Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение высшего образования
«Санкт-Петербургский государственный технологический институт
(технический университет)»

**Дисциплина «Разработка ПС»**

Отчёт по лабораторной работе №4.

«Работа с базами данных»

Вариант № 3

Выполнил: Гусев Антон Александрович,
студент 494 группы.

Преподаватели: Корниенко Иван Григорьевич,
Федин Алексей Константинович.

Санкт-Петербург
2021

## Постановка задачи

Необходимо написать приложение с использованием технологии WinForms реализующие вариант задания. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

## Исходные данные

В качестве исходных данных программа использует:

— Данные типа string, хранящиеся в базе данных

— Могут быть введены пользователем в соответствующие элементы интерфейса

## Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

— Если введенные пользователем данные отсутствуют.

— Если выбранный пользователем идентификатор записи отсутствует.

— Попытка пользователя обратиться к файлу с зарезервированным именем.

— Запись работы программы в уже существующий файл или создание недопустимого файла, попытка записи данных в файл, доступный только для чтения.

— Дата прибытия поезда меньше даты отправки

## Математические методы и алгоритмы решения задач

Работа с базой данных осуществляется с помощью подключения SQLite к среде разработки и использования EntityFramework для работы с сущностями, поддерживаемого SQLite.

## Формат представления данных

Исходные данные хранятся в базе данных, хранящейся на компьютере, путь к которой указан в коде программы

Таблица 1 – Основные переменные программы

| **Имя переменной** | **Тип данных** | **Описание** |
|---|---|---|
| `ID` | `int` | Идендефикатор поезда |
| `Name` | `string` | Наименование поезда |
| `Departure` | `string` | Дата и время отправления |
| `Arrival` | `string` | Дата и время прибытия |
| `Station_dep` | `string` | Пункт отправления |
| `Station_arr` | `string` | Пункт прибытия |
| `Cost` | `int` | Стоимость билета |

## Структура программы

Программа разбита на 5 класссов.

Основная последовательность работы программы: после запуска на экран выводится основная форма программы, на которой расположен весь основной рабочий интерфейс. Элементы интерфейса основной формы: таблица базы данных, кнопка добавления записи в БД, кнопка изменения записи в БД, кнопка удаления записи в БД, сохранение таблицы записей в файл, настройка необходимости вывода справочного окна при запуске программы, вызов справочного окна. В случае вызова справочного окна на экран выведется окно с информацией об авторе и назначении программы. Программа подключается к необходимой базе данных и выводит ее содержимое в таблицу на интерфейс основной формы. Работа с записями базы данных производится в соответствующих элементах интерфейса основной формы. При выборе добавления, изменения и удаления записей в базе данных выводятся соответствующие формы с необходимыми для реализации нужного действия элементами интерфейса.

Таблица 2 – Функции основного алгоритма

| **Имя** | **Описание** |
|---|---|
| `private void ButtonAdd_Click(object sender, EventArgs e)` | Функция добавления записи в таблицу |
| `private void ButtonDelete_Click(object sender, EventArgs e)` | Функция удаления записи из таблицы |
| `private void ButtonChange_Click(object sender, EventArgs e)` | Функция изменения записи в таблице |

