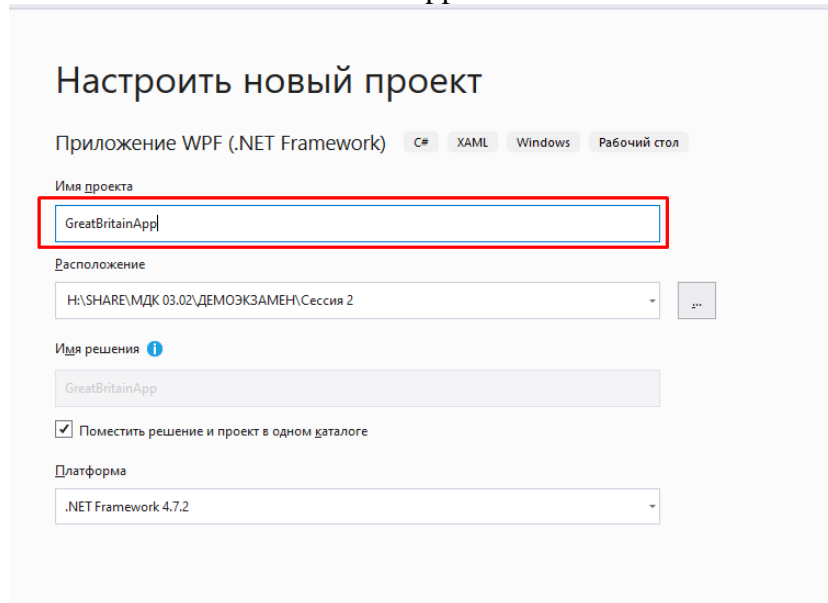


Сессия 2. Выполнение

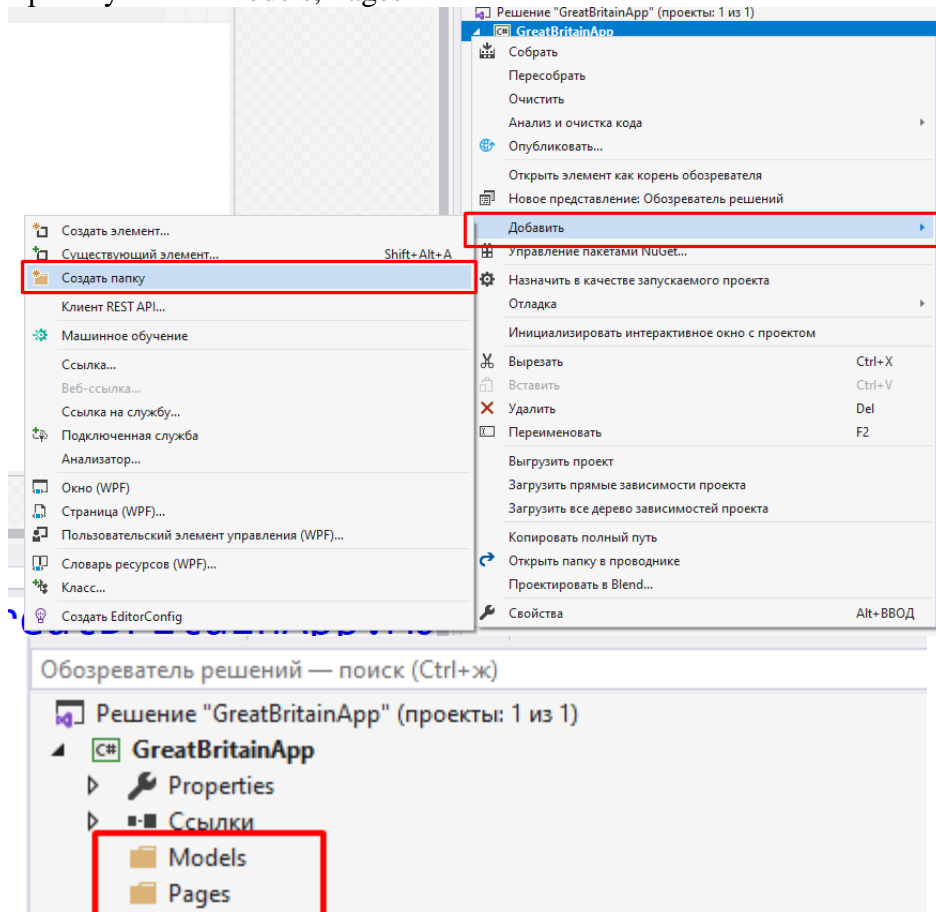
Создание приложения	2
Создание модели БД(Entity FrameWork).....	3
Добавление контекста подключения	6
Класс Good	7
Добавьте класс Manager.....	8
Проектирование каркаса приложения.....	9
Форма авторизации	11
Добавление дополнительных страниц в проект.....	11
Задание стилей	12
Авторизация - форма LoginWindow	14
Файл интерфейса LoginWindow.Xaml	14
Файл программного кода LoginWindow.cs.....	15
Создание папки Images с изображениями.....	16
Главная форма приложения - форма MainWindow	17
Файл интерфейса MainWindow.xaml	17
Файл программного кода MainWindow.cs.....	18
Страница каталога товаров CatalogOfGoodsPage	19
Файл интерфейса CatalogOfGoodsPage.xaml.....	19
Файл программного кода CatalogOfGoodsPage.cs	21
Страница товаров GoodsPage	23
Файл интерфейса GoodsPage.xaml	23
Файл программного кода GoodsPage.cs.....	24
Страница продажи товаров SellGoodsPage	27
Файл интерфейса SellGoodsPage.xaml	27
Файл программного кода SellGoodsPage.cs	29
Страница добавления и редактирования товара AddGoodPage	30
Файл интерфейса AddGoodPage.xaml	30
Файл программного кода AddGoodPage.cs	32
Страница дополнительных товаров AdditionalGoodsPage	37
Файл интерфейса AdditionalGoodsPage.xaml	37
Файл программного кода AdditionalGoodsPage.cs.....	39
Библиотека классов(dll)	42
Создание библиотеки	42
Программный код класса Analytics библиотеки CompanyCoreLib	44
Выполнение сборки библиотеки	45
Модульные тесты UnitTest	46
Создание модульных тестов	46
Добавление ссылки на тестируемую библиотеку.....	48
Программный код CompanyCoreLibTests.cs	49
Запуск тестов.....	51
Справочник	53

Создание приложения

1. Создайте проект WPF с именем GreatBritainApp



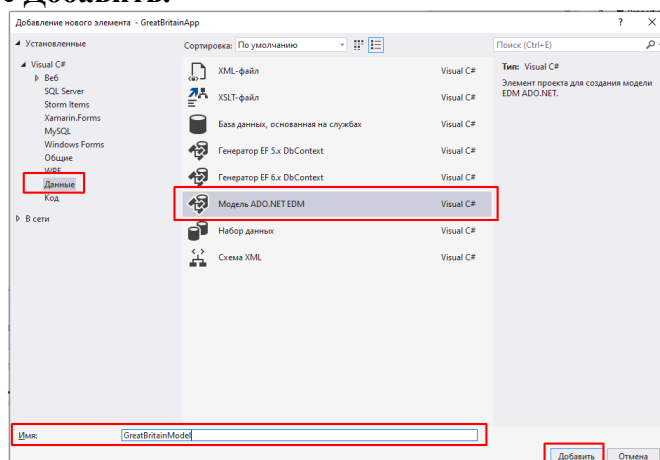
2. Добавьте к проекту папки Models, Pages



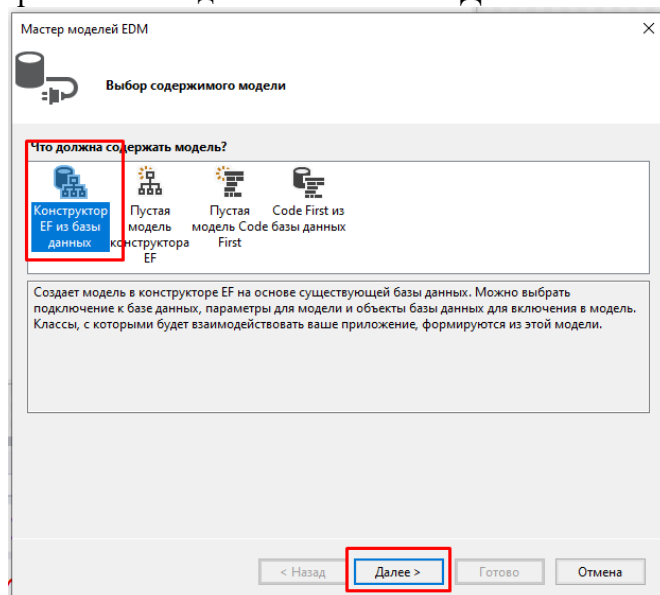
Создание модели БД(Entity FrameWork)

Предварительно у вас должна быть создана БД

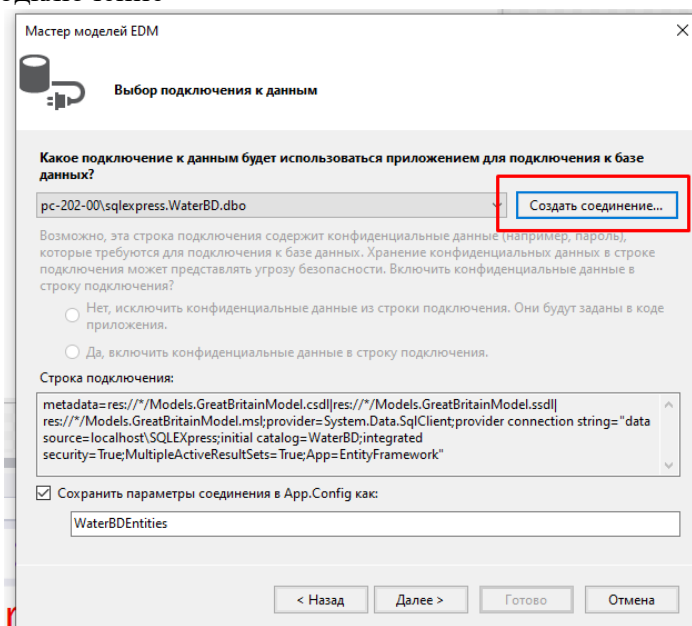
1. Выделите папку Models, щелкните правой кнопкой мыши по папке, выберите Добавить\Создать Элемент.
2. Слева выберите Данные\ Модель ADO.NET EDM. Задайте имя модели, например GreatBritainModel и нажмите **Добавить**.



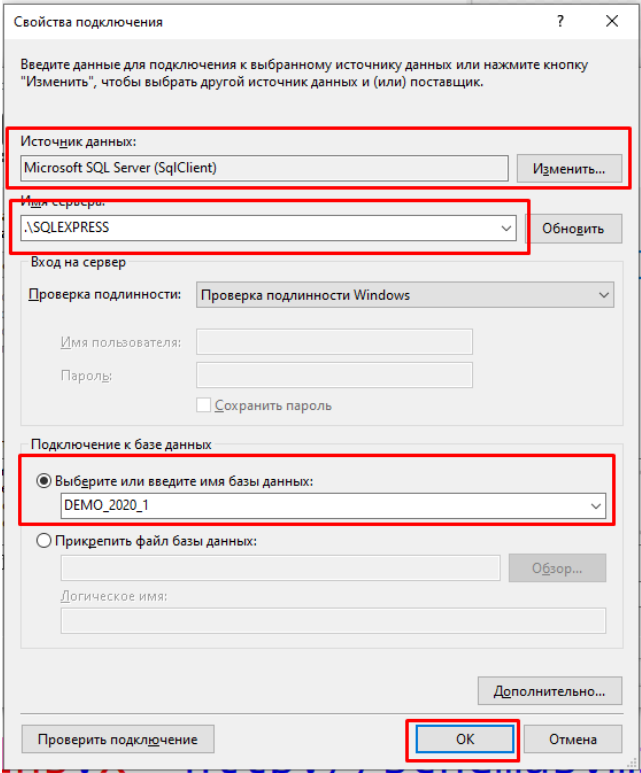
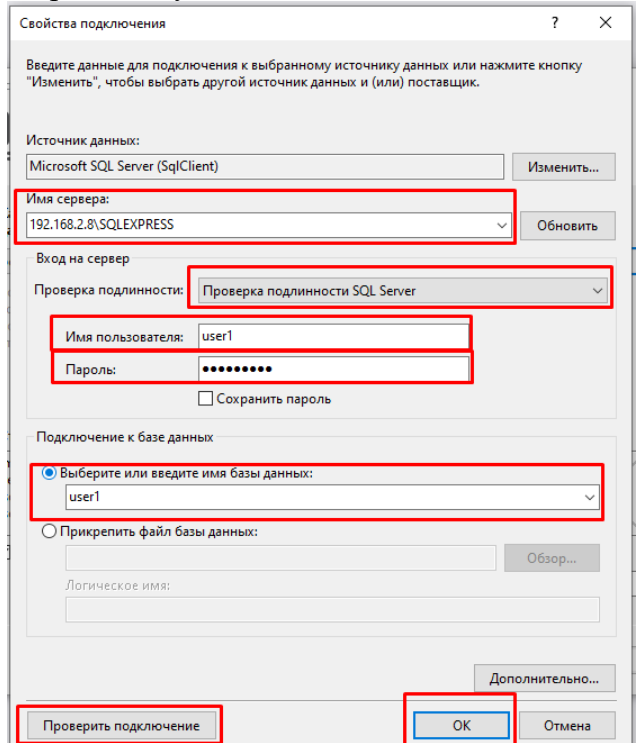
3. Выберите Конструктор EF из базы данных и нажмите Далее.



4. Нажмите Создать подключение



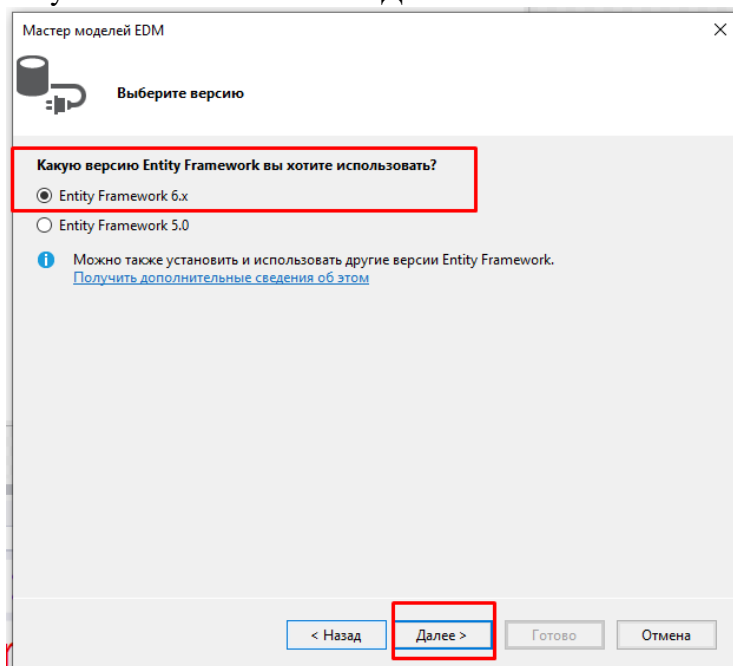
5.

