

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ (МИНОБРНАУКИ РОССИИ)**

**федеральное государственное автономное образовательное учреждение высшего  
образования «Санкт-Петербургский государственный университет аэрокосмиче-  
ского приборостроения»**

---

**ИНТЕРФЕЙСЫ ИНФОРМАЦИОННЫХ СИСТЕМ**

**Учебно-методическое пособие**

**Санкт-Петербург**

**2022 г.**

Составители: Богословская Н. В., Бржезовский А. В.

Рецензент:

В учебно-методическое пособие включены материалы, относящиеся к основам создания интерфейсов информационных систем, в том числе:

— основные сведения о библиотеках и подходах к созданию интерфейса пользователя (Windows Forms, Windows Presentation Foundation, ASP.NET и др.);

— типовые решения и шаблоны для организации доступа к данным (Transaction Script, Domain Model, Table Module, Row/Table Data Gateway, Record Set, Active record, Unit of Work и др.);

— описание основных возможностей библиотек для работы с базами данных (ADO.NET, Entity Framework и др.).

Учебно-методическое пособие предназначено для студентов, обучающихся по направлению 09.03.02 – Информационные системы и технологии.

# 1. Проектирование интерфейса

## 1.1 Windows Forms

### 1.1.1 Создание проекта Windows Forms

Для создания проекта Windows Forms в Windows 7 необходимо:

(i) запустить среду Microsoft Visual Studio (далее MSVS): **Пуск\Все программы\Microsoft Visual Studio ...\Visual Studio ...**;

(ii) создать проект Windows Forms (далее WF): **Файл\Создать\Проект ...\Visual C#\Приложение Windows Forms\Ok**, задав при необходимости имя решения (например, *MyPatterns*) и путь для хранения файлов проекта [<https://docs.microsoft.com/ru-ru/visualstudio/ide/create-csharp-winform-visual-studio?toc=%2Fvisualstudio%2Fget-started%2Fcsharp%2Ftoc.json&bc=%2Fvisualstudio%2Fget-started%2Fcsharp%2Fbreadcrumb%2Ftoc.json&view=vs-2019>].

После создания приложения на экране появляется форма, в которую можно добавлять диалоговые элементы (например, *TextBox*) посредством их перетаскивания из панели **Панель элементов** (рис. 1.1) [[https://msdn.microsoft.com/ru-ru/library/0h5y8567\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/0h5y8567(v=vs.110).aspx)]. Свойства элементов задаются в окне **Свойства**, расположенном по умолчанию в правой нижней части экрана (рис. 1.1).

Типовые приложения, работающие с базами данных, часто имеют в качестве основных формы, содержащие списки с объектами предметной области, на основе которых организуется диалог с пользователем посредством иерархических, контекстных или меню, расположенных на вкладках (рис. 1.2, 1.3).

Для ввода/редактирования детальных сведений об объектах предметной области создаются формы ввода, которые содержат набор типовых диалоговых элементов, пример формы ввода приведен на рис. 1.4. Полный перечень диалоговых элементов приведен в [[https://msdn.microsoft.com/ru-ru/library/3xdhey7w\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/3xdhey7w(v=vs.110).aspx)].

Для добавления формы к проекту необходимо в контекстном меню узла проекта в правой верхней части экрана (на рис. 1.1. — *MyPatterns*) выполнить команды: **Добавить\Форма Windows ...\Форма Windows Forms\Добавить**, задав при необходимости имя для новой формы.

Основные сведения о работе с формами в приложениях Windows Forms приведены в [[https://msdn.microsoft.com/ru-ru/library/ms229601\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/ms229601(v=vs.110).aspx)].

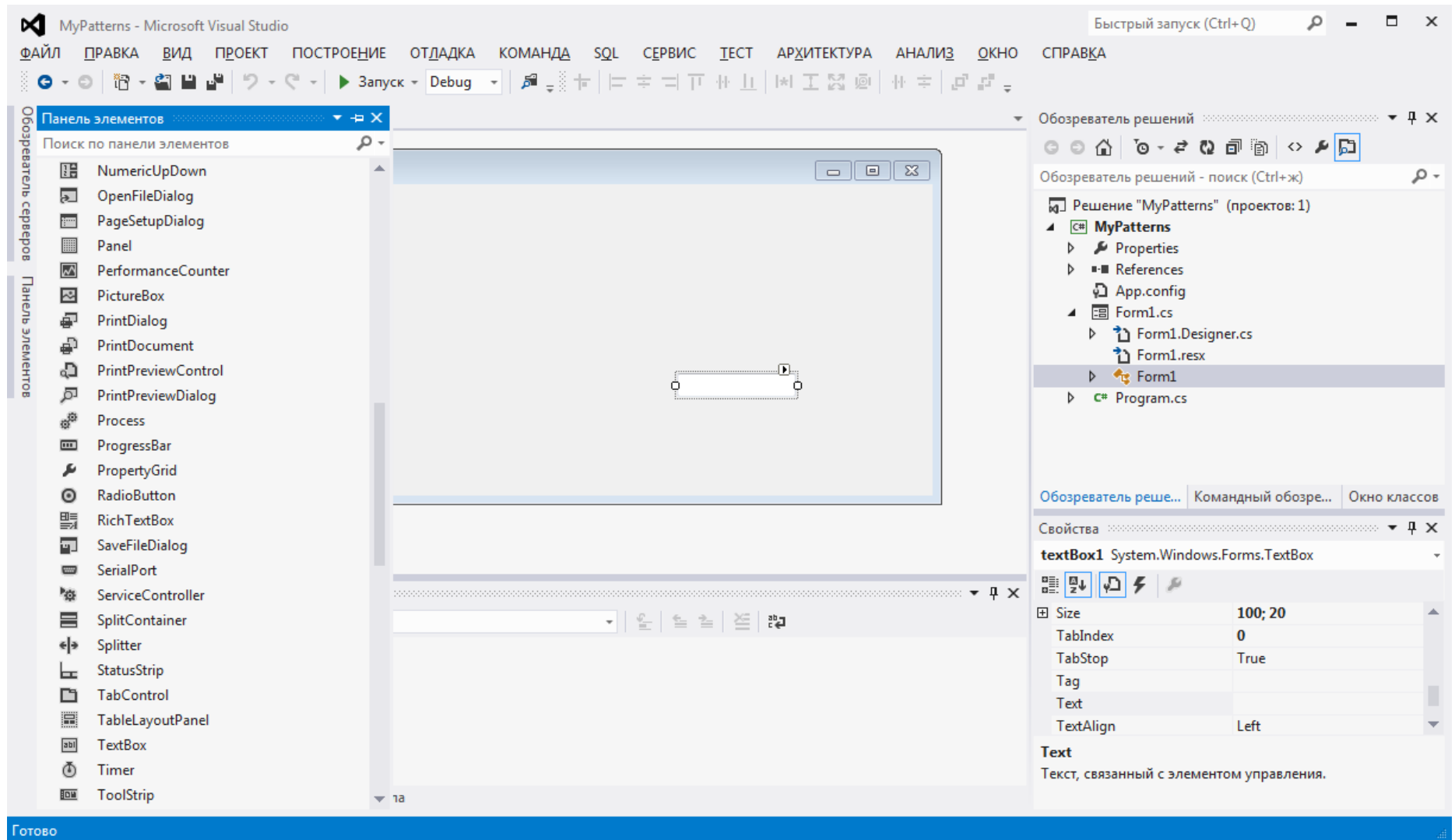


Рис. 1.1.