## Описание хода выполнения лабораторной работы

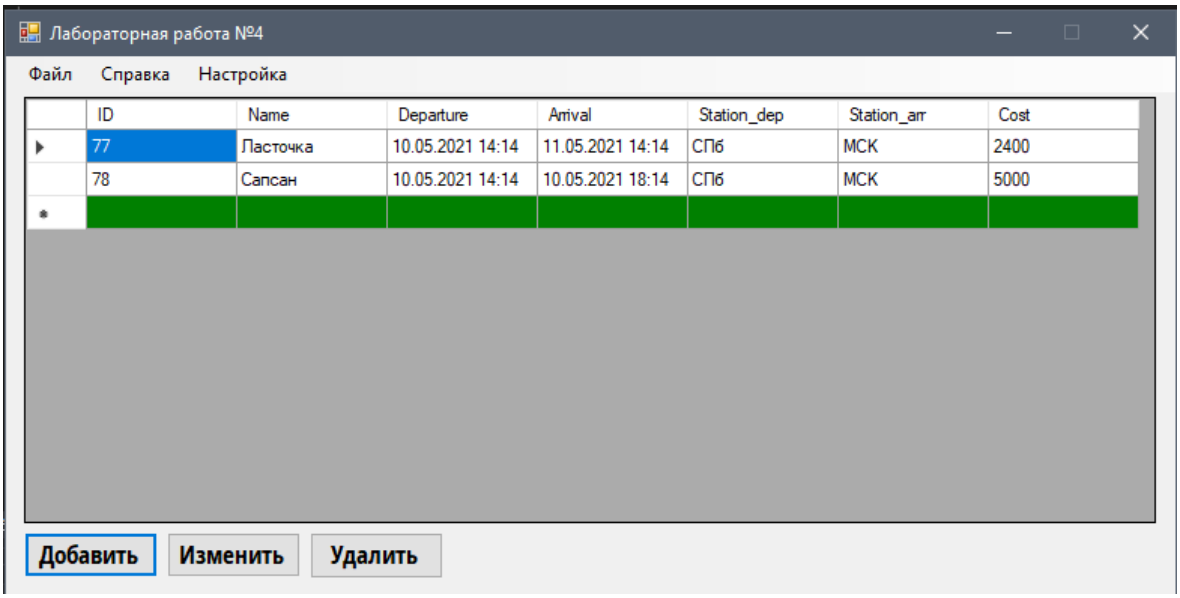
Первым шагом в выполнении данной лабораторной работы стало создание решения и проекта в среде разработки Microsoft Visual Studio C++ 2019.

Изначально был оформлен интерфейс основной формы, расчитанный на все необходимые функции программы. Затем были написан код для каждого элемента интерфейса основной формы, среди которых: добавление записи в БД, изменение записи в БД, удаление записи в БД, настройка отображения справочного окна при запуске программы, сохранение таблицы записей в файл, вызов справочного окна. Затем был создан пользовательский интерфейс, включающий в себя информацию о программе и авторе. В отдельных модулях был написан код работы с базой данных, описаны атрибуты сущностей.

В ходе работы над кодом программы были обработаны такие нестандартные случаи, как ввод пользователем некорректных данных, попытка использовать файл с зарезервированным OS Windows именем, попытка записи информации во внешний файл, имеющий атрибут «Только для чтения».

## Результаты работы программы

В качестве результата корректной работы программы выводится таблица записей базы данных на интерфейс.




Вывод записей базы данных в таблицу на интерфейсе.