Локальное подключение	Подключение к удаленному ПК
<p>В качестве источника данных укажите Microsoft SQL Server(SqlClient), имя сервера: .\SQLEXPRESS, база данных: DEMO_2020_1(имя вашей бд). Нажмите OK.</p> 	<p>В качестве источника данных укажите Microsoft SQL Server(SqlClient), имя сервера: IP-адрес сервера\SQLEXPRESS, Проверка подлинности SQL Server. Задайте ваши имя пользователя и пароль. Выберите вашу БД</p> 

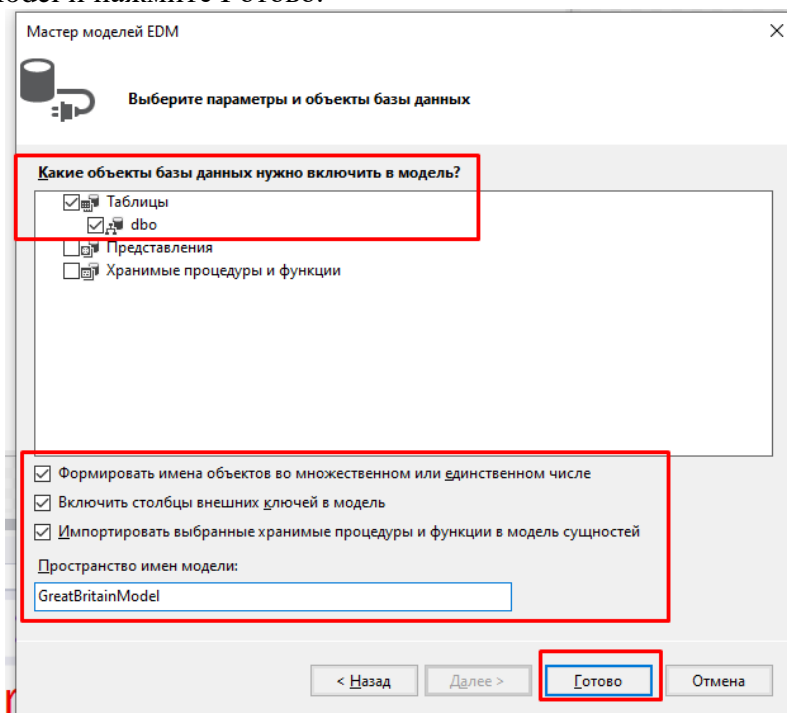
6. Проверьте настройки. Поставьте флажок возле пункта **Сохранить параметры соединения в файл App.config**.

Локальное подключение	Подключение к удаленному ПК
<p>Мастер моделей EDM</p> <p>Выбор подключения к данным</p> <p>Какое подключение к данным будет использоваться приложением для подключения к базе данных?</p> <p>pc-202-00\sql\express.DEMO_2020_1.dbo</p> <p>Возможно, эта строка подключения содержит конфиденциальные данные (например, пароль), которые требуются для подключения к базе данных. Хранение конфиденциальных данных в строке подключения может представлять угрозу безопасности. Включить конфиденциальные данные в строку подключения?</p> <p><input type="radio"/> Нет, исключить конфиденциальные данные из строки подключения. Они будут заданы в коде приложения.</p> <p><input type="radio"/> Да, включить конфиденциальные данные в строку подключения.</p> <p>Строка подключения:</p> <p>metadata=res://*/Models.GreatBritainModel.csdl res://*/Models.GreatBritainModel.ssdl res://*/Models.GreatBritainModel.msl;provider=System.Data.SqlClient;provider connection string="data source=.\SQLEXPRESS;initial catalog=DEMO_2020_1;integrated security=True;MultipleActiveResultSets=True;App=EntityFramework"</p> <p><input checked="" type="checkbox"/> Сохранить параметры соединения в App.Config как:</p> <p>GreatBritainEntities</p> <p>< Назад Далее > Готово Отмена</p>	<p>Поставьте галочку возле пункта Да, включить конфиденциальные данные в строку подключения</p> <p>Мастер моделей EDM</p> <p>Выбор подключения к данным</p> <p>Какое подключение к данным будет использоваться приложением для подключения к базе данных?</p> <p>zmk-r\sql\express.user1.dbo</p> <p>Возможно, эта строка подключения содержит конфиденциальные данные (например, пароль), которые требуются для подключения к базе данных. Хранение конфиденциальных данных в строке подключения может представлять угрозу безопасности. Включить конфиденциальные данные в строку подключения?</p> <p><input type="radio"/> Нет, исключить конфиденциальные данные из строки подключения. Они будут заданы в коде приложения.</p> <p><input checked="" type="radio"/> Да, включить конфиденциальные данные в строку подключения.</p> <p>Строка подключения:</p> <p>metadata=res://*/Models.GreatBritainModel.csdl res://*/Models.GreatBritainModel.ssdl res://*/Models.GreatBritainModel.msl;provider=System.Data.SqlClient;provider connection string="data source=192.168.2.8\SQLEXPRESS;initial catalog=user1;user id=user1;password=*****;MultipleActiveResultSets=True;App=EntityFramework"</p> <p><input checked="" type="checkbox"/> Сохранить параметры соединения в App.Config как:</p> <p>GreatBritainEntities</p> <p>< Назад Далее > Готово Отмена</p>

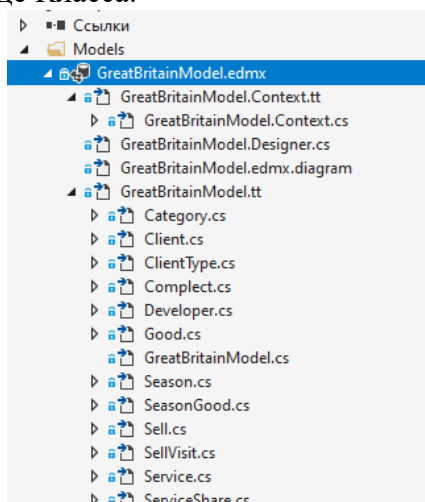
7. Укажите версию Entity Framework и нажмите Далее.



8. Выберите объекты – Таблицы, поставьте везде флажки и переименуйте Пространство имен модели в GreatBritainModel и нажмите Готово.

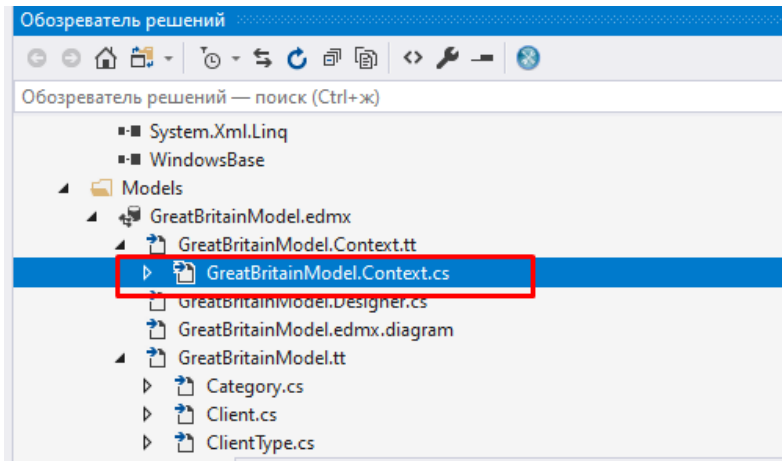


9. Дождитесь завершения импорта базы. В папке Models появится модель данной БД, в которой каждая таблица представляется в виде Класа.



Добавление контекста подключения

1. Откройте файл GreatBritainModel.Context.cs из папки Models.

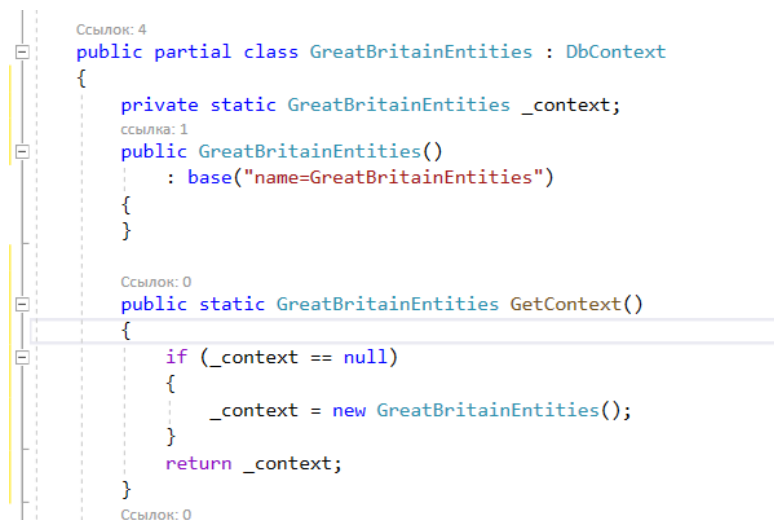


Будем использовать паттерн Singleton (Одиночка). Синглтон позволяет создать объект только при его необходимости. Если объект не нужен, то он не будет создан. В этом отличие синглтона от глобальных переменных.

Добавим внутрь класса **GreatBritainEntities : DbContext** следующий код

```
private static GreatBritainEntities _context;

public static GreatBritainEntities GetContext()
{
    if (_context == null)
    {
        _context = new GreatBritainEntities();
    }
    return _context;
}
```

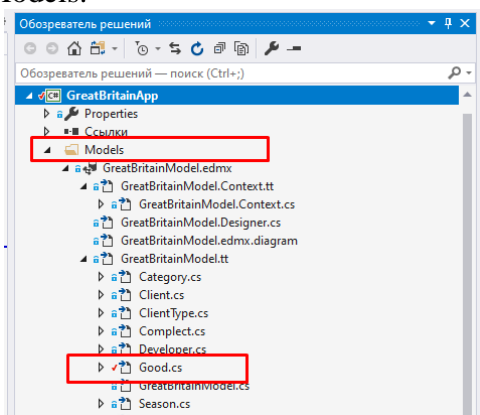


ПРИМЕЧАНИЕ.

Так как модель привязана к базе данных, то иногда может модель обновляться и добавленные строчки в этот файл могут исчезать. Сохраните резервную копию кода

Класс Good

Откройте файл Good.cs из папки Models.



Добавьте внутрь класса следующий программный код.

```
/// <summary>
/// Возвращает абсолютный путь к изображению
/// </summary>
public string GetPhoto
{
    get
    {
        if (MainPhoto is null)
            return null;
        return Directory.GetCurrentDirectory() + @"\Images\" + MainPhoto.Trim();
    }
}
/// <summary>
/// Задает цвет фона товара
/// </summary>
public string GetColor
{
    get
    {
        if (Active)
            return "#fff";
        else
            return "#8C8181";
    }
}
/// <summary>
/// Текстовое представление активности товара
/// </summary>
public string GetStatus
{
    get
    {
        if (Active)
            return "";
        else
            return "нет в наличии";
    }
}
/// <summary>
/// Возвращает количество дополнительных товаров
/// </summary>
```

```

public string GetCount
{
    get
    {
        if (Complects.Count > 0)
            return $"({Complects.Count})";
        else
            return "";
    }
}

```

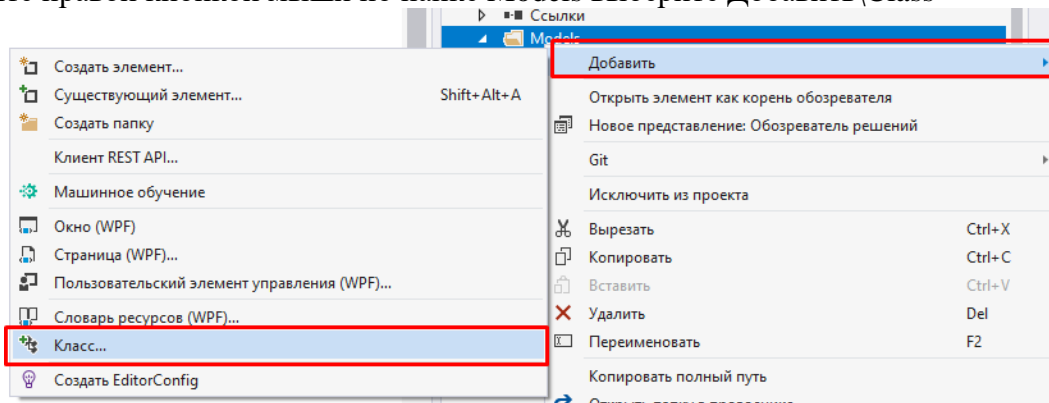
```

public string GetInfo
{
    get
    {
        return $"{GoodName} - {Price.ToString("c")} рублей." ;
    }
}

```

Добавьте класс Manager

1. Нажмите правой кнопкой мыши по папке Models выберите Добавить\Class



2. Дайте ему название **Manager**
3. Откройте файл Manager.cs и измените его код на следующий.

```
using System.Windows.Controls;
```

```

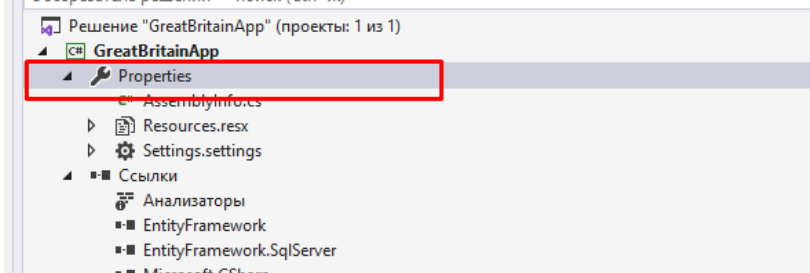
namespace GreatBritainApp.Models
{
    public class Manager
    {
        public static Frame MainFrame { get; set; }
    }
}

```

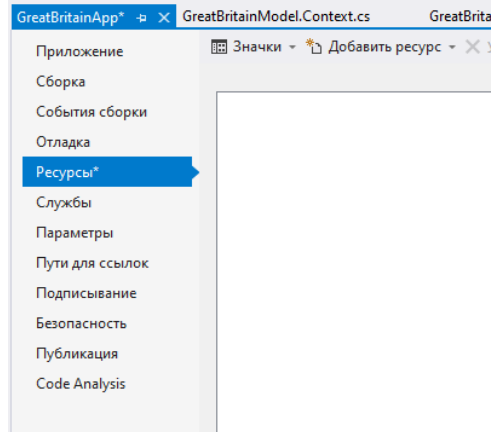

Проектирование каркаса приложения.

1. Добавим ресурсы: иконку и изображение.

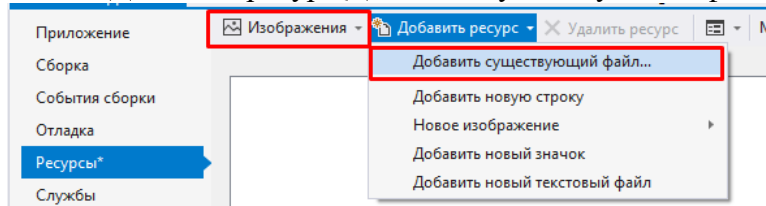
Дважды щелкните левой кнопкой мыши по пункту Properties.



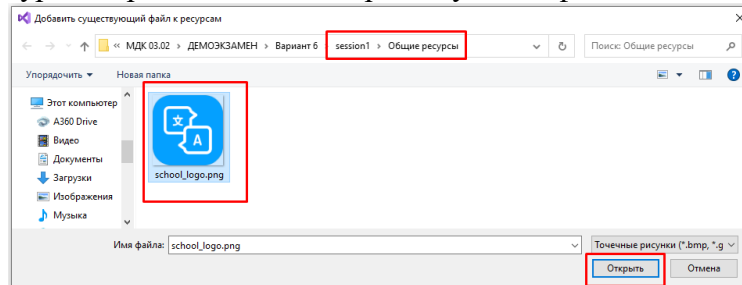
2. Выберите вкладку Ресурсы



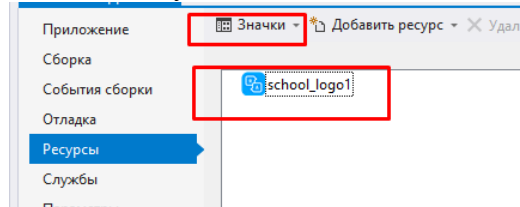
3. Выберите Изображение. Добавить ресурс\Добавить существующий файл.



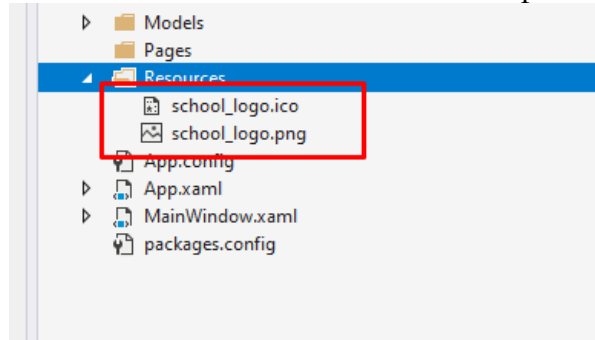
4. В папке Общие ресурсы первой сессии выберите нужный файл и нажмите Открыть.