MVOS Office

About

Orders Mail Items Suppliers Agents Ports Rates Budget Budget Items Units Users Admin Patterns

Refresh Create Edit Delete Create copy Search XLS Type: All Vessel: All Status: All From: 19.12.2017 To: 19.01.2018 Apply

	Order number	Type	Created on	Created for	Creation date	Status	User	Delivery date	Port	Agent	Supplier	Supply date	Received on board	Currency	Total	With discount	Invoice number	Invoice date	Invoice total	Comments
	1-00	Provision...	Office	Imi	30.12.2017	edit	avb												0,00	
*																				

*Puc. 1.2.*

MVOS Office

About

Orders Mail Items Suppliers Agents Ports Rates Budget Budget Items Units Users Admin Patterns

Refresh Add Delete Change Unbound Add Part Delete Part Change Part Search Part

Search

Provision/galley

- provision
- galley stores

Deck

- stores
  - diverse stores
  - lamps/bulbs
  - flags
  - electrical tools
  - paint equipment
  - mooring equipment
- maritime publications
- chemicals
- paint
- cargo handling equipment
- navigation equipment
- safety equipment
  - life rafts
  - fire equipment
  - LSA

Engine

- stores
- filters
  - main engine
  - auxiliary engine
  - separators
- spares
  - main engine
  - auxiliary engine
  - separators
  - propulsion
- diverse

Bunkers/luboil

- bunkers

Provisions

Name	Unit	Price	Currenc	Commor	Archive	Is Part	Equipme	Maker	Type	Serial N	Group N	Drawing N	Parts N	Add. Inf.
Beet roots	kg	0.29	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
Mushrooms mix frozen 600gr	piece	3.95	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
Pork neck - boneless	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
beef cuberoll	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
chicken legs quarters E.Q.	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
smocked salmon fillets sliced 200gr pack	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
fish fingers (oven-ready)	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
dumpings with pork and beef staffed E.Q.	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
salami lithuanian	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
doktor sausage	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
beef and pork sausages (sosiski) E.Q.	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
beef and pork thick sausages (sardelki) E.Q.	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
grill sausages	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
smocked natural ham	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
mackerel cured	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
low salted herring fillets in oil (2kgs pack)	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
Cheese, cottage 9% 200gr pack	cup	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
sour cream 20% (500gr pack)	cup	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
gouda cheese brick	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
edam cheese brick	kg	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
Milk LL 3.2%-3.5%	liter	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
milk L.L. 1%-1.5%	liter	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
processed cheese assorted in cup 150gr	cup	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								
philadelphia cheese assorted	cup	0.00	EUR	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>								

Puc. 1.3.

Edit order type of Provision/galley

Order number:  
1-03

Created on:  
Anmi

Created for:  
Anmi

Delivery date:

Supply date:

Received on board:

Supplier (dictionary):

Port (dictionary):

Agent (dictionary):

Supplier (text):

Port (text):

Agent (text):

Currency:

Total:  
0,00

With discount:  
0,00

Invoice number:

Invoice date:

Invoice total:  
0,00

Comments:

Add Item

Add Items

Delete Item

Delete Items

Change Item

Save

Close

Save and Close

onboard

Formed

Received

	Status	User	Moment	Comments
	onboard	avb	12.09.2017 12:41	
*				

	Name	Unit	Price	Qty	Qty (fact)	Amount	Discount (%)	With discount	Comments
*									

Рис. 1.4.

### 1.1.2 Обработка событий в формах

Примеры создания приложения WF без использования MSVS приведены в [3] и [\[https://msdn.microsoft.com/ru-ru/library/ms229597\(v=vs.110\).aspx\]](https://msdn.microsoft.com/ru-ru/library/ms229597(v=vs.110).aspx).

При создании проекта в MSVS автоматически создается ряд файлов, одним из которых является **Program.cs**, содержащий функцию **Main()** и обеспечивающий запуск приложения, а также реализующий цикл обработки событий приложения:

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

namespace MyPatterns
{
    static class Program
    {
        /// <summary>
        /// Главная точка входа для приложения.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Другим автоматически создаваемым файлом является **Form1.cs**, изначально содержащий конструктор для класса **Form1**:

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



```

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

```

Дополним форму на рис. 1.1. кнопкой, перетаскив из панели элементов **Button** внутрь формы. Установим свойство кнопки **Text** в **Clear**. Добавим обработчик события нажатия кнопки **button1\_Click** (двойной щелчок на кнопке в конструкторе формы), задав в нем установку значения свойства **Text** для **textBox1** в пустую строку. Дополним конструктор класса **Form1** инициализацией свойства **Text** для **textBox1**. Создадим обработчик события **FormClosing**, выбрав в окне свойств формы панель событий (пиктограмма в форме молнии) и выполнив двойной щелчок на событии **FormClosing**:

```

using ...

namespace MyPatterns
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            textBox1.Text = "Input text ...";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            textBox1.Text = "";
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            MessageBox.Show("Text: " + "" + textBox1.Text + "");
        }
    }
}

```

После запуска приложения (пиктограмма в форме зеленого треугольника *Запуск*), форма примет вид (рис. 1.5), при нажатии кнопки *Clear* будет осуществляться очистка текстового поля, а перед закрытием окна формы — вывод сообщения о введенном пользователем тексте.

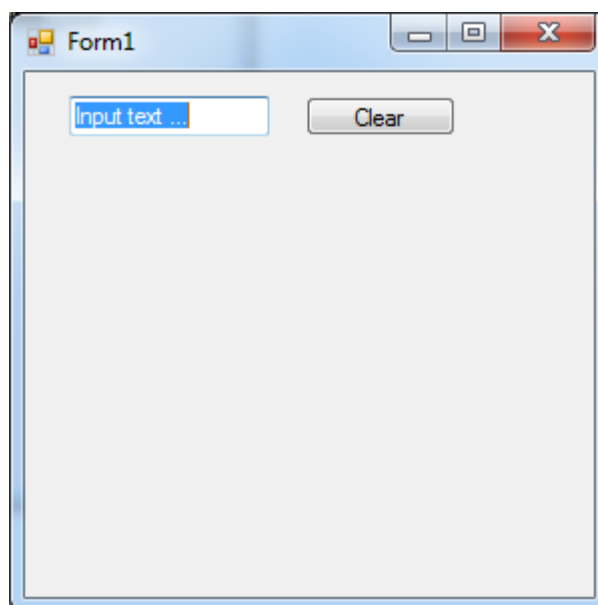


Рис. 1.5.