Рисунок 2 – Пример работы программы

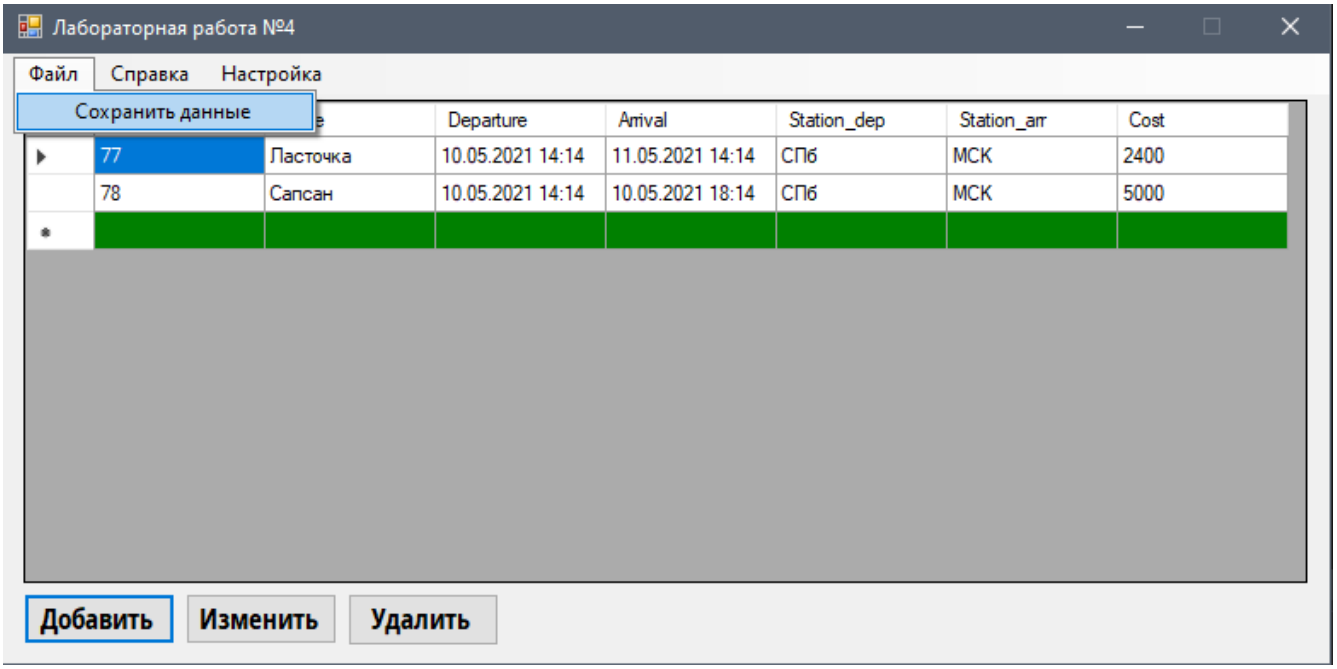



Рисунок 3 – Вызов меню работы с файлом

Сохраним таблицу записей в файл.

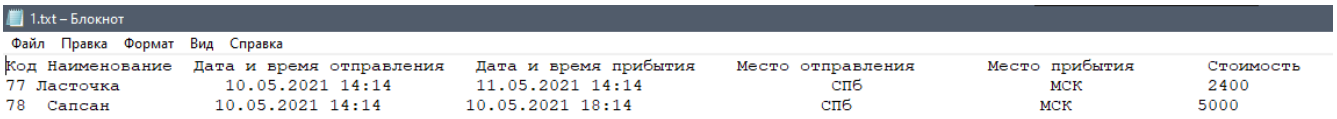
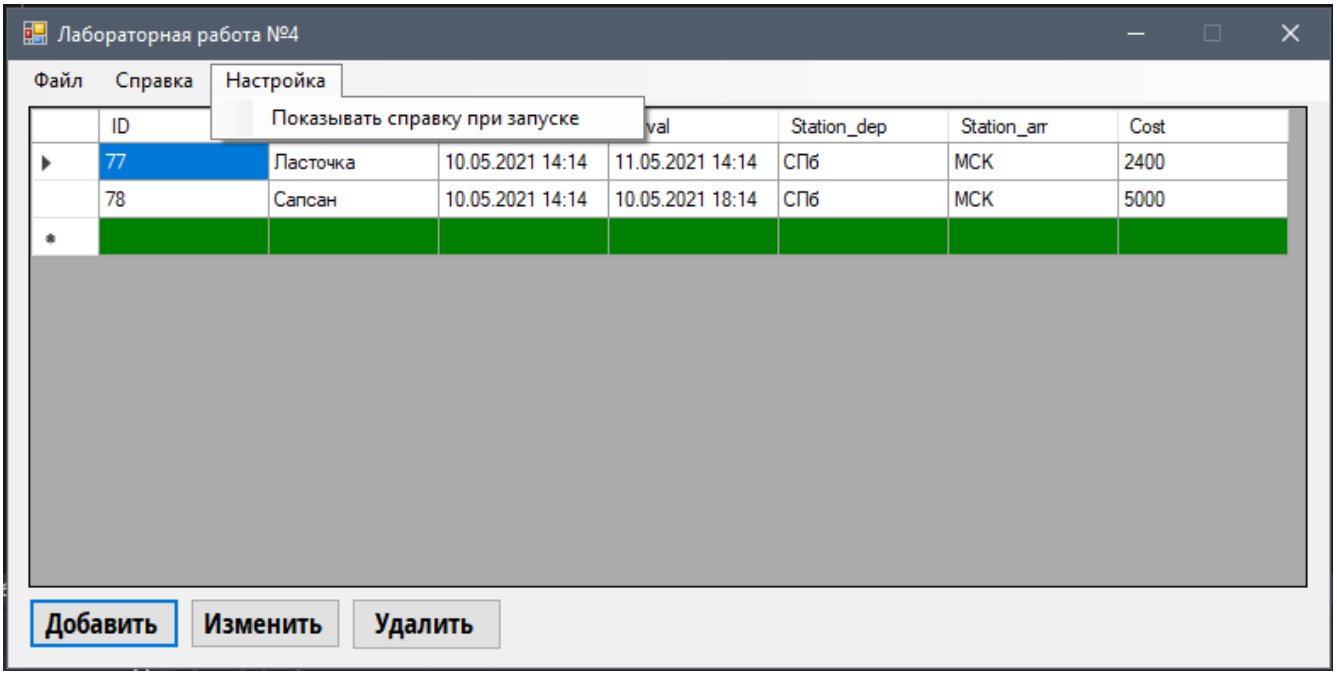