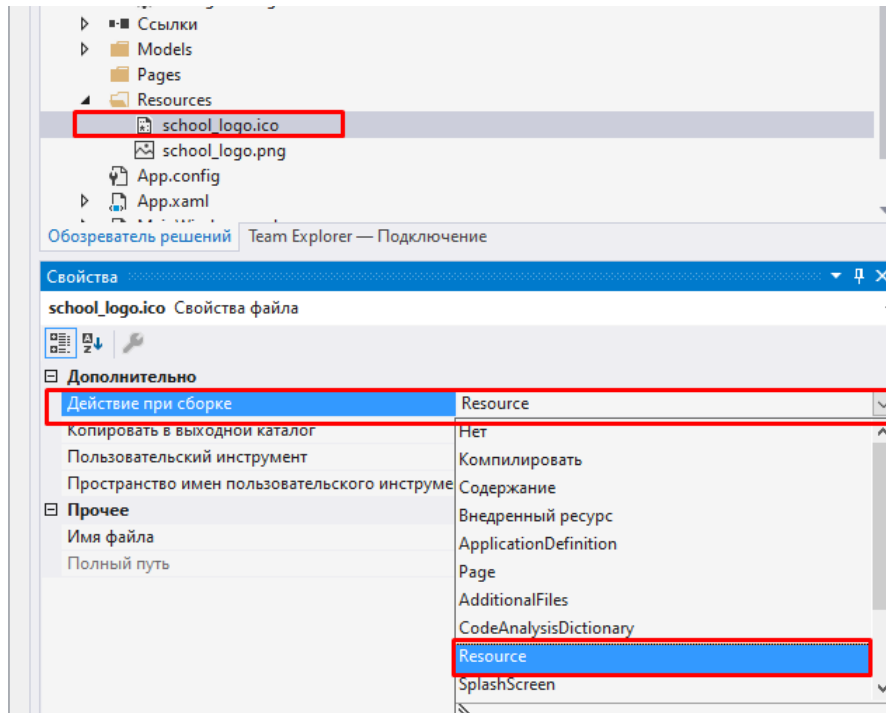
5. Аналогичным образом добавьте иконку



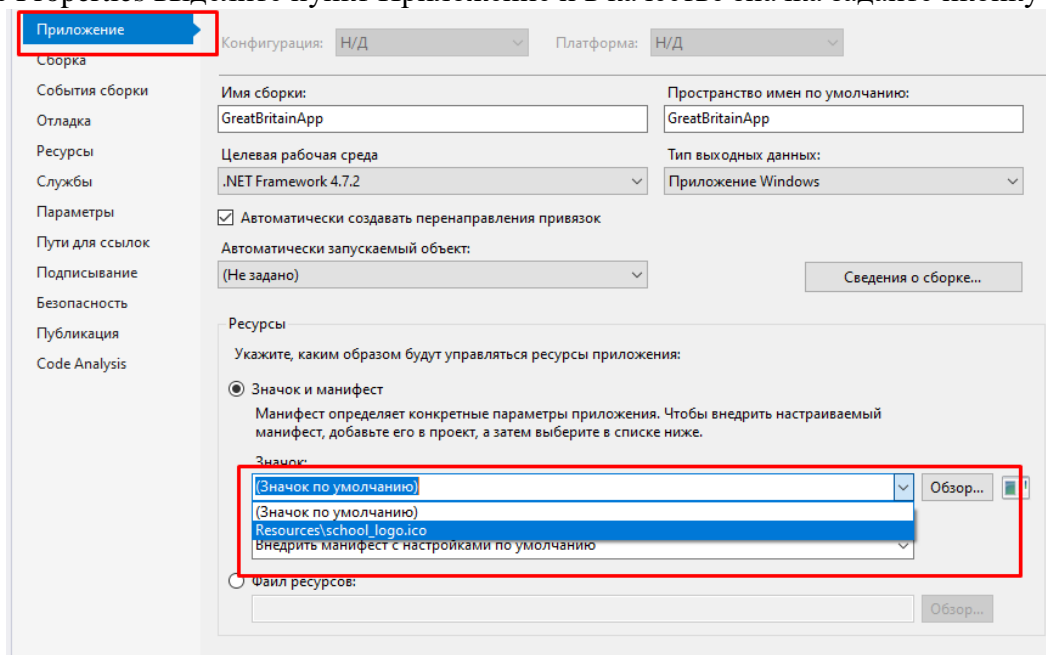
6. В обозревателе решений в папке Resources появятся данные файлы



7. В обозревателе решений выделите каждый из файлов и измените свойство Действие при сборке на **Resource**

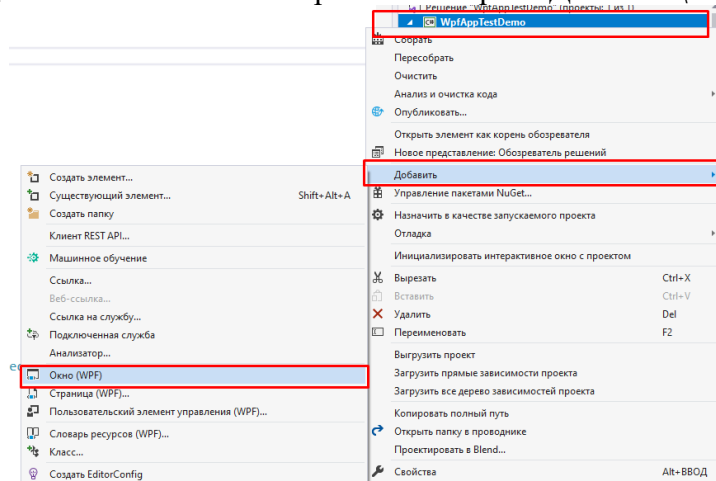


8. В окне Properties выделите пункт Приложение и в качестве значка задайте иконку



Форма авторизации

1. Добавьте к проекту новую форму для авторизации.
2. Правой кнопкой щелкните по названию проекта выберите **Добавить\Окно**

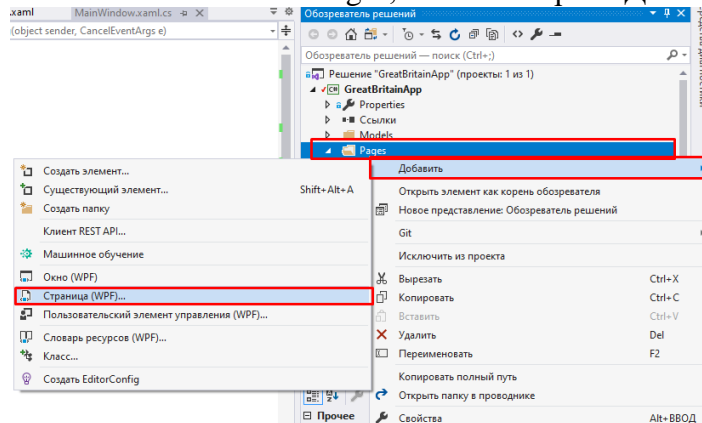


3. Назовите его, например LoginWindow.

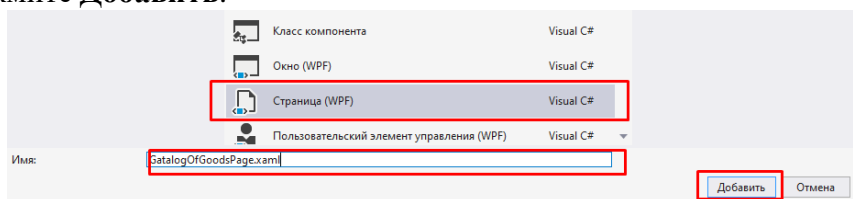
Добавление дополнительных страниц в проект.

Добавим в каталог Pages следующие страницы:

1. Нажмите правой кнопкой мыши по папке Pages, затем выберите **Добавить\Страница WPF**



2. Введите название страницы CatalogOfGoodsPage (эта страница будет отображать список товаров в виде плитки) и нажмите **Добавить**.



Аналогичным образом добавьте следующие страницы:

1. GoodsPage – для просмотра списка товаров, страница для перехода на страницы продаж и добавления или редактирования товара
2. AddGoodPage – страница добавления и редактирования товара
3. SellGoodPage – страница истории продаж товаров
4. AdditionalGoodsPage – страница для работы с дополнительными товарами.

Задание стилей

Откройте файл App.xaml и измените его код на следующий

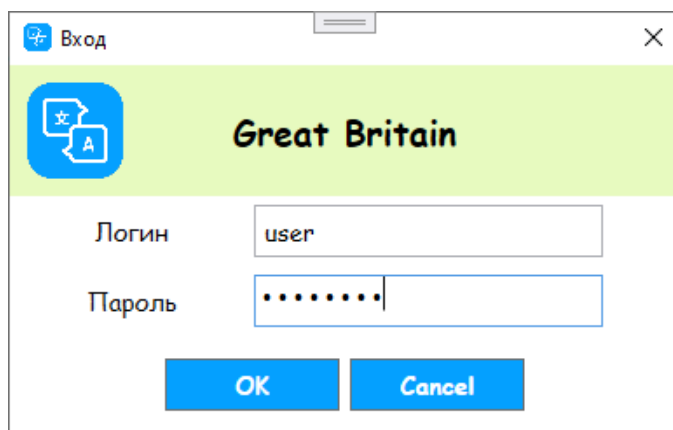
```
<Application x:Class="GreatBritainApp.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="clr-namespace:GreatBritainApp"
    StartupUri="LoginWindow.xaml">
    <!--строчка StartUpUrl отвечает за запускаемое со стартом приложения окно-->
    <!--Первое запускаемое окно будет главным в приложении.-->
    <Application.Resources>
        <!--цвета приложения для соответствия стилю-->
        <SolidColorBrush x:Key="main" Color="#FFF"/>
        <SolidColorBrush x:Key="additional" Color="#FFE7FABF"/>
        <SolidColorBrush x:Key="akcent" Color="#FF04A0FF"/>
        <!--Стиль для окна-->
        <Style TargetType="Window" x:Key="base_window">
            <Setter Property="FontFamily" Value="Comic Sans MS"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="MinHeight" Value="600"/>
            <Setter Property="MinWidth" Value="800"/>
            <Setter Property="Background" Value="{StaticResource main}"/>
        </Style>
        <!--стиль для страницы-->
        <Style TargetType="Page" x:Key="base_page">
            <Setter Property="FontFamily" Value="Comic Sans MS"/>
            <Setter Property="Background" Value="{StaticResource main}"/>
            <Setter Property="FontSize" Value="14"/>
        </Style>
        <!--стиль для полей ввода TextBox-->
        <Style TargetType="TextBox">
            <Setter Property="Height" Value="30"/>
            <Setter Property="VerticalAlignment" Value="Stretch"/>
            <Setter Property="VerticalContentAlignment" Value="Center"/>
            <Setter Property="Padding" Value="3 0"/>
            <Setter Property="Background" Value="{StaticResource main}"/>
        </Style>
        <!--стиль для полей кнопок Button-->
        <Style TargetType="Button">
            <Setter Property="Width" Value="auto"/>
            <Setter Property="Height" Value="auto"/>
            <Setter Property="Background" Value="{StaticResource akcent}"/>
            <Setter Property="Foreground" Value="{StaticResource main}"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="FontWeight" Value="Bold"/>
            <Setter Property="Margin" Value="3"/>
            <Setter Property="Padding" Value="5"/>
            <Setter Property="Height" Value="30"/>
        </Style>
        <!--стиль для меток TextBlock -->
        <Style TargetType="TextBlock" x:Key="base_textblock">
            <Setter Property="Height" Value="30"/>
            <Setter Property="VerticalAlignment" Value="Stretch"/>
            <Setter Property="HorizontalAlignment" Value="Stretch"/>
            <Setter Property="Padding" Value="5"/>
            <Setter Property="Background" Value="{StaticResource additional}"/>
        </Style>
```

```

<!--стиль для меток TextBlock -->
<Style TargetType="TextBlock" x:Key="item_textblock">
    <Setter Property="TextAlignment" Value="Center"/>
    <Setter Property="VerticalAlignment" Value="Top"/>
    <Setter Property="TextWrapping" Value="Wrap"/>
    <Setter Property="FontSize" Value="14"/>
    <Setter Property="Padding" Value="3"/>
</Style>
<!--стиль для меток ComboBoxItem -->
<Style TargetType="ComboBoxItem">
    <Setter Property="Background" Value="{StaticResource main}"/>
    <Setter Property="Height" Value="40"/>
</Style>
<!--стиль для меток ListViewItem -->
<Style TargetType="ListViewItem" x:Key="good_item">
    <Setter Property="Background" Value="{Binding GetColor}" />
    <Setter Property="BorderBrush" Value="{StaticResource akcent}"/>
    <Setter Property="BorderThickness" Value="1"/>
</Style>
</Application.Resources>
</Application>

```

Авторизация - форма LoginWindow



Файл интерфейса LoginWindow.Xaml

```
<Window x:Class="GreatBritainApp.LoginWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:GreatBritainApp"
    mc:Ignorable="d"
    Title="Вход" Style="{StaticResource base_window}" Height="250"
    Width="400" MinHeight="250" MinWidth="400"
    WindowStartupLocation="CenterScreen"
    Closing="WindowClosing"
    ResizeMode="NoResize">
    <!--NoResize - нужен, чтобы нельзя было изменить заданные размеры окна-->
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="75"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="40"/>
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid Background="{StaticResource additional}" />
        <Image Source="Resources/school_logo.png"
            Margin="10"
            HorizontalAlignment="Left" />
        <TextBlock Grid.Row="0" Text="Great Britain"
            HorizontalAlignment="Center"
            VerticalAlignment="Center"
            Background="{StaticResource additional}"
            FontSize="20"
            FontWeight="Bold" />
        <StackPanel Orientation="Horizontal" Grid.Row="1">
            <TextBlock Style="{StaticResource item_textblock}"
                Grid.Row="1"
                Text="Логин"
                Margin="20 0"
                VerticalAlignment="Center" Width="100"/>
```

```

        <TextBox x:Name="TbLogin"
            Grid.Column="1"
            Width="200"/>
    </StackPanel>
    <StackPanel Orientation="Horizontal" Grid.Row="2">
        <TextBlock Style="{StaticResource item_textblock}"
            Grid.Row="2" Grid.Column="0"
            Text="Пароль" Margin="20 0"
            VerticalAlignment="Center" Width="100"/>
        <PasswordBox x:Name="TbPass" Height="30" Width="200" />
    </StackPanel >
    <StackPanel Orientation="Horizontal" Grid.Row="3" HorizontalAlignment="Center">
        <Button Width="100" Height="30"
            Content="OK"
            x:Name="BtnOk"
            Click="BtnOkClick"/>
        <Button Width="100"
            Height="30"
            Content="Cancel"
            x:Name="BtnCancel"
            Click="BtnCancelClick"/>
    </StackPanel>
</Grid>
</Window>