### 1.1.3 Основные диалоговые элементы

Большинство диалоговых элементов наследуют от класса *Control*, для которого определено достаточно много свойств, задающих параметры отображения элементов управления, методов, их изменяющих и событий, возникающих при работе с диалоговыми элементами [\[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.control\(v=vs.110\).aspx\]](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.control(v=vs.110).aspx).

Класс *Button* наследуется от класса *ButtonBase*, основным событием, подлежащим обработке при использовании *Button*, является нажатие, пример обработки нажатия кнопки был приведен в п. 1.2 [\[https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button\(v=vs.110\).aspx\]](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.button(v=vs.110).aspx). В диалоговых окнах часто используются типовые кнопки *Yes/No*, *Ok/Cancel* и др., для которых предусмотрена стандартная обработка — закрытие формы и возврат в точку ее вызова результата, показываемого перечислением *DialogResult* [\[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.dialogresult\(v=vs.110\).aspx\]](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.dialogresult(v=vs.110).aspx). Закрытие формы часто осуществляется также при нажатии клавиш *Enter* или *Esc*, кнопка, соответствующая нажатию *Enter*, задается свойством формы *AcceptButton*, а нажатию *Esc* — свойством *CancelButton*.

Класс *CheckBox* также унаследован от *ButtonBase* [\[https://msdn.microsoft.com/ru-ru/library/system.windows.controls.checkbox\(v=vs.110\).aspx\]](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.checkbox(v=vs.110).aspx), он предназначен для отображения или ввода трех возможных состояний, задаваемых перечислением *CheckState*

[[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkstate\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkstate(v=vs.110).aspx)], текущее значение содержится в свойстве **CheckBox.CheckState** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.checkstate\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.checkstate(v=vs.110).aspx)]. Допустимость третьего состояния зависит от значения свойства **CheckBox.ThreeState** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.threestate\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.threestate(v=vs.110).aspx)], если оно установлено в *false*, допускается только два возможных состояния, установка/опрос которых осуществляется посредством свойства **CheckBox.Checked** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.checked\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkbox.checked(v=vs.110).aspx)].

Класс **RadioButton** тоже наследует от **ButtonBase** [[https://msdn.microsoft.com/ru-ru/library/system.windows.controls.radiobutton\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.radiobutton(v=vs.110).aspx)], как правило, используется в группе, если в одном контейнере расположено несколько радиокнопок, выбрана может быть только одна из них.

Классы **ComboBox** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.combobox\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.combobox(v=vs.110).aspx)], **ListBox** [[https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.controls.listbox(v=vs.110).aspx)], **CheckedListBox** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkedlistbox\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.checkedlistbox(v=vs.110).aspx)] наследуют от **ListControl**. **ListBox** и **CheckedListBox** предполагают множественный выбор, **ComboBox** используется для выбора единственного элемента (рис. 1.6).

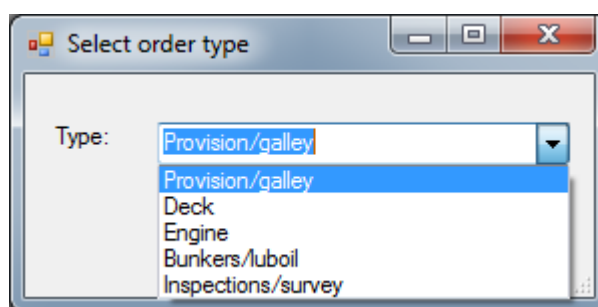


Рис. 1.6.

Классы имеют свойство **Items**, представляющее коллекцию из элементов списка, для ее заполнения используется метод **Add**. Альтернативным способом формирования списка является указание источника данных, приведенное в примере ниже.

```
private void GetOrderTypes()
{
    try
    {
        dsOrderType.Clear();
```

```

da.SelectCommand = new SqlCommand("select id, name from
                                   ord_r_tip", cnn);
da.Fill(dsOrderType, "ord_r_tip");

this.cbOrderType.DataSource =
    dsOrderType.Tables["ord_r_tip"];
this.cbOrderType.DisplayMember = "name";
this.cbOrderType.ValueMember = "id";
}
catch (Exception e)
{
    MessageBox.Show("Internal error: " + e.Message);
}
}

```

В данном случае свойство ***DataSource*** используется для указания источника данных для ***ComboBox***, в качестве которого выступает экземпляр класса ***DataSet***, содержащий данные, выбранные из БД. Свойство ***DisplayMember*** указывает, какой элемент должен отображаться в списке, а свойство ***ValueMember*** задает элемент, значение которого должно возвращаться как результат выбора.

Свойства ***SelectedItem*** и ***SelectedIndex*** позволяют получить выбранный элемент списка, в списках с множественным выбором для этого используется коллекция ***SelectedItems***. Пример использования ***CheckedListBox*** показан на рис. 1.7.

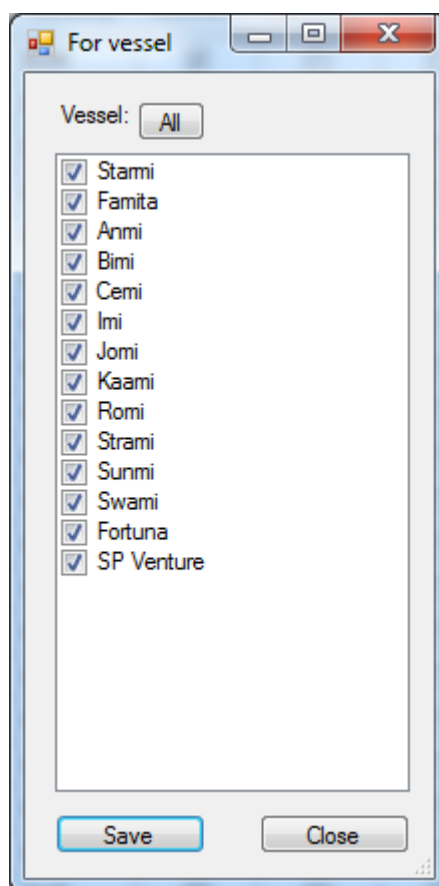


Рис. 1.7.

