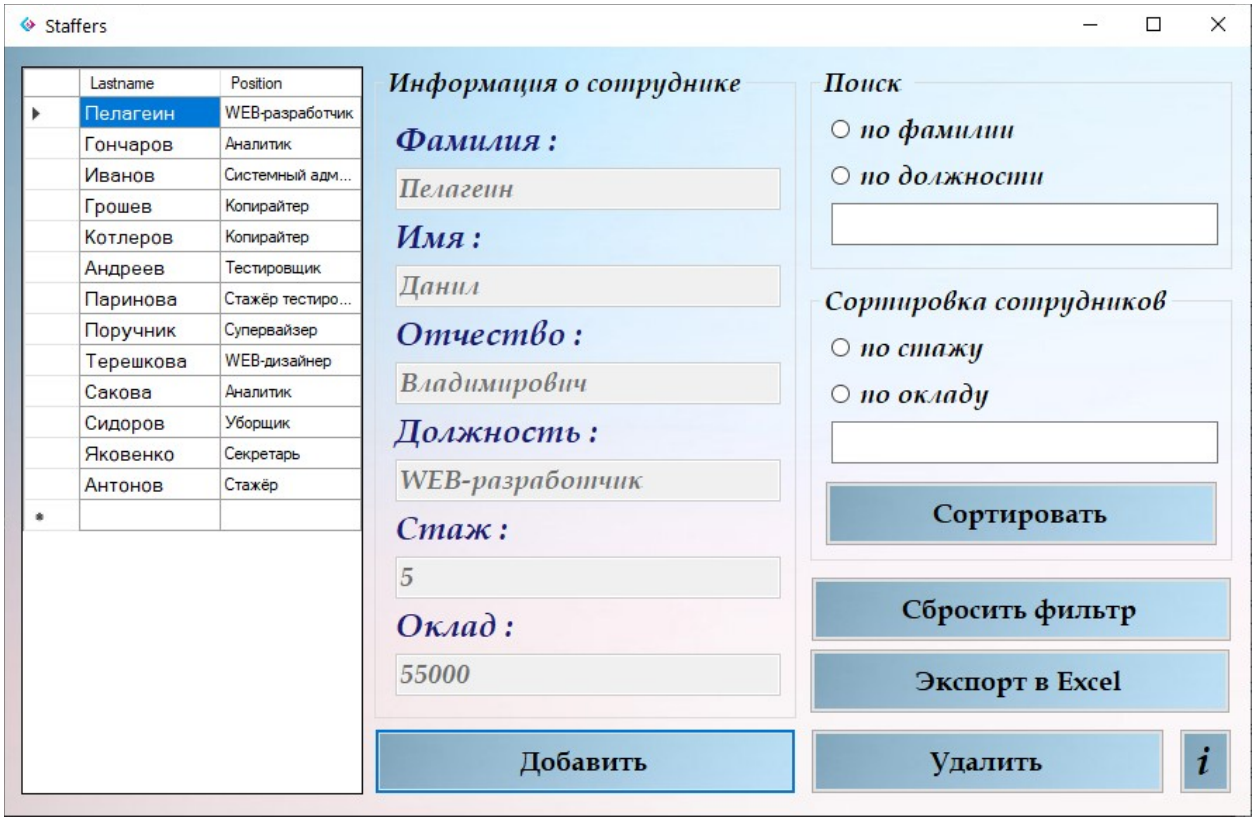


Рис.5. Главная форма.

Слева расположена таблица со списком сотрудников, 2 столбца – фамилия и должность. При выборе сотрудника из таблицы информация о нем появляется по центру (информацию изменить нельзя). Справа расположено поле быстрого поиска сотрудника по фамилии или должности. Ниже на форме расположено поле сортировки сотрудников по стажу или окладу, числовое значение этих данных вводится ниже в поле для ввода. В правом нижнем углу расположена кнопка информации о приложении.