Рисунок 4 – Содержимое файла с сохраненными данными




Вызов меню настроек

Рисунок 5 – Другой пример работы программы

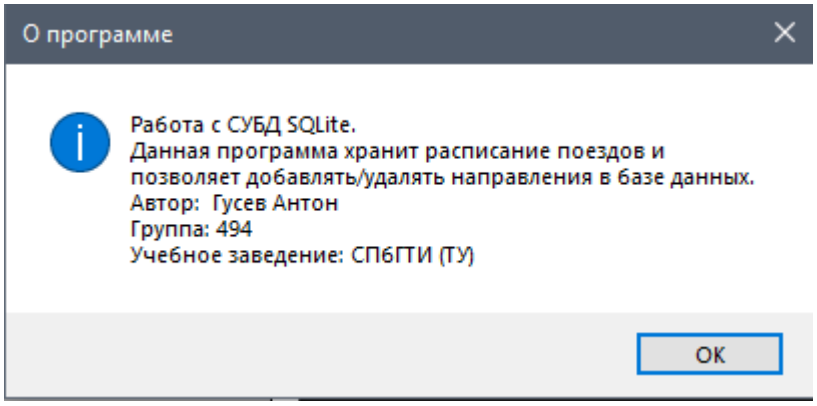



Рисунок 6 – Справочное окно