```

Файл программного кода LoginWindow.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
// для работы с моделями вашей БД нужно добавить путь вида
// НазваниеВашегоПриложения.Models
using GreatBritainApp.Models;
using System.Windows;
namespace GreatBritainApp
{
    /// <summary>
    /// Логика взаимодействия для LoginWindow.xaml
    /// </summary>
    public partial class LoginWindow : Window
    {
        public LoginWindow()
        {
            InitializeComponent();
        }

        private void BtnOkClick(object sender, RoutedEventArgs e)
        {
            try
            { //загрузка всех пользователей из БД в список
                List<User> users = GreatBritainEntities.GetContext().Users.ToList();
                //попытка найти пользователя с указанным паролем и логином
                //если такого пользователя не будет обнаружено то переменная u будет равна null
                User u = users.FirstOrDefault(p => p.password == TbPass.Password && p.username ==
TbLogin.Text);
            }
        }
    }
}

```

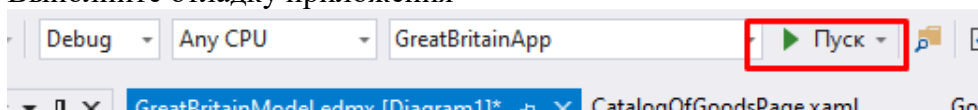
```

if (u != null)
{
    // логин и пароль корректные запускаем главную форму приложения
    MainWindow mainWindow = new MainWindow();
    mainWindow.Owner = this;
    this.Hide();
    mainWindow.Show();
}
else
{
    MessageBox.Show("Не верный логин или пароль");
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
}
//код кнопки Cancel
private void BtnCancelClick(object sender, RoutedEventArgs e)
{
    this.Close();
}
// попытка закрыть приложение
private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)
{
    // на экране отображается форма с двумя кнопками
    MessageBoxResult x = MessageBox.Show("Вы действительно хотите закрыть приложение?",
    "Выйти", MessageBoxButton.OKCancel, MessageBoxImage.Question);
    if (x == MessageBoxResult.Cancel)
        e.Cancel = true;
}
}
}

```

Создание папки Images с изображениями

1. Выполните отладку приложения



2. Перейдите в папку с исполняемым файлом

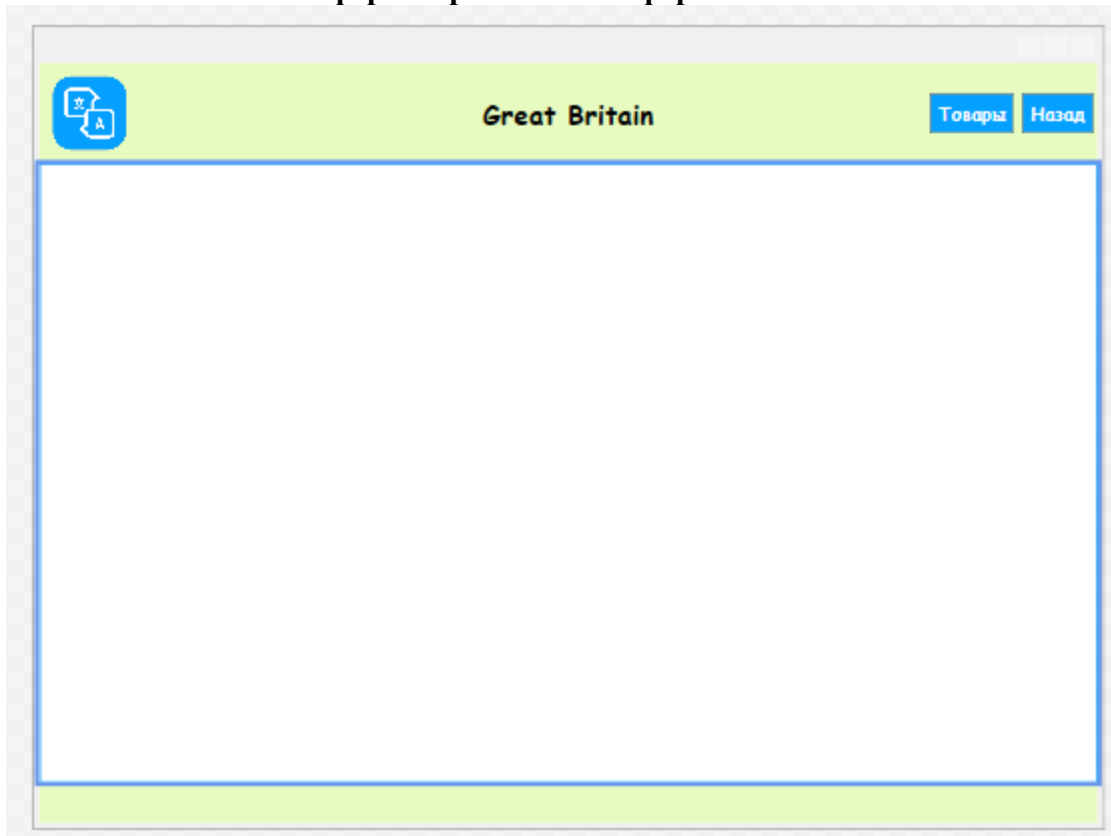
Сессия 2 > GreatBritainApp > bin > Debug >

3. Создайте папку Images

4. Распакуйте архив с изображениями в папку Images. Архив дан в Сессии 1(products_s_import.zip)



Главная форма приложения - форма MainWindow



Файл интерфейса MainWindow.xaml

```
<Window x:Class="GreatBritainApp.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:local="clr-namespace:GreatBritainApp"
    mc:Ignorable="d"
    Title="{ Binding ElementName=MainFrame, Path=Content.Title}"
    Style="{ StaticResource base_window}"
    Height="450" Width="800" Icon="Resources/school_logo.ico"
    Closing="WindowClosing"
    Closed="WindowClosed" >
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="75"/>
        <RowDefinition Height="*/>
        <RowDefinition Height="30"/>
    </Grid.RowDefinitions>
    <Grid Background="{ StaticResource additional }"/>
    <Image Source="Resources/school_logo.png"
        Margin="10"
        HorizontalAlignment="Left"/>
    <TextBlock Grid.Row="0" Text="Great Britain"
        HorizontalAlignment="Center"
        VerticalAlignment="Center"
        Background="{ StaticResource additional}"
        FontSize="20"
        FontWeight="Bold"/>
    <Frame x:Name="MainFrame" Grid.Row="1" BorderBrush="{ StaticResource akcent}"
        NavigationUIVisibility="Hidden"
```

```

        BorderThickness="1" ContentRendered="MainFrameContentRendered"/>
<StackPanel Grid.Row="0" Orientation="Horizontal" HorizontalAlignment="Right" >
    <Button x:Name="BtnEditGoods" Content="Товары" Click="BtnEditGoodsClick"/>
    <Button x:Name="BtnBack" Content="Назад" Click="BtnBackClick"/>
</StackPanel>
<Grid Grid.Row="2" Background="{StaticResource additional}"/>

</Grid>
</Window>

```

Файл программного кода MainWindow.cs

```

using GreatBritainApp.Models;
using GreatBritainApp.Pages;
using System;
using System.Windows;

namespace GreatBritainApp
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
            MainFrame.Navigate(new CatalogOfGoodsPage());
            Manager.MainFrame = MainFrame;
        }

        private void WindowClosed(object sender, EventArgs e)
        {
            // показать владельца формы
            Owner.Show();
            // или если надо, то можно закрыть приложение командой
            // App.Current.Shutdown();
        }
        //событие попытки закрытия окна,
        // если пользователь выберет Cancel, то форму не закроем
        private void WindowClosing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            MessageBoxResult x = MessageBox.Show("Вы действительно хотите выйти?",
            "Выйти", MessageBoxButton.OKCancel, MessageBoxImage.Question);
            if (x == MessageBoxResult.Cancel)
                e.Cancel = true;
        }
        // Кнопка назад
        private void BtnBackClick(object sender, RoutedEventArgs e)
        {
            Manager.MainFrame.GoBack();
        }
        // Кнопка навигации
        private void BtnEditGoodsClick(object sender, RoutedEventArgs e)
        {
            MainFrame.Navigate(new CatalogOfGoodsPage());
        }
    }
}

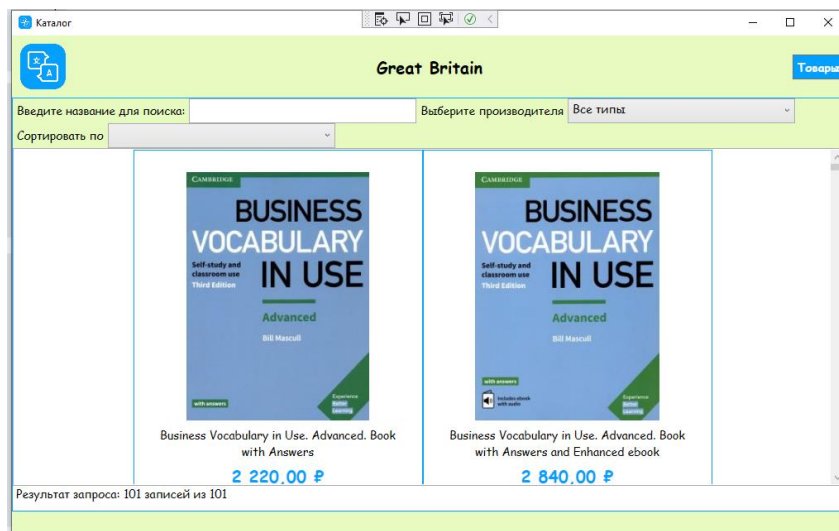
```

```

// Событие отрисовки страницы
// Скрываем или показываем кнопку Назад
// Скрываем или показываем кнопки Для перехода к остальным страницам
private void MainFrameContentRendered(object sender, EventArgs e)
{
    if (MainFrame.CanGoBack)
    {
        BtnBack.Visibility = Visibility.Visible;
        BtnEditGoods.Visibility = Visibility.Collapsed;
    }
    else
    {
        BtnBack.Visibility = Visibility.Collapsed;
        BtnEditGoods.Visibility = Visibility.Visible;
    }
}
}
}
}

```

Страница каталога товаров CatalogOfGoodsPage



Файл интерфейса CatalogOfGoodsPage.xaml

```

<Page x:Class="GreatBritainApp.Pages.CatalogOfGoodsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:GreatBritainApp.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="Каталог"
    IsVisibleChanged="PageIsVisibleChanged"
    Style="{StaticResource base_page}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="*/>
            <RowDefinition Height="30"/>
        </Grid.RowDefinitions>

```

```

<WrapPanel Grid.Row="0"
    HorizontalAlignment="Stretch"
    Background="{StaticResource additional}">
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Введите название для поиска:"
            Style="{StaticResource base_textblock}"/>
        <TextBox Width="275" x:Name="TBoxSearch"
            TextChanged="TBoxSearchTextChanged"/>
    </StackPanel>
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Выберите производителя"
            Style="{StaticResource base_textblock}" />
        <ComboBox Width="275" x:Name="ComboDeveloper"
            SelectionChanged="ComboTypeSelectionChanged"
            DisplayMemberPath="DeveloperName"/>
    </StackPanel>
    <StackPanel Orientation="Horizontal">
        <TextBlock Text="Сортировать по"
            Style="{StaticResource base_textblock}" />
        <ComboBox Width="275"
            x:Name="ComboSort"
            SelectionChanged="ComboSortSelectionChanged" >
            <ComboBoxItem Content="По возрастанию"/>
            <ComboBoxItem Content="По убыванию"/>
        </ComboBox>
    </StackPanel>
</WrapPanel>
<ListView x:Name="LViewGoods" Grid.Row="1"
    ScrollViewer.HorizontalScrollBarVisibility="Disabled"
    ItemContainerStyle="{StaticResource good_item}">
    <ListView.ItemsPanel >
        <ItemsPanelTemplate>
            <WrapPanel Orientation="Horizontal" HorizontalAlignment="Center"/>
        </ItemsPanelTemplate>
    </ListView.ItemsPanel>
    <ListView.ItemTemplate>
        <DataTemplate>
            <Grid Margin="20" Width="300">
                <Grid.RowDefinitions>
                    <RowDefinition Height="310" /> <RowDefinition Height="auto" />
                    <RowDefinition Height="20" />
                </Grid.RowDefinitions>
                <Image Width="240" Grid.Row="0" Stretch="Uniform"
                    HorizontalAlignment="Center" Margin="5"
                    Source="{Binding Path=GetPhoto}"/>
                <StackPanel Grid.Row="1" Height="100">
                    <TextBlock Padding="3" Height="Auto"
                        Style="{StaticResource item_textblock}">
                        <TextBlock.Text>
                            <MultiBinding StringFormat="{0}{1}">
                                <Binding Path="GoodName"/>
                                <Binding Path="GetCount"/>
                            </MultiBinding>
                        </TextBlock.Text>
                    </TextBlock>
                </StackPanel>
            </Grid>
        </DataTemplate>
    </ListView.ItemTemplate>

```

```

        <TextBlock Text="{Binding Price, StringFormat='c',
        ConverterCulture='ru-RU'}" Height="Auto"
        Style="{StaticResource item_textblock}"
        VerticalAlignment="Center"
        Foreground="{StaticResource akcent}"
        FontWeight="Bold" FontSize="20"/>
    </StackPanel>
    <TextBlock Text="{Binding GetStatus}" Height="Auto"
    Style="{StaticResource item_textblock}" Grid.Row="2"/>
</Grid>
</DataTemplate>
</ListView.ItemTemplate>
</ListView>
<TextBlock Grid.Row="2" x:Name="TextBlockCount" />
</Grid>
</Page>

```

Файл программного кода CatalogOfGoodsPage.cs

```

using GreatBritainApp.Models;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GreatBritainApp.Pages
{
    /// <summary>
    /// Логика взаимодействия для CatalogOfGoodsPage.xaml
    /// </summary>
    public partial class CatalogOfGoodsPage : Page
    {
        int _itemcount = 0;
        public CatalogOfGoodsPage()
        {
            InitializeComponent();
            // загрузка данных в combobox + добавление дополнительной строки
            var developers = GreatBritainEntities.GetContext().Developers.OrderBy(p =>
p.DeveloperName).ToList();
            developers.Insert(0, new Developer
            {
                DeveloperName = "Все типы"
            });
            ComboDeveloper.ItemsSource = developers;
            ComboDeveloper.SelectedIndex = 0;
            // загрузка данных в listview сортируем по названию
            LViewGoods.ItemsSource = GreatBritainEntities.GetContext().Goods.OrderBy(p =>
p.GoodName).ToList();
            _itemcount = LViewGoods.Items.Count;
            // отображение количества записей
            TextBlockCount.Text = $"Результат запроса: {_itemcount} записей из {_itemcount}";
        }
    }
}

```

```

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)
{
    //обновление данных после каждой активации окна
    if (Visibility == Visibility.Visible)
    {
        GreatBritainEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());
        LViewGoods.ItemsSource = GreatBritainEntities.GetContext().Goods.OrderBy(p =>
p.GoodName).ToList();
    }
}

// Поиск товаров, которые содержат данную поисковую строку
private void TBoxSearchTextChanged(object sender, TextChangedEventArgs e)
{
    UpdateData();
}

// Поиск товаров конкретного производителя
private void ComboTypeSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateData();
}

/// <summary>
/// Метод для фильтрации и сортировки данных
/// </summary>
private void UpdateData()
{
    // получаем текущие данные из бд
    var currentGoods = GreatBritainEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();
    // выбор только тех товаров, которые принадлежат данному производителю
    if (ComboDeveloper.SelectedIndex > 0)
        currentGoods = currentGoods.Where(p => p.DeveloperId == (ComboDeveloper.SelectedItem as
Developer).DeveloperId).ToList();
    // выбор тех товаров, в названии которых есть поисковая строка
    currentGoods = currentGoods.Where(p =>
p.GoodName.ToLower().Contains(TBoxSearch.Text.ToLower())).ToList();

    // сортировка
    if (ComboSort.SelectedIndex >= 0)
    {
        // сортировка по возрастанию цены
        if (ComboSort.SelectedIndex == 0)
            currentGoods = currentGoods.OrderBy(p => p.Price).ToList();
        // сортировка по убыванию цены
        if (ComboSort.SelectedIndex == 1)
            currentGoods = currentGoods.OrderByDescending(p => p.Price).ToList();
    }
    // В качестве источника данных присваиваем список данных
    LViewGoods.ItemsSource = currentGoods;
    // отображение количества записей
    TextBlockCount.Text = $" Результат запроса: {currentGoods.Count} записей из {_itemcount} ";
}