При нажатии на кнопку «Добавить» (в случае авторизации как администратор) открывается форма добавления сотрудников, представленная на рисунке 6:

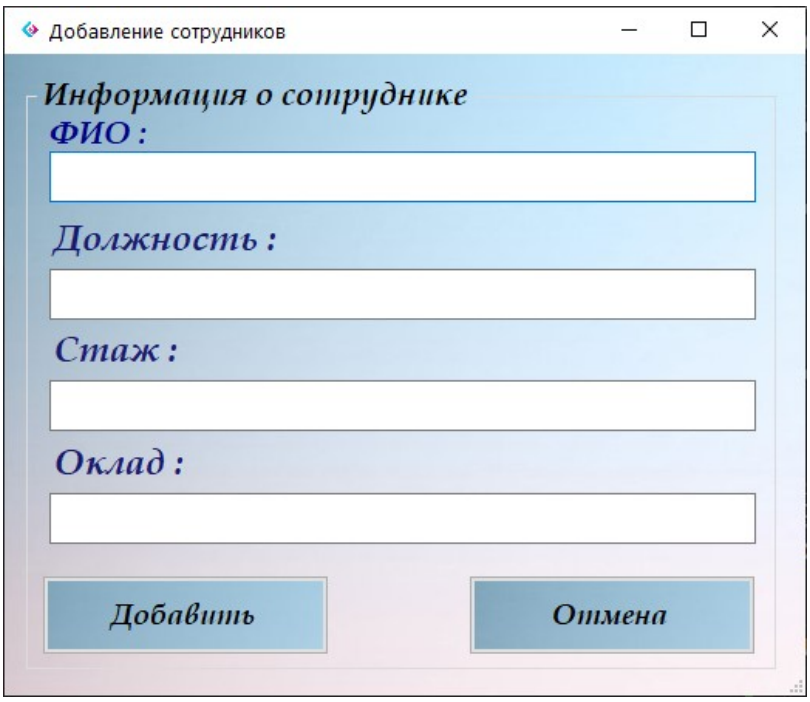


Рис.6. Форма добавления сотрудников.

При нажатии «Добавить», если заполнены все поля ввода, происходит добавление сотрудника в базу данных, в противном случае появляется диалоговое окно, представленное на рисунке 7:

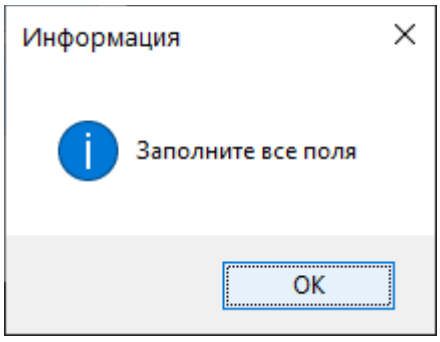


Рис.7. Диалоговое окно

Листинг программного модуля главной формы представлен в приложении 1.

2.2.3 РАЗРАБОТКА СПРАВОЧНОЙ ИНФОРМАЦИИ О ПРОГРАММНОМ ПРОДУКТЕ

Продукт разработан для просмотра, сортировки, поиска и добавления данных и вывода их в отдельный файл.

Программа была разработана на языке программирования C# в приложении Visual Studio 2022, студенткой 4 курса специальности 09.02.03 «Программирование в компьютерных системах» Поручник К.Ф.

Данная информация появляется на экране при нажатии кнопки «Информация», что представлено на рисунке 8:

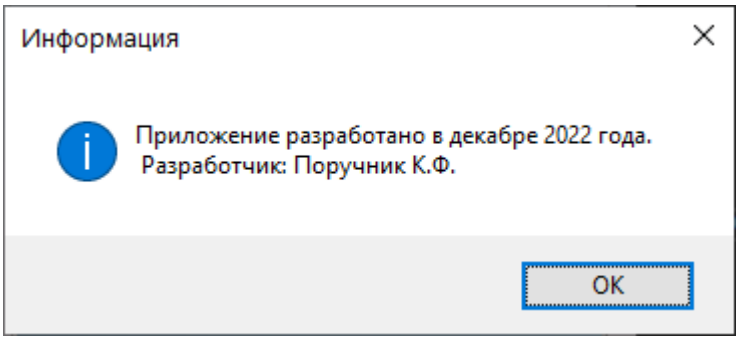


Рис.8. Окно информации

2.2.4 РАЗРАБОТКА ЗАПРОСОВ РАЗЛИЧНОЙ СЛОЖНОСТИ

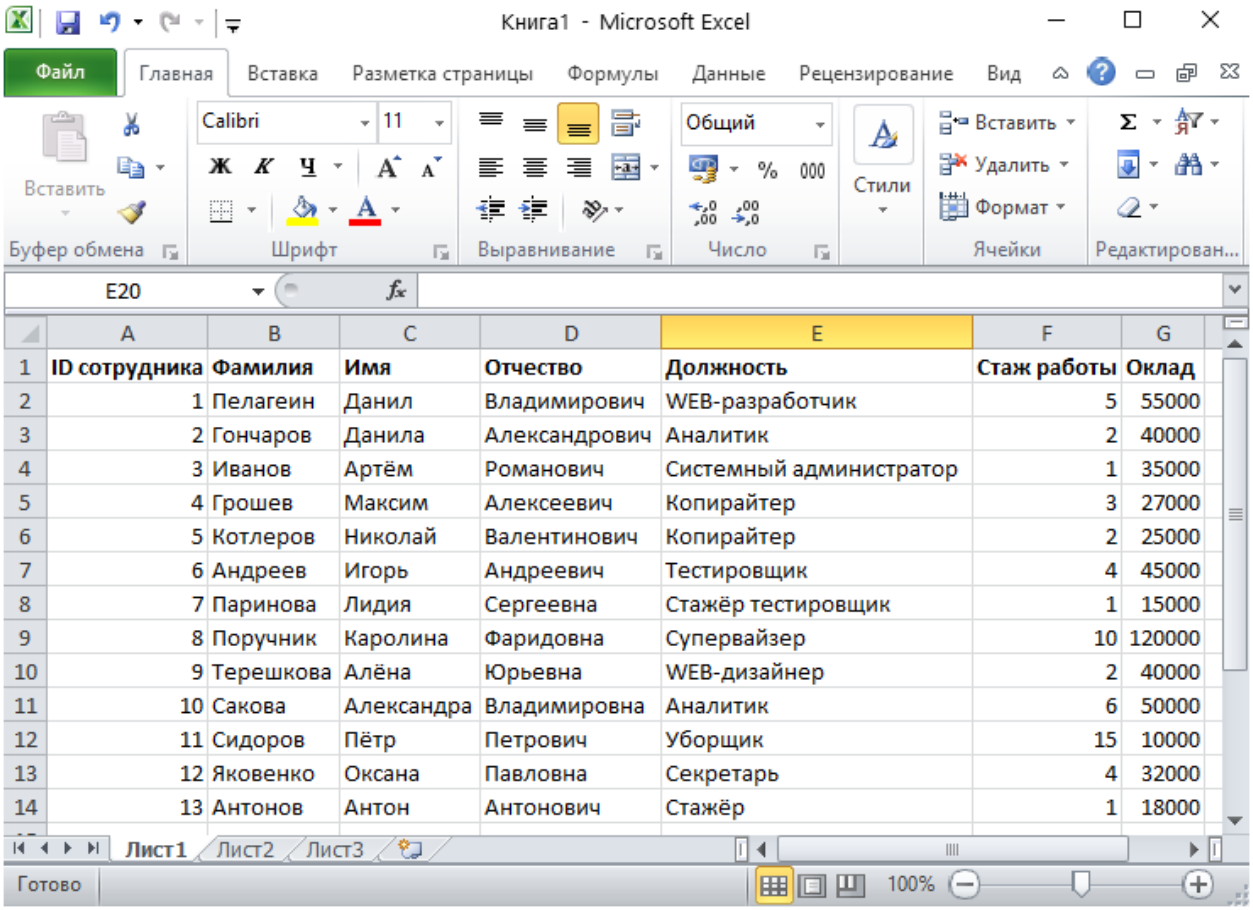
Запрос на добавление информации о сотруднике:

```
//Событие при нажатие на кнопку

private void addBtn_Click(object sender, EventArgs e) «добавить»
    {
      Add add = new Add(); //открытие окна добавления сотрудников
      add.ShowDialog();
      if (add.DialogResult == DialogResult.OK)
      {
        if (add.FIO.Text == "" ||
          add.position.Text == "" ||
          add.experience.Text == "" ||
          add.salary.Text == "")
//открытие диалогового окна, если поля ввода не заполнены
                    { MessageBox.Show("Заполните все поля", "Информация", MessageBoxButtons.OK,
MessageBoxIcon.Information); }
        else
        {
//объявление переменных, которые являются информацией о сотруднике
          int rs = dataGridView1.RowCount;
          string fio = add.FIO.Text;
          string[] subs = fio.Split(' ');
          string Lastname = subs[0];
          string Name = subs[1];
          string Fathername = subs[2];
          string position = add.position.Text;
          string experience = add.experience.Text;
          string salary = add.salary.Text;
//создание подключения к базе данных SQL
          conn = new SqlConnection();
          conn.ConnectionString = connStr;
          conn.Open();
          SqlCommand myComm = conn.CreateCommand();
//SQL запрос на добавление сотрудника
            myComm.CommandText = "insert into staffer (Lastname, Name, Fathername, Position, Experience,
Salary) values (@Lastname, @Name, @Fathername, @Position, @Experience, @Salary)";
//параметры для корректного добавления данных
          myComm.Parameters.Add("@Lastname", SqlDbType.NVarChar, 50);
          myComm.Parameters["@Lastname"].Value = Lastname;
          myComm.Parameters.Add("@Name", SqlDbType.NVarChar, 50);
          myComm.Parameters["@Name"].Value = Name;
          myComm.Parameters.Add("@Fathername", SqlDbType.NVarChar, 50);
          myComm.Parameters["@Fathername"].Value = Fathername;
          myComm.Parameters.Add("@Position", SqlDbType.NVarChar, 50);
          myComm.Parameters["@Position"].Value = position;
          myComm.Parameters.Add("@Experience", SqlDbType.Int, 4);
          myComm.Parameters["@Experience"].Value = experience;
          myComm.Parameters.Add("@Salary", SqlDbType.Int, 4);
          myComm.Parameters["@Salary"].Value = salary;
          myComm.ExecuteNonQuery();
          conn.Close();
          stafferTableAdapter.Fill(this.staffDataSet.staffer);
        }
      }
```