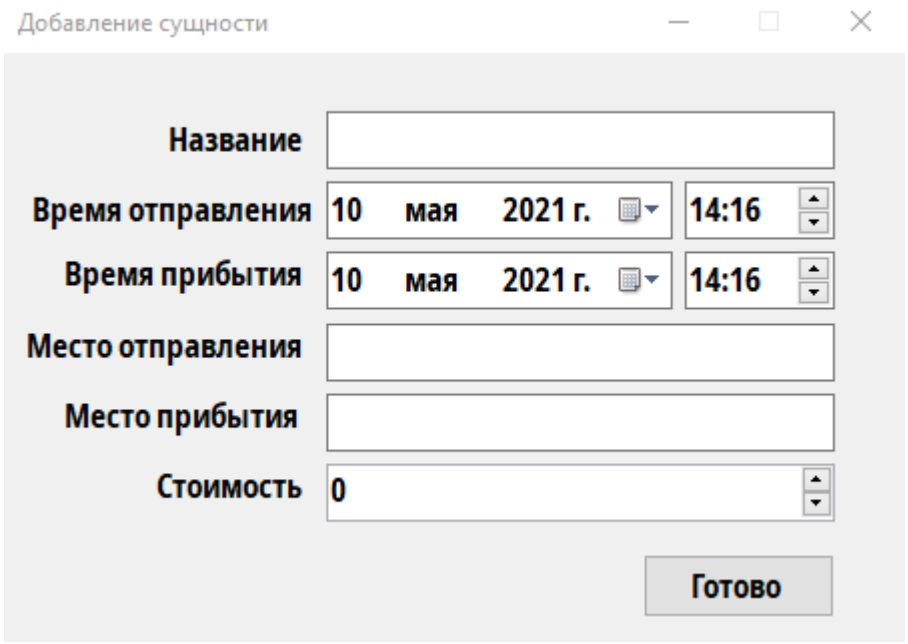



Рисунок 7 – Окно добавления записи

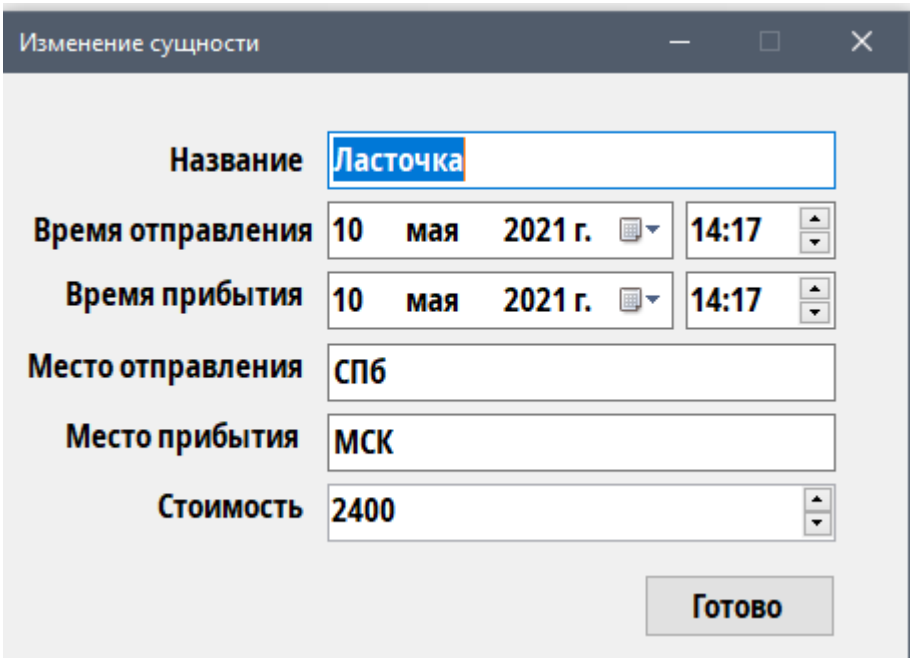



Рисунок 8 – Окно изменения записи

