Разработка приложения автоматизации работы диспетчера касс автовокзала

В данной работе описывается пошаговый процесс создания приложения, которое осуществляет автоматизацию работы диспетчера касс при продаже билетов на автовокзале.

Приложение реализовано в Microsoft Visual Studio на языке программирования C#. База данных Microsoft Access.

Условие задачи

Разработать систему, которая осуществляет автоматизацию работы диспетчера кассы автовокзала.

Входные данные для выполнения работы следующие:

- информация о маршрутах перевозки (номер маршрута, пункт назначения, район, область, расстояние, вес, время отправления, время прибытия);
- информация о проданных билетах на заданный рейс (место, стоимость, время приобретения, фамилия пассажира, паспортные данные пассажира, наличие или отсутствие льгот);
- информация о диспетчере, который обслуживает клиента (фамилия диспетчера, дата рождения, адрес);
- информация о водителе, который осуществляет перевозку за заданным маршрутом (фамилия водителя, дата рождения, паспортные данные);
- информация об автобусе на котором осуществляется перевозка (номер автобуса, модель, номерной знак, максимальное количество посадочных мест);
- номер перевозки.

В процессе работы с приложением формируется база данных перевозок, которые включают:

- данные о маршруте;
- данные о проданных билетах;
- данные о диспетчере, который вносит изменения в базу данных;
- данные об транспортном средстве (автобусе), на котором осуществляется перевозка;
- данные о водителе, который осуществляет перевозку.

Выполнение

1. Создание базы данных

Базу данных проектируем в системе Microsoft Access. Имя файла базы данных даем «Av_Bl.mdb«.
В соответствии с условием задачи, имеем 6 таблиц. Пять таблиц содержат данные о:

- маршрутах перевозки;
- проданных билетах;
- диспетчера, который вносит информацию в базу данных;
- транспортное средство (автобус, маршрутное такси), на котором осуществляется перевозка;
- водителя, который выполняет перевозку.

Назовем эти таблицы именами: «Маршрут«, «Билет«, «Диспетчер«, «Автобус«, «Водитель«.

Шестая таблица есть результатом работы приложения во взаимодействии с базой данных. В этой таблице непосредственно фиксируется перевозка. Назовем ее «Перевозки«.
Таблицы «Маршрут«, «Билет«, «Диспетчер«, «Автобус«, «Водитель» есть первичными по отношению к таблице «Перевозка«.
Структура таблиц следующая.

Таблица «Маршрут».

№ поля	Название поля	Тип	Описание
1	ID_Marshrut	Счетчик	Уникальный идентификатор маршрута
2	Номер маршрута	Текстовый	Номер маршрута
3	Пункт назначения	Текстовый	Пункт назначения (как в фильме)
4	Район	Текстовый	Район населенного пункта назначения
5	Область	Текстовый	Область населенного пункта назначения
6	Расстояние	Числовой	Расстояние между населенными пунктами
7	Вес	Числовой	Весовой коэффициент
8	Время отправки	Дата/Время	Время отправления из населенного пункта 1
9	Время прибытия	Дата/Время	Время прибытия к населенному пункту 2

Таблица «Билет».

№ поля	Название поля	Тип	Описание
1	ID_Bilet	Счетчик	Уникальный идентификатор билета
2	Место	Числовой	Место пассажира
3	Стоимость	Числовой	Стоимость билета
4	Время	Дата/Время	Время приобретения (покупки) билета
5	Ф_И_О	Текстовый	Ф.И.О. пассажира
6	Паспорт	Текстовый	Серия и номер паспорта
7	Льготы	Логический	Пользуется ли пассажир льготами

Таблица «Диспетчер».

№ поля	Название поля	Тип	Описание
1	ID_Dispetcher	Счетчик	Уникальный идентификатор диспетчера
2	Ф_И_О	Текстовый	Фамилия диспетчера
3	Дата рождения	Дата/Время	Дата рождения диспетчера
4	Адрес	Текстовый	Адрес проживания диспетчера

Таблица «Автобус».

№ поля	Название поля	Тип	Описание
1	ID_Avtobus	Счетчик	Уникальный идентификатор транспортного средства (ТС)
2	Номер	Текстовый	Номер (ТС)
3	Модель	Текстовый	Марка (модель) ТС
4	Номерной знак	Текстовый	Номерной знак
5	Количество мест	Числовой	Количество посадочных мест

Таблица «Водитель».

№ поля	Название поля	Тип	Описание
1	ID_Voditel	Счетчик	Уникальный идентификатор водителя
2	Ф_И_О	Текстовый	Фамилия водителя
3	Дата рождения	Дата/Время	Дата рождения водителя
4	Паспорт	Текстовый	Серия и номер паспорта водителя

Таблица «Перевозка».

Таблица "Перевозка".

№ поля	Название поля	Тип	Описание
1	ID_Perevozka	Счетчик	Уникальный идентификатор перевозки
2	Номер	Текстовый	Номер перевозки
3	ID_Marshrut	Числовой	Идентификатор маршрута для данной перевозки
4	ID_Bilet	Числовой	Идентификатор номера купленного билета для данной перевозки
5	ID_Dispetcher	Числовой	Идентификатор номера диспетчера, который заносит данную перевозку в базу данных
6	ID_Avtobus	Числовой	Идентификатор номера транспортного средства, на котором осуществляется перевозка
7	ID_Voditel	Числовой	Идентификатор номера водителя, который осуществляет перевозку

Схема взаимодействия между таблицами базы данных изображена на рисунке 1.

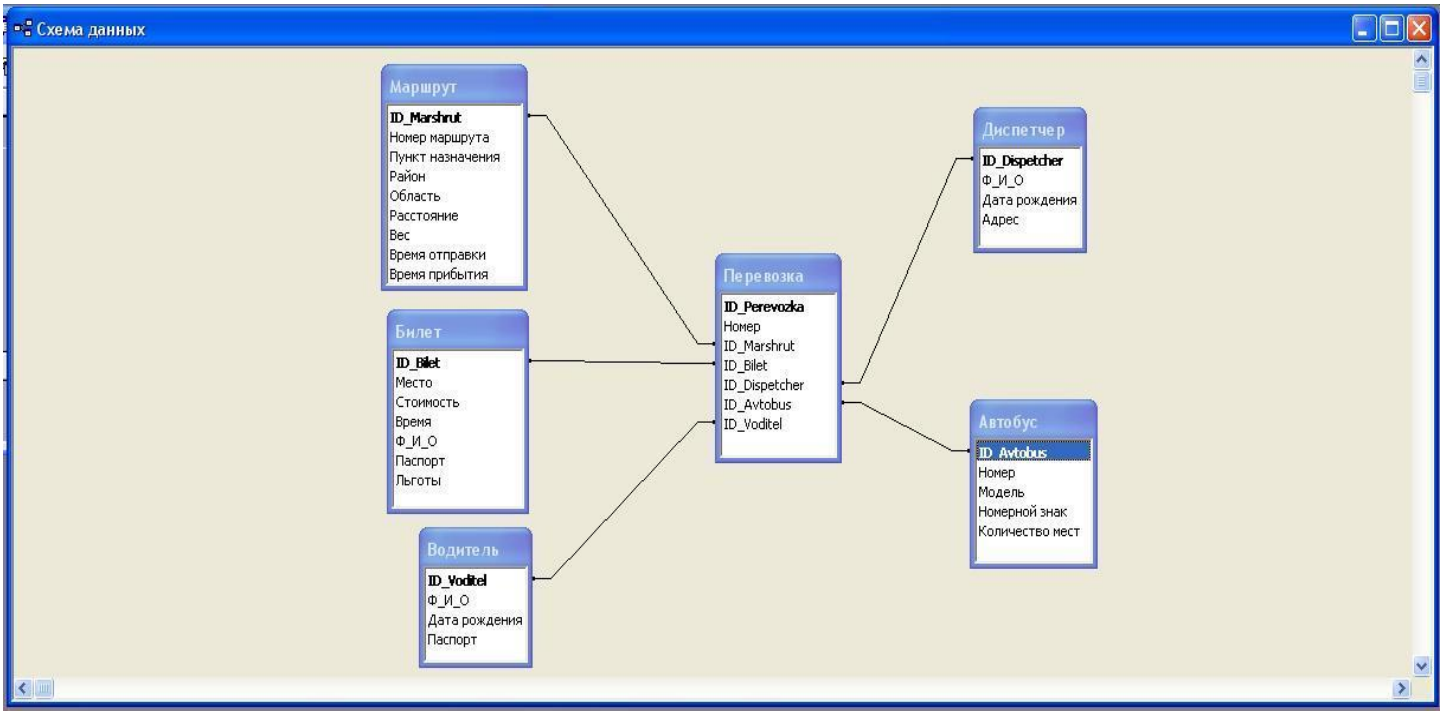

Рис. 1. Схема взаимодействия между таблицами базы данных

3. Создание проекта в Microsoft Visual Studio

Установить шаблон «Visual C#«, тип проекта Windows Forms Application. Сохранить проект в некоторой папке.

Имя приложения даем по умолчанию «WindowsFormsApplication1«.

В результате будет создана пустая форма приложения, которая изображена на рисунке 6.

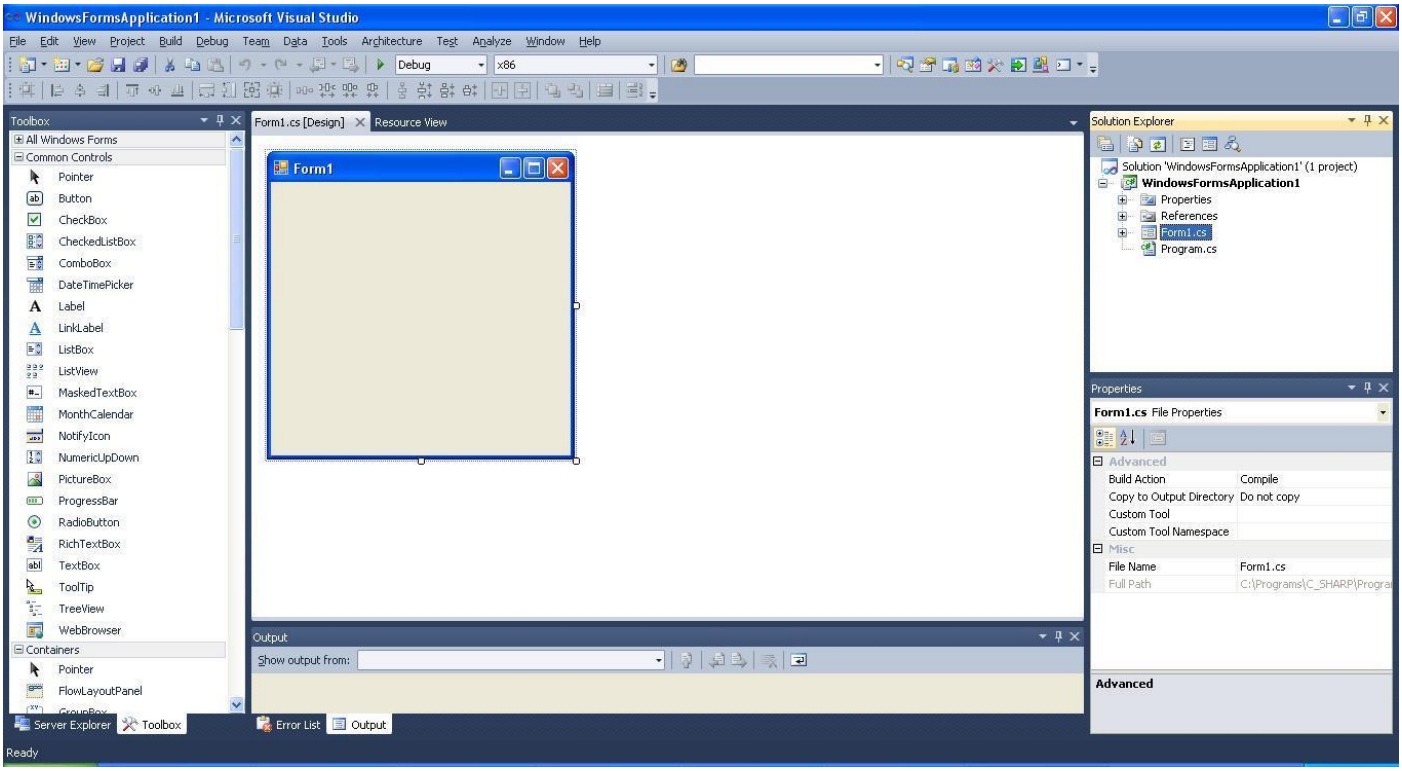

Рис. 6. Начальная форма приложения

4. Подключение базы данных

4.2. Подключение методов оперирования базой данных

Для того, чтобы программно управлять базой данных и получить доступ к методам, которые оперируют базой данных, нужно подключить пространство имен System.Data.OleDb.
Для этого, в основной форме Form1.cs в Solution Explorer выбираем режим просмотра кода (View Code) из контекстного меню (рис. 8) и в начале файла добавляем следующую строку:

```
using System.Data.OleDb;
```