2.2.5 РАЗРАБОТКА ПЕЧАТНОЙ ФОРМЫ ОТЧЕТНОЙ ДОКУМЕНТАЦИИ

В ходе разработки программы была разработана отчетная документация в программе «Excel» (отчет формируется при нажатии на кнопку «Экспорт в Excel»). Отчет представлен на рисунке 9.



	A	B	C	D	E	F	G
1	**ID сотрудника**	**Фамилия**	**Имя**	**Отчество**	**Должность**	**Стаж работы**	**Оклад**
2	1	Пелагеин	Данил	Владимирович	WEB-разработчик	5	55000
3	2	Гончаров	Данила	Александрович	Аналитик	2	40000
4	3	Иванов	Артём	Романович	Системный администратор	1	35000
5	4	Грошев	Максим	Алексеевич	Копирайтер	3	27000
6	5	Котлеров	Николай	Валентинович	Копирайтер	2	25000
7	6	Андреев	Игорь	Андреевич	Тестировщик	4	45000
8	7	Паринова	Лидия	Сергеевна	Стажёр тестировщик	1	15000
9	8	Поручник	Каролина	Фаридовна	Супервайзер	10	120000
10	9	Терешкова	Алёна	Юрьевна	WEB-дизайнер	2	40000
11	10	Сакова	Александра	Владимировна	Аналитик	6	50000
12	11	Сидоров	Пётр	Петрович	Уборщик	15	10000
13	12	Яковенко	Оксана	Павловна	Секретарь	4	32000
14	13	Антонов	Антон	Антонович	Стажёр	1	18000

Рис.9. Отчетная документация «Сотрудники».