// сортировка товаров
private void ComboSortSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    UpdateData();
}
}
}

```

Страница товаров GoodsPage

Great Britain										
Главное изоб	Наименование товара	Стоимость	Вес	Длина	Ширина	Высота	Производитель	Есть в наличии	Редактировать	Продажи
13	English for Everyone. English Vocabulary Builder	1 980,00 ₽					Dorling Kindersley	
14	English Grammar in Use. Book with Answers	1 950,00 ₽					Cambridge	
15	English Grammar in Use. Book without Answers	1 760,00 ₽					Cambridge	нет в наличии
16	English Grammar Today Book with Workbook	4 110,00 ₽					Cambridge	нет в наличии
17	English Vocabulary in Use. Advanced. Book with Answers	1 860,00 ₽					Cambridge	нет в наличии
18	English Vocabulary in Use. Elementary. Book with Answers and Enhanced eBook	2 380,00 ₽					Cambridge	
19	English Vocabulary in Use. Upper-Intermediate. Book with Answers	1 890,00 ₽					Cambridge	
20	Everyday Vocabulary + Grammar. For Intermediate Students. Учебное пособие	1 180,00 ₽					Антология	
21	Evolve. Level 1. Student's Book	1 840,00 ₽					Cambridge	нет в наличии
22	Exam Booster For Advanced Without Ans Key + Audio	1 300,00 ₽					Cambridge	
23	Eyes Open Level 1 Student's Book	1 340,00 ₽					Cambridge	
24	Eyes Open. Level 3. Student's Book	1 350,00 ₽					Cambridge	
25	Eyes Open. Level 3. Workbook with Online Practice	1 110,00 ₽					Cambridge	
26	fdgfdg	0,00 ₽	23	33	33	33	Dorling Kindersley	
27	Grammar in Use Intermediate Student's Book with Answers Self-study Reference and Practice	1 950,00 ₽					Cambridge	
28	Grammar in Use. Intermediate. Student's Book with Answers and Interactive eBook	2 380,00 ₽					Cambridge	нет в наличии
29	Grammar in Use. Intermediate. Student's Book without Answers	1 740,00 ₽					Cambridge	нет в наличии
30	Grammarway 1. Book with Answers. Beginner	1 270,00 ₽					Express Publishing	
31	Grammarway 2. Teacher's Book. Elementary	910,00 ₽					Express Publishing	
32	Legal English. Английский язык для юристов. Учебник	2 720,00 ₽					ИНФРА-М	
33	New Enterprise A2 - Grammar Book (with Digibooks App)	1 250,00 ₽					Express Publishing	
34	New Enterprise A2 Student's Book with DigiBooks App	1 650,00 ₽					Express Publishing	
35	New Enterprise A2. Workbook with digibook app	1 220,00 ₽					Express Publishing	
36	OK English! Все правила английского языка с упражнениями	370,00 ₽					АСТ	
37	Pocket English Grammar (Карманная грамматика английского языка). Справочное пособие	390,00 ₽					ИНФРА-М	
38	Prepare. Level 2. A2. Workbook with Audio Download	1 340,00 ₽					Cambridge	
39	Prepare. Level 2. Student's Book	1 780,00 ₽					Cambridge	нет в наличии
40	Prepare. Level 3. A2. Student's Book	1 810,00 ₽					Cambridge	

Файл интерфейса GoodsPage.xaml

```

<Page x:Class="GreatBritainApp.Pages.GoodsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:GreatBritainApp.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="Товары" IsVisibleChanged="PageIsVisibleChanged" Style="{StaticResource base_page}">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*"/>
            <RowDefinition Height="40"/>
        </Grid.RowDefinitions>
        <DataGrid x:Name="DataGridGood" Grid.Row="0"
            AutoGenerateColumns="False"
            IsReadOnly="True"
            RowHeight="30"
            SelectionMode="Single"
            LoadingRow="DataGridGoodLoadingRow">
            <DataGrid.Columns>
                <DataGridTemplateColumn Width="100" Header="Главное изображение">
                    <DataGridTemplateColumn.CellTemplate>
                        <DataTemplate>
                            <Image Source="{Binding Path=GetPhoto}" />
                        </DataTemplate>
                    </DataGridTemplateColumn.CellTemplate>
                </DataGridTemplateColumn>
            </DataGrid.Columns>
        </DataGrid>
    </Grid>

```

```

        <DataGridTextColumn Binding="{Binding GoodName}" Header="Наименование товара"
Width="10*"/>
        <DataGridTextColumn Binding="{Binding Price, StringFormat='c', ConverterCulture='ru-RU'}"
Header="Стоимость" Width="*"/>
        <DataGridTextColumn Binding="{Binding Weight}" Header="Вес" Width="*"/>
        <DataGridTextColumn Binding="{Binding Length}" Header="Длина" Width="*"/>
        <DataGridTextColumn Binding="{Binding Width}" Header="Ширина" Width="*"/>
        <DataGridTextColumn Binding="{Binding Height}" Header="Высота" Width="*"/>
        <DataGridTextColumn Binding="{Binding Developer.DeveloperName}" Header="Производитель"
Width="*"/>
        <DataGridTextColumn Binding="{Binding GetStatus}" Header="Есть в наличии" Width="*"/>
        <DataGridTemplateColumn Width="auto" Header="Редактировать">
            <DataGridTemplateColumn.CellTemplate>
                <DataTemplate>
                    <Button x:Name="BtnEdit" Content="..." Click="ButtonClick" Margin="0"/>
                </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
        </DataGridTemplateColumn>

        <DataGridTemplateColumn Width="auto" Header="Продажи">
            <DataGridTemplateColumn.CellTemplate>
                <DataTemplate>
                    <Button x:Name="BtnSell" Content="..." Click="BtnSellClick" Margin="0"/>
                </DataTemplate>
            </DataGridTemplateColumn.CellTemplate>
        </DataGridTemplateColumn>
    </DataGrid.Columns>
</DataGrid>
<Button x:Name="BtnAdd" Grid.Row="1" Width="100" HorizontalAlignment="Left" Margin="3"
Content="Добавить" Click="BtnAddClick"/>
<Button x:Name="BtnDelete" Grid.Row="1" Width="100" HorizontalAlignment="Right" Margin="3"
Content="Удалить" Click="BtnDeleteClick"/>
</Grid>
</Page>

```

Файл программного кода GoodsPage.cs

```

using GreatBritainApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GreatBritainApp.Pages
{
    /// <summary>
    /// Логика взаимодействия для GoodsPage.xaml
    /// </summary>
    public partial class GoodsPage : Page
    {
        public GoodsPage()
        {
            InitializeComponent();
        }
    }
}

```



```

private void ButtonClick(object sender, RoutedEventArgs e)
{
    // открытие редактирования товара
    // передача выбранного товара в AddGoodPage
    Manager.MainFrame.Navigate(new AddGoodPage((sender as Button).DataContext as Good));
}

private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)
{
    //событие отображения данного Page
    // обновляем данные каждый раз когда активируется этот Page
    if (Visibility == Visibility.Visible)
    {
        DataGridGood.ItemsSource = null;
        //загрузка обновленных данных
        GreatBritainEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload());
        List<Good> goods = GreatBritainEntities.GetContext().Goods.OrderBy(p => p.GoodName).ToList();
        DataGridGood.ItemsSource = goods;
    }
}

private void BtnAddClick(object sender, RoutedEventArgs e)
{
    // открытие AddGoodPage для добавления новой записи
    Manager.MainFrame.Navigate(new AddGoodPage(null));
}

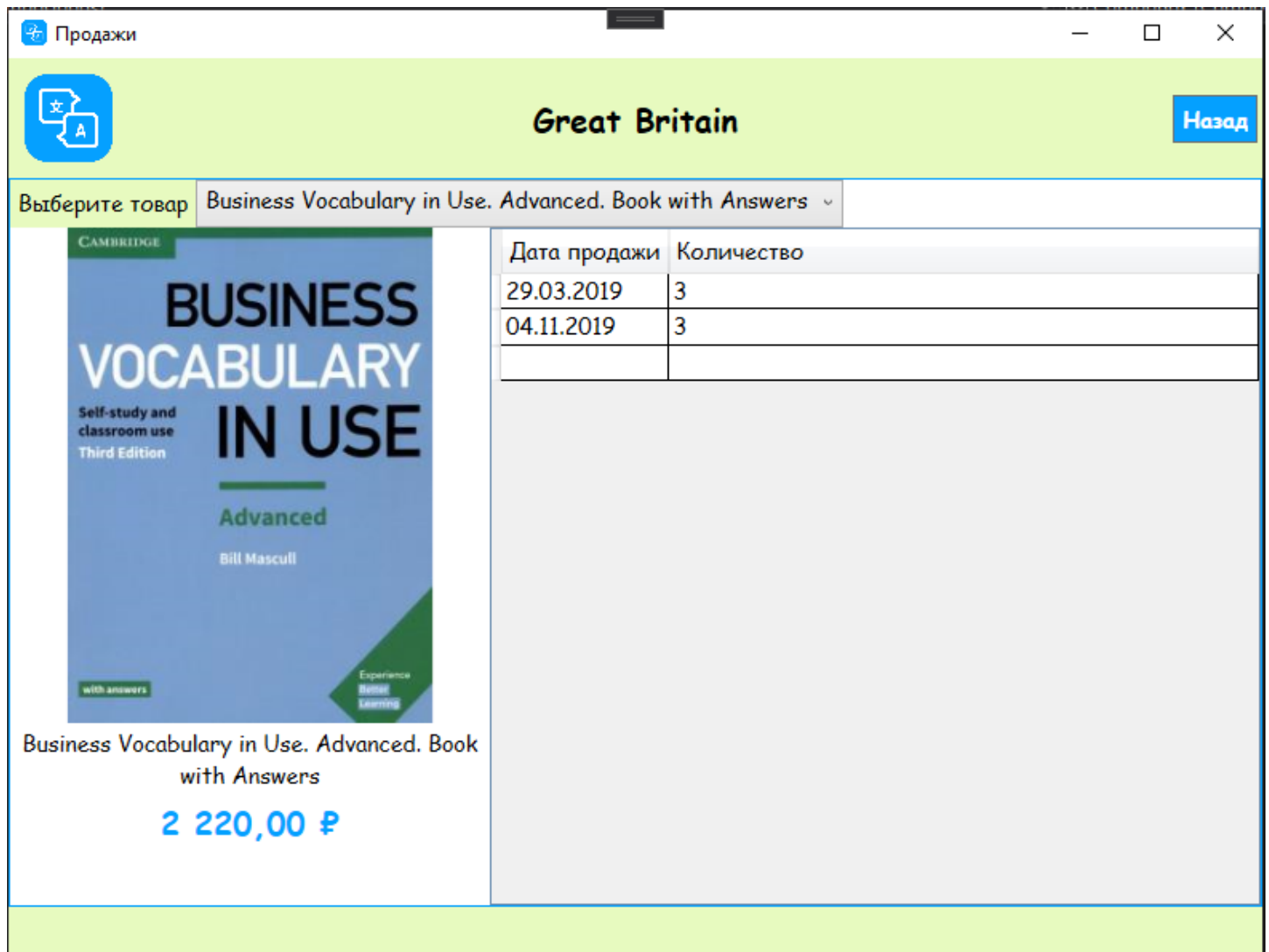
private void BtnDeleteClick(object sender, RoutedEventArgs e)
{
    // удаление выбранного товара из таблицы
    //получаем все выделенные товары
    var selectedGoods = DataGridGood.SelectedItems.Cast<Good>().ToList();
    // вывод сообщения с вопросом Удалить запись?
    MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить {selectedGoods.Count()}
записей???",
        "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);
    //если пользователь нажал ОК пытаемся удалить запись
    if (messageBoxResult == MessageBoxResult.OK)
    {
        try
        {
            // берем из списка удаляемых товаров один элемент
            Good x = selectedGoods[0];
            // проверка, есть ли у товара в таблице о продажах связанные записи
            // если да, то выбрасывается исключение и удаление прерывается
            if (x.Sells.Count > 0)
                throw new Exception("Есть записи в продажах");
            //ищем записи в таблице Complect, с которой связан этот товар
            var complects = GreatBritainEntities.GetContext().Complects.Where(p => p.MainGoodId ==
x.GoodId || p.SecondGoodId == x.GoodId).ToList();
            // удаляем эти записи
            GreatBritainEntities.GetContext().Complects.RemoveRange(complects);
            // удаляем товара
            GreatBritainEntities.GetContext().Goods.Remove(x);
            //сохраняем изменения
            GreatBritainEntities.GetContext().SaveChanges();
        }
    }
}

```

```

        MessageBox.Show("Записи удалены");
        List<Good> goods = GreatBritainEntities.GetContext().Goods.OrderBy(p =>
p.GoodName).ToList();
        DataGridGood.ItemsSource = null;
        DataGridGood.ItemsSource = goods;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message.ToString(), "Ошибка удаления", MessageBoxButton.OK,
MessageBoxImage.Error);
    }
}
}
private void BtnSellClick(object sender, RoutedEventArgs e)
{
    // открытие страницы о продажах SellGoodsPage
    // передача в него выбранного товара
    Manager.MainFrame.Navigate(new SellGoodsPage((sender as Button).DataContext as Good));
}
// отображение номеров строк в DataGrid
private void DataGridGoodLoadingRow(object sender, DataGridRowEventArgs e)
{
    e.Row.Header = (e.Row.GetIndex() + 1).ToString();
}
}
}

```



Файл интерфейса SellGoodsPage.xaml

```

<Page x:Class="GreatBritainApp.Pages.SellGoodsPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:GreatBritainApp.Pages"
    mc:Ignorable="d"
    d:DesignHeight="450" d:DesignWidth="800"
    Title="Продажи" Style="{StaticResource base_page}">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="300"/>
            <ColumnDefinition Width="*"/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>
        <StackPanel Orientation="Horizontal" Grid.ColumnSpan="2">
            <TextBlock Text="Выберите товар" Style="{StaticResource base_textblock}" />

```

```

<ComboBox HorizontalAlignment="Stretch" x:Name="ComboGoods"
    SelectionChanged="ComboGoodsSelectionChanged"
    SelectedValuePath="GoodId"
    DisplayMemberPath="GoodName"/>
</StackPanel>
<Grid x:Name="GridGood" Width="300" Grid.Row="1" Grid.Column="0">
    <Grid.RowDefinitions>
        <RowDefinition Height="310" />
        <RowDefinition Height="auto" />
        <RowDefinition Height="20" />
    </Grid.RowDefinitions>
    <Image Width="240" Grid.Row="0" Stretch="Uniform"
        HorizontalAlignment="Center"
        Source="{Binding Path=GetPhoto}"/>
    <StackPanel Grid.Row="1" Height="100">
        <TextBlock Padding="3" Height="Auto"
            Style="{StaticResource item_textblock}">
            <TextBlock.Text>
                <MultiBinding StringFormat="{0}{1}">
                    <Binding Path="GoodName"/>
                    <Binding Path="GetCount"/>
                </MultiBinding>
            </TextBlock.Text>
        </TextBlock>
        <TextBlock Text="{Binding Price, StringFormat='c',
            ConverterCulture='ru-RU'}" Height="Auto"
            Style="{StaticResource item_textblock}"
            VerticalAlignment="Center"
            Foreground="{StaticResource akcent}"
            FontWeight="Bold" FontSize="20"/>
    </StackPanel>
    <TextBlock Text="{Binding GetStatus}" Height="Auto"
        Style="{StaticResource item_textblock}" Grid.Row="2"/>
</Grid>
<DataGrid x:Name="DataGridSells" AutoGenerateColumns="False" Grid.Row="1" Grid.Column="1">
    <DataGrid.Columns>
        <DataGridTextColumn Header="Дата продажи" Binding="{Binding DateSell, StringFormat='d',
ConverterCulture='ru-RU'}"/>
        <DataGridTextColumn Header="Количество" Binding="{Binding Count}" Width="*"/>
    </DataGrid.Columns>
</DataGrid>
</Grid>
</Page>

```

Файл программного кода SellGoodsPage.cs

```
using GreatBritainApp.Models;
using System;
using System.Linq;
using System.Windows.Controls;

namespace GreatBritainApp.Pages
{
    /// <summary>
    /// Логика взаимодействия для SellGoodsPage.xaml
    /// </summary>
    public partial class SellGoodsPage : Page
    {
        public SellGoodsPage(Good good)
        {
            InitializeComponent();
            LoadData(good);
        }
        // загрузка данных в DataGrid и ComboBox
        void LoadData(Good good)
        {
            DataGridSells.ItemsSource = GreatBritainEntities.GetContext().Sells.Where(p => p.GoodId ==
good.GoodId).OrderBy(p => p.DateSell).ToList();
            ComboGoods.ItemsSource = GreatBritainEntities.GetContext().Goods.OrderBy(p =>
p.GoodName).ToList(); ;
            ComboGoods.SelectedIndex = 0;
            ComboGoods.SelectedValue = good.GoodId;
            GridGood.DataContext = good;
        }
        // фильтрация продаж по товару
        private void ComboGoodsSelectionChanged(object sender, SelectionChangedEventArgs e)
        {
            if (ComboGoods.SelectedIndex > 0)
            {
                int goodId = Convert.ToInt32(ComboGoods.SelectedValue);
                var x = GreatBritainEntities.GetContext().Sells.Where(p => p.GoodId == goodId).OrderBy(p =>
p.DateSell).ToList();
                DataGridSells.ItemsSource = x;
                GridGood.DataContext = ComboGoods.SelectedItem;
            }
        }
    }
}
```

Страница добавления и редактирования товара AddGoodPage

Добавление и редактирование
Great Britain
Назад

Название:
Advanced Grammar in Use. Book without Answers

Описание товара:

Вес	100,00	Длина	1500,00	Ширина	300,00	Высота	40,00
-----	--------	-------	---------	--------	--------	--------	-------


Стоимость:
1 850,00 ₽

Производитель:
Cambridge

В продаже
☒


Главное изображение:

Загрузить



Дополнительные товары

Редактировать



Сохранить

Файл интерфейса AddGoodPage.xaml