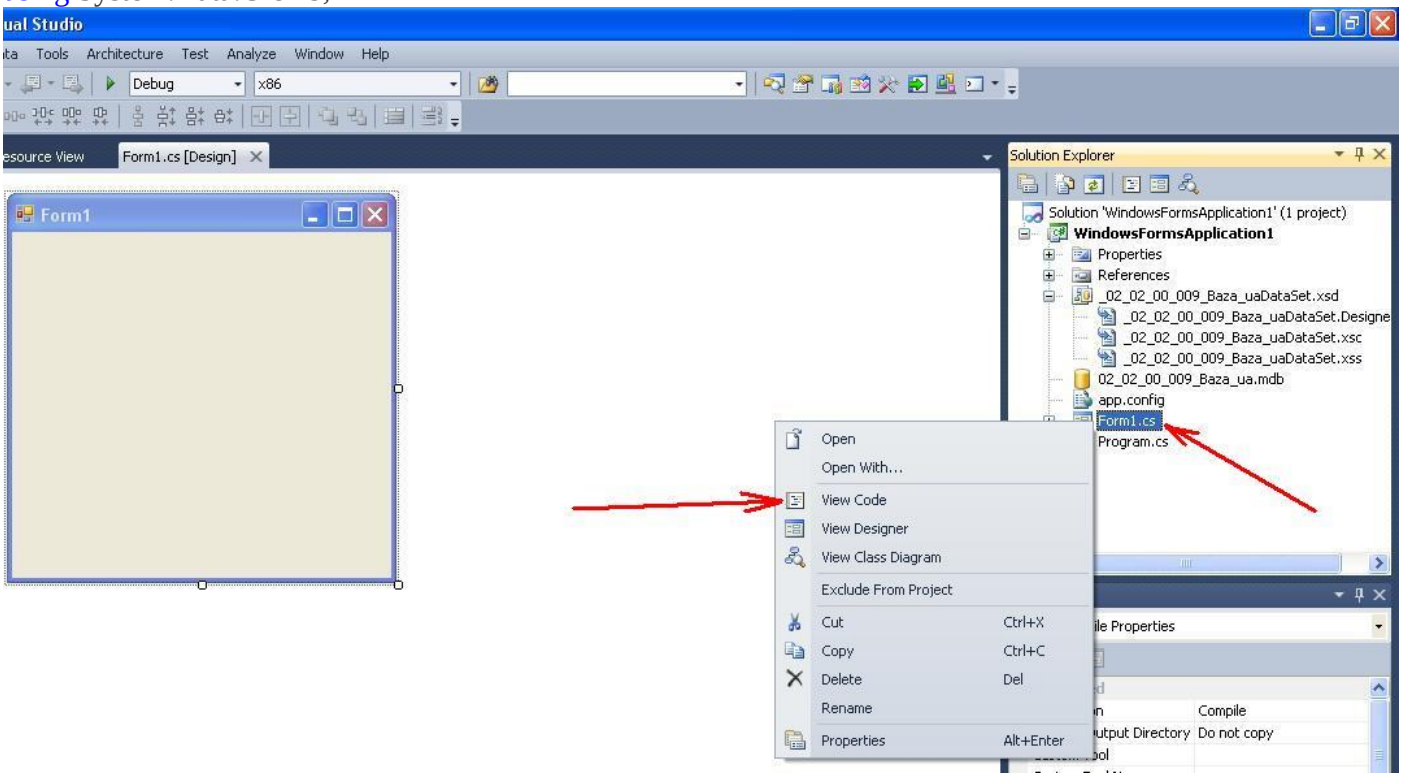

Рис. 8. Вызов режима программного кода формы Form1.cs

Общий вид верхней части файла Form1.cs уже имеет вид:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
```

5. Проектирование формы приложения

Для проектирования формы используются элементы управления (controls) из панели Toolbox (рис. 9).

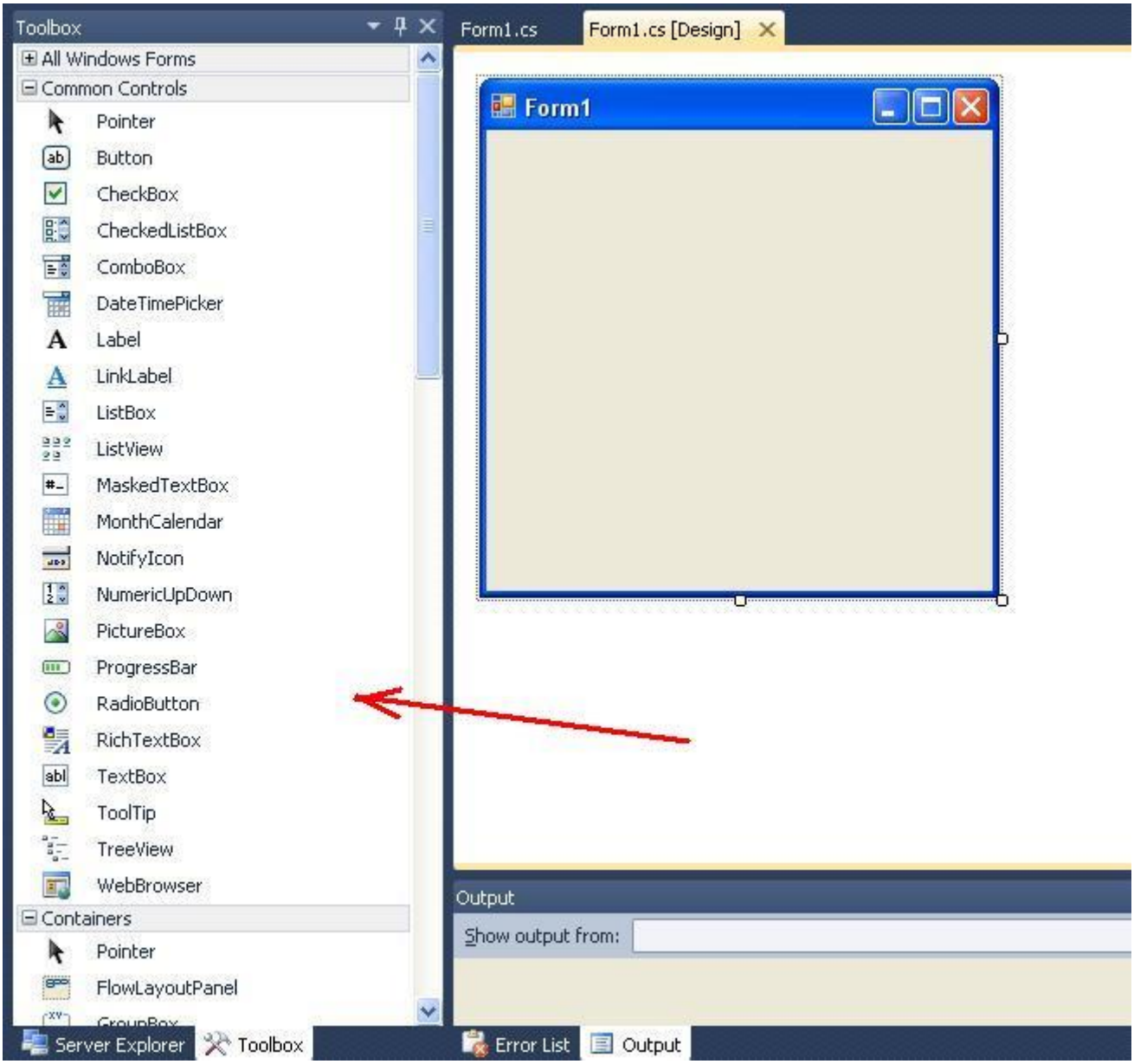

Рис. 9. Панель Toolbox

5.1. Разработка меню

Для построения меню используется элемент управления menuStrip из вкладки «Menus & Toolbars» панели ToolBox (рис. 10).

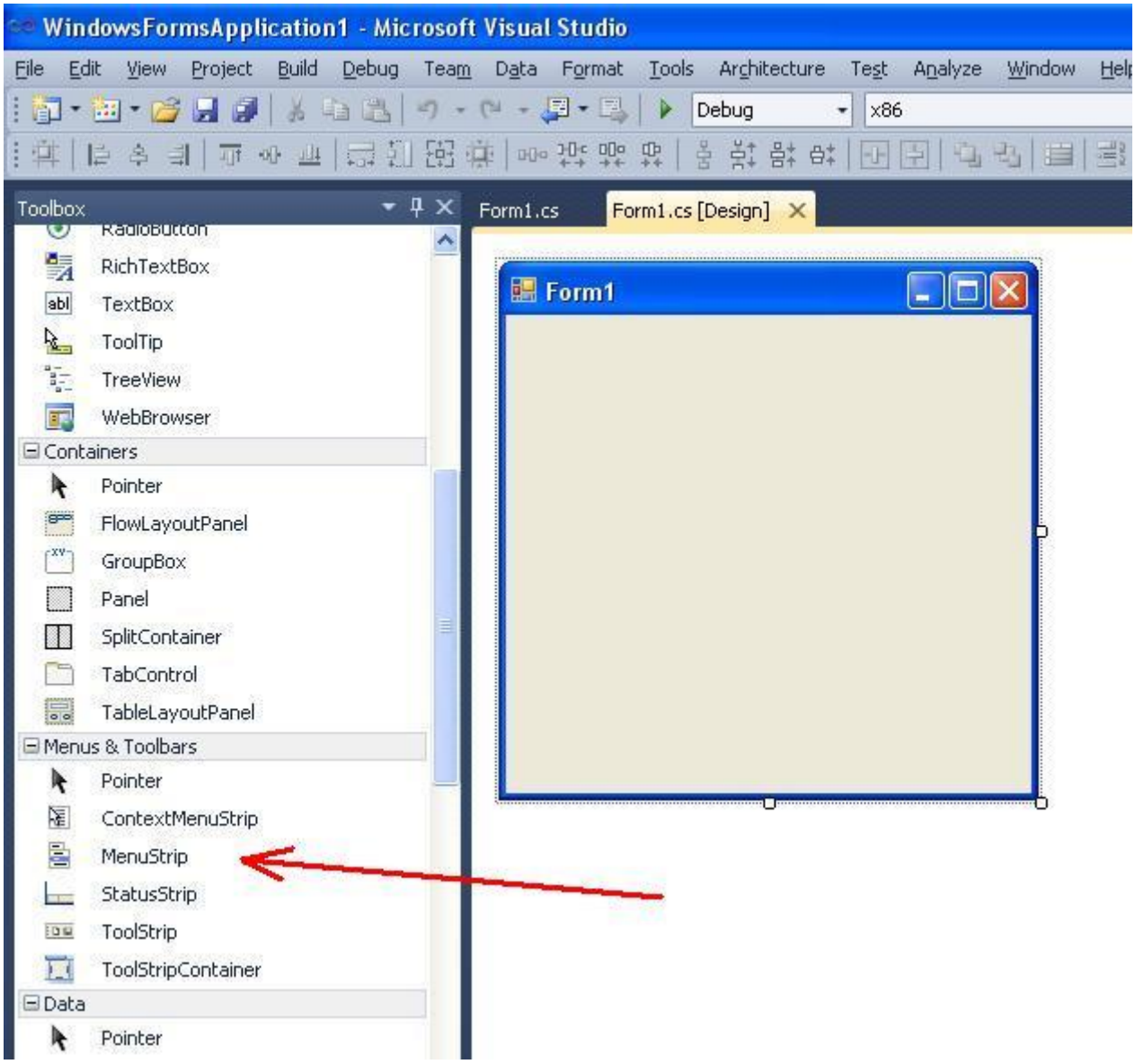

Рис. 10. Элемент управления MenuStrip

Разработка меню с помощью элемента управление MenuStrip подробно описывается здесь. Разрабатываем меню как показано на рисунке 11.

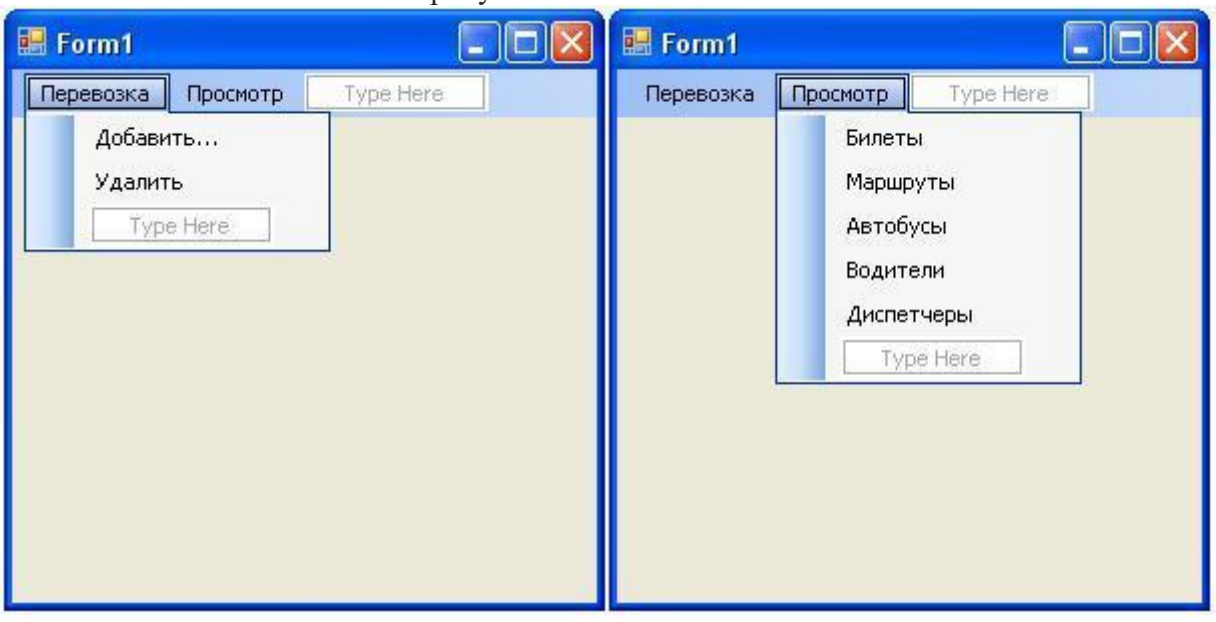

Рис. 11. Разработка меню «Перевозка» и «Просмотр«

5.2. Элементы управления DataGridView

Изменяем размеры формы и выносим на форму два элемента управления DataGridView из вкладки Data панели Toolbox (рис. 12).

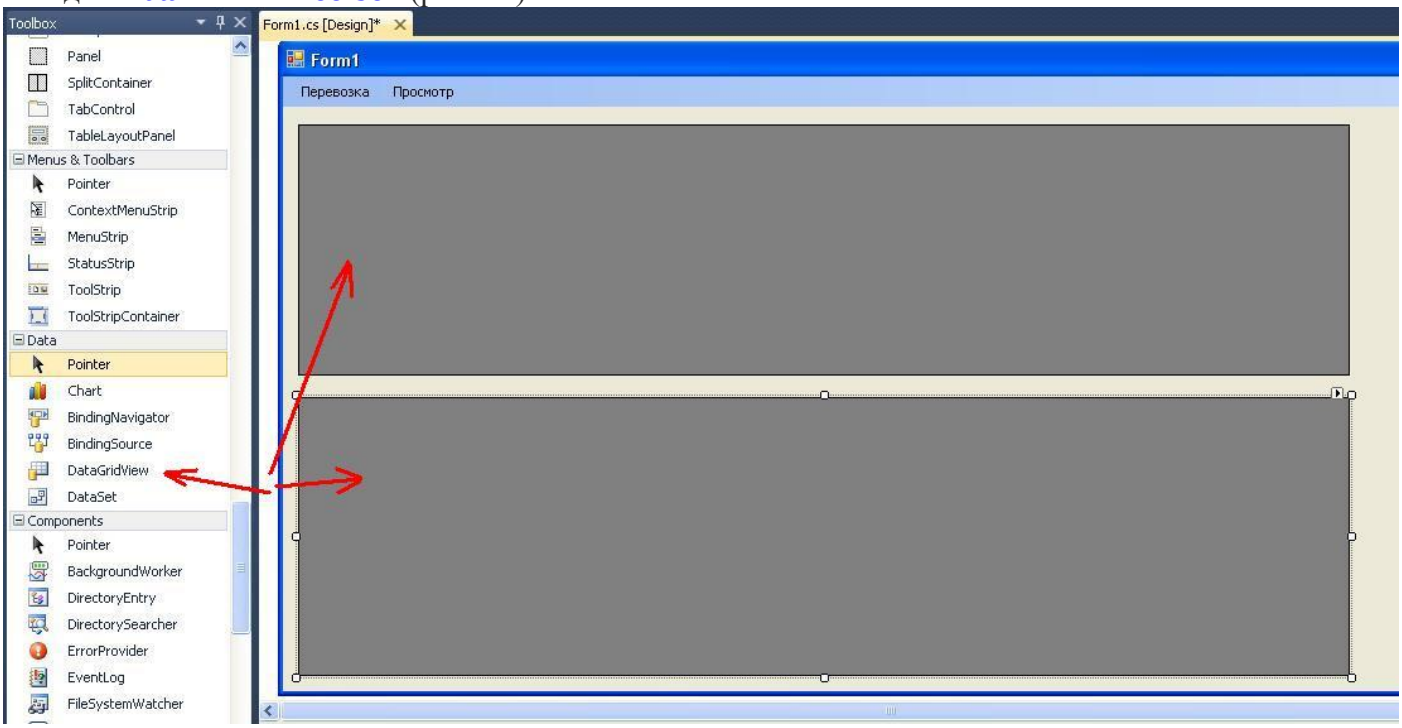

Рис. 12. Элементы управления DataGridView

5.3. Элементы управления типа GroupBox

Активируем вкладку Containers из панели инструментов Toolbox.

Размещаем на форме 4 элемента управления типа GroupBox (рис. 13). В результате получаем 4 объекта с именами groupBox1, groupBox2, groupBox3, groupBox4. С помощью этих имен можно иметь доступ к свойствам (properties) и методам этих объектов.

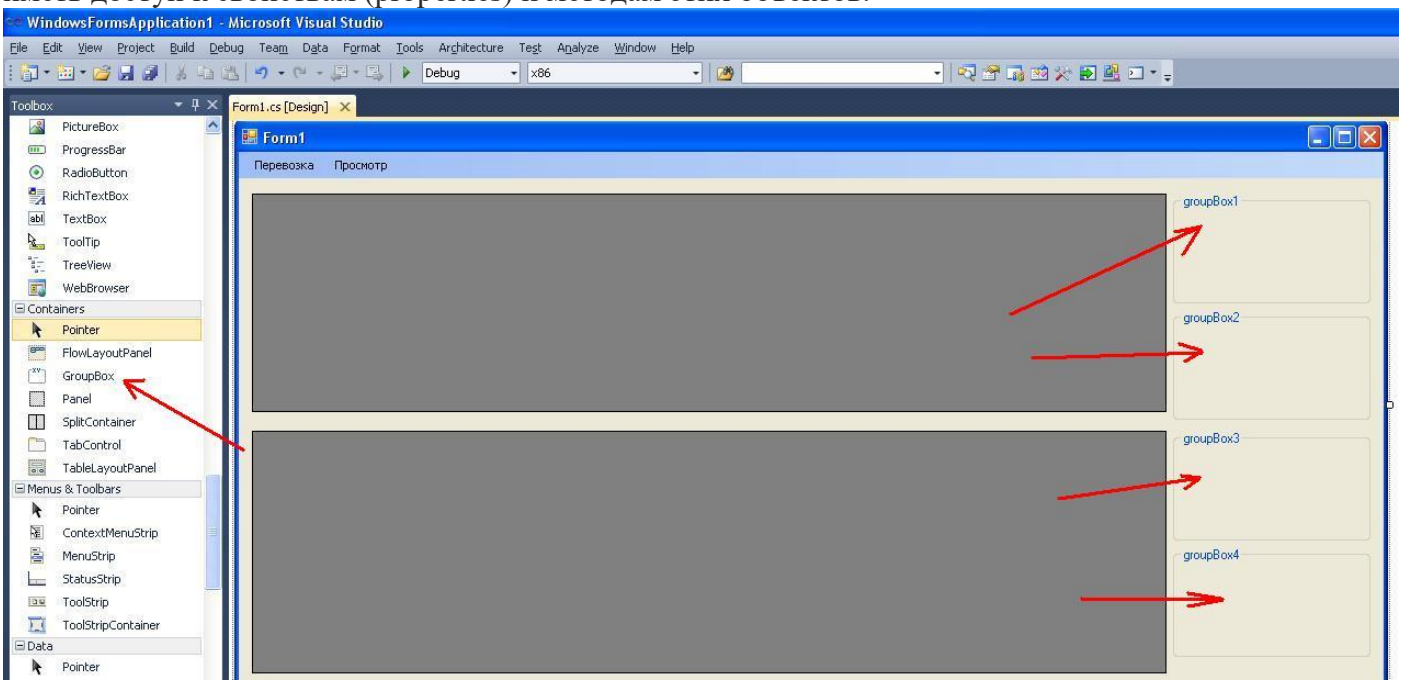

Рис. 13. Элементы управления типа GroupBox

Устанавливаем подпись для объекта groupBox1. Это осуществляется с помощью свойства Text как показано на рисунке 14. Значение свойства Text равно «Фильтр«. В этой группе будут размещаться объекты, которые отвечают за фильтрацию данных в базе данных.

Таким самым образом устанавливаем значения других трех объектов:

– в объекте groupBox2 свойство Text = «Перевозка«;

– в объекте groupBox3 свойство Text = «Просмотр«;

– в объекте groupBox4 свойство Text = «Команды«.

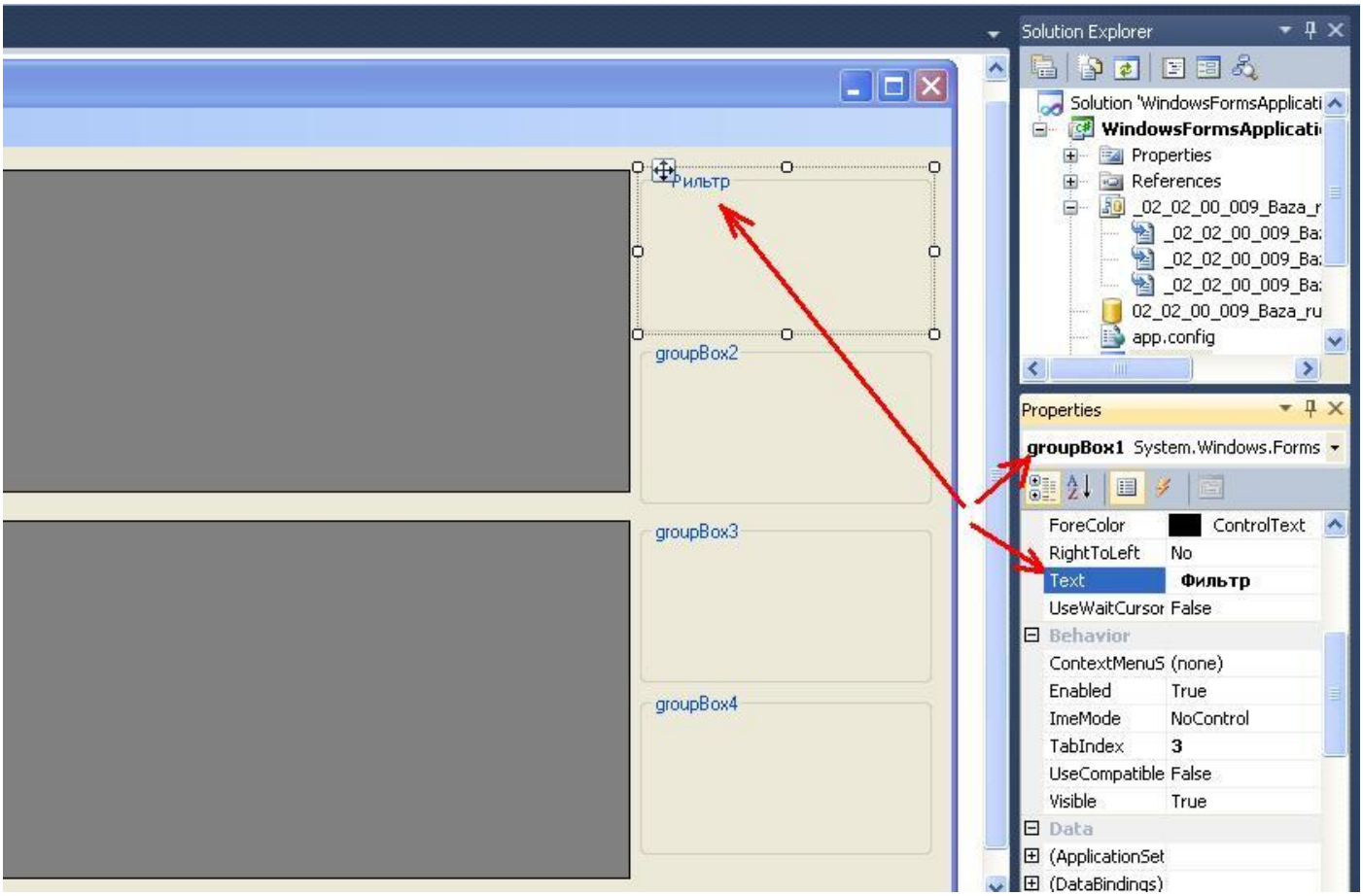

Рис. 14. Свойство Text объекта groupBox1

После внесенных изменений, форма приложения будет иметь вид как показано на рисунке 15.

Рис. 15. Форма приложения после формирования
объектов groupBox1, groupBox2, groupBox3, groupBox4

5.4. Размещение элементов управления типа Label, Button, TextBox и ComboBox.

Элементы управления типа Button и TextBox размещаются в вкладке «Common Controls» панели инструментов Toolbox (рис. 16).

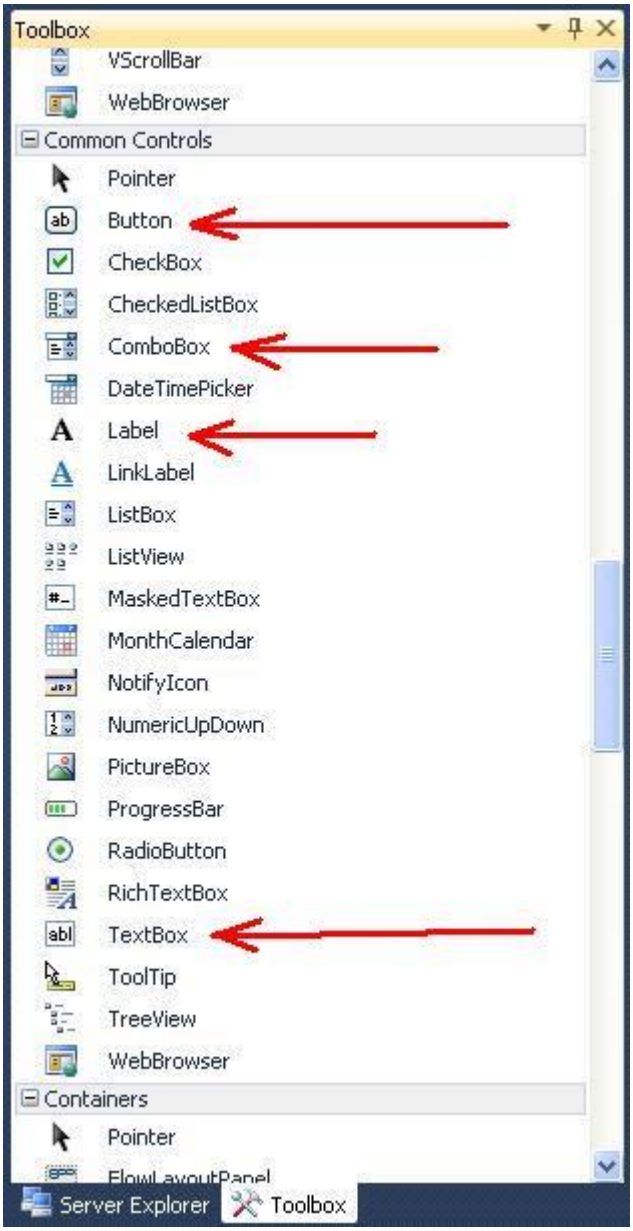

Рис. 16. Элементы управления Label, Button, TextBox и ComboBox из панели инструментов Toolbox

5.5. Размещение элементов управления из группы «Фильтр«

В области объекта groupBox1 («Фильтр«) размещаем один элемент управления типа ComboBox, два элемента типа Label, один элемент управления типа Button и один элемент управления типа TextBox (рис. 17).

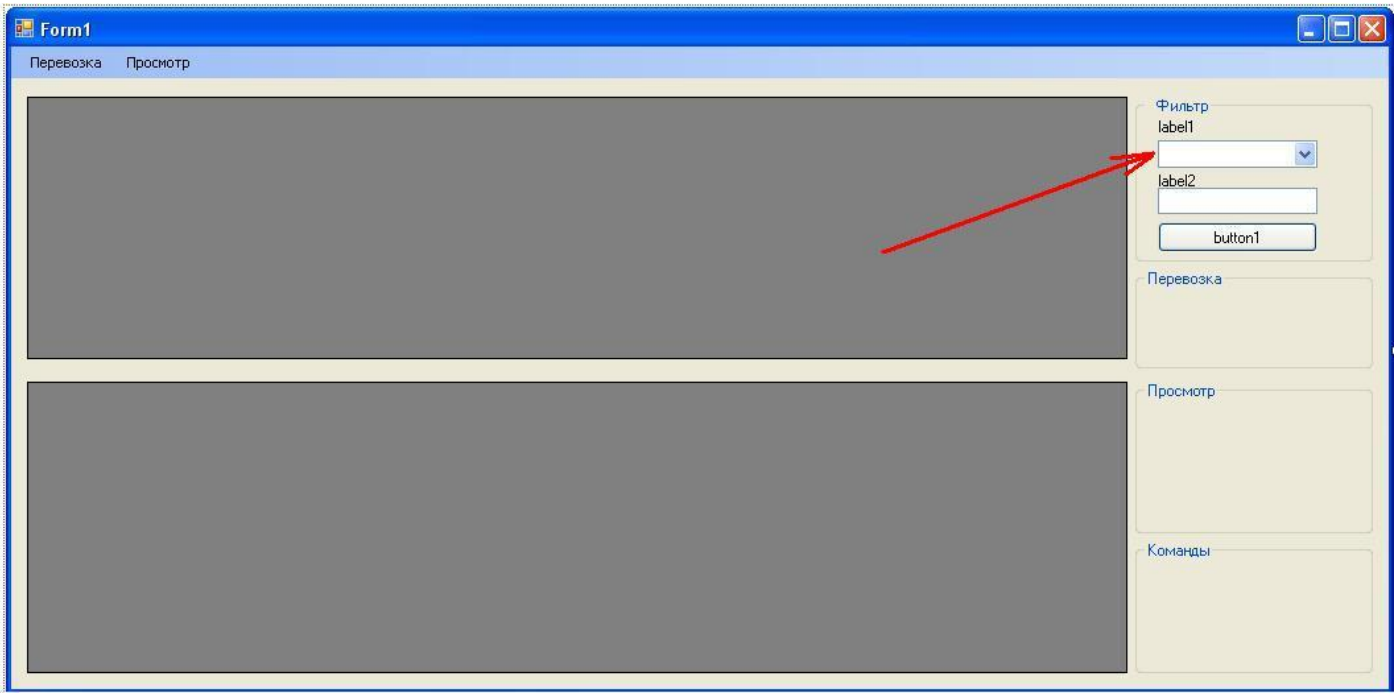

Рис. 17. Элементы управления label1, label2, button1, comboBox1, textBox1 в группе «Фильтр«

Устанавливаем следующие свойства элементов управления:

- в элементе управления label1 свойство Text равно значению «Поле«;
- в элементе управления label2 свойство Text = «Маска«;
- в button1 свойство Text = «Применить«.

Дополнительно изменяем размеры элементов управления button1 и textBox1 для более наглядного вида.

После изменения внесенных изменений окно формы имеет вид (рис. 18).

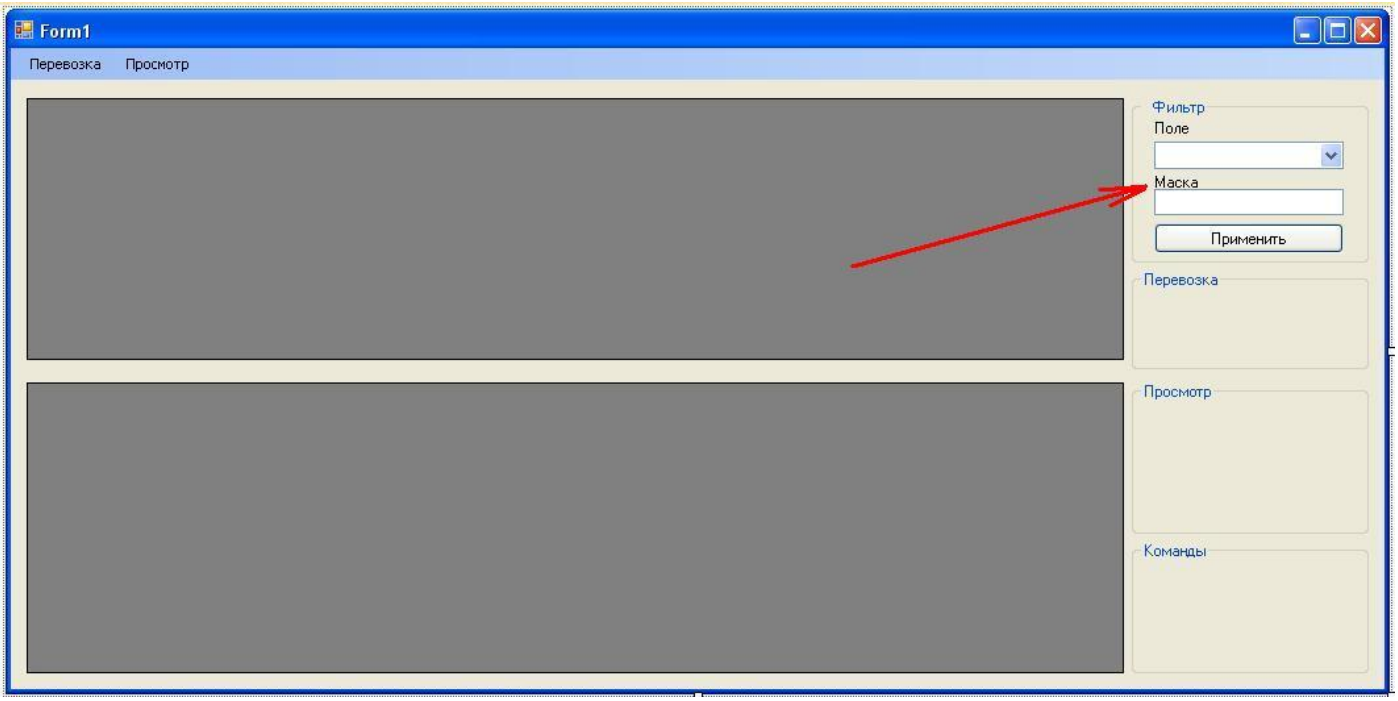

Рис. 18. Окно формы после формирования элементов управление с groupBox1

5.6. Проектирование элементов управления, которые размещаются в группе «Перевозка«

Таким же образом (п. 5.5) формируем группу «Перевозка» (groupBox2). Из панели Toolbox размещаем два элемента управления типа Button и формируем их свойства Text как показано на рисунке 19. Будет создано два объекта типа Button с именами button2 и button3.
Свойство Text объекта button2 устанавливаем в значение «Добавить …«.
Свойство Text объекта button3 устанавливаем в значение «Удалить«.

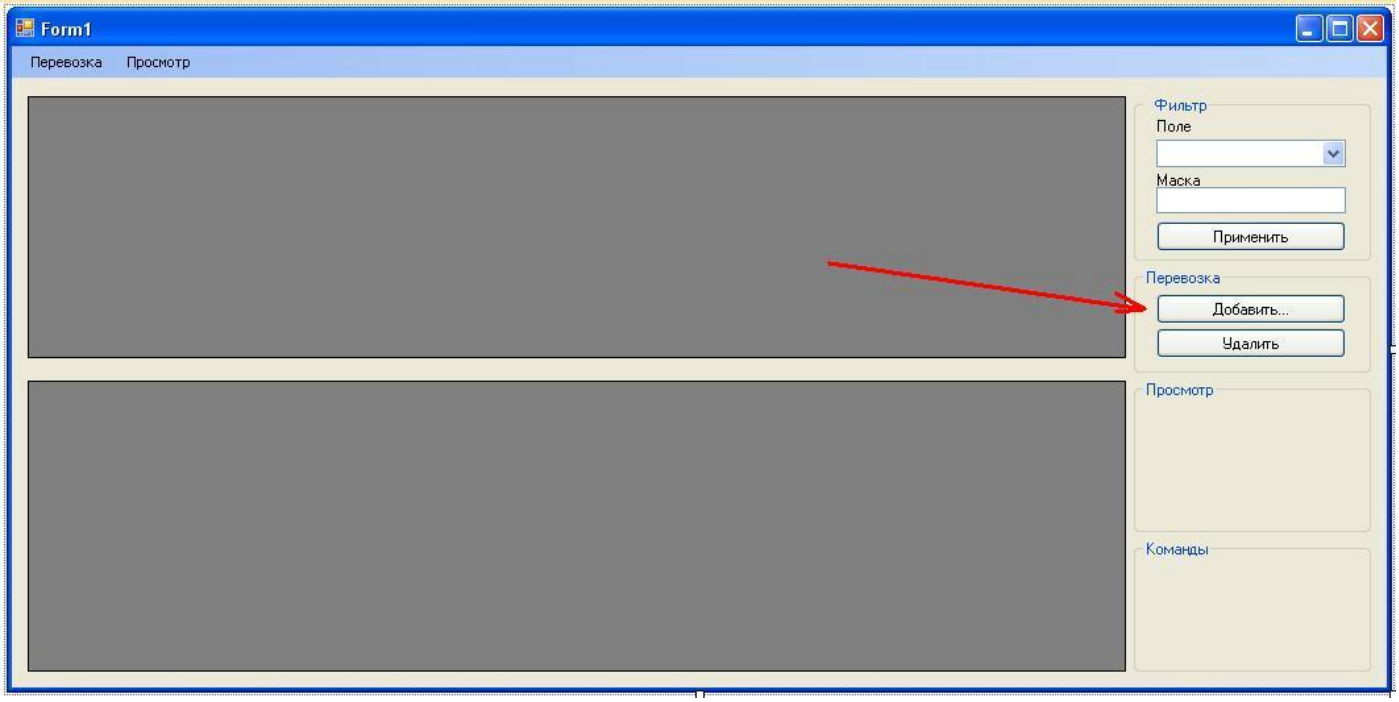

Рис. 19. Разработка элементов управления, которые размещаются в группе «Перевозка«

5.7. Проектирование элементов управления, которые размещаются в группе «Просмотр«

Из панели Toolbox размещаем 5 элементов управления типа Button в области группы «Просмотр» (объект groupbox3).