2.2.6 РАЗРАБОТКА ДОПОЛНИТЕЛЬНЫХ ФУНКЦИЙ

Справа вверху на форме расположено поле быстрого поиска по фамилии или должности сотрудника. Пользователь начинает вводить данные в поле для ввода – моментально происходит поиск по примерному неполному или точному полному совпадению. Ниже на форме расположено поле сортировки сотрудников по стажу или окладу, числовое значение этих данных вводится ниже в поле для ввода. При нажатии на кнопку «сортировать» в таблице слева предоставляется отсортированный список сотрудников. При нажатии кнопки «Сбросить фильтр» таблица слева принимает исходный вид. Пример выполнения поиска представлен на рисунке 10. Пример выполнения сортировки сотрудников представлен на рисунке 11.

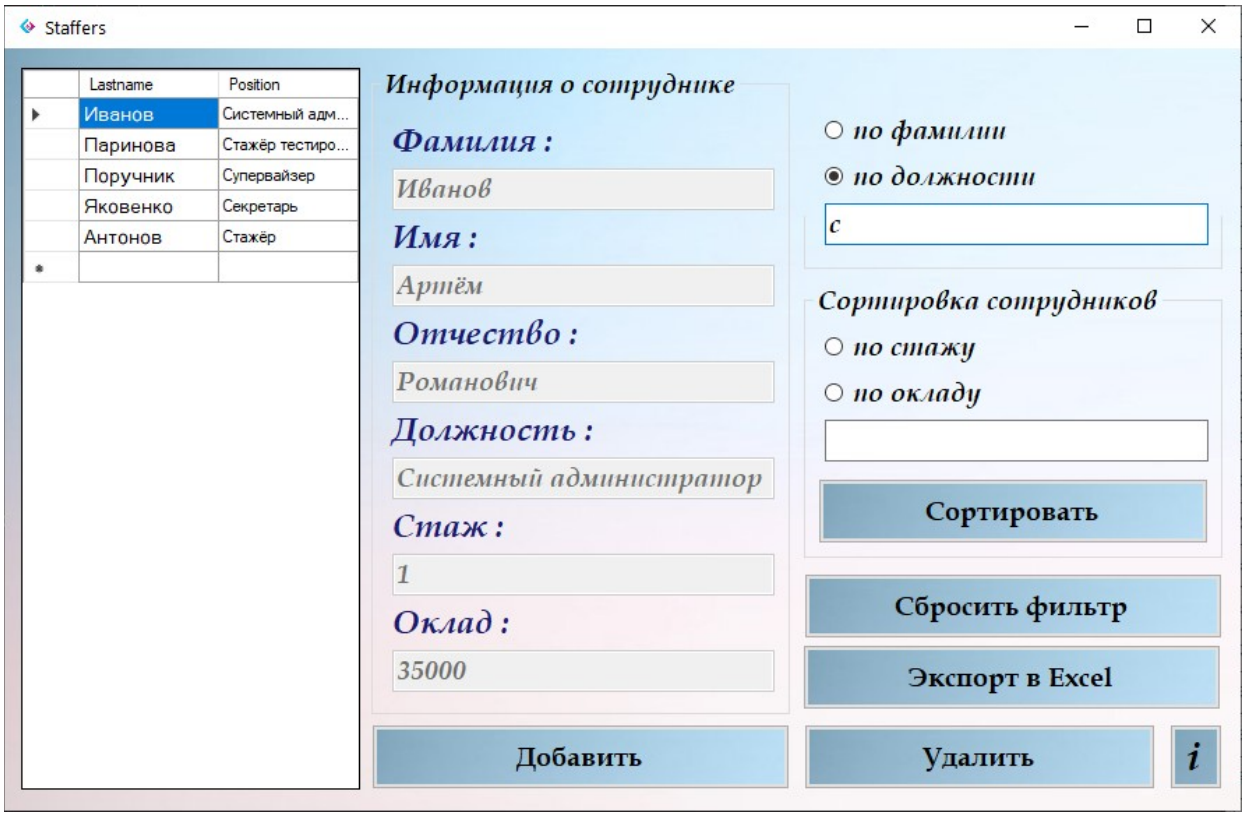


Рис.10. Пример выполнения поиска сотрудника по должности.

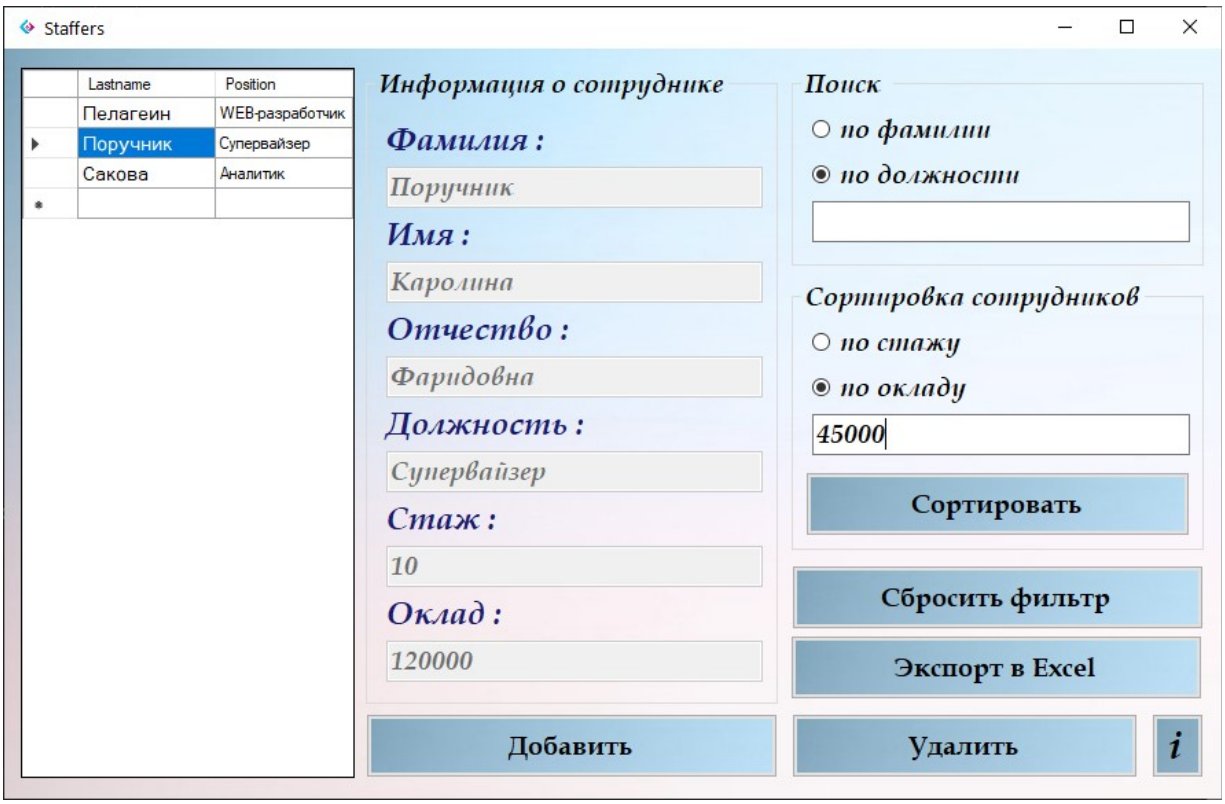


Рис.11. Пример выполнения сортировки сотрудников по окладу.