```

<Page x:Class="GreatBritainApp.Pages.AddGoodPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:local="clr-namespace:GreatBritainApp.Pages"
    mc:Ignorable="d" d:DesignHeight="450" d:DesignWidth="800"
    Title="Добавление и редактирование" Style="{StaticResource base_page}"
    IsVisibleChanged="PageIsVisibleChanged">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="auto"/>
            <RowDefinition Height="30"/>
            <RowDefinition Height="40"/>
            <RowDefinition Height="*/>
            <RowDefinition Height="100"/>
            <RowDefinition Height="auto"/>
        </Grid.RowDefinitions>
    </Grid>

```

```

<Grid.ColumnDefinitions>
    <ColumnDefinition Width="200"/>
    <ColumnDefinition/>
</Grid.ColumnDefinitions>
<Rectangle Grid.Row="0" Grid.ColumnSpan="2" />
<StackPanel Grid.Row="2" Orientation="Horizontal" Grid.Column="1">
    <TextBlock Text="Вес" VerticalAlignment="Center" Style="{StaticResource item_textblock}"/>
    <TextBox Width="100" x:Name="TextBoxWeight"
        Text="{Binding Weight, StringFormat={}{0:f2}, ConverterCulture='ru-RU'}"/>
    <TextBlock Text="Длина"
        VerticalAlignment="Center"
        Style="{StaticResource item_textblock}"/>
    <TextBox Width="100" x:Name="TextBoxLength"
        Text="{Binding Length, StringFormat={}{0:f2}, ConverterCulture='ru-RU'}"/>
    <TextBlock Text="Ширина" VerticalAlignment="Center" Style="{StaticResource item_textblock}"/>
    <TextBox Width="100" x:Name="TextBoxWidth"
        Text="{Binding Width, StringFormat={}{0:f2}, ConverterCulture='ru-RU'}"/>
    <TextBlock Text="Высота" VerticalAlignment="Center" Style="{StaticResource item_textblock}"/>
    <TextBox Width="100" x:Name="TextBoxHeight"
        Text="{Binding Height, StringFormat={}{0:f2}, ConverterCulture='ru-RU'}"/>
</StackPanel>

<Button x:Name="BtnSave" Grid.Row="8" Content="Сохранить" HorizontalAlignment="Center"
Margin="140,5,340,0" VerticalAlignment="Top" Width="120"
Click="BtnSaveClick" Grid.Column="1"/>

<TextBlock x:Name="TextBlockGoodId" Grid.Row="0" Grid.Column="0" Text="ID:" Margin="20 0"/>
<TextBlock Grid.Row="1" Grid.Column="0" Text="Название:" Margin="20 0"/>
<TextBlock Grid.Row="2" Grid.Column="0" Text="Описание товара:" Margin="20 0"/>
<TextBlock Grid.Row="3" Grid.Column="0" Text="Стоимость:" Margin="20 0"/>
<TextBlock Grid.Row="4" Grid.Column="0" Text="Производитель:" Margin="20 0"/>
<TextBlock Grid.Row="5" Grid.Column="0" Text="В продаже" Margin="20 0"/>
<TextBlock Grid.Row="6" Grid.Column="0" Text="Главное изображение:"
Margin="20 0" Height="30" VerticalAlignment="Top"/>

<TextBlock Grid.Row="7" Grid.Column="0" Text="Дополнительные товары:"
Margin="20,0,8,0" Height="30" VerticalAlignment="Top"/>

<Button x:Name="BtnEditAdditional" Grid.Row="7" Grid.Column="0" Margin="20,30,0,0"
HorizontalAlignment="Left" VerticalAlignment="Top"
Click="BtnEditAdditionalClick" Content="Редактировать"/>

<TextBox x:Name="TextBoxGoodId" Text="{Binding GoodId}" Grid.Row="0" Grid.Column="1"
Padding="0,2" IsReadOnly="True"/>
<TextBox x:Name="TextBoxGoodName" Text="{Binding GoodName}" Grid.Row="1"
Grid.Column="1" Padding="0,2"/>
<ComboBox x:Name="ComboDeveloper" SelectedItem="{Binding Developer}" Grid.Row="4"
Grid.Column="1" DisplayMemberPath="DeveloperName"
SelectedValuePath="DeveloperId" />
<ListView x:Name="ListViewAdditional"
ScrollViewer.VerticalScrollBarVisibility="Hidden"
ScrollViewer.HorizontalScrollBarVisibility="Visible"
Grid.Row="7" Grid.Column="1"
PreviewMouseLeftButtonUp="ListViewAdditionalPreviewMouseLeftButtonUp" >

```



```

<ListView.ItemsPanel >
    <ItemsPanelTemplate>
        <WrapPanel Orientation="Horizontal" HorizontalAlignment="Center"/>
    </ItemsPanelTemplate>
</ListView.ItemsPanel>
<ListView.ItemTemplate >
    <DataTemplate>
        <Image Width="30" Grid.Row="0" Stretch="Uniform" ToolTip="{Binding GetInfo}"
            HorizontalAlignment="Center" Margin="5"
            Source="{Binding Path=GetPhoto}" />
    </DataTemplate>
</ListView.ItemTemplate>
</ListView>

<Image x:Name="ImagePhoto" Source="{Binding GetPhoto}" Grid.Row="6" Grid.Column="1"
    VerticalAlignment="Stretch" HorizontalAlignment="Stretch"/>
<Button x:Name="BtnLoad" Grid.Row="6" Grid.Column="0" Margin="20,30,0,0"
    HorizontalAlignment="Left" VerticalAlignment="Top"
    Click="BtnLoadClick" Content="Загрузить"/>
<TextBox x:Name="TextBoxPrice" Text="{Binding Price, StringFormat='c', ConverterCulture='ru-RU'}"
    Grid.Row="3" Grid.Column="1" Padding="0,2"/>
<CheckBox x:Name="CheckBoxActive" Grid.Row="5" Grid.Column="1"
    VerticalAlignment="Center" Margin="20,12,0,12" IsChecked="{Binding Active}"/>
</Grid>
</Page>

```

Файл программного кода AddGoodPage.cs

```

using GreatBritainApp.Models;
using Microsoft.Win32;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media.Imaging;

namespace GreatBritainApp.Pages
{
    /// <summary>
    /// Логика взаимодействия для AddGoodPage.xaml
    /// </summary>
    public partial class AddGoodPage : Page
    {
        //текущий товар
        private Good _currentGood = new Good();
        // путь к файлу
        private string _filePath = null;
        // название текущей главной фотографии
        private string _photoName = null;
        // текущая папка приложения
        private static string _currentDirectory = Directory.GetCurrentDirectory() + @"Images\";
    }
}

```



```

// передача в AddGoodPage товара
public AddGoodPage(Good selectedGood)
{
    InitializeComponent();
    // если передано null, то мы добавляем новый товар
    if (selectedGood != null)
    {
        _currentGood = selectedGood;
        TextBoxGoodId.Visibility = Visibility.Hidden;
        TextBlockGoodId.Visibility = Visibility.Hidden;
        int x = selectedGood.GoodId;
        // загрузка комплементарных товаров
        List<Complect> additional = GreatBritainEntities.GetContext().Complects.Where(p =>
p.MainGoodId == selectedGood.GoodId).ToList();
        List<Good> goods = new List<Good>();
        foreach (Complect item in additional)
        {
            goods.Add(item.Good1);
        }
        ListViewAdditional.ItemsSource = goods;
        _filePath = _currentDirectory + _currentGood.MainPhoto;
    }
    DataContext = _currentGood;
    _photoName = _currentGood.MainPhoto;
    //загрузка производителей
    ComboDeveloper.ItemsSource = GreatBritainEntities.GetContext().Developers.ToList();
}
// проверка полей
private StringBuilder CheckFields()
{
    StringBuilder s = new StringBuilder();
    if (string.IsNullOrEmpty(_currentGood.GoodName))
        s.AppendLine("Поле название пустое");
    if (_currentGood.Developer == null)
        s.AppendLine("Выберите производителя");
    if (_currentGood.Price < 0)
        s.AppendLine("Цена не может быть отрицательной");
    if (!string.IsNullOrEmpty(TextBoxHeight.Text))
    {
        double x = 0;
        if (!double.TryParse(TextBoxHeight.Text, out x))
            s.AppendLine("Высота только число");
        else if (x < 0)
        {
            s.AppendLine("Высота не может быть отрицательной");
        }
    }
    if (!string.IsNullOrEmpty(TextBoxLength.Text))
    {
        double x = 0;
        if (!double.TryParse(TextBoxLength.Text, out x))
            s.AppendLine("Длина только число");
        else if (x < 0)
        {
            s.AppendLine("Длина не может быть отрицательной");
        }
    }
}

```

```

if (!string.IsNullOrEmpty(TextBoxWidth.Text))
{
    double x = 0;
    if (!double.TryParse(TextBoxWidth.Text, out x))
        s.AppendLine("Ширина только число");
    else if (x < 0)
    {
        s.AppendLine("Ширина не может быть отрицательной");
    }
}
if (!string.IsNullOrEmpty(TextBoxWeight.Text))
{
    double x = 0;
    if (!double.TryParse(TextBoxWeight.Text, out x))
        s.AppendLine("Вес только число");
    else if (x < 0)
    {
        s.AppendLine("Вес не может быть отрицательным");
    }
}
if (string.IsNullOrEmpty(_photoName))
    s.AppendLine("фото не выбрано пустое");
return s;
}
// сохранение
private void BtnSaveClick(object sender, RoutedEventArgs e)
{
    StringBuilder _error = CheckFields();
    // если ошибки есть, то выводим ошибки в MessageBox
    // и прерываем выполнение
    if (_error.Length > 0)
    {
        MessageBox.Show(_error.ToString());
        return;
    }
    // проверка полей прошла успешно
    if (_currentGood.GoodId == 0)
    {
        // добавление нового товара
        // формируем новое название файла картинки,
        // так как в папке может быть файл с тем же именем
        string photo = ChangePhotoName();
        // путь куда нужно скопировать файл
        string dest = _currentDirectory + photo;
        File.Copy(_filePath, dest);
        _currentGood.MainPhoto = photo;
        // добавляем товар в БД
        GreatBritainEntities.GetContext().Goods.Add(_currentGood);
    }
}

```

```

try
{
    if (_filePath != null)
    {
        string photo = ChangePhotoName();
        string dest = _currentDirectory + photo;
        File.Copy(_filePath, dest);
        _currentGood.MainPhoto = photo;
    }
    // Сохраняем изменения в БД
    GreatBritainEntities.GetContext().SaveChanges();
    MessageBox.Show("Запись Изменена");
    // Возвращаемся на предыдущую форму
    Manager.MainFrame.GoBack();
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message.ToString());
}
}
// открытие этой же страницы
// в качестве параметра передаем выделенный товар в комплементарных товарах
private void ListViewAdditionalPreviewMouseLeftButtonUp(object sender, MouseButtonEventArgs e)
{
    if (ListViewAdditional.SelectedItems.Count > 0)
    {
        Good good = (sender as ListView).SelectedItem as Good;
        Manager.MainFrame.Navigate(new AddGoodPage(good));
    }
}
// открытие окна редактирования комплементарных товаров
private void BtnEditAdditionalClick(object sender, RoutedEventArgs e)
{
    if (_currentGood.GoodId != 0)
    {
        Manager.MainFrame.Navigate(new AdditionalGoodsPage(_currentGood));
    }
}
// Событие визуализации страницы
// после визуализации окна данные в listView подгружаются снова
private void PageIsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e)
{
    // загрузка комплементарных товаров
    List<Complect> additional = GreatBritainEntities.GetContext().Complects.Where(p => p.MainGoodId
== _currentGood.GoodId).ToList();
    List<Good> goods = new List<Good>();
    foreach (Complect item in additional)
    {
        goods.Add(item.Good1);
    }
    ListViewAdditional.ItemsSource = goods;
}

```

```

// загрузка фото
private void BtnLoadClick(object sender, RoutedEventArgs e)
{
    try
    {
        //Диалог открытия файла
        OpenFileDialog op = new OpenFileDialog();
        op.Title = "Select a picture";
        op.Filter = "JPEG Files (*.jpeg)|*.jpeg|PNG Files (*.png)|*.png|JPG Files (*.jpg)|*.jpg|GIF Files (*.gif)|*.gif";
        // диалог вернет true, если файл был открыт
        if (op.ShowDialog() == true)
        {
            // проверка размера файла
            // по условию файл должен быть не более 2Мб.
            FileInfo fileInfo = new FileInfo(op.FileName);
            if (fileInfo.Length > (1024 * 1024 * 2))
            {
                // размер файла меньше 2Мб. Поэтому выбрасывается новое исключение
                throw new Exception("Размер файла должен быть меньше 2Мб");
            }
            ImagePhoto.Source = new BitmapImage(new Uri(op.FileName));
            _photoName = op.SafeFileName;
            _filePath = op.FileName;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
        _filePath = null;
    }
}

//подбор имени файла
string ChangePhotoName()
{
    string x = _currentDirectory + _photoName;
    string photoname = _photoName;
    int i = 0;
    if (File.Exists(x))
    {
        while (File.Exists(x))
        {
            i++;
            x = _currentDirectory + i.ToString() + photoname;
        }
        photoname = i.ToString() + photoname;
    }
    return photoname;
}
}
}

```

Страница дополнительных товаров AddGoodPage



Файл интерфейса AdditionalGoodsPage.xaml

```
<Page x:Class="GreatBritainApp.Pages.AdditionalGoodsPage"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:GreatBritainApp.Pages"
      mc:Ignorable="d"
      d:DesignHeight="450" d:DesignWidth="800"
      Title="Дополнительные товары" Style="{StaticResource base_page}">

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="*" />
            <ColumnDefinition Width="*" />
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition Height="auto" />
            <RowDefinition Height="auto" />
            <RowDefinition Height="*" />
        </Grid.RowDefinitions>
        <StackPanel Orientation="Horizontal" Grid.ColumnSpan="2">
            <TextBlock Text="Выберите товар" Style="{StaticResource base_textblock}" />
            <ComboBox HorizontalAlignment="Stretch" x:Name="ComboGoods"
                      SelectionChanged="ComboGoodsSelectionChanged"
                      SelectedValuePath="GoodId"
                      DisplayMemberPath="GoodName" />
        </StackPanel>
    </Grid>
```

```

<TextBlock Text="Дополнительные товары" Grid.Row="1" Grid.Column="0"/>
<ListBox x:Name="ListBoxAdditional" ScrollViewer.VerticalScrollBarVisibility="Visible"
    HorizontalAlignment="Stretch" SelectedValuePath="GoodId"
    Grid.Row="2" Grid.Column="0">
<ListBox.ItemTemplate>
<DataTemplate>
    <StackPanel Margin="5" Orientation="Horizontal">
<Button x:Name="BtnDelete" Content="-" Width="20"
    HorizontalAlignment="Right" Click="BtnDeleteClick"/>
<Image Width="60" Height="60" Source="{Binding Path=GetPhoto}" />
<StackPanel>
    <TextBlock FontSize="14"
        Text="{Binding Path=GoodName,
            StringFormat={ } Товар: {0} }"
        Width="350" HorizontalAlignment="Left" />
    <TextBlock FontSize="14" Text="{Binding Path=Price,
        StringFormat={ } {0:f2} руб."
        Width="300" HorizontalAlignment="Left" />
    </StackPanel>
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>