Корректируем размеры этих элементов управления.

В результате будет создано 5 объектов с именами button4, button5, button6, button7, button8.
В каждом из этих объектов формируем свойство Text:

- в объекте button4 свойство Text = «Билеты«;
- в объекте button5 свойство Text = «Маршруты«;
- в объекте button6 свойство Text = «Автобусы«;
- в объекте button7 свойство Text = «Водители«;
- в объекте button8 свойство Text = «Диспетчеры«.

После проектирования форма приложения с сформированной группой «Просмотр» будет иметь вид как показано на рисунке 20.

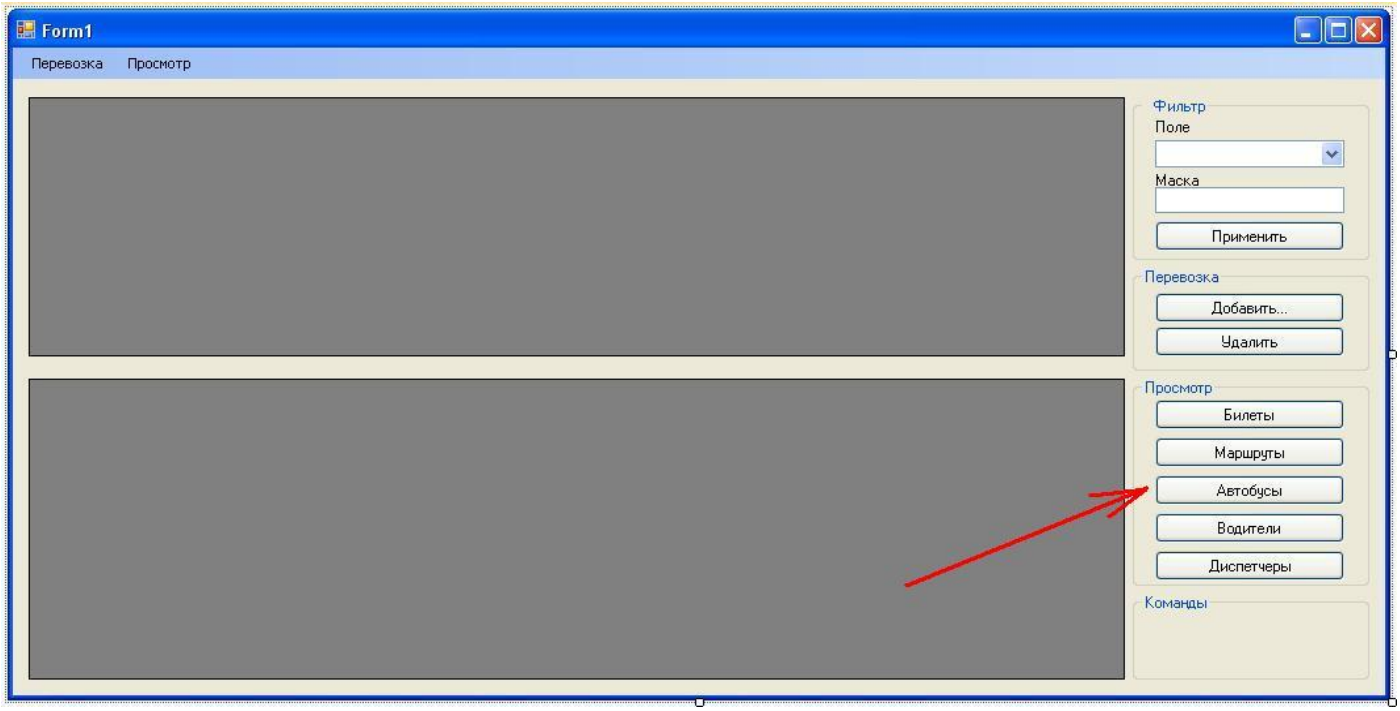

Рис. 20. Элементы управления из группы «Просмотр«

5.8. Проектирование элементов управления, которые размещаются в группе «Команды«

В группе «Команды» разместим два элемента управления типа Button.
Будет создано два объекта с именами button9 и button10.
Установим свойство Text этих объектов соответственно у значения «Добавить…» и «Удалить«.
В результате, форма приложения будет иметь вид как показано на рисунке 21.

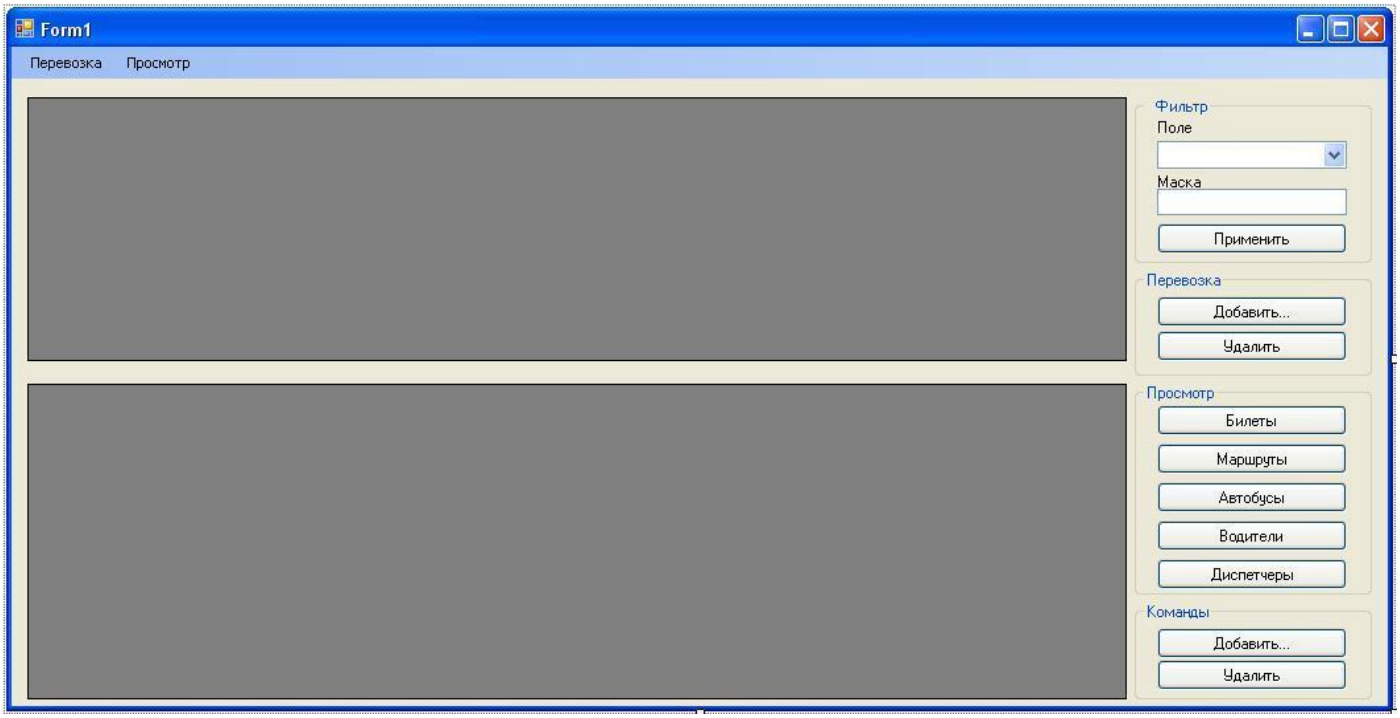

Рис. 21. Форма приложения после размещения всех элементов управления

6. Формирование списка в элементе управления comboBox1

Элемент управления (объект) comboBox1, что используется в группе «Фильтр«, должен содержать поля, к которым будет применяться фильтр для удобного вывода данных из базы данных.
В приложении, в будущем, фильтр будем применять к таким полям:

- номер перевозки;
- номер маршрута;

- пункт назначения;
- пассажир;
- водитель.

Для вызова процесса формирования списка вызовем редактор строк элемента управления comboBox1 (рис. 22). Для этого, после активирования (выделения) comboBox1 делаем клик на кнопке «>» (стрелка вправо). Откроется меню, в котором выбираем «Edit Items…«.

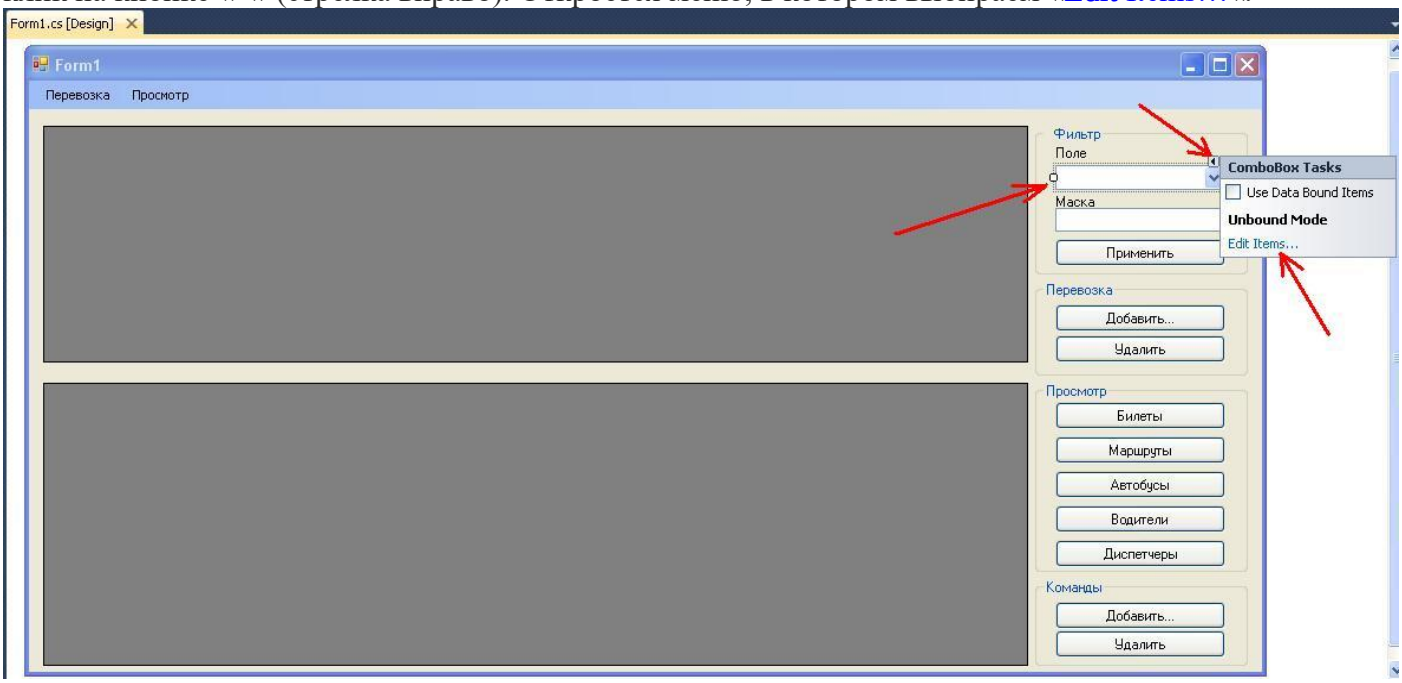

Рис. 22. Вызов меню формирования списка в comboBox1

В результате откроется окно «String Collection Editor«, в котором заносим названия полей, для которых будет установлен фильтр.

Рис. 23. Окно редактора строк элемента управления comboBox1

После нажатия на «OK» будут сформированы строки элемента управление comboBox1.

7. Получение свойства ConnectionString

В следующих программных кодах, где выполняются операции с базой данных, фигурирует свойство «ConnectionString«. Это свойство нужно уметь прочитать правильно.

Свойство «ConnectionString» содержит информацию о провайдере данных и о размещении файла базы данных на диске.

Для получения текста свойства ConnectionString нужно выполнить такие шаги (рис. 24):

- в окне Server Explorer выделить базу данных «02_02_00_009_Baza_ru.mdb«;

- в окне Properties выбрать свойство Connection String и скопировать его в буфер обмена Clipboard.

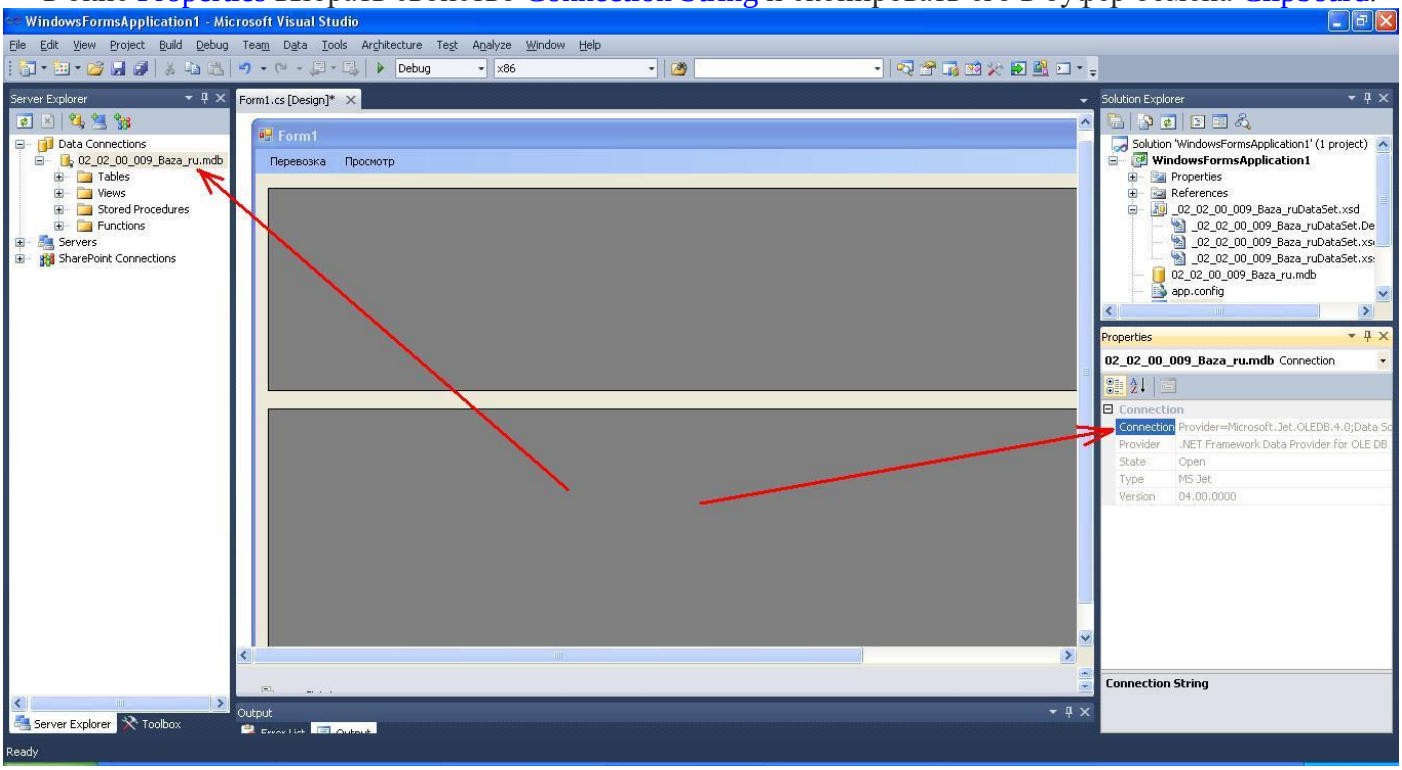

Рис. 24. Чтение свойства «ConnectionString«

В нашем случае, значение свойства «ConnectionString» равно:
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Programs\C_SHARP\Program_02_02_00_009r\WindowsFormsApplication1\WindowsFormsApplication1\02_02_00_009_Baza_ru.mdb
Следует заметить следующее. При использовании строки «ConnectionString» в тексте программы на языке C#, символ '\' нужно заменить на '\\' в соответствии с синтаксисом языка.

8. Добавление внутренних переменных общего доступа к форме «Form1.cs«

Основной форме приложения соответствует класс Form1. При написании программного кода в классе будут использованы некоторые внутренние переменные общего назначения. Это переменные «ConnectionString» и act_table.
Первая переменная будет содержать строку соединения с базой данных «ConnectionString«.
Вторая переменная содержит номер активной таблицы, которая будет выводиться в dataGridView2.
Соответствие между номерами и таблицами базы данных следующее:

- 1 – таблица «Билет«;
- 2 – таблица «Маршрут«;
- 3 – таблица «Автобус«;
- 4 – таблица «Водитель«;
- 5 – таблица «Диспетчер«.

Осуществим добавление текста свойства «ConnectionString» в текст класса главной формы программы.
С помощью Solution Explorer откроем файл Form1.cs командой «View Code» (рис. 25).

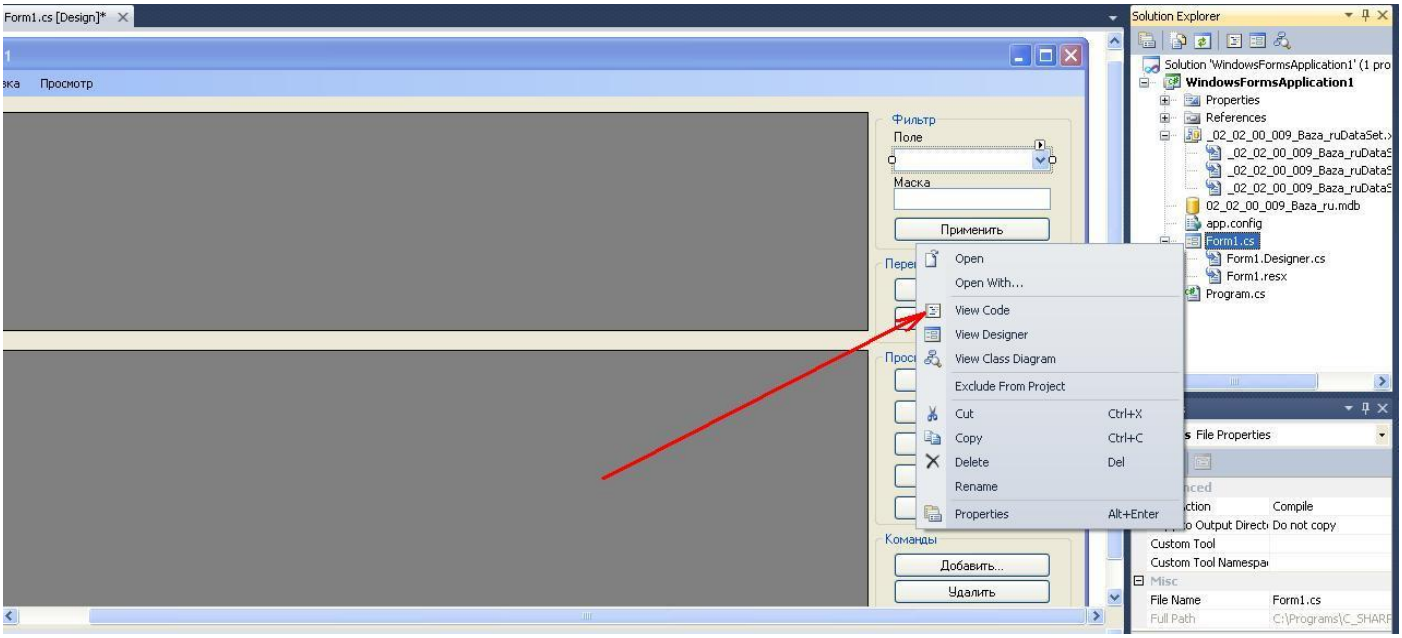

Рис. 25. Вызов команды перехода к просмотру программного кода формы Form1

В результате будет создана вкладка с программным кодом, который описан в файле «Form1.cs«. Листинг файла формы следующий:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }
   }
}
```