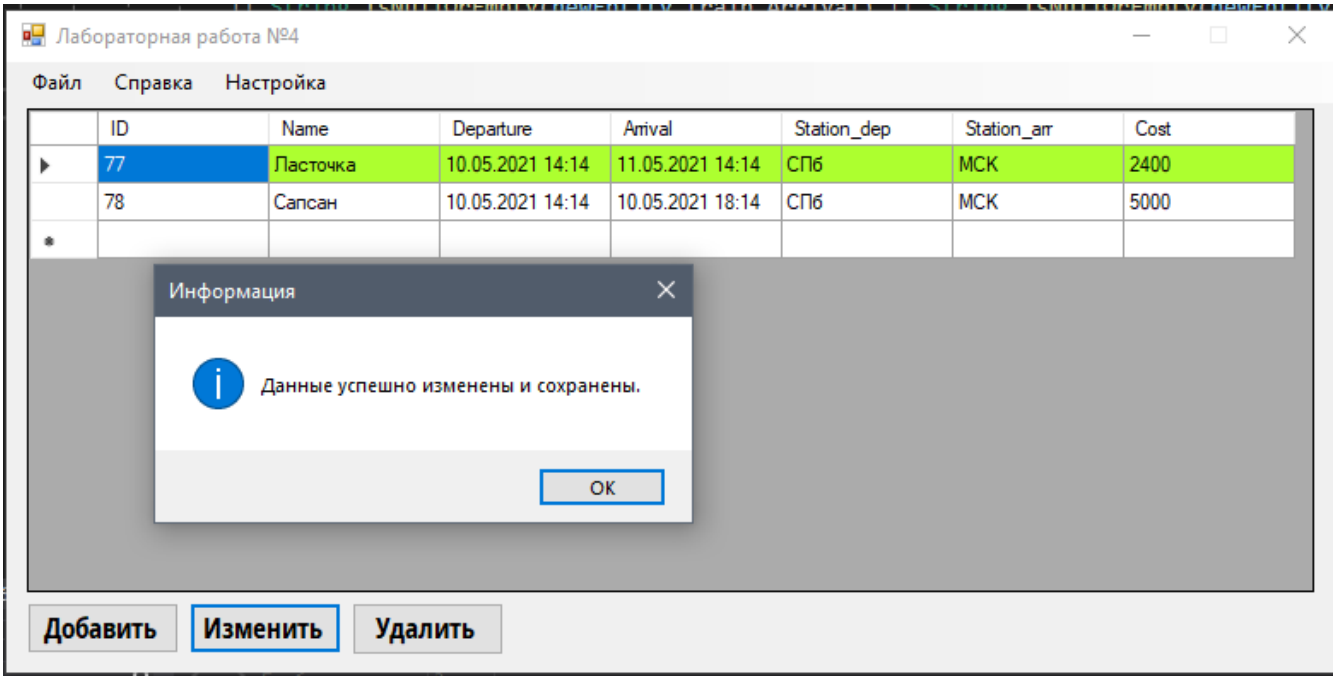



Рисунок 9 – Изменения записи

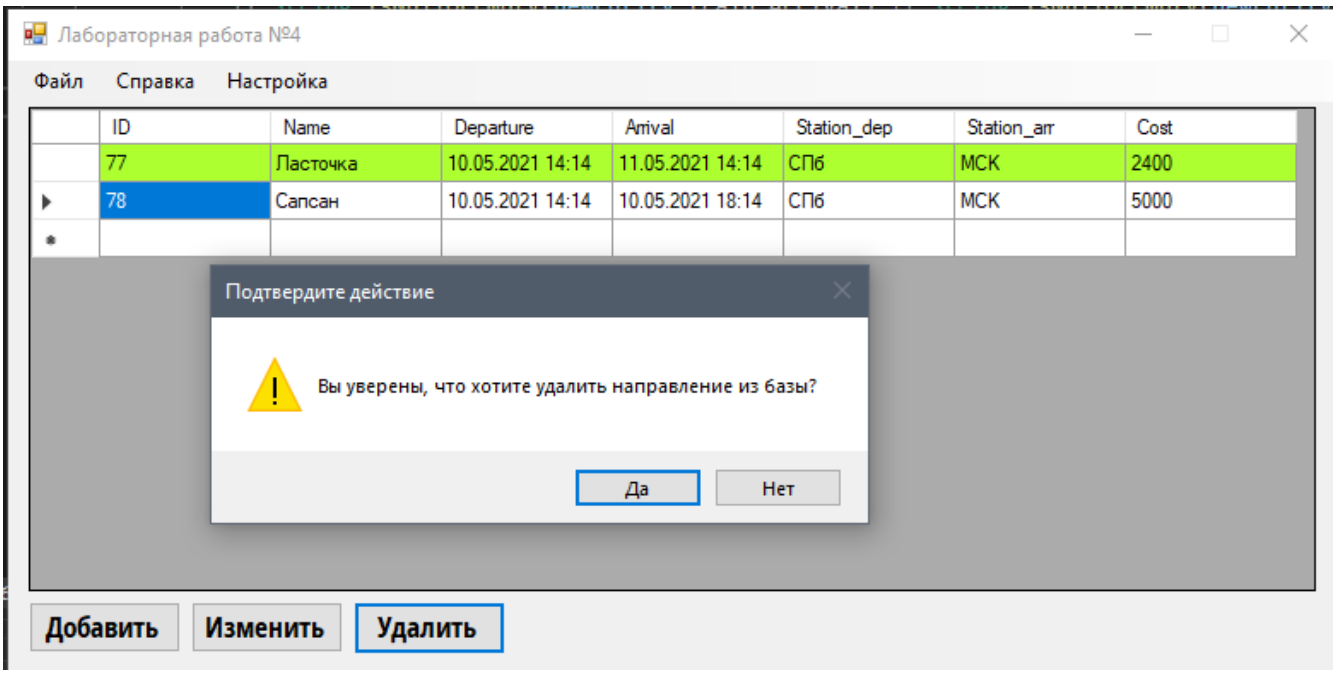



Рисунок 10 – Удаление записи

## Исходный текст программы

[Начало программы ---]