Класс ***DataGridView*** [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datagridview\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datagridview(v=vs.110).aspx)] служит для отображения табличных данных, источниками которых могут быть объекты классов ***Array***, ***DataTable***, ***DataView***, ***DataSet***, а также компоненты, реализующие интерфейсы ***IList*** и ***IListSource***. Примеры использования ***DataGridView*** приведены на рис. 1.2, 1.3, 1.4. В приложениях, работающих с базами данных чаще всего источниками данных служат объекты класса ***DataSet***, пример заполнения ***DataGridView*** на основе ***DataSet*** приведен ниже.

```
private void showStates()
{
    try
    {
        dsStates.Clear();
        da.SelectCommand = new SqlCommand("select id as Number,
                                           name as Name, color as Color
                                           from ord_r_state order by name", cnn);
        da.Fill(dsStates, "ord_r_state");
        dgvStates.DataSource = dsStates.Tables["ord_r_state"];
    }
    catch (Exception e)
```

```

    {
        MessageBox.Show("Internal error: " + e.Message);
    }
}

```

Здесь, также как в примере с *ComboBox* используется свойство *DataSource*, в качестве значения которого указывается экземпляр *DataSet*.

Функции сортировки и фильтрации строк в наборе данных реализует класс *DataGridView* [[https://msdn.microsoft.com/ru-ru/library/system.data.dataview\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.data.dataview(v=vs.110).aspx)], он не сохраняет данные, и является надстройкой над *DataTable*. Фильтрация реализуется установкой значения свойства *RowFilter*, синтаксис задаваемого здесь условия аналогичен используемому в разделе *where* запросов на языке SQL.

Фильтрация строк возможна также по их состоянию с помощью свойства *RowStateFilter*, возможные состояния определяются перечислением *DataGridViewRowState* [[https://msdn.microsoft.com/ru-ru/library/system.data.dataviewrowstate\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.data.dataviewrowstate(v=vs.110).aspx)].

Для сортировки строк используется свойство *Sort*, синтаксис выражения, определяющего порядок сортировки аналогичен используемому в разделе *order by* запросов на языке SQL.

Класс *DateTimePicker* позволяет выбирать значения даты/времени в различных форматах, формат задается значением свойства *Format* посредством его выбора из перечисления *DateTimePickerFormat* [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepickerformat\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepickerformat(v=vs.110).aspx)], при выборе варианта *Custom* для задания формата используется свойство *CustomFormat* [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker.customformat\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.datetimepicker.customformat(v=vs.110).aspx)]. Свойства *Text* и *Value* позволяют получить выбранное значение в символьном представлении и в виде объекта *DateTime*. Свойства *MinDate* и *MaxDate* позволяют ограничить диапазон, из которого осуществляется выбор.

Класс *ImageList* используется для отображения списка изображений [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.imagelist\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.imagelist(v=vs.110).aspx)]. Свойство *Images* представляет коллекцию изображений, добавление в которую производится методом *Add*. Размер изображений регулируется свойством *ImageSize*, принимающим значения структуры *Size* (диапазон допустимых значений 1..256, по умолчанию 16x16) [[https://msdn.microsoft.com/ru-ru/library/system.drawing.size\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.drawing.size(v=vs.110).aspx)], а свойство *ColorDepth* задает глубину цвета (4..32 бит), принимая значения из одноименного перечисления [[https://msdn.microsoft.com/ru-ru/library/system.windows.forms.colordepth\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.windows.forms.colordepth(v=vs.110).aspx)].

Класс ***Label*** применяется для задания имен/пояснений к другим диалоговым элементам

...

## 1.2 Windows Presentation Foundation

### 1.2.1 Язык XAML

В связи с бурным развитием интернет-приложений и приложений для мобильных устройств популярным подходом к проектированию интерфейса пользователя в последнее время стало использование языков разметки. Их применение позволяет разделить работы по дизайну и реализации интерфейса между профессиональными дизайнерами и профессиональными программистами. В связи с этим Microsoft был реализован класс приложений Windows Presentation Foundation (WPF).

В приложениях WPF используется язык разметки интерфейсов eXtensible Application Markup Language (XAML) [<https://docs.microsoft.com/ru-ru/windows/uwp/xaml-platform/xaml-overview>].

### 1.2.2 Создание проекта WPF

Создание приложений WPF в MSVS осуществляется аналогично приложениям WF [<https://docs.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-wpf?view=vs-2019>].

Набор диалоговых элементов, используемых в WPF схож с диалоговыми элементами WF.

## 1.3 Приложения ASP.NET

### 1.3.1 Создание проекта ASP.NET

В отличие от WF и WPF, ориентированных на создание настольных бизнес-приложений, технология ASP.NET обеспечивает разработку Web-приложений, где в качестве языка разметки интерфейса используется HTML, расширенный серверными элементами управления [<https://docs.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-aspnet-core?view=vs-2019>].

### 1.3.2 Диалоговые элементы ASP.NET

Серверные элементы ASP.NET по своему составу и возможностям схожи с диалоговыми элементами WF и WPF [[https://msdn.microsoft.com/ru-ru/library/system.web.ui.webcontrols\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.web.ui.webcontrols(v=vs.110).aspx)].

## 1.4 Лабораторная работа 1

Для выбранного варианта задания спроектировать интерфейс информационной системы (ИИС), обеспечивающий работу с основными объектами предметной области (ПрО), хранимыми в базе данных (БД) информационной системы (ИС).

Содержание отчета:

- краткие функциональные требования к ИС;
- основные формы приложения для работы с БД (дизайн форм в выбранной системе программирования без программирования обработчиков событий);
- описание сценариев и логики диалога с пользователем (проект).

Варианты заданий приведены в ПРИЛОЖЕНИИ.



## 2. Программирование интерфейса

Современные СУБД предоставляют достаточно развитые возможности в части выборки данных, организации многопользовательской работы и др. механизмы. Вместе с тем, при разработке приложений баз данных (информационных систем) возникает ряд вопросов относительно организации кэширования данных между БД и приложением, организации пользовательских транзакций, блокировок и пр. Для решения этих вопросов в литературе [1] [2] [3] [4] [5] [6] [7] предлагаются типовые решения (шаблоны проектирования). Выбор шаблона зависит от природы решаемых задач, сложности системы, используемых библиотек для доступа к данным.

К числу шаблонов, используемых для реализации работы приложений с базами данных, относятся:

- (i) Оптимистическая автономная блокировка (Optimistic Offline Lock)
- (ii) Пессимистическая автономная блокировка (Pessimistic Offline Lock)
- (iii) Блокировка с низкой степенью детализации (Coarse-Grained Lock)
- (iv) Неявная блокировка (Implicit Lock)
- (v) Отображение метаданных (Metadata Mapping)
- (vi) Отображение с помощью таблицы ассоциаций (Association Table Mapping)
- (vii) Отображение внешних ключей (Foreign Key Mapping)
- (viii) Поле идентификации (Identity Field)
- (ix) Загрузка по требованию (Lazy Load)
- (x) Коллекция объектов (Identity Map)
- (xi) Единица работы (Unit of Work)
- (xii) Преобразователь данных (Data Mapper)
- (xiii) Активная запись (Active Record)
- (xiv) Шлюз записи данных (Row Data Gateway)
- (xv) Шлюз таблицы данных (Table Data Gateway)
- (xvi) Модуль таблицы (Table Module)
- (xvii) Модель предметной области (Domain Model)
- (xviii) Сценарий транзакции (Transaction Script)
- (xix) Объект запроса (Query Object)Specification
- (xx) Множество записей (Record Set)

## 2.1 Библиотеки и типовые решения для доступа к данным

В продуктах Microsoft предлагаются следующие возможности для доступа к данным, хранящимся под управлением СУБД:

- (xxi) ADO.NET (Active Data Objects и платформа .NET Framework) — набор классов для установления соединений, выборки и сохранения данных в БД [<https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/>];
- (xxii) ADO.NET Entity Framework — продукт класса ORM (Object-Relational Mapping) [<https://docs.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/>] [<https://docs.microsoft.com/ru-ru/ef/>].