Модифицируем этот листинг таким образом, чтобы в нем были добавлены две переменные ConnectionString и act_table. Переменную ConnectionString описываем как public, так как она будет использоваться в других классах (формах). Для удобного отображения в окне проектирования, строка переменной ConnectionString разбита на части.

```
public string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" +
          "Data Source=C:\\Programs\\C_SHARP\\Program_02_02_00_009r\\" +
         "WindowsFormsApplication1\WindowsFormsApplication1\02_02_00_009_Baza_ru.mdb";

private int act_table = 1; // активная таблица (1-билеты, 2-маршруты, 3-автобусы, 4-водители, 5 - диспетчеры
```

Листинг класса Form1 основной формы приложения имеет вид:

```
...
public partial class Form1 : Form
{
 string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" +
   "Data Source=C:\\Programs\\C_SHARP\\Program_02_02_00_009r\\" +
```

```
"WindowsFormsApplication1\\WindowsFormsApplication1\\02_02_00_009_Baza_ru.mdb";
  private int act_table = 1; // активная таблица (1-билеты, 2-маршруты, 3-автобусы, 4-водители, 5 -
диспетчеры
  public Form1()
  {
    InitializeComponent();
  }
}
...
```

9. Вывод данных в dataGridView1

Данные которые будут выводиться в компоненте dataGridView1 будут получены из разных таблиц. Чтобы происходило фильтрование данных, нужно учитывать строку, которая выделена в comboBox1 и маску которая набрана в элементе управления textBox1. В comboBox1 пользователь имеет возможность выбирать поле, на основании которого будет происходить фильтрация данных.

9.1. Текст SQL-запроса

В таблице приведены поля, которые будут выводиться в dataGridView1.

№ перевозки	№ маршрута	Пункт назначения	Время отправки	Время прибытия	Место	Ф.И.О. пассажира	Стоимость	Водитель
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...

Поля, которые нужно выводить в соответствии с синтаксисом языка SQL следующие:

```
[Перевозка].[Номер]
[Маршрут].[Номер маршрута]
[Маршрут].[Пункт назначения]
[Маршрут].[Время отправки]
[Маршрут].[Время прибытия]
[Билет].[Место]
[Билет].[Ф_И_О]
[Билет].[Стоимость]
[Водитель].[Ф_И_О]
```

Запрос на языке SQL для построения новой таблицы имеет следующий вид:

```
SELECT
  [Перевозка].[Номер]
  [Маршрут].[Номер маршрута]
  [Маршрут].[Пункт назначения]
  [Маршрут].[Время отправки]
  [Маршрут].[Время прибытия]
  [Билет].[Место]
  [Билет].[Ф_И_О]
  [Билет].[Стоимость]
  [Водитель].[Ф_И_О]
FROM
  [Перевозка],
  [Маршрут],
  [Билет],
  [Водитель]
WHERE
  ([Перевозка].[ID_Marshrut]=[Маршрут].[ID_Marshrut]) AND
  ([Перевозка].[ID_Bilet] = [Билет].[ID_Bilet]) AND
  ([Перевозка].[ID_Voditel] = [Водитель].[ID_Voditel])
```

В этом запросе не учитывается фильтр. Для учета фильтра, к запросу нужно добавить соответствующую строку. В приложении эта строка будет добавляться программно.

Например, для добавления фильтра по номеру перевозки, к тексту SQL-запроса нужно добавить следующую строку:

```
 AND ([Перевозка].[Номер] = '" + textBox1.Text + "')
```
где textbox1.Text – текст фильтра в компоненте textbox1.

9.2. Программирование события клика на кнопке «Применить»

Вызов вывода данных на основе фильтра осуществляется после нажатия на кнопке «Применить». Процесс программирования события в MS Visual Studio подробно описан здесь. Вывод базы данных в элементе управления dataGridView описывается здесь. Программируем событие клика на кнопке «Применить«.

Обработчик события клика на кнопке «Применить» имеет следующий вид.

```
private void button1_Click(object sender, EventArgs e)
{
  string CommandText = "SELECT " +
    "[Перевозка].[Номер], " +
    "[Маршрут].[Номер маршрута], " +
    "[Маршрут].[Пункт назначения], " +
    "[Маршрут].[Время отправки], " +
    "[Маршрут].[Время прибытия], " +
    "[Билет].[Место], " +
    "[Билет].[Ф_И_О], " +
    "[Билет].[Стоимость], " +
    "[Водитель].[Ф_И_О] " +
  "FROM " +
    "[Перевозка], " +
    "[Маршрут], " +
    "[Билет], " +
    "[Водитель] " +
  "WHERE " +
    "([Перевозка].[ID_Marshrut]=[Маршрут].[ID_Marshrut]) AND " +
    "([Перевозка].[ID_Bilet] = [Билет].[ID_Bilet]) AND " +
    "([Перевозка].[ID_Voditel] = [Водитель].[ID_Voditel]) ";

  if (textBox1.Text != "")  // если набран текст в поле фильтра
  {
    if (comboBox1.SelectedIndex == 0) // № перевозки
      CommandText = CommandText + " AND ([Перевозка].[Номер] = '" + textBox1.Text + "')";
    if (comboBox1.SelectedIndex == 1) // № маршрута
      CommandText = CommandText + " AND (Маршрут.[Номер маршрута] = '" + textBox1.Text + "') ";

    if (comboBox1.SelectedIndex == 2) // Пункт назначения
      CommandText = CommandText + " AND (Маршрут.[Пункт назначения] LIKE '" + textBox1.Text + "%')
";
    if (comboBox1.SelectedIndex == 3) // Пассажир
      CommandText = CommandText + " AND (Билет.[Ф_И_О] LIKE '" + textBox1.Text + "%') ";
    if (comboBox1.SelectedIndex == 4) // Водитель
      CommandText = CommandText + " AND ([Водитель].[Ф_И_О] LIKE '" + textBox1.Text + "%') ";
  }

  OleDbDataAdapter dataAdapter = new   OleDbDataAdapter(CommandText, ConnectionString);
  DataSet ds = new DataSet();
  dataAdapter.Fill(ds, "[Перевозка]");
  dataGridView1.DataSource = ds.Tables["[Перевозка]"].DefaultView;
}
```

Сначала, в переменную CommandText, добавляется строка SQL-запроса которая выводит все данные из базы данных. Потом, в зависимости от фильтра, в значение CommandText дописывается соответствующая строка, завершающая SQL-команду.

После построения CommandText, создается объект типа OleDbDataAdapter, который пересылает наборы данных с вызывающего процесса и обратно. Адаптеры данных содержат набор с четырех внутренних объектов команд. Это команды чтения, вставки, изменения и удаления информации. Как видно из программного кода, конструктор объекта получает входными параметрами строку запроса на языке SQL и строку подключения к базе данных. Таким образом объект адаптера данных связан с нашей базой данных.
После создания адаптера данных (OleDbDataAdapter) создаем объект типа DataSet (набор данных):
DataSet ds = new DataSet();
Набор данных представляет собой своего рода промежуточный буфер для данных, которые могут отображаться. Набор данных представляет удобный механизм чтения и обновления данных и инкапсулирует множество таблиц и связей между ними.

Следующая команда – это заполнение набора данных (переменная ds) значениями записей из базы данных на основании SQL-запроса, который помещается в адаптере данных ds помощью метода Fill():
dataAdapter.Fill(ds, "[Перевозка]");
Для отображения (визуализации) данных, сформированных на основе SQL-запроса, нужно чтобы свойство DataSource компонента dataGridView1 ссылалось на некоторую таблицу набора данных ds. Например, таблицу «Перевозка«. Программный код операции отображения данных имеет следующий вид:
dataGridView1.DataSource = ds.Tables["[Перевозка]"].DefaultView;
После этого данные из таблицы «Перевозка» отобразятся на форме.

9.3. Добавление метода button1_Click() в метод Form1_Load()

Метод Form_Load() вызывается в момент загрузки формы. Дописываем в этот метод вызов метода обработки события button1_Click().
На данный момент листинг метода следующий.

```
private void Form1_Load(object sender, EventArgs e)
{
   comboBox1.SelectedIndex = 0;
   button1_Click(sender, e);
}
```

10. Разработка формы добавления перевозки и создание его программного кода

Добавление новой перевозки выполняется нажатием кнопки «Добавить…» из группы «Перевозка«. Для того, чтобы создавать поля, которые используются в отображении перевозки (элемент управление dataGridView1), нужно создать соответствующую форму.

10.1. Создание формы «Form2.cs«

Процесс создания новой формы в MS Visual Studio на языке C# подробно описан здесь.
Создание новой формы выполняется командой

```
Project -> Add Windows Form...
```

В открывшемся окне, выбираем «Windows Form«. Имя формы оставляем «Form2.cs«.
В результате, в «Solution Explorer» получим файлы «Form2.cs» и «Form2.Designer.cs«.
На рисунке 26 отображена форма, отображающая данные для перевозки.

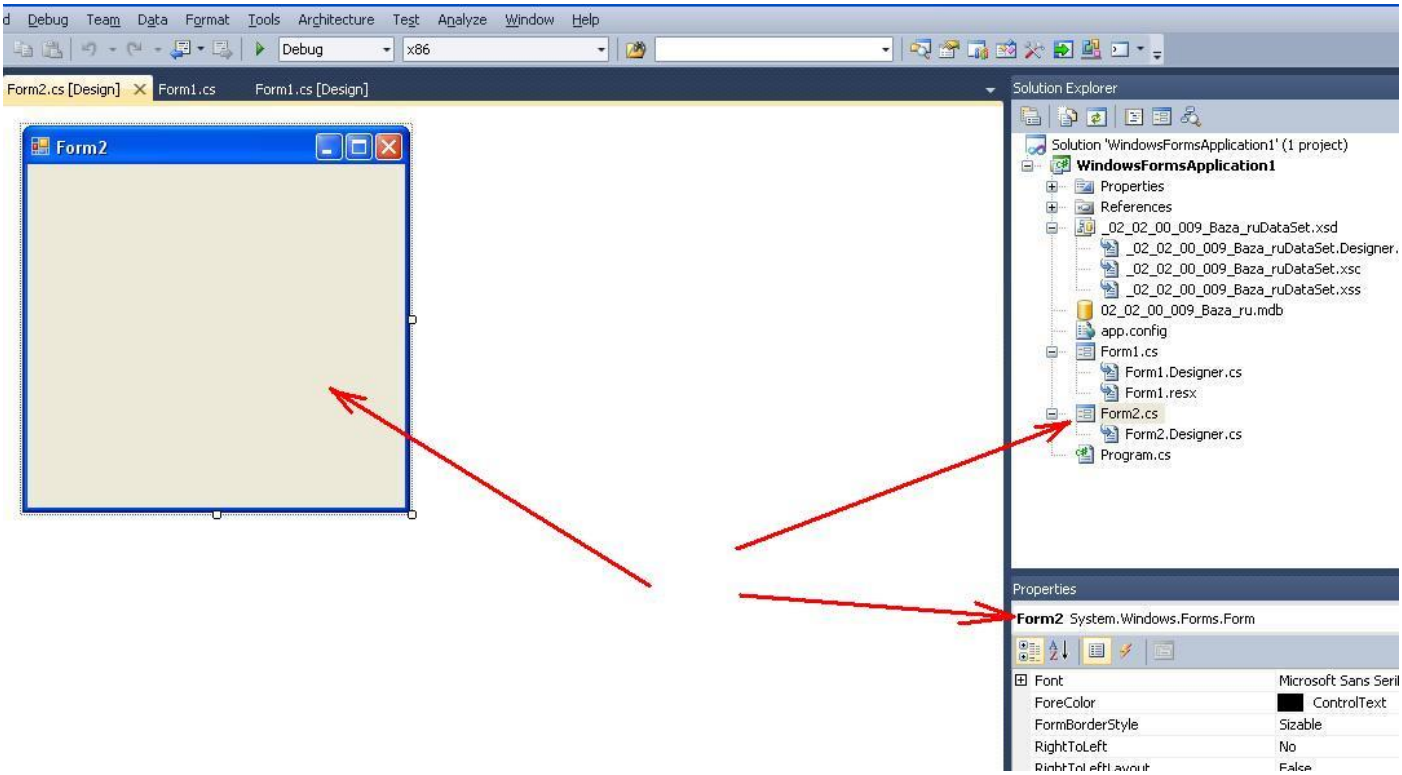

Рис. 26. Создание формы «Form2.cs«

Изменяем размеры формы «Form2.cs» и размещаем на форме элементы управления так как изображено на рисунке 27. Красным цветом выделены названия элементов управления, которые будут использоваться в программном коде.