[Начало Program.cs ---]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace rps4
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainWindow());
        }
    }
}
```

[Конец Program.cs ---]

[Начало MainForm.cs---]

```
using System;
using System.ComponentModel;
using System.Data;
using System.Data.Entity;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace rps4
{
    public partial class MainWindow : Form
    {
        public ApplicationContext db;
        public BindingList<Train> Trains;
        public MainWindow()
        {
            InitializeComponent();
            saveFileDialog.Filter = @"Text files(*.txt)|*.txt";
            MaximizeBox = false;

            db = new ApplicationContext();
            db.Trains.Load();

            Trains = db.Trains.Local.ToBindingList();

            TrainsGrid.DataSource = Trains;
            if (Settings.Default.Show == true)
            {
                InfoToolStripMenuItem_Click(null, null);
                ShowInfoToolStripMenuItem.Checked = true;
            }
            else ShowInfoToolStripMenuItem.Checked = false;
            if (TrainsGrid.RowCount == 1)
            {
                ButtonChange.Enabled = false;
                ButtonDelete.Enabled = false;
```

```
            }
        }

        private void ButtonAdd_Click(object sender, EventArgs e)
        {
            try
            {
                var newEntity = new Adding(new Train());

                foreach (DataGridViewRow row in TrainsGrid.Rows)
                {
                    row.DefaultCellStyle.BackColor = Color.White;
                }
                newEntity.ShowDialog();

                if (String.IsNullOrEmpty(newEntity.train.Name) ||
String.IsNullOrEmpty(newEntity.train.Departure)
                    || String.IsNullOrEmpty(newEntity.train.Arrival) ||
String.IsNullOrEmpty(newEntity.train.Station_dep)
                    || String.IsNullOrEmpty(newEntity.train.Station_arr) ||
String.IsNullOrEmpty(newEntity.train.Cost.ToString()))
                {
                    throw new NullReferenceException();
                }
                else
                {
                    db.Trains.Add(newEntity.train);
                    db.SaveChanges();
                    int newRowIndex = TrainsGrid.Rows.Count - 1;
                    TrainsGrid.Rows[newRowIndex].DefaultCellStyle.BackColor = Color.Green;
                    MessageBox.Show("Данные успешно добавлены и сохранены.", "Информация",
                                    MessageBoxButtons.OK, MessageBoxIcon.Information);
                    ButtonChange.Enabled = true;
                    ButtonDelete.Enabled = true;
                }

            }
            catch (NullReferenceException)
            {
                MessageBox.Show("Вы не ввели данные.", "Ошибка!",
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }

        private void NewEntity_FormClosed(object sender, FormClosedEventArgs e)
        {
            throw new NotImplementedException();
        }

        private void ButtonDelete_Click(object sender, EventArgs e)
        {
            try
            {
                string Choose = TrainsGrid.CurrentCell.OwningColumn.Name;
                if (Choose == "ID")
                {
                    if (DialogResult.Yes == MessageBox.Show("Вы уверены, что хотите удалить направление
из базы?", "Подтвердите действие",
                        MessageBoxButtons.YesNo, MessageBoxIcon.Warning))
                    {
                        int deleting = int.Parse(TrainsGrid.CurrentCell.Value.ToString());
                        db.Trains.Remove(db.Trains.Find(deleting));
                        db.SaveChanges();
                    }
                }
                if (TrainsGrid.RowCount == 1)
                {
                    ButtonChange.Enabled = false;
                    ButtonDelete.Enabled = false;
```

```
            }
        }
        catch (NullReferenceException)
        {
            MessageBox.Show("Нет строк для удаления.",
                            "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

    }

    private void ButtonChange_Click(object sender, EventArgs e)
    {
        try
        {
            Train columnNameOfChosenCell = TrainsGrid.CurrentRow.DataBoundItem as Train;

            foreach (DataGridViewRow row in TrainsGrid.Rows)
            {
                row.DefaultCellStyle.BackColor = Color.White;
            }

            // Вывод вспомогательной формы
            var newEntity = new Adding(columnNameOfChosenCell);
            newEntity.Text = "Изменение сущности";
            newEntity.ShowDialog();