Некоторые авторы более удачным решением для 2) считают библиотеку Hibernate [<http://hibernate.org>], изначально разработанную для платформы Java и портированную на платформу .NET под названием NHibernate [<https://nhibernate.info>].

## 2.2 Шаблон сценарий транзакции (Transaction Script)

Сценарий транзакции используется в несложных бизнес-приложениях [1] [2], в которых бизнес-логика может быть сформулирована в виде набора логически законченных действий над базой данных.

В форме на рис. 1.2 пользователю отображается перечень заказов с учетом выбранного типа, судна, статуса, временного периода. Выборку данных и заполнение *DataGridView* осуществляет следующая функция, работающая по сценарию транзакции:

```

private void showOrders()
{
    try
    {
        dsOrders.Clear();

        string orderTypeId = GetOrderTypeId().ToString();
        string shipId = "-1";
        if (thisNodeId > 0) { shipId = thisNodeId.ToString(); }
        if (thisNodeId == 0) { shipId = GetShipId().ToString(); }
        string stateId = GetStateId().ToString();
        string fromDateTxt = tbFrom.Text;
        string toDateTxt = tbTo.Text;

        DateTime fromDate;
        if (String.IsNullOrEmpty(fromDateTxt))
        {
            MessageBox.Show("Date from is null or empty"); return; }
        else if (!DateTime.TryParse(fromDateTxt, out fromDate))
        {
            MessageBox.Show("Invalid date from '" + fromDateTxt + "'");
            return; }
        fromDateTxt = "'" + fromDate.Year.ToString() + "-" + fromDate.Month.ToString() +
            "-" + fromDate.Day.ToString() + "'";

        DateTime toDate;
        if (String.IsNullOrEmpty(toDateTxt))
        {
            MessageBox.Show("Date to is null or empty"); return; }
        else if (!DateTime.TryParse(toDateTxt, out toDate))
        {
            MessageBox.Show("Invalid date to '" + toDateTxt + "'"); return; }
        toDateTxt = "'" + toDate.Year.ToString() + "-" + toDate.Month.ToString() +
            "-" + toDate.Day.ToString() + "'";
    }
}

```

```

string procName;
if (thisNodeId == 0) { procName = "exec ord_rsel " + orderTypeId + ", " +
    shipId + ", " + stateId + ", " + fromDateTxt + ", " + toDateTxt; }
else if (thisNodeId > 0) { procName = "exec ord_rsel_ship " + orderTypeId +
    ", " + shipId + ", " + stateId + ", " + fromDateTxt + ", " + toDateTxt; }
else { MessageBox.Show("No this"); return; }
//MessageBox.Show(procName);
da.SelectCommand = new SqlCommand(procName, cnn);
da.Fill(dsOrders, "ordr");
dgvOrders.DataSource = dsOrders.Tables["ordr"];
dgvOrders.Columns["id"].Visible = false;
dgvOrders.Columns["ship_id"].Visible = false;
dgvOrders.Columns["supplier_id"].Visible = false;
dgvOrders.Columns["supplier_text"].Visible = false;
dgvOrders.Columns["port_id"].Visible = false;
dgvOrders.Columns["port_text"].Visible = false;
dgvOrders.Columns["agent_id"].Visible = false;
dgvOrders.Columns["agent_text"].Visible = false;
dgvOrders.Columns["ordr_state_id"].Visible = false;
dgvOrders.Columns["ordr_tip_id"].Visible = false;
dgvOrders.Columns["Total"].DefaultCellStyle.Format = "#,###,##0.00";
dgvOrders.Columns["With discount"].DefaultCellStyle.Format = "#,###,##0.00";
dgvOrders.Columns["Invoice total"].DefaultCellStyle.Format = "#,###,##0.00";
setOrderColor();
}
catch (Exception e)
{
    MessageBox.Show("Internal error: " + e.Message);
}
}

```

Для заполнения *DataSet*, являющегося источником данных для *DataGridView*, используется команда SQL для *DataAdapter*, соответствующая вызову хранимой процедуры *ordr\_sel*:

```
create proc ordr_sel @ordr_tip_id tinyint, @ship_id tinyint, @ordr_state_id tinyint, @from_date date,
    @to_date date
as
select      ordr.id as 'id',
    ordr.ship_id as 'ship_id',
    ordr.number as 'Order number',
    case
        when (ordr.ordr_tip_id is null) then ''
        else (select grup.name from grup where grup.id = ordr.ordr_tip_id)
    end as 'Type',
    ship_on.name as 'Created on',
    ship_for.name as 'Created for',
    ordr.creation_date as 'Creation date',
    ordr_state.name as 'Status',
    (select usr.name from usr, ordr_log where ordr_log.ordr_id = ordr.id and ordr_log.ordr_ship_id =
        ordr.ship_id and ordr_log.usr_id = usr.id and ordr_log.ordr_state_id = ordr.ordr_state_id and
        ordr_log.moment = (select max(moment) from ordr_log where ordr_log.ordr_id = ordr.id and
        ordr_log.ordr_ship_id = ordr.ship_id and ordr_log.ordr_state_id = ordr.ordr_state_id)
    ) as 'User',
    delivery_date as 'Delivery date',
    case
        when (ordr.port_id is null) then port
        else (select port.name from port where port.id = ordr.port_id)
    end as 'Port',
    case
        when (ordr.agent_id is null) then agent
        else (select agent.name from agent where agent.id = ordr.agent_id)
    end as 'Agent',
    case
```

```

        when (ordr.supplier_id is null) then supplier
        else (select supplier.name from supplier where supplier.id = ordr.supplier_id)
    end as 'Supplier',
    ordr.supplier_date as 'Supply date',
    ordr.implement_date as 'Received on board',
    ordr.currency as 'Currency',
    ordr.total as 'Total',
    ordr.totald as 'With discount',
    ordr.account_number as 'Invoice number',
    ordr.account_date as 'Invoice date',
    ordr.account_total as 'Invoice total',
    ordr.comment as 'Comments',
    ordr.supplier_id as 'supplier_id',
    ordr.supplier as 'supplier_text',
    ordr.port_id as 'port_id',
    ordr.port as 'port_text',
    ordr.agent_id as 'agent_id',
    ordr.agent as 'agent_text',
    ordr.ordr_state_id as 'ordr_state_id',
    ordr.ordr_tip_id as 'ordr_tip_id'