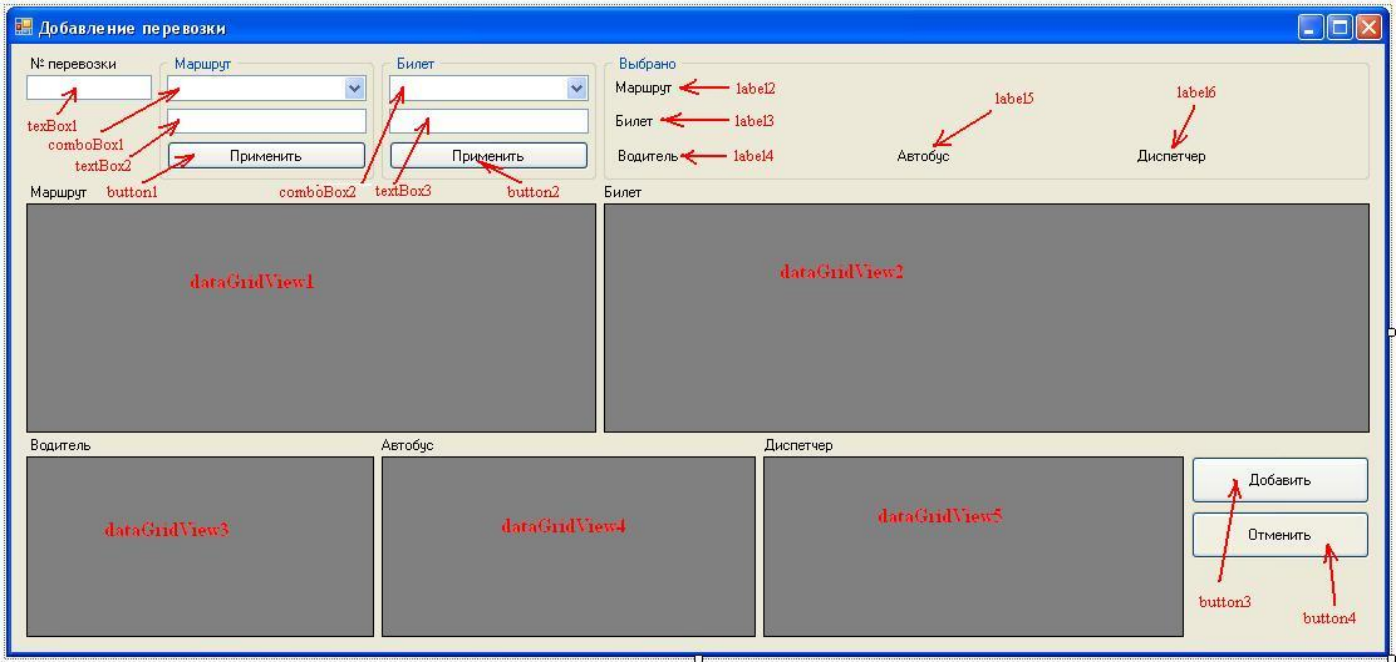

Рис. 27. Форма «Form2.cs» с описанием названий элементов управления

В целом, форма «Form2.cs» имеет вид, как показано на рисунке 28.

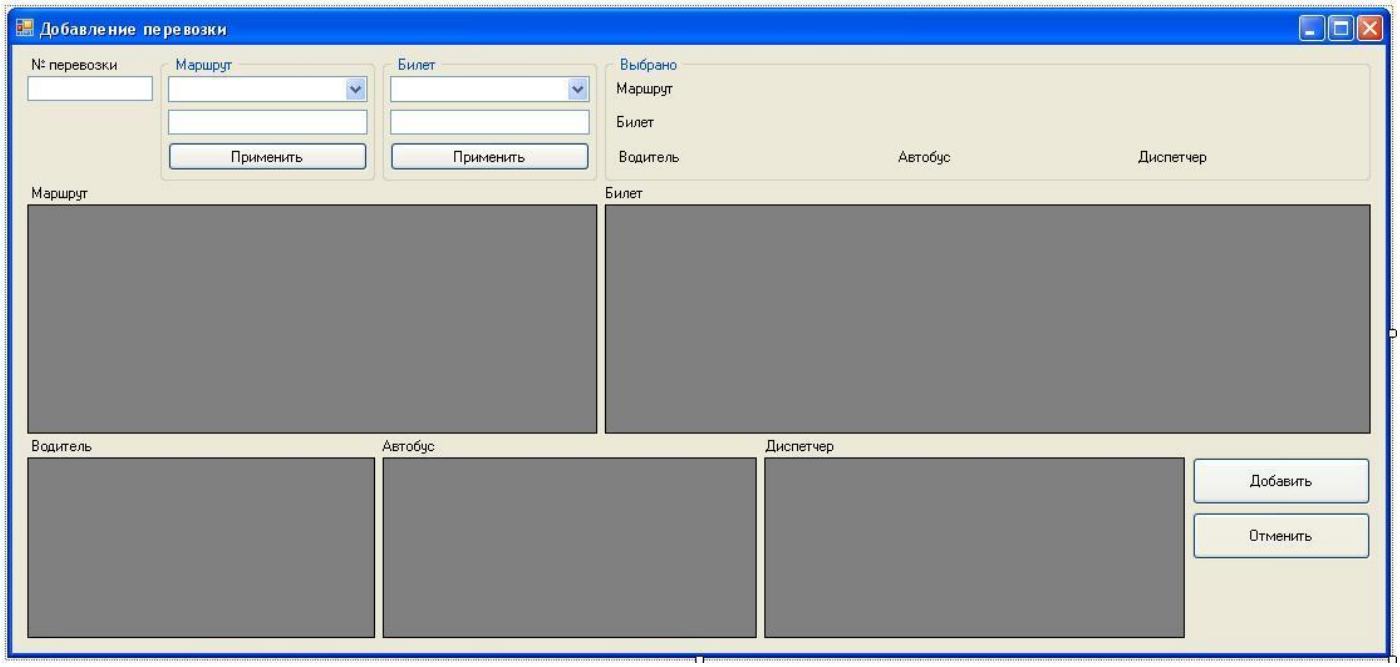

Рис. 28. Форма «Form2.cs«

После создания самой формы будет создан класс формы с именем Form2. Листинг описания класса формы Form2 следующий:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
   public partial class Form2 : Form
   {
      public Form2()
      {
         InitializeComponent();
      }
}
```

10.2. Программирование событий клика на кнопках «Добавить» и «Отменить«

Вызовем событие Click, которое генерируется при клику на кнопке «Добавить» формы Form2. Процесс программирования нового события подробно описан здесь.

В результате, в классе формы будет сгенерирован код, в котором нужно вставить текст обработчика события.

Листинг метода обработки события клика на кнопке «Добавить» следующий:

```
private void button3_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.OK;
}
```

Листинг метода обработки события клика на кнопке «Отменить«:

```
private void button4_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.No;
}
```

10.3. Программирование методов отображения данных в элементах управления dataGridView1, dataGridView2, dataGridView3, dataGridView4, dataGridView5

Для того, чтобы иметь доступ к методам и свойствам которые используются при работе с базами данных, в файле «Form2.cs» подключаем пространство имен (см. **п. 4.2**):

```
using System.Data.OleDb;
```

Для отображения данных в элементах управления типа DataGridView создаем дополнительный метод Get_Table(), который получает входящими два параметра. Параметр table_name задает название таблицы в базе данных. Параметр num_dG задает номер элемента управления типа DataGridView – от 1 до 5. Например, если num_dG=2, то это значит, что отображаются данные в элементе управления dataGridView2.

Листинг метода Get_Table() следующий:

```
private void Get_Table(string table_name, int num_dG)
{
    Form1 f1 = new Form1();
    string CommandText = "SELECT * FROM ";
    CommandText += table_name;
    OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, f1.ConnectionString);
    DataSet ds = new DataSet();
    dataAdapter.Fill(ds, table_name);
    if (num_dG == 1) dataGridView1.DataSource = ds.Tables[table_name].DefaultView;
    if (num_dG == 2) dataGridView2.DataSource = ds.Tables[table_name].DefaultView;
    if (num_dG == 3) dataGridView3.DataSource = ds.Tables[table_name].DefaultView;
    if (num_dG == 4) dataGridView4.DataSource = ds.Tables[table_name].DefaultView;
    if (num_dG == 5) dataGridView5.DataSource = ds.Tables[table_name].DefaultView;
}
```

Следующим шагом программируем событие Load, которое вызовется при загрузке формы Form2. Этому событию буде соответствовать метод Form2_Load().

Листинг метода Form2_Load() следующий:

```
private void Form2_Load(object sender, EventArgs e)
{
   Get_Table("Маршрут", 1); // заполняем таблицу "Маршрут"
   Get_Table("Билет", 2);   // заполняем таблицу "Билет"
   Get_Table("Водитель", 3);
   Get_Table("Автобус", 4);
   Get_Table("Диспетчер", 5);
   textBox1.Text = "0";
   comboBox1.SelectedIndex = 0;
   comboBox2.SelectedIndex = 0;
}
```

В методе Form2_Load() поочередно вызовется метод Get_Table(), что заполняет все 5 элементов управления типа DataGridView.

10.4. Программирование обработчиков событий изменения активной ячейки в компонентах dataGridView1, dataGridView2, dataGridView3, dataGridView4, dataGridView5

Во время изменения активной ячейки в элементах управления типа DataGridView, нужно выводить соответствующий текст в элементах управления (см. рис. 27):

- label2 (Маршрут);
- label3 (Билет);
- label4 (Водитель);
- label5 (Автобус);
- label6 (Диспетчер).

При изменении активной ячейки в элементах управления типа DataGridView генерируется событие CellEnter (рис. 29).

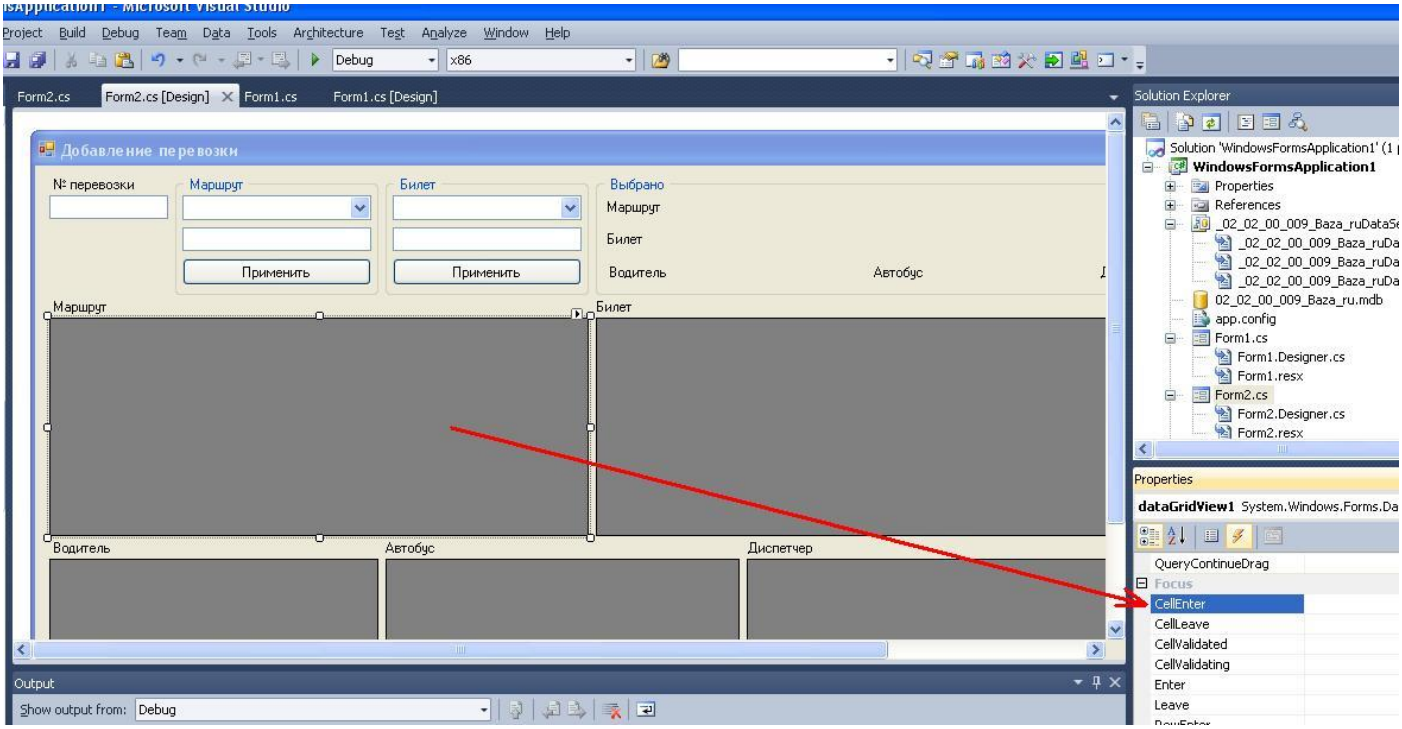

Рис. 29. Событие CellEnter элемента управления dataGridView1

Листинг обработчиков событий CellEnter для элементов управления dataGridView1, dataGridView2, dataGridView3, dataGridView4, dataGridView5 следующий.

```
private void dataGridView1_CellEnter(object sender, DataGridViewCellEventArgs e)
{
  int row;
  // Берем данные из ячеек таблицы "Маршрут"
  row = dataGridView1.CurrentCell.RowIndex;
  label2.Text = "Маршрут - " + "№" + Convert.ToString(dataGridView1[1, row].Value) +
      "/" + Convert.ToString(dataGridView1[2, row].Value) +
      "/" + Convert.ToString(dataGridView1[3, row].Value) +
      "/" + Convert.ToString(dataGridView1[4, row].Value) +
      "/" + Convert.ToString(dataGridView1[7, row].Value) +
     " - " + Convert.ToString(dataGridView1[8, row].Value);
}

private void dataGridView2_CellEnter(object sender, DataGridViewCellEventArgs e)
{
  int row;
  // так же берем данные из таблицы "Билет"
  row = dataGridView2.CurrentCell.RowIndex;
  label3.Text = "Билет: место №" + Convert.ToString(dataGridView2[1, row].Value) +
     " / " + Convert.ToString(dataGridView2[2, row].Value) +
     " / " + Convert.ToString(dataGridView2[3, row].Value) +
     " / " + Convert.ToString(dataGridView2[4, row].Value) +
     " / " + Convert.ToString(dataGridView2[5, row].Value) +
     " / " + Convert.ToString(dataGridView2[6, row].Value);
}

private void dataGridView3_CellEnter(object sender, DataGridViewCellEventArgs e)
{
  int row;
  // Данные о водителе
  row = dataGridView3.CurrentCell.RowIndex;
  label4.Text = "Водитель: " + Convert.ToString(dataGridView3[1, row].Value);
}
```

```
private void dataGridView4_CellEnter(object sender, DataGridViewCellEventArgs e)
{
  int row;
  // Информация об автобусе
  row = dataGridView4.CurrentCell.RowIndex;
  label5.Text = "Автобус: №" + Convert.ToString(dataGridView4[1, row].Value) +
     " / " + Convert.ToString(dataGridView4[2, row].Value) +
     " / " + Convert.ToString(dataGridView4[3, row].Value) +
     " / " + Convert.ToString(dataGridView4[4, row].Value);
}

private void dataGridView5_CellEnter(object sender, DataGridViewCellEventArgs e)
{
  int row;
  // Информация о диспетчере
  row = dataGridView5.CurrentCell.RowIndex;
  label6.Text = "Диспетчер: " + Convert.ToString(dataGridView5[1, row].Value);
}
```

10.5. Построение строк полей для фильтра в элементах управления comboBox1 и comboBox2

В компоненте comboBox1 пользователь имеет возможность выбирать поля из таблицы «Маршрут», которые будут использоваться при фильтрации данных.

Процесс формирования строк в элементах управления типа ComboBox приведен в п. 6.

Формируем такие строки в comboBox1:

- № маршрута;
- Пункт назначения;
- Район;
- Область.

В comboBox2 формируем такие строки:

- Место;
- Ф. И. О. пассажира.

10.6. Программирование фильтра к таблице «Маршрут«

В форме Form2 можно применять фильтр к таблицам «Маршрут» и «Билет«.

Применение фильтра к таблице «Маршрут» реализовано с помощью компонент comboBox1, textBox2 и button1.

Листинг метода обработки события клика на кнопке «Применить» при выборе фильтра для таблицы «Маршрут«:

```
private void button1_Click(object sender, EventArgs e)
{
  // Фильтр к таблице "Маршрут"
  string CommandText = "SELECT * FROM [Маршрут]";
  // формируем переменную CommandText
  if (textBox2.Text == "")
    CommandText = "SELECT * FROM [Маршрут]";
  else
  if (comboBox1.SelectedIndex == 0) // № перевозки
    CommandText = "SELECT * FROM [Маршрут] WHERE [Номер маршрута] = '" + textBox2.Text + "'"; //
работает
  else
  if (comboBox1.SelectedIndex == 1) //
    CommandText = "SELECT * FROM [Маршрут] WHERE [Пункт назначения] LIKE '" + textBox2.Text +
"%'";
  else
  if (comboBox1.SelectedIndex == 2) // Район
    CommandText = "SELECT * FROM [Маршрут] WHERE Район LIKE '" + textBox2.Text + "%'";
  else
  if (comboBox1.SelectedIndex == 3) // Область
```

```
  CommandText = "SELECT * FROM [Маршрут] WHERE Область LIKE '" + textBox2.Text + "%'";

 Form1 f = new Form1();
 OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, f.ConnectionString);
 DataSet ds = new DataSet();
 dataAdapter.Fill(ds, "[Маршрут]");
 dataGridView1.DataSource = ds.Tables[0].DefaultView;
}
```

10.7. Программирование фильтра к таблице «Билет«

Фильтр к таблице «Билет» применяется при нажатии на кнопке «Применить» в группе «Билет«. Элемент управления кнопки имеет название button2. Кроме того, для организации фильтра используются элементы управления comboBox2 и textBox3.

В поле textBox3 указывается значение фильтра. В comboBox2 указывается поле из таблицы «Билет«, к которому применяется фильтр.