            // Изменение сущности

            if (String.IsNullOrEmpty(newEntity.train.Name) ||
String.IsNullOrEmpty(newEntity.train.Departure)
            || String.IsNullOrEmpty(newEntity.train.Arrival) ||
String.IsNullOrEmpty(newEntity.train.Station_dep)
            || String.IsNullOrEmpty(newEntity.train.Station_arr) ||
String.IsNullOrEmpty(newEntity.train.Cost.ToString()))
            {
                throw new NullReferenceException();
            }
            // Сохранение изменений
            db.SaveChanges();
            int changedRowIndex = TrainsGrid.CurrentCell.RowIndex;
            TrainsGrid.Rows[changedRowIndex].DefaultCellStyle.BackColor = Color.GreenYellow;
            MessageBox.Show("Данные успешно изменены и сохранены.", "Информация",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);

        }
        catch (NullReferenceException)
        {
            MessageBox.Show("Вы не ввели данные.", "Ошибка!",
                            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void InfoToolStripMenuItem_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Работа с СУБД SQLite.\n" +
                        "Данная программа хранит расписание поездов и \n" +
                        "позволяет добавлять/удалять направления в базе данных.\n" +
                        "Автор:  Гусев Антон\n" +
                        "Группа: 494\n" +
                        "Учебное заведение: СПбГТИ (ТУ)", "О программе",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void ShowInfoToolStripMenuItem_Click(object sender, EventArgs e)
    {
        if (ShowInfoToolStripMenuItem.Checked)
        {
            ShowInfoToolStripMenuItem.Checked = false;
            Settings.Default.Show = false;
            Settings.Default.Save();
```

```
            }
            else
            {
                ShowInfoToolStripMenuItem.Checked = true;
                Settings.Default.Show = true;
                Settings.Default.Save();

            }
        }

        private void SaveToolStripMenuItem_Click(object sender, EventArgs e)
        {

            if (saveFileDialog.ShowDialog() == DialogResult.Cancel)
                return;

            string fileOutputPath = saveFileDialog.FileName;
            saveFileDialog.FileName = string.Empty;

            string text = SaveInFile.MakeResult(Trains);

            SaveInFile.SaveToFile(fileOutputPath, text);
        }
    }
}
```

[Конец MainForm.cs---]

[Начало Train.cs---]

```
using System;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;

namespace rps4
{
    [Table("trains")]
    public class Train
    {
        [Key]
        public int ID { get; set; }
        public string Name { get; set; }
        public string Departure { get; set; }
        public string Arrival { get; set; }
        public string Station_dep { get; set; }
        public string Station_arr { get; set; }
        public int Cost { get; set; }
    }
}
```

[Конец Train.cs---]

[Начало ApplicationContext.cs---]

```
using System.Data.Entity;

namespace rps4
{
    public class ApplicationContext : DbContext
    {
        public ApplicationContext() : base("DefaultConnection")
        {
        }
        public DbSet<Train> Trains { get; set; }
    }
}
```

[Конец ApplicationContext.cs---]

[Начало SaveInFile.cs---]

```
using System;
using System.ComponentModel;

namespace rps4
{
    class SaveInFile
    {
        public static void SaveToFile(string fileOutputPath, string text)
        {
            System.IO.File.WriteAllText(fileOutputPath, text);
        }


        public static string MakeResult(BindingList<Train> Trains)
        {
            string text = String.Format("{0} {1, 7} {2, 25} {3, 23} {4, 20} {5, 20} {6, 15}\n",
                "Код", "Наименование", "Дата и время отправления", "Дата и время прибытия", "Место
отправления", "Место прибытия", "Стоимость");
            foreach (Train trainRow in Trains)
            {
                text += String.Format("{0} {1, 7} {2, 25} {3, 23} {4, 20} {5, 20} {6, 15}\n",
                    trainRow.ID, trainRow.Name, trainRow.Departure, trainRow.Arrival,
trainRow.Station_dep, trainRow.Station_arr, trainRow.Cost);
            }
            return text;
        }
    }
}
```

[Конец SaveInFile.cs---]

[Конец программы ---]