```

```

from ordr, ship ship_on, ship ship_for, ordr_state

```

```

where ordr.ship_id = ship_on.id and ordr.for_ship_id = ship_for.id and ordr_state_id = ordr_state.id and
    (ordr.ordr_state_id > 1 or (ordr.ordr_state_id = 1 and ordr.ship_id = 0)) and
    (ordr.for_ship_id = @ship_id or @ship_id = 0) and (ordr.ordr_tip_id = @ordr_tip_id or
    @ordr_tip_id = 0) and (ordr.ordr_state_id = @ordr_state_id or @ordr_state_id = 0) and
    (ordr.creation_date between @from_date and @to_date)

```

```

order by moment desc

```

```

go

```

Хранимая процедура обращается к ранее созданным таблицам базы данных:

```
create table ship (  
    id smallint primary key,  
    name varchar(20) not null,  
    number char(2) not null default '00',  
    email varchar(50) null,  
    type varchar(20) null,  
    flag varchar(20) null,  
    class varchar(50) null,  
    built smallint null,  
    red bit not null default 0,  
    yellow bit not null default 0,  
    this bit not null default 0,  
    seq smallint null,  
    rv bigint default 0,  
    ship_id uniqueidentifier ) -- from maintenance system
```

```
go
```

```
create table port (  
    id smallint primary key,  
    name varchar(150) not null,  
    country varchar(4) null,  
    countryn varchar(4) null,  
    archive bit not null default 0,  
    rv bigint default 0 )
```

```
go
```

```
create table supplier (  
    id int primary key,  
    name varchar(150) not null,  
    country varchar(4) null,  
    countryn varchar(4) null,  
    phone varchar(150) null,
```

```
        email varchar(150) null,  
        archive bit not null default 0,  
        port_id smallint null,  
        rv bigint default 0 )  
go  
  
alter table supplier add address varchar(150) null  
go  
  
create table grup (  
    id smallint primary key,  
    name varchar(50) not null,  
    seq int null,  
    rv bigint default 0 )  
go  
  
create table usr (  
    id smallint not null primary key,  
    name varchar (50) not null,  
    ref_name varchar (100) not null,  
    archive bit not null default 0,  
    rv bigint default 0 )  
go  
  
alter table usr add email varchar(150) null  
go  
  
create table agent (  
    id int primary key,  
    name varchar(150) not null,  
    country varchar(4) null,  
    countryn varchar(4) null,
```



```

        address varchar(150) null,
        phone varchar(150) null,
        email varchar(150) null,
        archive bit not null default 0,
        port_id smallint,
        rv bigint default 0 )
go

alter table agent add mobile varchar(150) null
go

alter table agent add city varchar(150) null
go

create table ord_r_state (
    id tinyint primary key,
    name varchar(20) not null,
    color int not null default 0xFF000000,
    background int not null default 0xFFFFFFFF,
    rv bigint default 0 )
go

create table ord_r (
    id int not null,
    ship_id smallint not null,
    for_ship_id smallint not null,
    creation_date date,
    delivery_date date,
    delivery_address varchar (150), -- ???
    supplier_id int,
    supplier varchar (150),
    port_id smallint,

```

```

port varchar (150),
agent_id int,
agent varchar (150),
total money,
totald money,    -- with discount
currency varchar(3),
number varchar(20) not null unique,
supplier_date date,
implement_date date,
comment varchar (150),
ordr_state_id tinyint not null,
ordr_tip_id tinyint,
account_date date,
account_number varchar(20),
account_total money,
account_currency varchar(3),    -- <> currency ???
chck_in_usr_id smallint,
rv bigint default 0,
moment datetime not null default GetDate(),
primary key (id, ship_id) )

go

create table ordr_log (
    id bigint not null,
    ship_id smallint not null,
    ordr_id int not null,
    ordr_ship_id smallint not null,
    usr_id smallint not null,
    ordr_state_id tinyint not null,
    moment datetime not null default GetDate(),
    comment varchar (150),
    primary key (id, ship_id) )

```

go

Еще одним действием, выполняемым по сценарию транзакции, является перевод заказа (рис. 1.4) из состояния в состояние кнопкой *Formed/Change state*:

```
private void btnChangeState_Click(object sender, EventArgs e)
{
    try
    {
        if ((orderStateId == 1) && (dgvItems.RowCount == 1))
            { MessageBox.Show("Order contain no items"); return; }

        byte state = 0;
        byte oldState = 0;
        short user = -1;
        string comment = "";

        oldState = orderStateId;
        if (thisNodeId == 0) { state = GetOrderState(); }
        else { state = 2; }
        if (state == 0) { MessageBox.Show("Undefined order status '" + cbState.Text + "'"); return; }

        DateTime dt;
        string receivedOnBoard = GetDeliveryDate();
        if ((String.IsNullOrEmpty(receivedOnBoard)) && (state == 2))
            { MessageBox.Show("Empty delivery date, set and save order"); return; }

        receivedOnBoard = GetSupplyDate();
        if ((String.IsNullOrEmpty(receivedOnBoard)) && (state == 7))
            { MessageBox.Show("Empty supply date, set and save order"); return; }

        SqlCommand cmd = new SqlCommand("usr_sel_id", cnn);
```

```

cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@name", SqlDbType.VarChar, 100);
cmd.Parameters["@name"].Value = userNameWin;

cmd.Parameters.Add("@id", SqlDbType.SmallInt);
cmd.Parameters["@id"].Direction = ParameterDirection.Output;

cmd.Parameters.Add("RETURN_VALUE", SqlDbType.Int);
cmd.Parameters["RETURN_VALUE"].Direction = ParameterDirection.ReturnValue;

cnn.Open();
cmd.ExecuteScalar();
cnn.Close();

int i = Convert.ToInt32(cmd.Parameters["RETURN_VALUE"].Value);

if (i == 1) { MessageBox.Show("Empty name"); return; }
else if (i == 2) { MessageBox.Show("User '" + userNameWin + "' not found"); return; }

user = Convert.ToInt16(cmd.Parameters["@id"].Value);
if (user == -1) { MessageBox.Show("Internal error"); return; }

cmd.Dispose();

InputDialog idComments = new InputDialog();
idComments.nameLabel1 = "Comments:";
idComments.objects = "Change status";
idComments.operation = "ok";
idComments.ShowDialog();
if (idComments.DialogResult == DialogResult.OK)
    { comment = idComments.valueLabel1; } else { return; };