```
private void button2_Click(object sender, EventArgs e)
{
 // Фильтр к таблице "Билет"
 string CommandText = "SELECT * FROM [Билет]";
 // формируем переменную CommandText
 if (textBox3.Text == "")
  CommandText = "SELECT * FROM [Билет]";
 else
 if (comboBox2.SelectedIndex == 0) // № перевозки
 CommandText = "SELECT * FROM [Билет] WHERE [Место] = " + textBox3.Text;
 else
 if (comboBox2.SelectedIndex == 1) //
  CommandText = "SELECT * FROM [Билет] WHERE [П_І_Б пассажира] LIKE '" + textBox3.Text + "%'";
 Form1 f = new Form1();
 OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, f.ConnectionString);
 DataSet ds = new DataSet();
 dataAdapter.Fill(ds, "[Маршрут]");
 dataGridView2.DataSource = ds.Tables[0].DefaultView;
}
```

10.8. Установление доступа к некоторым полям формы Form2

Для того, чтобы из формы Form1 считывать значения некоторых элементов управления формы Form2, нужно изменить директиву видимости с private на public.

Чтобы осуществить это, вызовем файл «Form2.Designer.cs» (рис. 30).

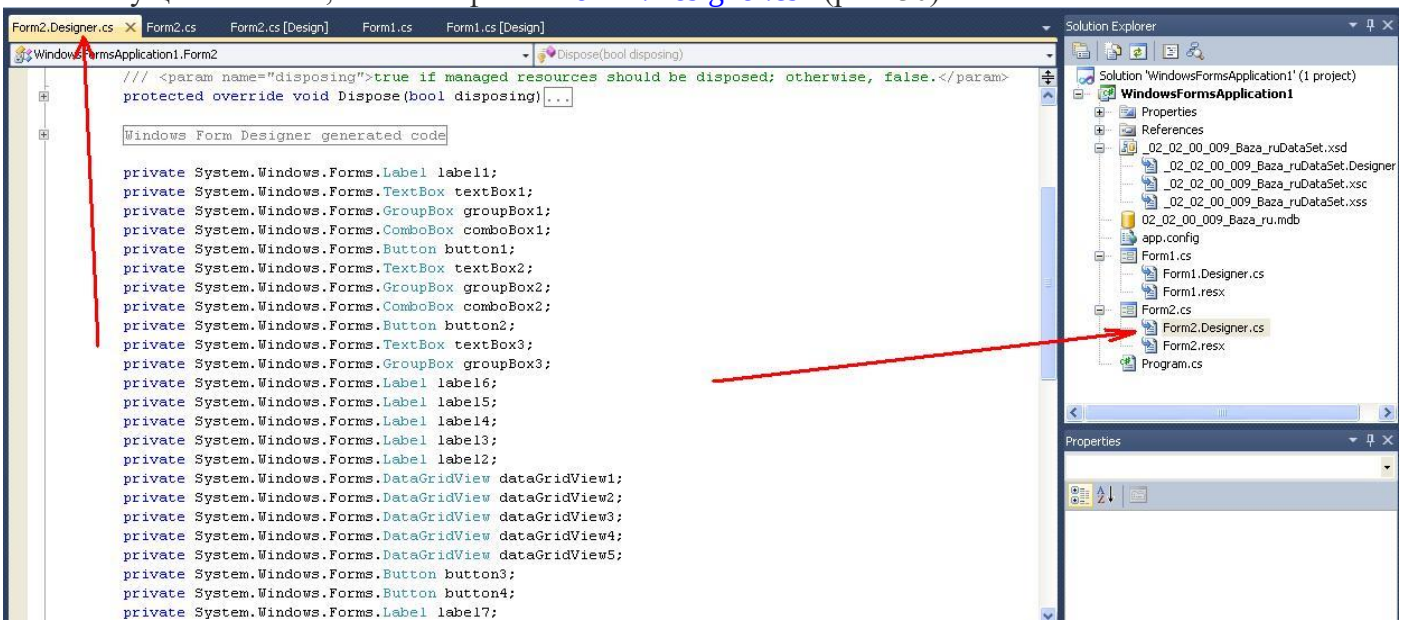

Рис. 30. Файл «Form2.Designer.cs«

В результате, получим текст файла, который имеет приблизительно следующий вид:

```
namespace WindowsFormsApplication1
{
   partial class Form2
   {
      ...
      private System.Windows.Forms.Label label1;
      private System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.GroupBox groupBox1;
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.TextBox textBox2;
      private System.Windows.Forms.ComboBox comboBox1;
      private System.Windows.Forms.GroupBox groupBox2;
      private System.Windows.Forms.Button button2;
      private System.Windows.Forms.TextBox textBox3;
      private System.Windows.Forms.ComboBox comboBox2;
      private System.Windows.Forms.GroupBox groupBox3;
      private System.Windows.Forms.Label label6;
      private System.Windows.Forms.Label label5;
      private System.Windows.Forms.Label label4;
      private System.Windows.Forms.Label label3;
      private System.Windows.Forms.Label label2;
      private System.Windows.Forms.Label label7;
      private System.Windows.Forms.DataGridView dataGridView1;
      private System.Windows.Forms.DataGridView dataGridView2;
      private System.Windows.Forms.Label label8;
      private System.Windows.Forms.Label label9;
      private System.Windows.Forms.DataGridView dataGridView3;
      private System.Windows.Forms.Label label10;
      private System.Windows.Forms.DataGridView dataGridView4;
      private System.Windows.Forms.Label label11;
      private System.Windows.Forms.DataGridView dataGridView5;
      private System.Windows.Forms.Button button3;
      private System.Windows.Forms.Button button4;
   }
}
```

Изменяем доступ с private на public для таких свойств:

- textBox1 (№ перевозки);
- dataGridView1;
- dataGridView2;
- dataGridView3;
- dataGridView4;
- dataGridView5;

Таким образом, фрагмент листинга файла «Form2.Designer.cs» имеет вид:

```
namespace WindowsFormsApplication1
{
   ...

   partial class Form2
   {
      private System.Windows.Forms.Label label1;
      public System.Windows.Forms.TextBox textBox1;
      private System.Windows.Forms.GroupBox groupBox1;
      private System.Windows.Forms.ComboBox comboBox1;
      private System.Windows.Forms.Button button1;
      private System.Windows.Forms.TextBox textBox2;
```

```
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.ComboBox comboBox2;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        public System.Windows.Forms.DataGridView dataGridView1;
        public System.Windows.Forms.DataGridView dataGridView2;
        public System.Windows.Forms.DataGridView dataGridView3;
        public System.Windows.Forms.DataGridView dataGridView4;
        public System.Windows.Forms.DataGridView dataGridView5;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Label label11;
    }
}
```

На этом создание формы «Form2.cs» завершено.

10.9. Листинг файла «Form2.cs«

В сокращенном виде текст файла «Form2.cs» следующий:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace WindowsFormsApplication1
{
  public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            ...
        }

        private void button4_Click(object sender, EventArgs e)
        {
            ...
        }
```

```
        private void Get_Table(string table_name, int num_dG)
        {
            ...
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            ...
        }

        private void dataGridView1_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            ...
        }

        private void dataGridView2_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            ...
        }

        private void      dataGridView3_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            ...
        }

        private void dataGridView4_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            ...
        }

        private void dataGridView5_CellEnter(object sender, DataGridViewCellEventArgs e)
        {
            ...
        }

        private void button1_Click(object sender, EventArgs e)
        {
            ...
        }

        private void button2_Click(object sender, EventArgs e)
        {
            ...
        }
    }
}
```

11. Программирование события вызова формы Form2 из формы Form1

Форма Form2 вызовется при выборе пользователем кнопки «Добавить…» из группы «Перевозка» (см. рис. 21). Элемент управления типа «кнопка», отвечающий команде «Добавить…«, имеет название button2.

Листинг события вызова формы Form2 из формы Form1 имеет вид:

```
private void button2_Click(object sender, EventArgs e)
{
  string CommandText;
  string num_per, ID_M, ID_B, ID_D, ID_A, ID_V;
  int row;
```

```
  Form2 f = new Form2(); // создали новую форму

  if (f.ShowDialog() == DialogResult.OK)
  {
    // добавляем данные
    // Номер перевозки
    if (f.textBox1.Text == "") num_per = "0";
    else num_per = f.textBox1.Text;
    // добавляем ID_Marshrut
    row = f.dataGridView1.CurrentCell.RowIndex; // взяли строку с dataGridView1
    ID_M = Convert.ToString(f.dataGridView1[0, row].Value);
    // добавляем ID_Bilet
    row = f.dataGridView2.CurrentCell.RowIndex; // взяли строку с dataGridView2
    ID_B = Convert.ToString(f.dataGridView2[0, row].Value);
    // добавляем ID_Dispetcher
    row = f.dataGridView3.CurrentCell.RowIndex; // взяли строку с dataGridView3
    ID_D = Convert.ToString(f.dataGridView3[0, row].Value);
    // добавляем ID_Avtobus
    row = f.dataGridView4.CurrentCell.RowIndex; // взяли строку с dataGridView4
    ID_A = Convert.ToString(f.dataGridView4[0, row].Value);
    // добавляем ID_Voditel
    row = f.dataGridView5.CurrentCell.RowIndex; // взяли строку с dataGridView5
    ID_V = Convert.ToString(f.dataGridView5[0, row].Value);
    // формируем CommandText
    CommandText = "INSERT INTO [Перевозка] (Номер, ID_Marshrut, ID_Bilet, ID_Dispetcher, ID_Avtobus,
ID_Voditel) " +
    "VALUES (" + num_per + ", " + ID_M + ", " + ID_B + ", " +
           ID_D + ", " + ID_A + ", " + ID_V + ")";

    // выполняем SQL-команду
    My_Execute_Non_Query(CommandText);
    // перерисовываем dataGridView1
    button1_Click(sender, e);
  }
}
```

В методе button2_Click() вызовется метод My_Execute_Non_Query(), листинг которого приведенн ниже:

```
// выполнение SQL-запроса для команд INSERT, UPDATE, DELETE
public void My_Execute_Non_Query(string CommandText)
{
  OleDbConnection conn = new OleDbConnection(ConnectionString);
  conn.Open();
  OleDbCommand myCommand = conn.CreateCommand();
  myCommand.CommandText = CommandText;
  myCommand.ExecuteNonQuery();
  conn.Close();
}
```

В методе My_Execute_Non_Query() создается объект, который осуществляет соединение с базой данных. Затем происходит выполнение команды, которая вносит изменения в базу данных. Для этого, в C# предназначен метод ExecuteNonQuery().

Команды на языке SQL делятся на два типа:

- те, которые читают данные из базы данных;
- те, что вносят изменения в данные базы данных.

После этого можно протестировать приложение. Можно вызвать команду «Добавить…» из группы «Перевозка» и добавить строку перевозки в таблицу, которая отображается в dataGridView1.

12. Листинг модуля «Form1.cs«

На данный момент, листинг файла «Form1.cs» в сокращенном виде следующий:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;

namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      public string ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;" +
         "Data Source=C:\\Programs\\C_SHARP\\Program_02_02_00_009u\\" +
         "WindowsFormsApplication1\\WindowsFormsApplication1\\02_02_00_009_Baza_ua.mdb";

      private int act_table = 1; // активная таблица: 1-билеты, 2-маршруты, 3-автобусы, 4-водители, 5 -
диспетчеры

      public Form1()
      {
         InitializeComponent();
      }

      private void button1_Click(object sender, EventArgs e)
      {
          ...
      }

       private void Form1_Load(object sender, EventArgs e)
       {
          ...
       }

       private void button2_Click(object sender, EventArgs e)
       {
         ...
       }

       public void My_Execute_Non_Query(string CommandText)
       {
         ...
       }
   }
}
```

13. Создание формы удаления записи в dataGridView1

В случае, если пользователь выбирает кнопку «Удалить» (button3) из группы «Перевозка«, нужно запрограммировать метод обработки события клика на этой кнопке (рис. 31).

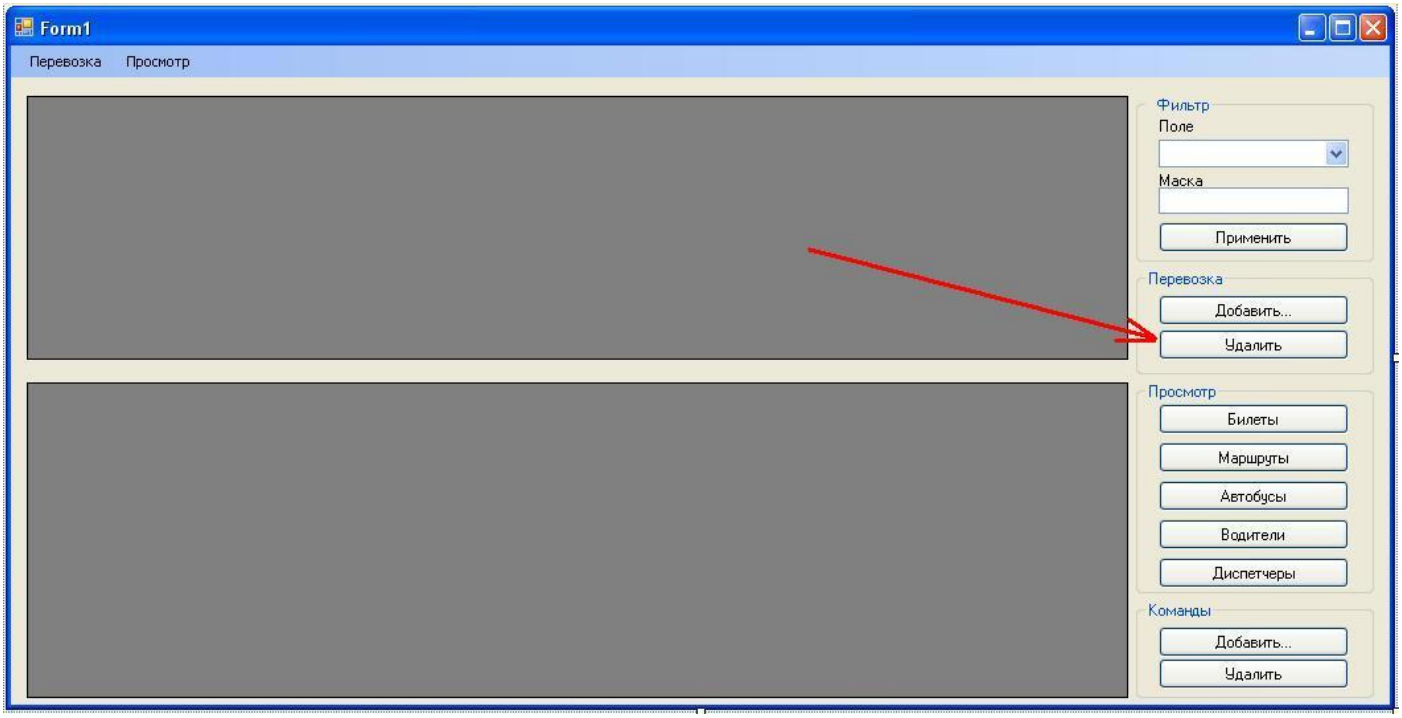

Рис. 31. Кнопка «Удалить» из основной формы «Form1.cs«

Создаем форму стандартным способом. Процесс создания формы в MS Visual Studio C# подробно описан здесь.

Предварительно нужно подготовить форму по образцу, как показано на рисунке 32.

При создании формы настраиваем такие свойства формы:

- свойство «Text» устанавливаем в значение «Удаление перевозки«;
- свойство FormBorderStyle = «Fixed3D»;
- свойство StartPosition = «CenterScreen»;
- свойство ControlBox = false.

Размещаем на форме следующие элементы управления:

- элемент управления типа «Label» (текст «Вы действительно желаете удалить данную запись?«);
- два элемента управления типа «Button» (кнопки «Да» и «Нет«).

В результате, в «Solution Explorer» получим файлы «Form3.cs» и «Form3.Designer.cs«.

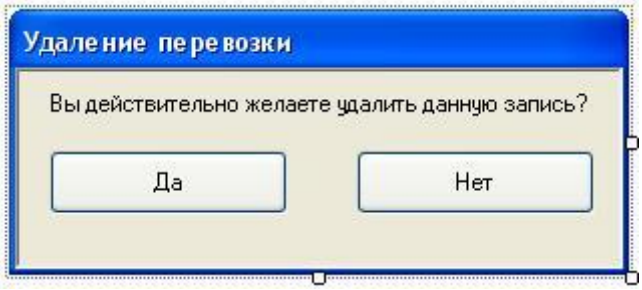

Рис. 32. Форма подтверждения удаления перевозки

В форме «Form3.cs» программируем обработку события клика на кнопке «Да» стандартным способом:

```
private void button1_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.OK;
}
```

Также программируем событие клику на кнопке «Нет«:

```
private void button2_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.No;
}
```

14. Программирование события вызова команды «Удалить» из группы «Перевозка«

При вызове команды «Удалить» из группы «Перевозка» нужно вызвать форму «Form3.cs«.
Листинг метода обработки события следующий:

```
private void button3_Click(object sender, EventArgs e)
{
  Form3 f = new Form3();

  if (f.ShowDialog() == DialogResult.OK)
  {
    int index, index_old;
    string ID;
    string CommandText = "DELETE FROM ";

    index = dataGridView1.CurrentRow.Index; // № по порядку в таблице представления
    index_old = index;
    ID = Convert.ToString(dataGridView1[0, index].Value); // ID подаем в запрос как строку

    // Формируем строку CommandText
    CommandText = "DELETE FROM [Перевозка] WHERE [Перевозка].[Номер] = '" + ID + "'";

    // выполняем SQL-запрос
    My_Execute_Non_Query(CommandText);

    // перерисовывание dbGridView1
    button1_Click(sender, e);

    if (index_old >= 0)
    {
      dataGridView1.ClearSelection();
      dataGridView1[0, index_old].Selected = true;
    }
  }
}
```

15. Программирование команд из группы «Просмотр«

15.1. Просмотр таблицы «Билет» в dataGridView2

Чтобы вывести таблицу «Билет» в dataGridView2 пользователь должен сделать клик на кнопке «Билеты» (button4) из группы «Просмотр» (рис. 33).

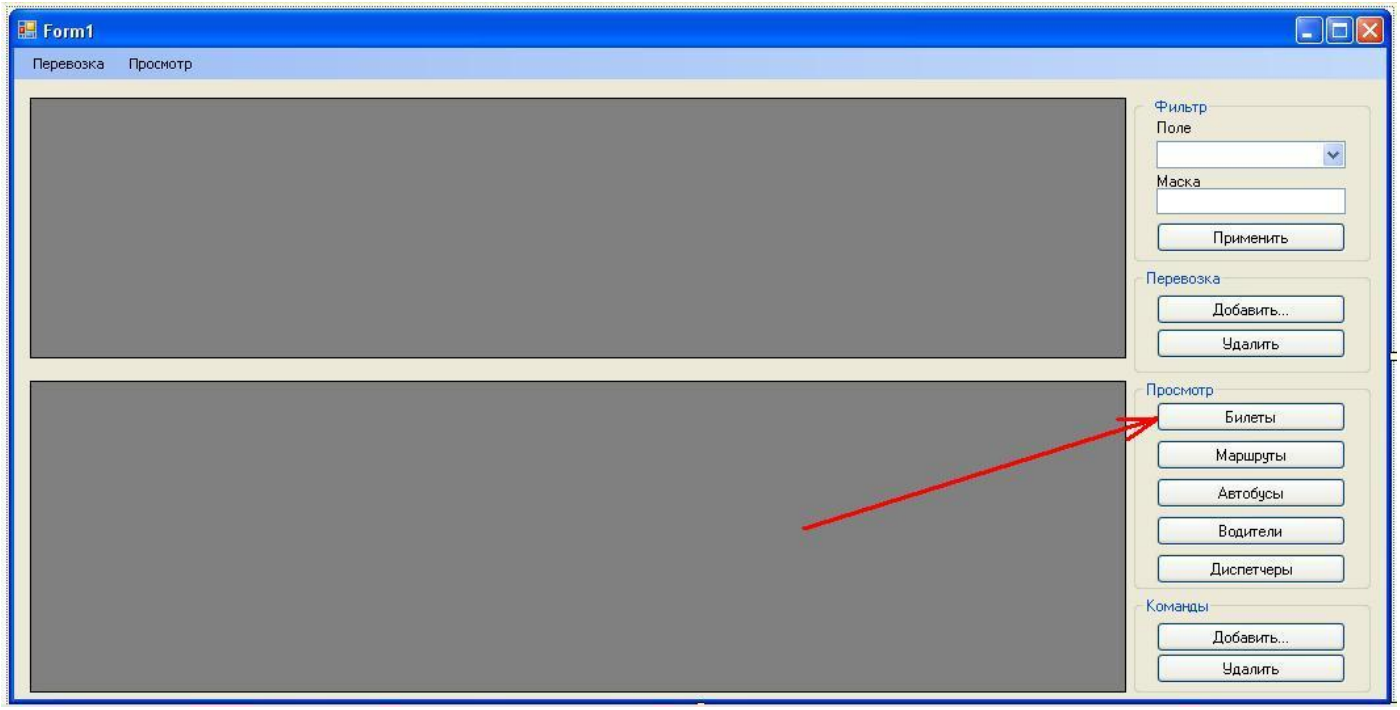

Рис. 33. Вызов команды просмотра списка купленных билетов

Листинг вспомогательной функции Get_Bilets() и обработчика события button4_Click() клика на кнопке button4 следующий.