ЗАКЛЮЧЕНИЕ

В ходу работы были последовательно пройдены все этапы разработки программного модуля, начиная с написания технического задания, создания логической и физической модели и заканчивая созданием корректно работающего приложения.

Каждый этап сопровождался подробным анализом проделанной работы, что позволило свести к минимуму ошибки.

В процессе выполнения проекта была создана программа «Сотрудники».

В результате выполнения данного проекта был создан программный модуль связанный напрямую с базой данных SQL, позволяющий обрабатывать, добавлять, искать и фильтровать данные.

СПИСОК ЛИТЕРАТУРЫ

1. «C# in Depth» - Jon Skeet. 2019 г. – 209с.
2. «Объектно-ориентированное программирование в действии» - Бадд Тимоти / [Пер. с англ. А Бердникова; Гл. ред. В. Усманов]. - СПб: Питер, 2019. - 460 с.
3. «Высокоуровневые методы информатики и программирования» Учебное пособие. Бутаков С.В. Министерство образования и науки Российской Федерации, Федеральное агентство по образованию. Барнаул: изд-во ААЭП, 2020. - 72с.
4. Технологии объектно-ориентированного программирования Хореев П.Б. Учебное пособие; - М.: Academia, 2018. - 447 с.
5. «ПРОГРАММИРОВАНИЕ НА C#» - М. А. МЕДВЕДЕВ А. Н. МЕДВЕДЕВ. Учебное пособие / М. А. Медведев, А.Н. Медведев. — Екатеринбург: Изд-во Урал. 2021. — 64 с.
6. https://learn.microsoft.com/ru-ru/dotnet/csharp/ - официальный сайт Microsoft «Документация по C#» Обновлено: 2022.

ПРИЛОЖЕНИЕ 1.