<TextBlock Text="Все товары" Grid.Row="1" Grid.Column="1"/>
<ListBox x:Name="ListBoxAllGoods"
    ScrollViewer.VerticalScrollBarVisibility="Visible"
    HorizontalAlignment="Stretch" SelectedValuePath="GoodId"
    Grid.Row="2" Grid.Column="1" >
<ListBox.ItemTemplate>
<DataTemplate>
    <StackPanel Margin="5" Orientation="Horizontal">
<Button x:Name="BtnAdd" Content="+" Width="20"
    HorizontalAlignment="Right" Click="BtnAddClick"/>
<Image Width="60" Height="60" Source="{Binding Path=GetPhoto}" />
<StackPanel>
    <TextBlock FontSize="14"
        Text="{Binding Path=GoodName,
            StringFormat={ } Товар: {0} }"
        Width="350" HorizontalAlignment="Left" />
    <TextBlock FontSize="14" Text="{Binding Path=Price,
        StringFormat={ } {0:f2} руб."
        Width="300" HorizontalAlignment="Left" />
    </StackPanel>
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>
</Grid>
</Page>

```

Файл программного кода AdditionalGoodsPage.cs

```
using GreatBritainApp.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows;
using System.Windows.Controls;

namespace GreatBritainApp.Pages
{
    /// <summary>
    /// Логика взаимодействия для AdditionalGoodsPage.xaml
    /// </summary>
    public partial class AdditionalGoodsPage : Page
    {
        // текущий выбранный товар
        Good _currentGood = null;
        public AdditionalGoodsPage(Good good)
        {
            InitializeComponent();
            _currentGood = good;
            LoadGoodsIntoCombo(good);
            LoadData(good);
        }

        //Загрузка в Combo товаров
        void LoadGoodsIntoCombo(Good good)
        {
            ComboGoods.ItemsSource = GreatBritainEntities.GetContext().Goods.OrderBy(p =>
p.GoodName).ToList();
            ComboGoods.SelectedIndex = 0;
            ComboGoods.SelectedValue = good.GoodId;
        }

        // загрузка данных в два ListBox
        void LoadData(Good good)
        {
            List<Complect> additional = GreatBritainEntities.GetContext().Complects.Where(p => p.MainGoodId
== good.GoodId).ToList();
            List<Good> goods = new List<Good>();
            List<Good> allGoods = GreatBritainEntities.GetContext().Goods.Where(p=> p.Active ==
true).ToList();
            foreach (Complect item in additional)
            {
                goods.Add(item.Good1);
                allGoods.Remove(item.Good1);
            }
            allGoods.Remove(good);
            ListBoxAdditional.ItemsSource = goods;
            ListBoxAllGoods.ItemsSource = allGoods;
        }
    }
}
```

```

// фильтрация дополнительных товаров товаров
private void ComboGoodsSelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (ComboGoods.SelectedIndex >= 0)
    {
        int goodId = Convert.ToInt32(ComboGoods.SelectedValue);
        var x = GreatBritainEntities.GetContext().Goods.FirstOrDefault(p => p.GoodId == goodId);
        _currentGood = x;
        LoadData(x);
    }
}

//Добавление комплементарного товара
private void BtnAddClick(object sender, RoutedEventArgs e)
{
    var g = (sender as Button).DataContext as Good;
    Complect complect = new Complect();
    complect.MainGoodId = _currentGood.GoodId;
    complect.SecondGoodId = g.GoodId;
    // вывод сообщения с вопросом Добавить запись?
    MessageBoxResult messageBoxResult = MessageBox.Show($"Добавить запись???",
        "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);
    //если пользователь нажал ОК пытаемся добавить запись
    if (messageBoxResult == MessageBoxResult.OK)
    {
        try
        {
            //добавляем комплементарный товар
            GreatBritainEntities.GetContext().Complects.Add(complect);
            //сохраняем изменения
            GreatBritainEntities.GetContext().SaveChanges();
            MessageBox.Show("Запись добавлена");
            //обновляем данные в ListBox
            LoadData(_currentGood);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Ошибка добавления", MessageBoxButton.OK,
                MessageBoxImage.Error);
        }
    }
}

```



```

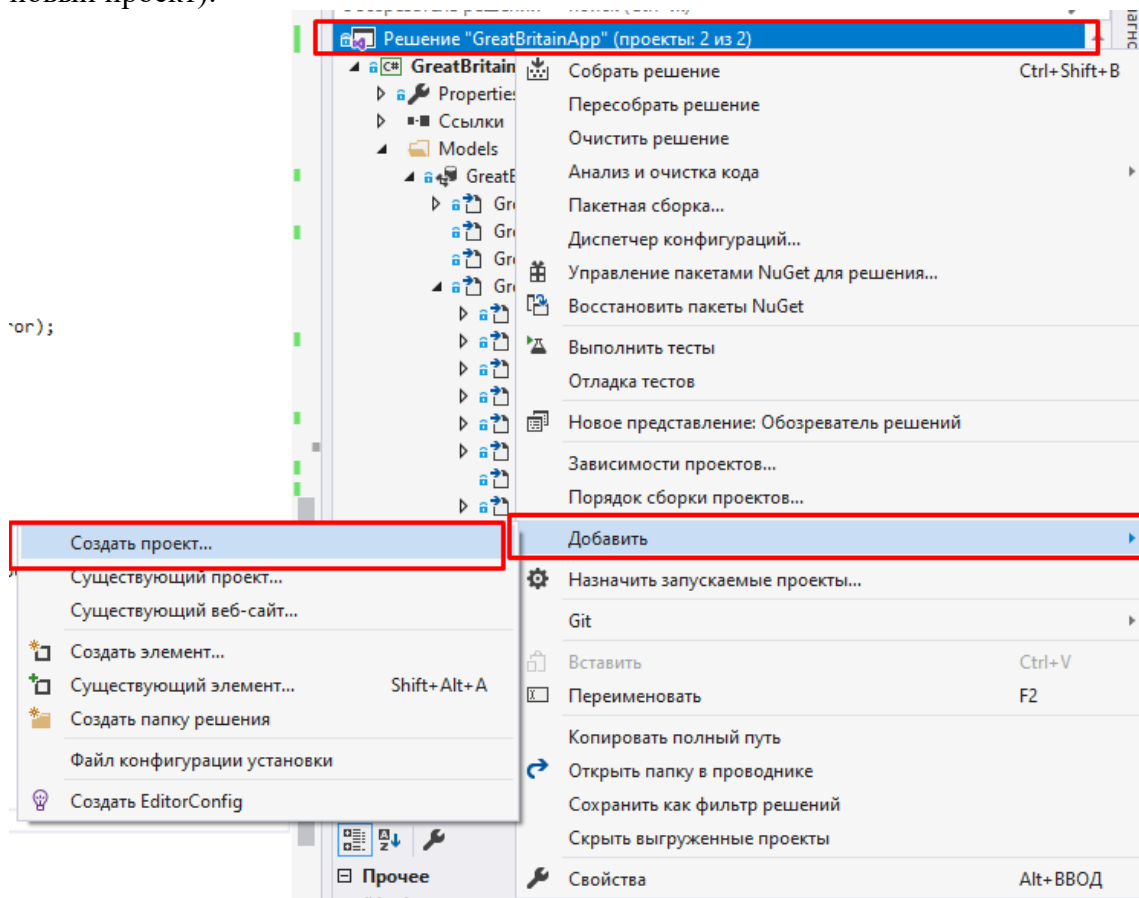
//удаление комплементарного товара
private void BtnDeleteClick(object sender, RoutedEventArgs e)
{
    var g = (sender as Button).DataContext as Good;
    var deletedItem = GreatBritainEntities.GetContext().Complects.FirstOrDefault(p => p.MainGoodId ==
_currentGood.GoodId && p.SecondGoodId == g.GoodId);
    // вывод сообщения с вопросом Удалить запись?
    MessageBoxResult messageBoxResult = MessageBox.Show($"Удалить запись???",
        "Удаление", MessageBoxButton.OKCancel, MessageBoxImage.Question);
    //если пользователь нажал ОК пытаемся удалить запись
    if (messageBoxResult == MessageBoxResult.OK)
    {
        try
        {
            // удаляем запись о комплементарном товаре
            GreatBritainEntities.GetContext().Complects.Remove(deletedItem);
            //сохраняем изменения
            GreatBritainEntities.GetContext().SaveChanges();
            MessageBox.Show("Запись удалена");
            //обновляем данные в ListBox
            LoadData(_currentGood);
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message.ToString(), "Ошибка удаления", MessageBoxButton.OK,
            MessageBoxImage.Error);
        }
    }
}
}
}
}

```

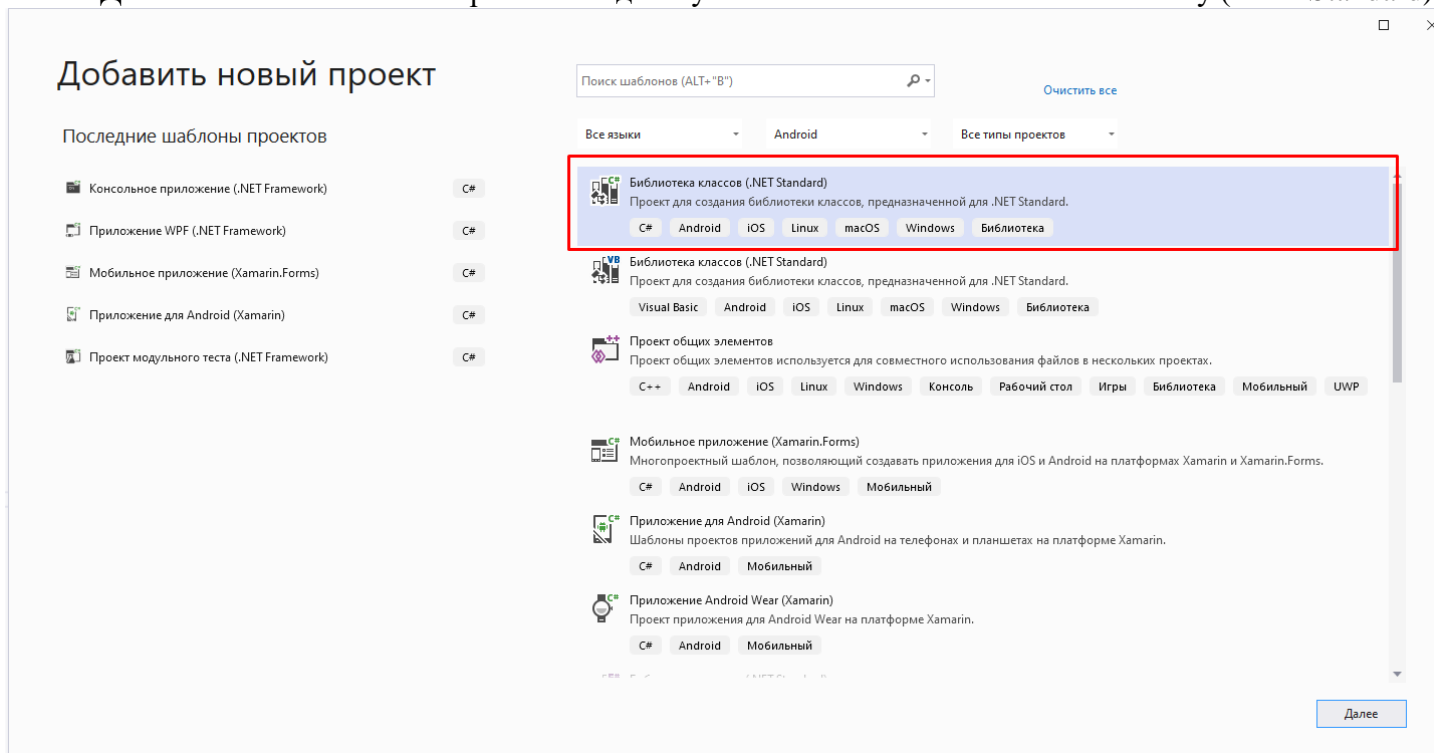
Библиотека классов(dll)

Создание библиотеки

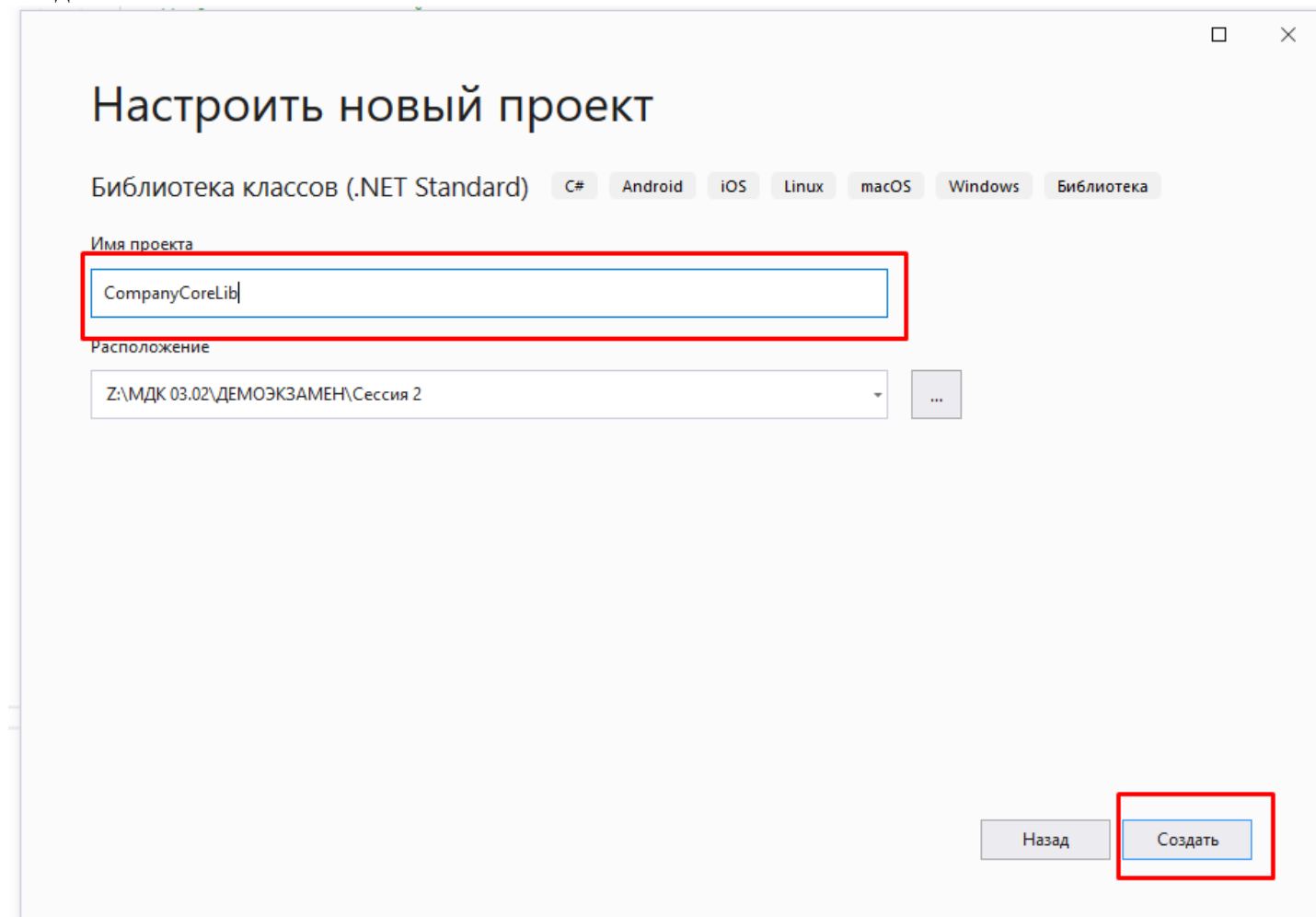
Возьмем имеющийся проект WPF приложения. В структуре проекта нажмем правой кнопкой на название решения и далее в появившемся контекстном меню выберем Add -> New Project... (Добавить новый проект):



Далее в списке шаблонов проекта найдем пункт Библиотека классов Class Library (.NET Standard):

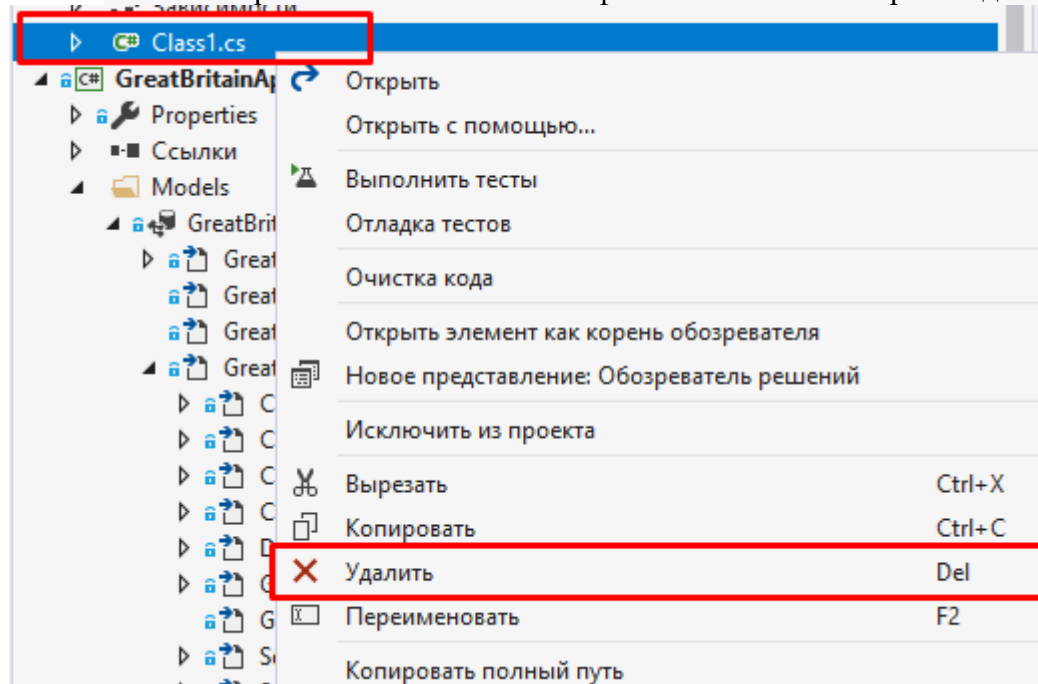


Дадим название проекта CompanyCoreLib. Название дано в файле спецификации метода. Нажмите Создать.

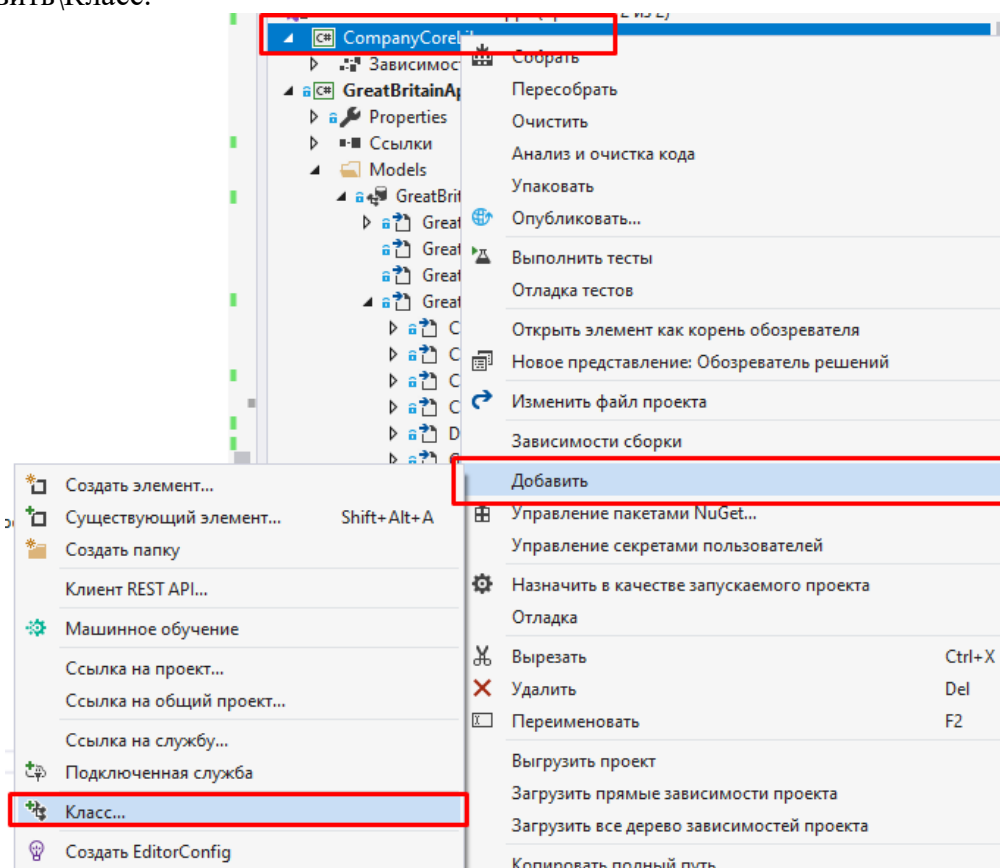


По умолчанию новый проект имеет один пустой класс Class1 в файле Class1.cs. Мы можем этот файл удалить или переименовать, как нам больше нравится.

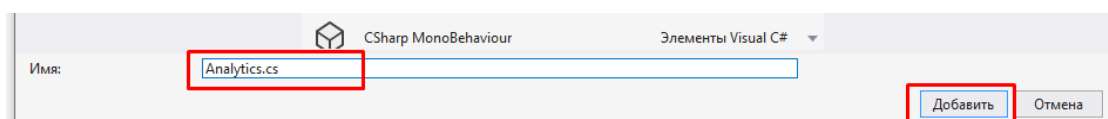
Нажмите правой кнопкой мыши на файл Class1.cs и выберите Удалить



Нажмите правой кнопкой мыши по проекту CompanyCoreLib. Затем в выпадающем меню выберите Добавить\Класс.



В качестве названия класса напишите Analytics.cs и нажмите **Добавить**.



Программный код класса Analytics библиотеки CompanyCoreLib

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace CompanyCoreLib
{
    public class Analytics
    {
        public List<DateTime> PopularMonth(List<DateTime> dates)
        {
            // если список был пуст, вернем пустой список
            if (dates.Count == 0)
                return new List<DateTime>();
            // создаем словарь вида 2019.12.01 5
            // Ключом является месяц, значением счетчик, который считает, сколько раз встречалась дата
            // месяца
            Dictionary<DateTime, int> monthCount = new Dictionary<DateTime, int>();
        }
    }
}
```