```
private void Get_Bilets()  // читает все поля из таблицы "Билет"
{
  string CommandText = "SELECT ID_Bilet, [Место], [Стоимость], [Время], [Ф_И_О], [Паспорт], [Льготы] FROM [Билет]";
  OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, ConnectionString);

  // создаем объект DataSet
  DataSet ds = new DataSet();
  // заполняем dataGridView1 данными из таблицы "Билет" базы данных
  dataAdapter.Fill(ds, "[Билет]");
  dataGridView2.DataSource = ds.Tables[0].DefaultView;
  dataGridView2.Columns[0].Visible = false; // Прячем поле ID_Bilets
}

// Клик на кнопке "Билеты" из группы "Просмотр"
private void button4_Click(object sender, EventArgs e)
{
  Get_Bilets();
  act_table = 1;
}
```

Также добавляем вызов функции Get_Bilets() в метод Form1_Load(). Таким образом, листинг метода Form1_Load() следующий:

```
private void Form1_Load(object sender, EventArgs e)
{
  comboBox1.SelectedIndex = 0;
  button1_Click(sender, e);
  Get_Bilets();
}
```

Теперь можно протестировать работу приложения.

15.2. Просмотр таблицы «Маршрут» в dataGridView2

Вывод таблицы «Маршрут» в dataGridView2 осуществляется с помощью вызова команды «Маршруты» из группы «Просмотр«.

Листинг вспомогательного метода Get_Marshruts() и обработчика события клика на кнопке «Маршруты» (button5) следующий:

```
private void Get_Marshruts()  // читает все поля из таблицы "Маршрут"
{
  string CommandText = "SELECT * FROM [Маршрут]";
  OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, ConnectionString);
  DataSet ds = new DataSet();  // создаем объект DataSet
  dataAdapter.Fill(ds, "[Маршрут]");
  dataGridView2.DataSource = ds.Tables[0].DefaultView;
}

private void button5_Click(object sender, EventArgs e)
{
  Get_Marshruts();
  act_table = 2;
}
```

15.3. Просмотр таблицы «Автобус» в dataGridView2

Чтобы вывести таблицу «Автобус» в dataGridView2 вызвать команду «Автобусы» (button6) из группы «Просмотр«.

Листинг дополнительного метода Get_Bus() и обработчика события клика на кнопке «Автобусы» следующий.

```
private void Get_Avtobus()  // читает все поля из таблицы "Автобус"
{
  string CommandText = "SELECT * FROM Автобус";
  OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, ConnectionString);
  DataSet ds = new DataSet();  // создаем объект DataSet
  dataAdapter.Fill(ds, "Автобус"); // заполняем набор данных данными из таблицы "Автобус"
  dataGridView2.DataSource = ds.Tables[0].DefaultView;
  dataGridView2.Columns[0].Visible = false; // спрятать нулевой столбец (поле ID_Avtobus)
}

private void button6_Click(object sender, EventArgs e)
{
  Get_Avtobus();
  act_table = 3;
}
```

15.4. Просмотр таблицы «Водитель» в dataGridView2

Таблица «Водитель» выводится в dataGridView2 после нажатия на кнопке «Водители» из группы «Просмотр«.

Листинг вспомогательного метода Get_Voditel() и метода обработки события клика на кнопке button7 следующий:

```
private void Get_Voditel()  // читает все поля из таблицы "Водитель"
{
  string CommandText = "SELECT * FROM Водитель";
  OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, ConnectionString);
  DataSet ds = new DataSet();
  dataAdapter.Fill(ds, "Водитель");
  dataGridView2.DataSource = ds.Tables[0].DefaultView;
  dataGridView2.Columns[0].Visible = false;
}

private void button7_Click(object sender, EventArgs e)
{
  Get_Voditel();
  act_table = 4;
}
```

15.5. Просмотр таблицы «Диспетчер» в dataGridView2

Для вывода таблицы «Диспетчер» в элементе управления dataGridView2 нужно выбрать команду «Диспетчеры» из группы «Просмотр«.
Листинг вспомогательного метода Get_Dispetcher() и обработчика события клика на кнопке button8 следующий.

```
// заполняет dataGridView2 данными из таблицы "Диспетчер"
private void Get_Dispetcher()
{
  string CommandText = "SELECT * FROM [Диспетчер]";
  OleDbDataAdapter dataAdapter = new OleDbDataAdapter(CommandText, ConnectionString);
  DataSet ds = new DataSet();
  dataAdapter.Fill(ds, "Диспетчер");
  dataGridView2.DataSource = ds.Tables[0].DefaultView;
  dataGridView2.Columns[0].Visible = false;
}

private void button8_Click(object sender, EventArgs e)
{
  Get_Dispetcher();
  act_table = 5;
}
```

16. Разработка форм добавления данных в таблицы

16.1. Разработка формы добавления нового билета

Разработка новой формы в MS Visual Studio C# подробно описывается здесь.
Добавляем новую форму. Создаем форму добавления нового билета по примеру, изображенному на рисунке 34.

Файлы, которые соответствуют форме: «Form4.cs» и «Form4.Designer.cs«.
Настраиваем следующие свойства формы:

- свойство Text = «Билеты»;
- свойство FormBorderStyle = «Fixed3D»;
- свойство ControlBox = «false» (спрятать системное меню);
- свойство StartPosition = «CenterScreen» (окно появляется по центру экрана).

Размещаем элементы управления на форме. Значимые элементы управления имеют такие имена:

- textBox1 – поле «Место» в таблице «Билет«;
- textBox2 – поле «Стоимость«;
- textBox3 – «Время«;
- textBox4 – «Ф_И_О«;
- textBox5 – «Паспорт«;
- checkBox1 – «Льготы«;
- button1 – кнопка «Добавить«;
- button2 – кнопка «Отменить«.

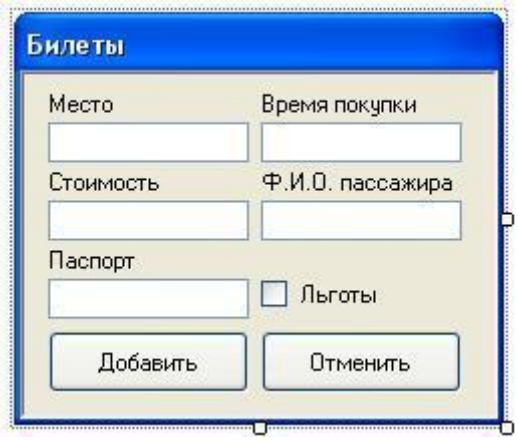

Рис. 34. Форма добавления нового билета

После создания формы нужно настроить видимость извне для некоторых полей класса Form4. Для этого переходим в файл «Form4.Designer.cs» (см. рисунок 30 для формы «Form2.cs«).

Устанавливаем доступ public для значимых полей формы «Form4.cs» – textBox1, textBox2, textBox3, textBox4, textBox5, checkBox1. Это можно сделать и другим путем изменив поле Modifiers в окне Properties.

Фрагмент соответствующего кода класса Form4 имеет вид:

```
partial class Form4
{
  ...

  #endregion

  private System.Windows.Forms.Label label1;
  public System.Windows.Forms.TextBox textBox1;
  private System.Windows.Forms.Label label2;
  public System.Windows.Forms.TextBox textBox2;
  private System.Windows.Forms.Label label3;
  private System.Windows.Forms.Label label4;
  public System.Windows.Forms.TextBox textBox3;
  private System.Windows.Forms.Label label5;
  public System.Windows.Forms.TextBox textBox4;
  public System.Windows.Forms.TextBox textBox5;
  public System.Windows.Forms.CheckBox checkBox1;
  private System.Windows.Forms.Button button1;
  private System.Windows.Forms.Button button2;
}
```

Программируем событие клика на кнопках button1 и button2 так же, как описано п. 11.2.

Листинг обработчиков событий следующий:

```
private void button1_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.OK;
}

private void button2_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.No;
}
```

16.2. Разработка формы добавления данных в таблицу «Маршрут«

По образцу создания формы добавления данных в таблицу «Билет» создаем форму добавления данных в таблицу «Маршрут» (рис. 35). Построение новой формы подробно описывается здесь.

В результате, в Solution Explorer, получим файлы с именами «Form5.cs» и «Form5.Designer.cs«.

Настраиваем следующие свойства формы:

- свойство Text = «Добавление маршрута«;
- свойство FormBorderStyle = «Fixed3D»;
- свойство ControlBox = «false»;
- свойство StartPosition = «CenterScreen».

Значимые элементы управления имеют такие имена:

- textBox1 – «№ маршрута«;
- textBox2 – «Пункт назначения«;
- textBox3 – «Район«;
- textBox4 – «Область«;
- textBox5 – «Расстояние«;
- textBox6 – «Вес«.
- dateTimePicker1 – время отправки;
- dateTimePicker2 – время прибытия;
- button1 – кнопка «Добавить«;
- button2 – кнопка «Отменить«.

В элементах управления dateTimePicker1 и dateTimePicker2 свойство Format устанавливаем у значение «Time».

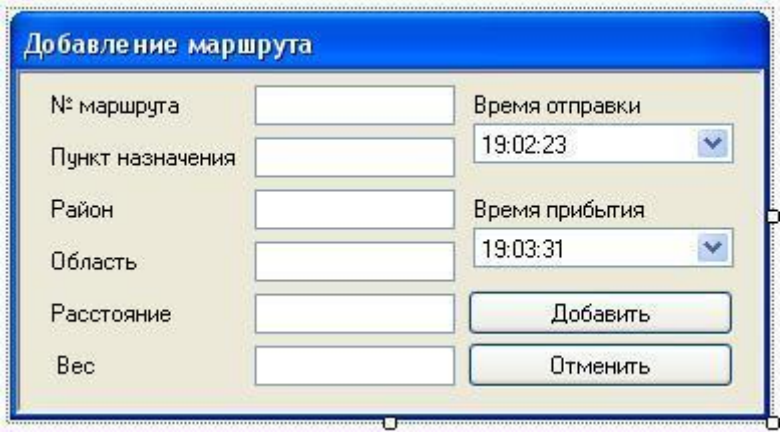

Рис. 35. Форма добавления нового маршрута

Устанавливаем доступ public для значимых полей формы «Form5.cs» – textBox1, textBox2, textBox3, textBox4, textBox5, textBox6, dateTimePicker1, dateTimePicker2.

Фрагмент соответствующего кода класса Form5 имеет вид:

```
...
partial class Form5
{
  #endregion
  private System.Windows.Forms.Label label1;
  public System.Windows.Forms.TextBox textBox1;
  private System.Windows.Forms.Label label2;
  public System.Windows.Forms.TextBox textBox2;
  public System.Windows.Forms.TextBox textBox3;
  private System.Windows.Forms.Label label3;
  public System.Windows.Forms.TextBox textBox4;
  private System.Windows.Forms.Label label4;
  public System.Windows.Forms.TextBox textBox5;
  private System.Windows.Forms.Label label5;
  public System.Windows.Forms.TextBox textBox6;
  private System.Windows.Forms.Label label6;
  private System.Windows.Forms.Label label7;
  public System.Windows.Forms.DateTimePicker dateTimePicker1;
  private System.Windows.Forms.Label label8;
  public System.Windows.Forms.Button button1;
```

```
  private System.Windows.Forms.Button button2;
  public System.Windows.Forms.DateTimePicker dateTimePicker2;
}
...
```

Программируем события клика на кнопках button1 и button2 так же, как описывается в п. 11.2.
Листинг обработчиков событий следующий:

```
private void button1_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.OK;
}

private void button2_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.No;
}
```

16.3. Разработка формы добавления данных в таблицу «Автобус«

Создание новой формы подробно описывается здесь.
Форма добавления нового транспортного средства изображена на рисунке 36. В приложении этой форме отвечают файлы «Form6.cs» и «Form6.Designer.cs«.
Настраиваем следующие свойства формы:

- свойство Text = «Добавление автобуса«;
- свойство FormBorderStyle = «Fixed3D»;
- свойство ControlBox = «false»;
- свойство StartPosition = «CenterScreen».

Значимые элементы управления имеют такие имена:

- textBox1 (Номер);
- textBox2 (Модель);
- textBox3 (Номерной знак);
- textBox4 (Количество мест).

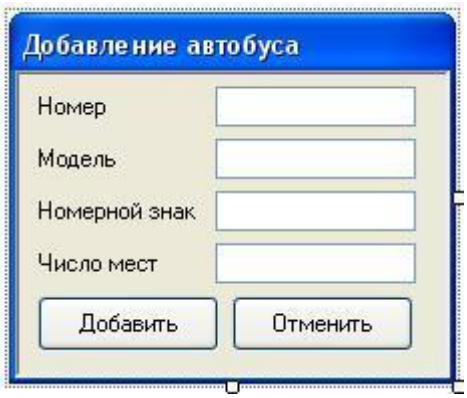

Рис. 36. Форма добавления нового автобуса

Устанавливаем доступ public для значимых полей формы «Form6.cs«: textBox1, textBox2, textBox3, textBox4.

```
...
partial class Form6
{
  ...
  private System.Windows.Forms.Label label1;
  public System.Windows.Forms.TextBox textBox1;
  private System.Windows.Forms.Label label2;
  public System.Windows.Forms.TextBox textBox2;
```

```
  public System.Windows.Forms.TextBox textBox3;
  private System.Windows.Forms.Label label3;
  public System.Windows.Forms.TextBox textBox4;
  private System.Windows.Forms.Label label4;
  private System.Windows.Forms.Button button1;
  private System.Windows.Forms.Button button2;
}
```

...

Программируем событие клика на кнопках button1 и button2 таким же образом, как описано в п. 11.2. Листинг обработчиков событий следующий:

```
private void button1_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.OK
}

private void button2_Click(object sender, EventArgs e)
{
  this.DialogResult = DialogResult.No;
}
```

16.4. Разработка формы добавления нового водителя

Разработка формы добавления данных в таблицу «Водитель» осуществляется стандартным способом. Форма изображена на рисунке 37.

В Solution Explorer новосозданной форме соответствуют файлы «Form7.cs«, «Form7.Designer.cs«. Свойства формы настраиваем так же, как и в предшествующих формах.

Значимые элементы управления имеют такие имена:

- textBox1 (Фамилия);
- textBox2 (Паспорт);
- dateTimePicker1 (Дата рождения);
- button1 – кнопка «Добавить«;
- button2 – кнопка «Отменить«.

В элементе управления dateTimePicker1 свойство Format = «Short».

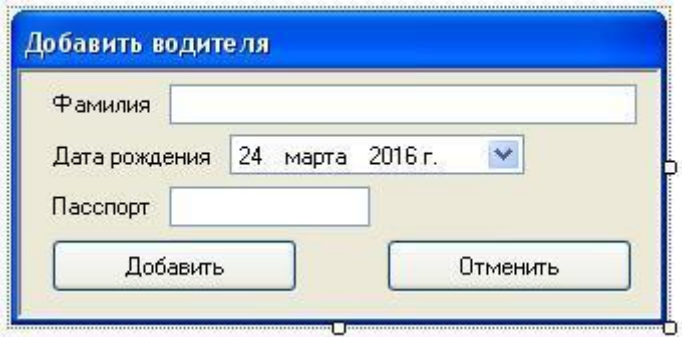

Рис. 37. Форма добавления данных в таблицу «Водитель«

В окне «Properties» устанавливаем доступ public в свойстве Modifiers (рис. 38) для таких элементов управления:

- textBox1 (Фамилия);
- dateTimePicker1 (Дата рождения);
- textBox2 (Паспорт).

Автоматически будут внесены изменения в файле «Form7.Designer.cs«.

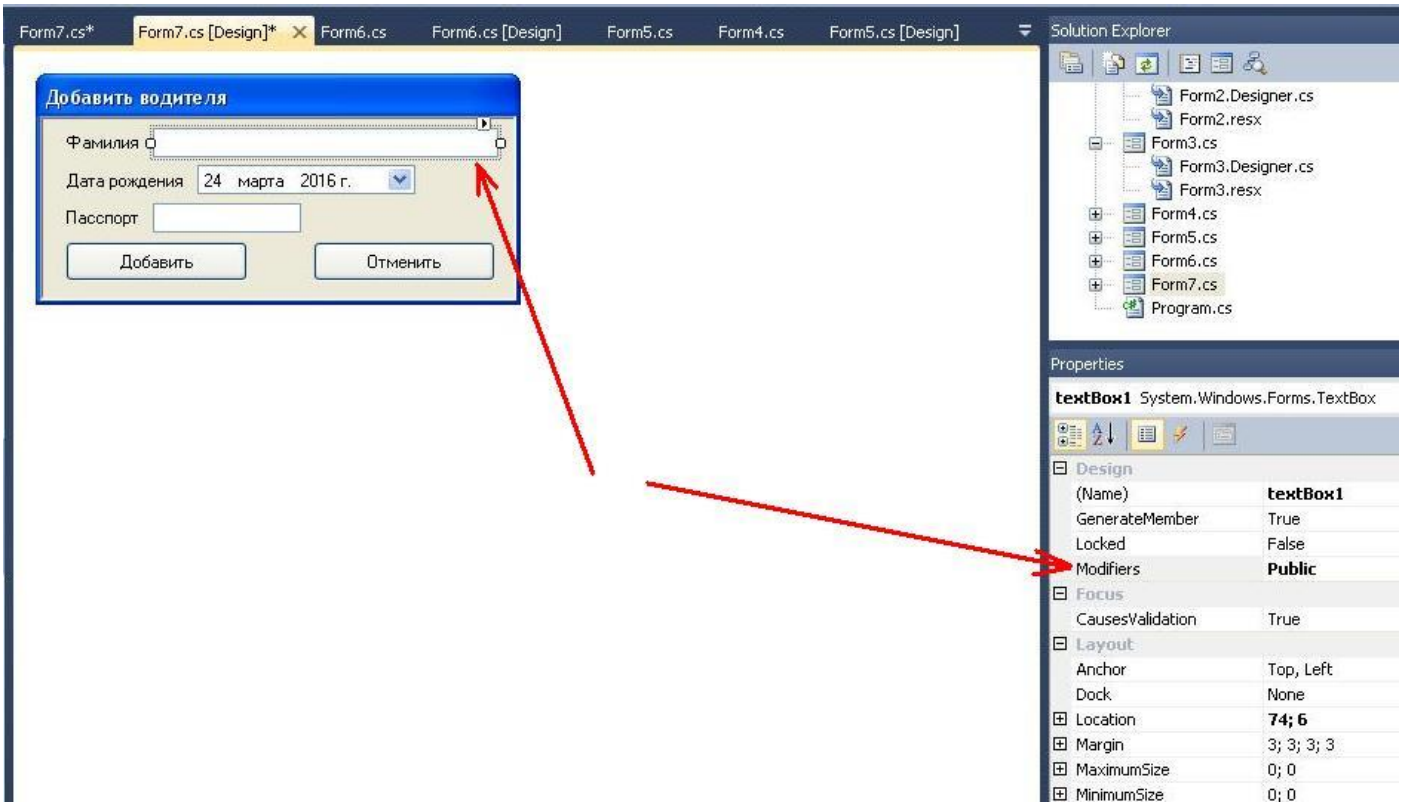

Рис. 38. Установление доступа public к элементу управления textBox1

Программируем событие клика на кнопках button1 и button2 так же, как описывается в п. 11.2. Листинг обработчиков событий следующий:

```
private void button1_Click(object sender, EventArgs e)
{
 this.DialogResult = DialogResult.OK;
}

private void button2_Click(object sender, EventArgs e)
{
 this.DialogResult = DialogResult.No;
}
```

16.5. Разработка формы добавления нового диспетчера

Форма добавления нового диспетчера изображена на рисунке 39. Файлы, которые соответствуют форме, имеют названия «Form8.cs» и «Form8.Designer.cs«.
Свойства формы настраиваем так же как и в предшествующих формах.

Значимые элементы управления имеют такие имена:

- textBox1 (Ф_И_О.);
- dateTimePicker1 (Дата рождения);
- textBox2 (Адрес).

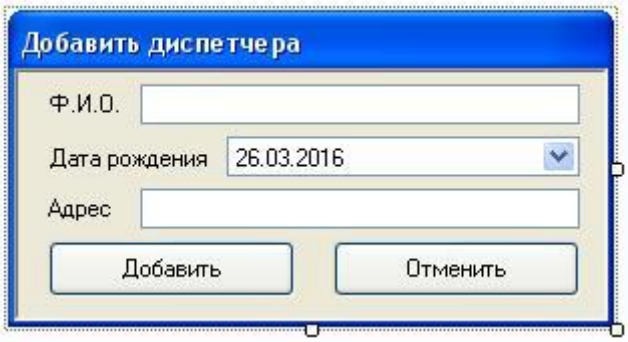

Рис. 39. Форма добавления данных в таблицу «Диспетчер«

В окне «Properties» устанавливаем доступ public в свойстве Modifiers (см. рис. 38) для таких элементов управления:

- textBox1 (Ф_И_О.);
- dateTimePicker1 (Дата рождения);
- textBox2 (Адрес).

Автоматически будут внесены изменения в файл «Form8.Designer.cs«.

Программируем события клика на кнопках button1 и button2 так же, как описано в предшествующих пунктах.

Листинг обработчиков событий следующий:

```
private void button1_Click(object sender, EventArgs e)
{
 this.DialogResult = DialogResult.OK;
}

private void button2_Click(object sender, EventArgs e)
{
 this.DialogResult = DialogResult.No;
}
```

17. Программирование методов добавления данных в таблицы «Билет«, «Маршрут«, «Автобус«, «Водитель«, «Диспетчер«

Все следующие шаги будут выполнятся в главной форме приложения Form1.cs.

Поэтому, сначала активируем форму «Form1.cs«.

17.1. Добавление данных в таблицу «Билет«

Листинг метода Add_Bilet() добавления данных в таблицу «Билет» следующий:

```
// добавление Билета через ConnectionString и запрос ExecuteNonQuery()
private void Add_Bilet(string mesto, string stoimost, DateTime vremja, string name, string passport, bool lgota)
{
 string CommandText;
 string s_vremja;
 string s_stoimost;
 s_vremja = Convert.ToString(vremja); // переводим время в строку
 s_stoimost = stoimost.Replace(',', '.'); // меняем разделитель разрядов '.' на
 // разделитель понятный синтаксису SQL '.'