```

```
cmd = new SqlCommand("ordr_log_ins", cnn);
cmd.CommandType = CommandType.StoredProcedure;

cmd.Parameters.Add("@ordr_id", SqlDbType.Int);
cmd.Parameters["@ordr_id"].Value = orderId;

cmd.Parameters.Add("@ordr_ship_id", SqlDbType.SmallInt);
cmd.Parameters["@ordr_ship_id"].Value = orderShipId;

cmd.Parameters.Add("@usr_id", SqlDbType.SmallInt);
cmd.Parameters["@usr_id"].Value = user;

cmd.Parameters.Add("@ordr_state_id", SqlDbType.TinyInt);
cmd.Parameters["@ordr_state_id"].Value = state;

cmd.Parameters.Add("@comment", SqlDbType.VarChar, 150);
cmd.Parameters["@comment"].Value = comment;

cmd.Parameters.Add("RETURN_VALUE", SqlDbType.Int);
cmd.Parameters["RETURN_VALUE"].Direction = ParameterDirection.ReturnValue;

cnn.Open();
cmd.ExecuteScalar();
cnn.Close();

i = Convert.ToInt32(cmd.Parameters["RETURN_VALUE"].Value);

if (i == 2) { MessageBox.Show("Empty name"); }
else
{
    orderStateId = state;
}
```

```

        ShowHideControls();
        if (thisNodeId != 0) { cbState.Text = "formed"; }
    }

    cmd.Dispose();
}
catch (Exception ee)
{
    cnn.Close();
    MessageBox.Show("Internal error: " + ee.Message);
}
}

```

Функция вызывает хранимые процедуры *usr\_sel\_id* для получения идентификатора пользователя и *ordr\_log\_ins* для записи в лог, фиксирующий изменение статуса заказа:

```

create proc usr_sel_id @name varchar(50), @id smallint output
as

```

```

    begin

```

```

        select @name = isnull(@name, '')
        if (@name = '') return 1
        if (not exists (select * from usr where name = @name)) return 2
        else select @id = id from usr where name = @name

```

```

    end

```

```

go

```

```

create proc ordr_log_ins @ordr_id int, @ordr_ship_id smallint, @usr_id smallint, @ordr_state_id tinyint,
    @comment varchar (150), @id bigint = null output, @ship_id smallint = null output,
@state_name varchar(20) = null output
as

```

```

    begin

```

```

        declare @ship smallint, @max int, @max_back int

```

```

begin tran
    select @ship = id from ship where this = 1
    select @ship = isnull(@ship, -1)
    if @ship = -1
        begin
            commit
            return 2
        end

    select @max = max(id) from ord_r_log where ship_id = @ship
    select @max = isnull(@max, 0)
    if @ship > 0
        begin
            select @max_back = max(id) from node00b_02..ord_r_log where ship_id = @ship
            select @max_back = isnull(@max_back, 0)
            if @max_back > @max
                select @max = @max_back
            insert into node00b_02..ord_r_log (id, ship_id, ord_r_id, ord_r_ship_id, usr_id,
                                             ord_r_state_id, comment)
            values (@max + 1, @ship, @ord_r_id, @ord_r_ship_id, @usr_id, @ord_r_state_id,
                  @comment)

            update node00b_02..ord_r set ord_r_state_id = @ord_r_state_id where id = @ord_r_id and
                                             ship_id = @ord_r_ship_id

            if @ord_r_state_id = 8
                begin
                    insert into node00b_02..item
                    (id, ship_id, ord_r_id, ord_r_ship_id, goods_id, goods_ship_id,
                     unit_id, price, qty, qty_fact, amount, discount, amountd, comment)
                    select
                    id, ship_id, ord_r_id, ord_r_ship_id, goods_id, goods_ship_id,

```

```

        unit_id, price, qty, qty_fact, amount, discount, amountd, comment
from item
where      (ordr_id = @ordr_id and ordr_ship_id = @ordr_ship_id) and
          (not exists (select * from node00b_02..item it
                      where item.id = it.id and item.ship_id = it.ship_id))

delete node00b_02..item
where      (ordr_id = @ordr_id and ordr_ship_id = @ordr_ship_id) and
          (not exists (select * from item it where item.id = it.id and
                      item.ship_id = it.ship_id))

update node00b_02..item
set
  goods_id = (select it.goods_id from item it
              where item.id = it.id and item.ship_id = it.ship_id),
  goods_ship_id = (select it.goods_ship_id from item it
                  where item.id = it.id and item.ship_id = it.ship_id),
  unit_id = (select it.unit_id from item it
             where item.id = it.id and item.ship_id = it.ship_id),
  price = (select it.price from item it
           where item.id = it.id and item.ship_id = it.ship_id),
  qty = (select it.qty from item it
         where item.id = it.id and item.ship_id = it.ship_id),
  qty_fact = (select it.qty from item it
              where item.id = it.id and item.ship_id = it.ship_id),
  amount = (select it.amount from item it
            where item.id = it.id and item.ship_id = it.ship_id),
  discount = (select it.discount from item it
              where item.id = it.id and item.ship_id = it.ship_id),
  amountd = (select it.amountd from item it
             where item.id = it.id and item.ship_id = it.ship_id),
  comment = (select it.comment from item it

```



```

        where item.id = it.id and item.ship_id = it.ship_id)
    where
        ord_r_id = @ord_r_id and ord_r_ship_id = @ord_r_ship_id

    if (not exists (select * from node00b_02..ord_r where id = @ord_r_id and
        ship_id = @ord_r_ship_id))
        insert into node00b_02..ord_r (id, ship_id, for_ship_id,
            creation_date, delivery_date, supplier_id, supplier, port_id,
            port, agent_id, agent, total, totald, currency,
            number, supplier_date, implement_date, comment, ord_r_state_id,
            account_date, account_number, account_total, ord_r_tip_id)
        select
            id, ship_id, for_ship_id, creation_date, delivery_date,
            supplier_id, supplier, port_id, port, agent_id, agent,
            total, totald, currency, number, supplier_date,
            GetDate(), comment, @ord_r_state_id, account_date,
            account_number, account_total, ord_r_tip_id
        from ord_r where id = @ord_r_id and ship_id = @ord_r_ship_id

    end
end
else
    begin
        insert into ord_r_log (id, ship_id, ord_r_id, ord_r_ship_id, usr_id,
            ord_r_state_id, comment)
        values (@max+1, @ship, @ord_r_id, @ord_r_ship_id, @usr_id, @ord_r_state_id, @comment)
        update ord_r set ord_r_state_id = @ord_r_state_id
            where id = @ord_r_id and ship_id = @ord_r_ship_id
    end
select @id = @max + 1
select @ship_id = @ship

```

```

        select @state_name = name from ordr_state where id = @ordr_state_id
    commit tran
end
go

```

Процедуры работают с таблицами, структура которых была приведена выше, а также таблицей, в которой хранятся позиции заказов:

```

create table item (
    id bigint not null,
    ship_id smallint not null,
    ordr_id int not null,          -- Order
    ordr_ship_id smallint not null, -- composite key
    goods_id int not null,        -- Goods
    goods_ship_id smallint not null, -- composite key
    unit_id smallint,
    price money,
    qty real,
    qty_fact real,
    amount money,
    discount tinyint default 0,
    amountd money, -- with discount
    comment varchar (150),
    primary key (id, ship_id) )
go

```

### **2.3 Шаблон модель предметной области (Doman Model)**

Модель предметной области используется в сложных приложениях и представляет собой систему классов, отображающих в оперативной памяти часть информации, выбираемой из БД [1] [2]. Разработка модели предметной области детально обсуждается в литературе [3] [4] [7].