Листинг главной формы программного модуля «Сотрудники»:

```
//добавление библиотек
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;
using Excel = Microsoft.Office.Interop.Excel;
namespace Staffer
{
//создание класса Сотрудники
public partial class Staffers : Form
   {
      public Staffers()
      {
         InitializeComponent();
      }
//объявление глобальных переменных
      private Excel.Sheets excelsheets;
      private Excel.Worksheet excelworksheet;
      private Excel.Range excelcells;
      private Excel.Application excelapp;
//создание переменной для подключения к базе данных
      string connStr = @"Data Source=localhost;Initial Catalog=staff;Integrated Security=True";
      SqlConnection conn = null;
      private void Staffers_Load(object sender, EventArgs e)
      {
  // TODO: данная строка кода позволяет загрузить данные в таблицу "staffDataSet.staffer". При необходимости она может быть перемещена или удалена.
         this.stafferTableAdapter.Fill(this.staffDataSet.staffer);
      }
//кнопка Добавления сотрудников
      private void addBtn_Click(object sender, EventArgs e)
      {
         Add add = new Add();
         add.ShowDialog();
         if (add.DialogResult == DialogResult.OK)
         {
            if (add.FIO.Text == "" ||
               add.position.Text == "" ||
               add.experience.Text == "" ||
               add.salary.Text == "")
                        { MessageBox.Show("Заполните все поля", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information); }
            else
```

```
            {
                int rs = dataGridView1.RowCount;
                string fio = add.FIO.Text;
                string[] subs = fio.Split(' ');
                string Lastname = subs[0];
                string Name = subs[1];
                string Fathername = subs[2];
                string position = add.position.Text;
                string experience = add.experience.Text;
                string salary = add.salary.Text;
                conn = new SqlConnection();
                conn.ConnectionString = connStr;
                conn.Open();
                SqlCommand myComm = conn.CreateCommand();
                    myComm.CommandText = "insert into staffer (Lastname, Name, Fathername, Position, Experience,
Salary) values (@Lastname, @Name, @Fathername, @Position, @Experience, @Salary)";
                myComm.Parameters.Add("@Lastname", SqlDbType.NVarChar, 50);
                myComm.Parameters["@Lastname"].Value = Lastname;
                myComm.Parameters.Add("@Name", SqlDbType.NVarChar, 50);
                myComm.Parameters["@Name"].Value = Name;
                myComm.Parameters.Add("@Fathername", SqlDbType.NVarChar, 50);
                myComm.Parameters["@Fathername"].Value = Fathername;
                myComm.Parameters.Add("@Position", SqlDbType.NVarChar, 50);
                myComm.Parameters["@Position"].Value = position;
                myComm.Parameters.Add("@Experience", SqlDbType.Int, 4);
                myComm.Parameters["@Experience"].Value = experience;
                myComm.Parameters.Add("@Salary", SqlDbType.Int, 4);
                myComm.Parameters["@Salary"].Value = salary;
                myComm.ExecuteNonQuery();
                conn.Close();
                stafferTableAdapter.Fill(this.staffDataSet.staffer);
            }
        }
    }
//реализация быстрого поиска
    private void Search_TextChanged_1(object sender, EventArgs e)
    {
        if (lastnameRbtn.Checked == true)
        {
            stafferBindingSource.Filter = "Lastname LIKE  '" + Search.Text + "*'";
            if (Search.Text == "")
                stafferBindingSource.Filter = null;
        }
        else if (positionRbtn.Checked == true)
        {
            stafferBindingSource.Filter = "Position LIKE  '" + Search.Text + "*'";
            if (Search.Text == "")
                stafferBindingSource.Filter = null;
        }
    }
//реализация сортировки
    private void SortBtn_Click_1(object sender, EventArgs e)
    {
        if (Sort.Text != "")
        {
            if (experienceBbtn.Checked == true)
            {
                stafferBindingSource.Filter = "Experience > " + Sort.Text;
            }
            if (salaryBbtn.Checked == true)
            {
```

```
                stafferBindingSource.Filter = "Salary > " + Sort.Text;
            }
        }
    }
//реализация события кнопки Сброс фильтра
    private void OffFilterBtn_Click(object sender, EventArgs e)
    {
        stafferBindingSource.Filter = null;
    }
//кнопки выбора фамилии
    private void lastnameRbtn_CheckedChanged(object sender, EventArgs e)
    {
        Search.Clear();
        Sort.Clear();
    }
//реализация закрытия программы при закрытии главной формы
    private void Staffers_FormClosed(object sender, FormClosedEventArgs e)
    {
        Application.Exit();
    }
//реализация события кнопки удаления сотрудников из БД
    private void DeleteBtn_Click(object sender, EventArgs e)
    {
        conn = new SqlConnection();
        conn.ConnectionString = connStr;
        conn.Open();
        SqlCommand myComm = conn.CreateCommand();
        int index = dataGridView1.CurrentCell.RowIndex;
        string k = dataGridView1.Rows[index].Cells[0].Value.ToString();
        myComm.CommandText = $"Delete from staffer where StaffID = {k}";
        myComm.ExecuteNonQuery();
        conn.Close();
        this.stafferTableAdapter.Fill(this.staffDataSet.staffer);
    }
//реализация экпорта данных в Excel
    private void ExportBtn_Click(object sender, EventArgs e)
    {
        excelapp = new Excel.Application();
        string put = Application.StartupPath.ToString();
        excelapp.Workbooks.Open(put + @"\Staffer.xlsx");
        excelsheets = excelapp.ActiveWorkbook.Worksheets;
        excelworksheet = (Excel.Worksheet)excelsheets.get_Item(1);
        excelworksheet.Activate();
        for (int i=0; i < dataGridView1.RowCount-1; i++)
        {
            for (int j = 0; j<dataGridView1.ColumnCount; j++)
            {
                excelcells = (Excel.Range)excelworksheet.Cells[i + 2, j + 1];
                excelcells.Borders.ColorIndex = 1;
                excelcells.HorizontalAlignment = Excel.Constants.xlCenter;
                excelcells.VerticalAlignment = Excel.Constants.xlCenter;
                excelcells.Value2 = dataGridView1.Rows[i].Cells[j].Value;
            }
        }
        excelapp.Visible = true;
    }
```