 CommandText = "INSERT INTO [Билет] (Место, Стоимость, [Время], [Ф_И_О], Паспорт, Льготы) " +
"VALUES ('" + mesto + "', " + s_stoimost + ", '" + s_vremja + "', '"
 + name + "', '" + passport + "', " + lgota + ")";
 My_Execute_Non_Query(CommandText);
}
```

В вышеприведенном листинге метод Add_Bilet() получает входным параметром переменные: mesto, stoimost, vremja, name, passport, lgota. Потом формируется запрос на языке SQL. В этом запросе, в переменной CommandText формируется команда

INSERT INTO

которая выполняется в методе My_Execute_Non_Query().

17.2. Добавление данных в таблицу «Маршрут«

Листинг метода Add_Marshrut() следующий.

```
private void Add_Marshrut(string num_marsh, string punkt, string rajon, string oblast, double rasst, double ves,
DateTime vremja_otpr, DateTime vremja_prib)
{
 string CommandText;
 string s_otpr, s_prib;
 string s_ves, s_rasst;

 s_otpr = Convert.ToString(vremja_otpr); // переводим время отправки в строку
 s_prib = Convert.ToString(vremja_prib); // переводим время прибытия в строку
 s_ves = Convert.ToString(ves); // переводим вес из double в строку
 s_ves = s_ves.Replace(',','.'); // меняем запятую на точку согласно синтаксису SQL
 s_rasst = Convert.ToString(rasst); // переводим расстояние из double в string
 s_rasst = s_rasst.Replace(',', '.'); // меняем запятую на точку

 CommandText = "INSERT INTO [Маршрут] ([Номер маршрута], [Пункт назначения], Район, Область,
Расстояние, Вес, [Время отправки], [Время прибытия])"
 + " VALUES ('" + num_marsh + "', '" + punkt + "', '" + rajon + "', '" + oblast + "', "
 + s_rasst + ", " + s_ves + ", '" + s_otpr + "', '" + s_prib + "')";
 My_Execute_Non_Query(CommandText);
}
```

Метод Add_Marshrut() получает входными параметрами значения, которые отвечают полям таблицы «Маршрут» базы данных.

17.3. Добавление данных в таблицу «Автобус«

Листинг метода Add_Avtobus() следующий.

```
void Add_Avtobus(string num, string model, string znak, string k_mest)
{
 string CommandText;
 CommandText = "INSERT INTO [Автобус] ([Номер], [Модель], [Номерной знак], [Количество мест])"
 + " VALUES ('" + num + "', '" + model + "', '" + znak + "', " + k_mest + ")";
 My_Execute_Non_Query(CommandText);
}
```

Метод Add_Avtobus() получает входные параметры, которые соответствуют полям таблицы «Автобус» базы данных.

17.4. Добавление данных в таблицу «Водитель«

Листинг метода Add_Voditel() добавления данных в таблицу «Водитель» следующий.

```
void Add_Voditel(string f_i_o, string d_r, string passport) // добавить водителя
{
 string CommandText;
 CommandText = "INSERT INTO [Водитель] ([Ф_И_О], [Дата рождения], [Паспорт])"
 + " VALUES ('" + f_i_o + "', '" + d_r + "', '" + passport + "')";
 My_Execute_Non_Query(CommandText);
}
```

17.5. Добавление данных в таблицу «Диспетчер«

Листинг метода Add_Dispetcher() следующий.

```
void Add_Dispetcher(string f_i_o, string d_r, string adres) // добавить диспетчера
{
 string CommandText;
 CommandText = "INSERT INTO [Диспетчер] ([Ф_И_О], [Дата рождения], [Адрес])"
```

```
 + " VALUES ('" + f_i_o + "', '" + d_r + "', '" + adres + "')";
 My_Execute_Non_Query(CommandText);
}
```

18. Программирование события клика на кнопке «Добавить…» из группы «Команды«

В основной форме «Form1.cs«, при выборе кнопки «Добавить…» из группы «Команды» нужно выводить одно из окон форм, которые соответствуют добавлению данных в таблицы «Билет«, «Маршрут«, «Автобус«, «Водитель«, «Диспетчер» (рис. 40).

Для этого, средствами языка C# должна вызываться (создаваться) соответствующая форма. После внесения данных в форму и выбора подтверждающего запроса, данные из формы должны добавляться в соответствующую таблицу базы данных.

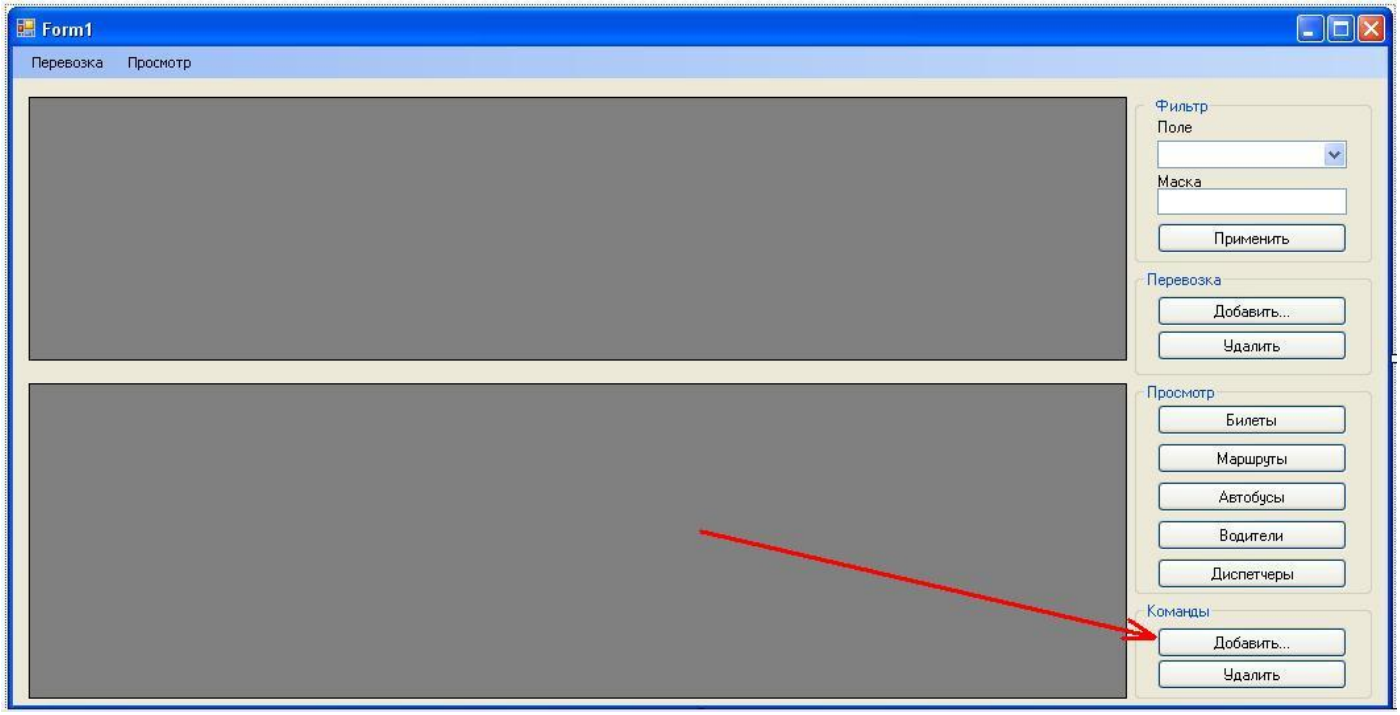

Рис. 40. Кнопка «Добавить…» главной формы

Листинг обработчика события клика на кнопке «Добавить…» следующий:

```
private void button9_Click(object sender, EventArgs e)
{
   if (act_table == 1) // обрабатываем таблицу "Билет"
   {
      Form4 f = new Form4();

      if (f.ShowDialog() == DialogResult.OK)
      {
         // добавляем данные в таблицу "Билеты"
         Add_Bilet(f.textBox1.Text, f.textBox2.Text, Convert.ToDateTime(f.textBox3.Text),
 f.textBox4.Text, f.textBox5.Text, f.checkBox1.Checked);
         Get_Bilets();
      }
   }
   else
   if (act_table == 2) // обрабатываем таблицу "Маршрут"
   {
      Form5 f = new Form5();
      if (f.ShowDialog() == DialogResult.OK)
      {
         // добавляем данные в таблицу "Маршрут"
         Add_Marshrut(f.textBox1.Text, f.textBox2.Text, f.textBox3.Text, f.textBox4.Text,
```

```
            Convert.ToDouble(f.textBox5.Text), Convert.ToDouble(f.textBox6.Text),
 f.dateTimePicker1.Value, f.dateTimePicker2.Value);
            Get_Marshruts();
        }
    }
    else
    if (act_table == 3) // обрабатываем таблицу "Автобус"
    {
        Form6 f = new Form6();
        if (f.ShowDialog() == DialogResult.OK)
        {
            // добавляем данные в таблицу "Автобус"
            Add_Avtobus(f.textBox1.Text, f.textBox2.Text, f.textBox3.Text, f.textBox4.Text);
            Get_Avtobus();
        }
    }
    else
    if (act_table == 4) // обрабатываем таблицу "Водитель"
    {
        Form7 f = new Form7();
        if (f.ShowDialog() == DialogResult.OK)
        {
            // добавляем данные в таблицу "Водитель"
            Add_Voditel(f.textBox1.Text, Convert.ToString(f.dateTimePicker1.Value), f.textBox2.Text);
            Get_Voditel();
        }
    }
    else
    if (act_table == 5) // обрабатываем таблицу "Диспетчер"
    {
        Form8 f = new Form8();
        if (f.ShowDialog() == DialogResult.OK)
        {
            // добавляем данные в таблицу "Диспетчер"
            Add_Dispetcher(f.textBox1.Text, Convert.ToString(f.dateTimePicker1.Value), f.textBox2.Text);
            Get_Dispetcher();
        }
    }
}
```

Как видно из листинга, на основе переменной act_table (активная таблица) вызовется соответствующий метод добавления данных в базу данных.

19. Программирование события клика на кнопке «Удалить» из группы «Команды«

Листинг обработчика события клика на кнопке «Удалить» из группы «Команды» следующий.

```
private void button10_Click(object sender, EventArgs e)
{
 Form3 f = new Form3();
 f.Text = " ";
 if (f.ShowDialog() == DialogResult.OK)
 {
  int index, index_old;
  string ID;
  string CommandText = "DELETE FROM ";
  index = dataGridView2.CurrentRow.Index; // № по порядку в таблице представления
  index_old = index;
  ID = Convert.ToString(dataGridView2[0, index].Value); // ID подаем в запрос как строку
```

```
  // Формируем строку CommandText
  if (act_table == 1) // обрабатываем таблицу "Билет"
  CommandText = "DELETE FROM Билет WHERE Билет.ID_Bilet = " + ID;
  if (act_table == 2) // обрабатываем таблицу "Маршрут"
  CommandText = "DELETE FROM Маршрут WHERE Маршрут.ID_Marshrut = " + ID;
  if (act_table == 3) // обрабатываем таблицу "Автобус"
  CommandText = "DELETE FROM Автобус WHERE Автобус.ID_Avtobus = " + ID;
  if (act_table == 4) // обрабатываем таблицу "Водитель"
  CommandText = "DELETE FROM Водитель WHERE Водитель.ID_Voditel = " + ID;
  if (act_table == 5) // обрабатываем таблицу "Диспетчер"
  CommandText = "DELETE FROM Диспетчер WHERE Диспетчер.ID_Dispetcher = " + ID;
  // выполняем SQL-запрос
  My_Execute_Non_Query(CommandText);

  // перерисовывание dbGridView2
  if (act_table == 1) Get_Bilets();
  else
  if (act_table == 2) Get_Marshruts();
  else
  if (act_table == 3) Get_Avtobus();
  else
  if (act_table == 4) Get_Voditel();
  else
  if (act_table == 5) Get_Dispetcher();

  if (index_old >= 0)
  {
    dataGridView2.ClearSelection();
    dataGridView2[0, index_old].Selected = true;
  }
 }
}
```

20. Программирование клика по командах меню (см. п. 5.1.)

Последним шагом осталось назначить командам меню (см. п. 5.1., рис. 11) выполнение соответствующих обработчиков событий клика на кнопках, которые размещаются на главной форме приложения.

Так, например, команда «Добавить…» из меню «Перевозка» должна выполнять ту же работу, что и команда «Добавить…» из группы «Перевозка» главной формы (рис. 41). Поэтому, нецелесообразно два раза писать код обработчика события добавления перевозки. Система Microsoft Visual Studio позволяет удобно назначать метод выполнения (если он уже запрограммирован) другой команде (см. рис. 42).

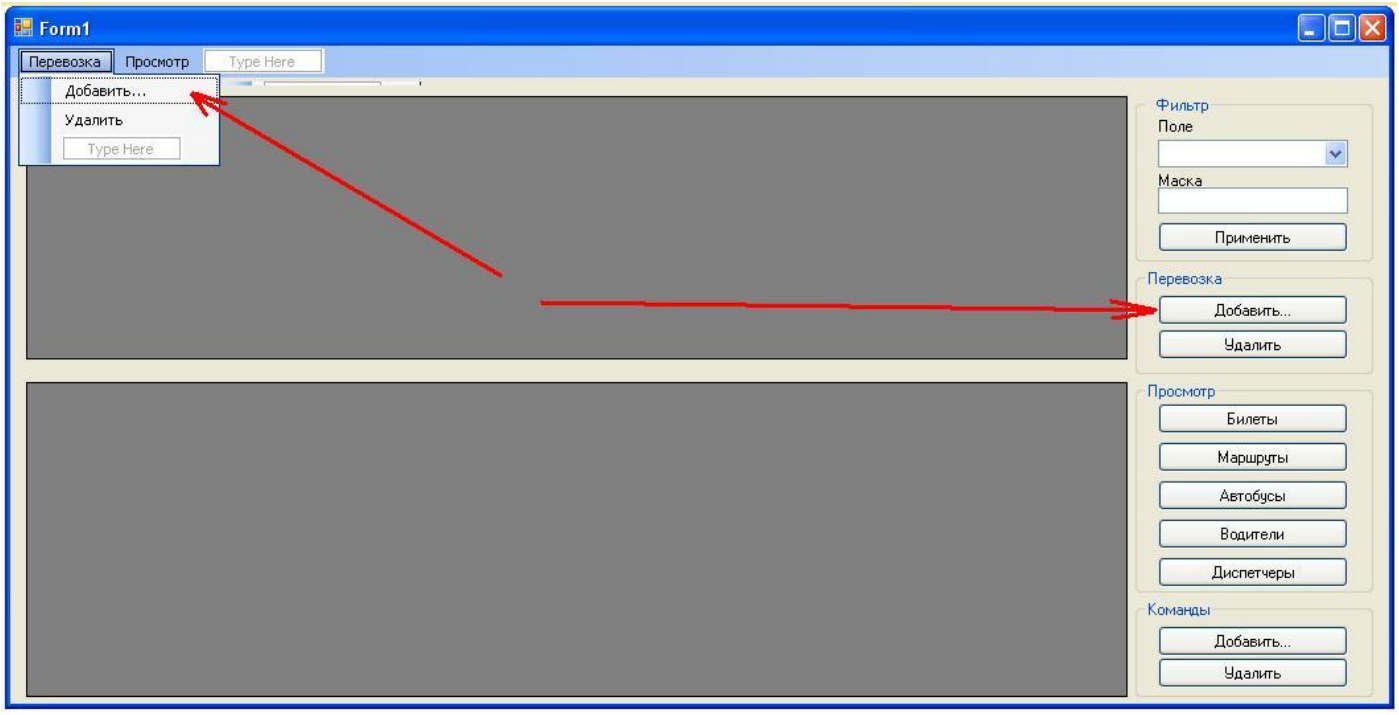

Рис. 41. Соответствие команды «Добавить…» из меню «Перевозка» одноименной команде из группы «Перевозка«

Команда добавления перевозки из основной формы вызовется на кнопке «Добавить…» из группы «Перевозка«. Этой кнопке соответствует объект с именем button2. Метод обработки события клика на кнопке button2 в приложении имеет название button2_Click() (см. п. 11).

Для назначения метода обработки события команде «Добавить…» из меню «Перевозка» нужно:

- активировать (выделить мышкой) команду «Добавить…» из меню «Перевозка«;
- в окне «Properties» активировать вкладку «Events»;
- в поле «Click» выбрать метод button2_Click (из выпадающего списка).

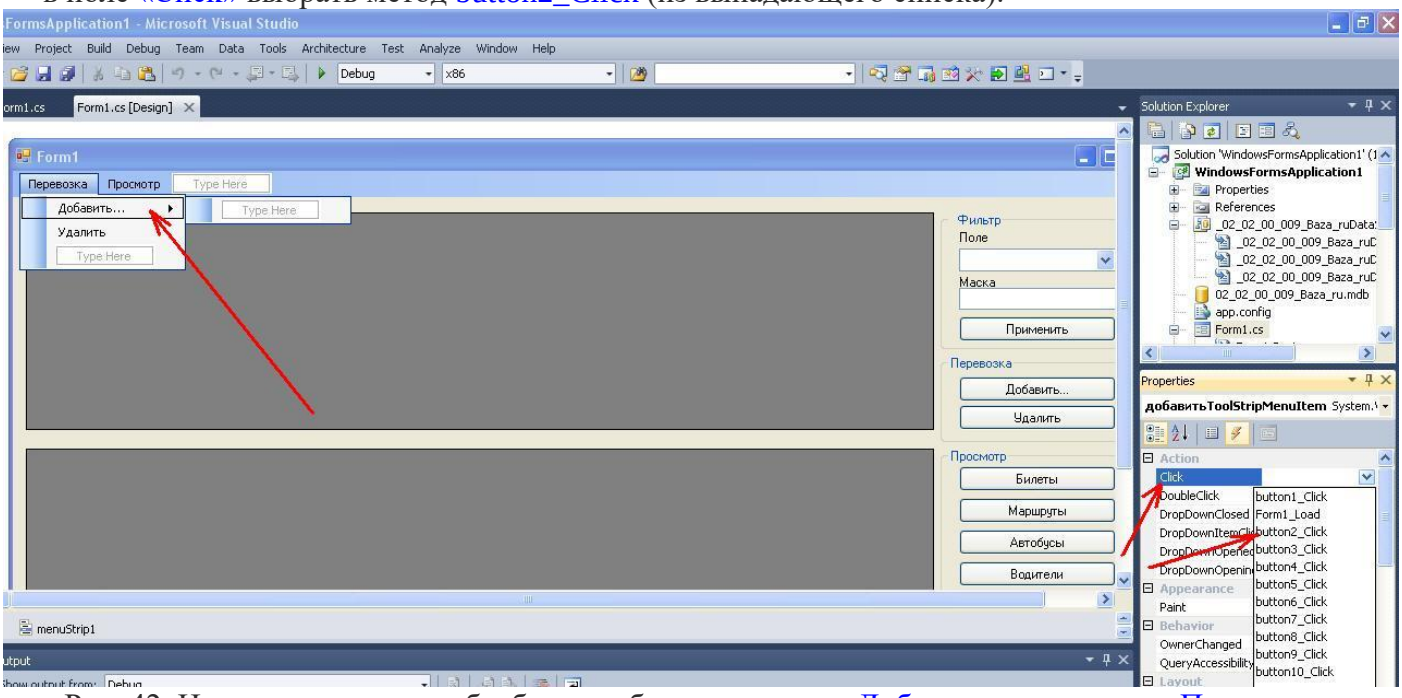

Рис. 42. Назначение метода обработки события команде «Добавить…» из меню «Перевозка«

Точно также назначаем другим командам меню следующие обработчики событий:

- команде «Удалить» из меню «Перевозка» назначаем обработчик события button3_Click;
- команде «Билеты» из меню «Просмотр» назначаем обработчик события button4_Click;
- команде «Маршруты» из меню «Просмотр» назначаем обработчик события button5_Click;
- команде «Автобусы» из меню «Просмотр» назначаем обработчик события button6_Click;
- команде «Водители» из меню «Просмотр» назначаем обработчик события button7_Click;

- команде «Диспетчеры» из меню «Просмотр» назначаем обработчик события button8_Click.