### **2.4 Лабораторная работа 2**

Для выбранного варианта задания спроектировать слой доступа к данным информационной системы с использованием шаблонов проектирования. Слой доступа к данным должен обеспечивать, выборку, вставку, модификацию и удаление записей в таблицах базы данных.

Содержание отчета:

- обоснование выбора шаблонов проектирования для реализации доступа к БД;
- описание основных принципов работы выбранных шаблонов;
- тексты программ с описаниями.

Варианты заданий приведены в ПРИЛОЖЕНИИ.

## Библиографический список

1. Фаулер М. Архитектура корпоративных программных приложений: Пер. с англ. — М.: Издательский дом "Вильямс", 2006. — 544 с.: ил.
2. Фаулер М. Архитектура корпоративных программных приложений: Пер. с англ. — М.: Издательский дом "Вильямс", 2016. — 544 с.: ил.
3. Эванс Э. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем: Пер. с англ. — М.: Издательский дом "Вильямс", 2011. — 448 с.: ил.
4. Эванс Э. Предметно-ориентированное проектирование (DDD). Структуризация сложных программных систем: Пер. с англ. — М.: Издательский дом "Вильямс", 2018. — 448 с.: ил.
5. Миллет С., Тьюн Н. Предметно-ориентированное проектирование. Паттерны, принципы и методы: Пер. с англ. — СПб.: Издательский дом "Питер", 2017. — 832 с.: ил.
6. Вернон В. Предметно-ориентированное проектирование. Самое основное: Пер. с англ. — М.: Издательский дом "Вильямс", 2017. — 160 с.: ил.
7. Нильсон Дж. Применение DDD и шаблонов проектирования . Проблемно-ориентированное проектирование приложений с примерами на С# и .NET: Пер. с англ. — М.: Издательский дом "Вильямс", 2008. — 560 с.: ил.
8. Нагел К. и др. С# 4.0 и платформа .NET 4 для профессионалов: Пер. с англ. — М.: Издательский дом "Вильямс", 2011. — 1440 с.: ил.
9. Нагел К. и др. С# 5.0 и платформа .NET 4.5 для профессионалов: Пер. с англ. — М.: Издательский дом "Вильямс", 2014. — 1440 с.: ил.

## Содержание

<b>1. Проектирование интерфейса</b>	<b>3</b>
<b>1.1 Windows Forms</b>	<b>3</b>
1.1.1 Создание проекта Windows Forms	3
1.1.2 Обработка событий в формах	8
1.1.3 Основные диалоговые элементы	10
<b>1.2 Windows Presentation Foundation</b>	<b>15</b>
1.2.1 Язык XAML	15
1.2.2 Создание проекта WPF	15
<b>1.3 Приложения ASP.NET</b>	<b>15</b>
1.3.1 Создание проекта ASP.NET	15
1.3.2 Диалоговые элементы ASP.NET	15
<b>1.4 Лабораторная работа 1</b>	<b>16</b>
<b>2. Программирование интерфейса</b>	<b>17</b>
2.1 Библиотеки и типовые решения для доступа к данным	18
2.2 Шаблон сценарий транзакции (Transaction Script)	18
2.3 Шаблон модель предметной области (Domain Model)	35
2.4 Лабораторная работа 2	35
<b>Библиографический список</b>	<b>36</b>
<b>Содержание</b>	<b>37</b>
<b>ПРИЛОЖЕНИЕ</b>	<b>38</b>
<b>Варианты заданий</b>	<b>38</b>

## ПРИЛОЖЕНИЕ

## Варианты заданий

1. Интерфейс информационной системы железной дороги.
2. Интерфейс информационной системы авиакомпании.
3. Интерфейс информационной системы аэропорта.
4. Интерфейс информационной системы морского порта.
5. Интерфейс информационной системы автобусного вокзала.
6. Интерфейс информационной системы школы.
7. Интерфейс информационной системы библиотеки.
8. Интерфейс информационной системы университета.
9. Интерфейс информационной системы службы занятости.
10. Интерфейс информационной системы службы социальной защиты.
11. Интерфейс информационной системы поликлиники.
12. Интерфейс информационной системы обязательного медицинского страхования.
13. Интерфейс информационной системы пенсионного фонда.
14. Интерфейс информационной системы выставочного комплекса.
15. Интерфейс информационной системы для организации НИОКР.
16. Интерфейс информационной системы издательства.
17. Интерфейс информационной системы редакции газеты.
18. Интерфейс информационной системы типографии.
19. Интерфейс информационной системы гостиницы.
20. Интерфейс информационной системы киноцентра.
21. Интерфейс информационной системы фирмы по прокату автомобилей.
22. Интерфейс информационной системы букмекерской фирмы.
23. Интерфейс информационной системы фондовой биржи.
24. Интерфейс информационной системы банка.
25. Интерфейс информационной системы лизинговой компании.
26. Интерфейс информационной системы туристического агентства.
27. Интерфейс информационной системы фильмотеки.
28. Интерфейс информационной системы агентства недвижимости.
29. Интерфейс информационной системы страховой организации.
30. Интерфейс информационной системы автошколы.
31. Интерфейс информационной системы оператора связи.

32. Интерфейс информационной системы автосервиса.
33. Интерфейс информационной системы для оказания госуслуг.
34. Интерфейс информационной системы фирмы по сборке и продаже компьютеров и комплектующих.
35. Интерфейс информационной системы транспортной фирмы.
36. Интерфейс информационной системы супермаркета.
37. Интерфейс информационной системы книжного магазина.
38. Интерфейс информационной системы ломбарда.
39. Интерфейс информационной системы ГИБДД.
40. Интерфейс информационной системы спортивного клуба.
41. Интерфейс информационной системы интернет-провайдера.
42. Интерфейс информационной системы интернет-магазина.
43. Интерфейс информационной системы интернет-аукциона.
44. Интерфейс информационной системы почтовой службы.
45. Интерфейс информационной системы предприятия ЖКХ.
46. Интерфейс информационной системы рекламного агентства.
47. Интерфейс информационной системы курьерской фирмы.
48. Интерфейс информационной системы ресторанного комплекса.
49. Интерфейс информационной системы службы такси.
50. Интерфейс информационной системы службы технической поддержки.
51. Интерфейс информационной системы <свой вариант> (необходимо сформулировать тему).