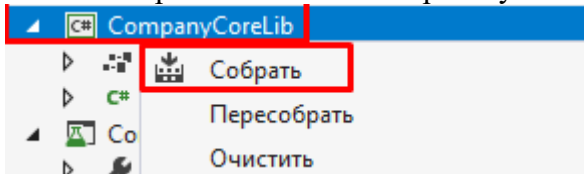
```

// проходим по списку дат
foreach (DateTime x in dates)
{
    // получение месяца из текущей даты
    int month = x.Month;
    // формируем ключ вида 2019.month.01
    DateTime newDate = new DateTime(x.Year, month, 1);
    if (monthCount.ContainsKey(newDate))
    {
        // если в словаре есть такой ключ, то значение счетчика увеличиваем на 1
        monthCount[newDate] += 1;
    }
    else
    {
        // если в словаре нет такого ключа, то создаем новый
        // элемент словаря вида Key: 2019.month.01 Value: 1
        monthCount[newDate] = 1;
    }
}
List<DateTime> returnDates = new List<DateTime>();
// пробегаемся по отсортированному словарю
// для сортировки используем Linq
// в начале сортируем словарь по значению в порядке убывания,
// затем по месяцам от меньшего к большему, если например значения совпадают
foreach (var item in monthCount.OrderByDescending(i => i.Value).ThenBy(i => i.Key.Month))
{
    // наполняем список возвращаемых дат
    returnDates.Add(item.Key);
}
return returnDates;
}
}
}

```

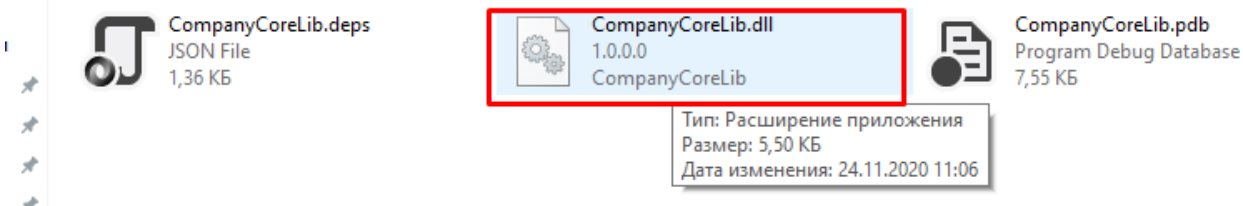
Выполнение сборки библиотеки

Нажмите правой кнопкой по проекту CompanyCoreLib и затем нажмите Собрать(Build)



В папке Debug появится библиотека

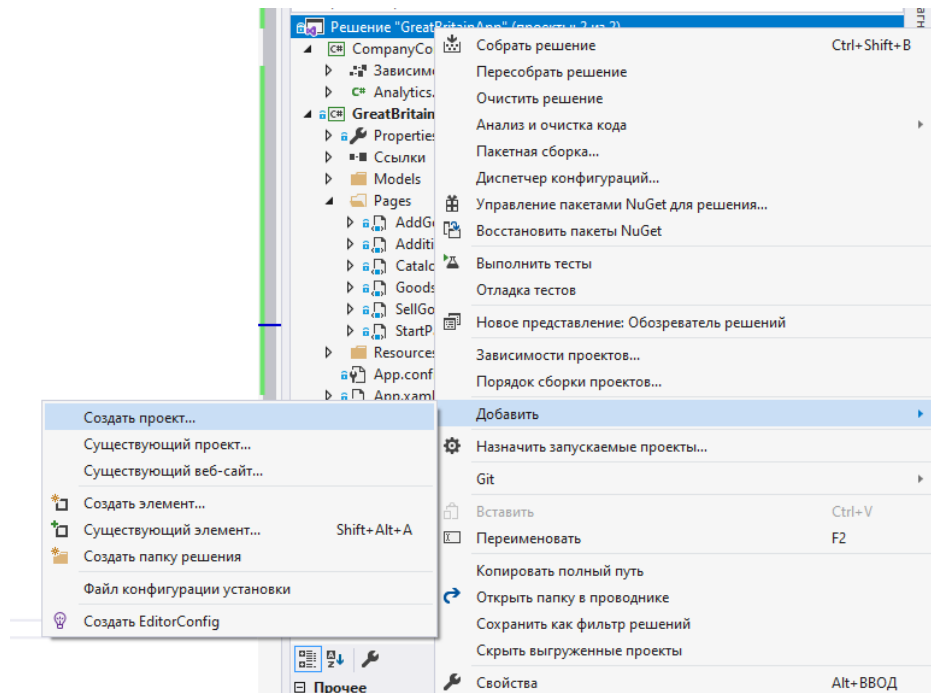
Этот компьютер > Сетевой диск (Z:) > МДК 03.02 > ДЕМОЭКЗАМЕН > Сессия 2 > CompanyCoreLib > bin > Debug > netstandard2.0



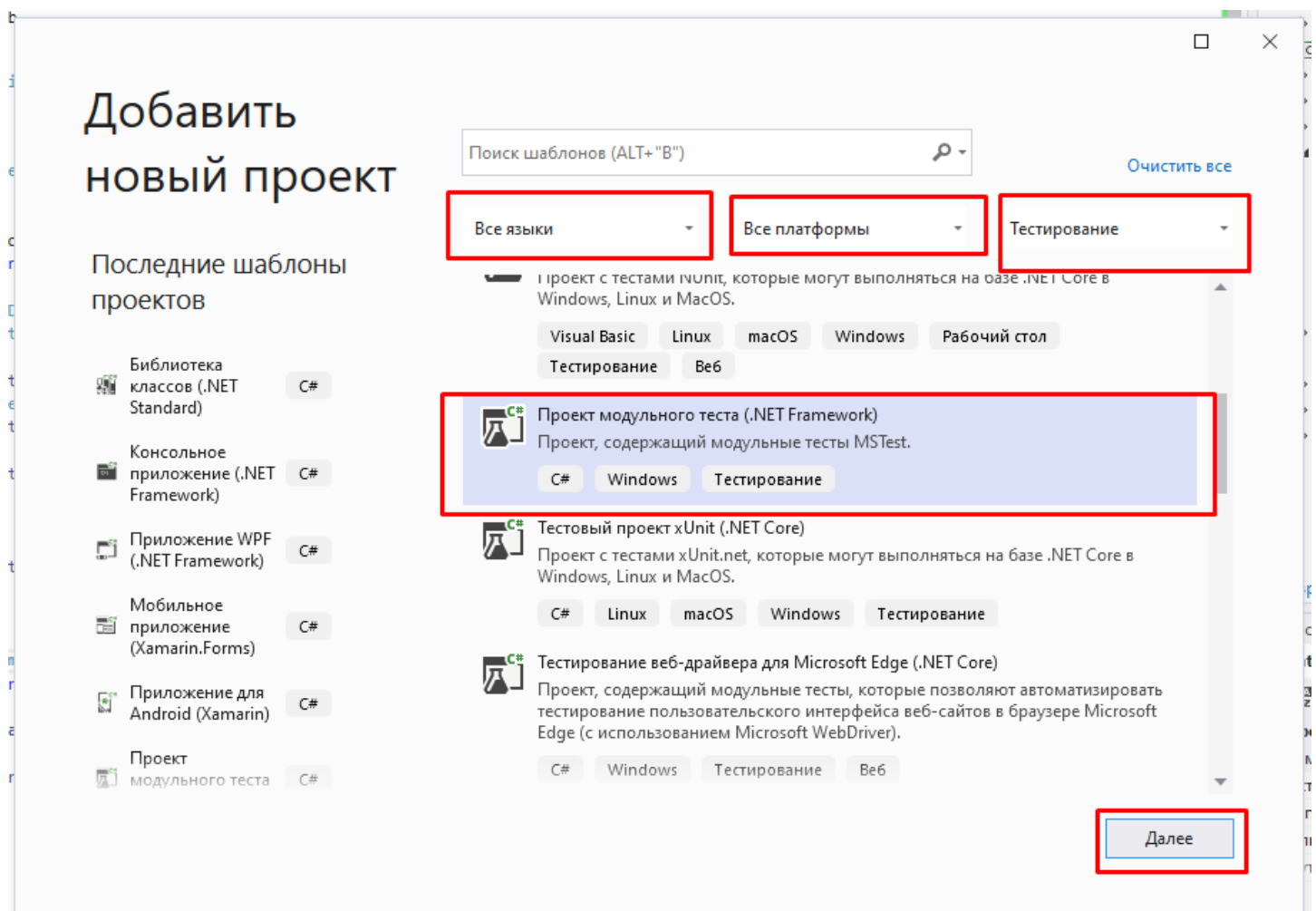
Модульные тесты UnitTest

Создание модульных тестов

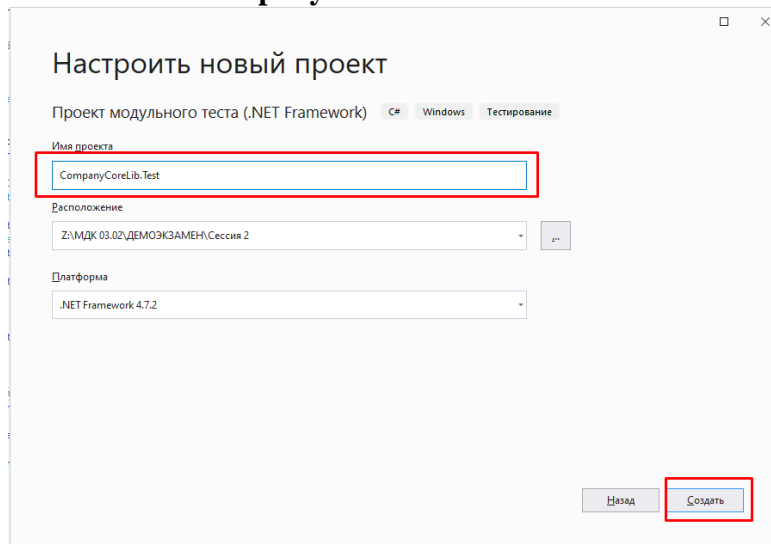
Возьмем имеющийся проект WPF приложения. В структуре проекта нажмем правой кнопкой на название решения и далее в появившемся контекстном меню выберем Add -> New Project... (Добавить новый проект):



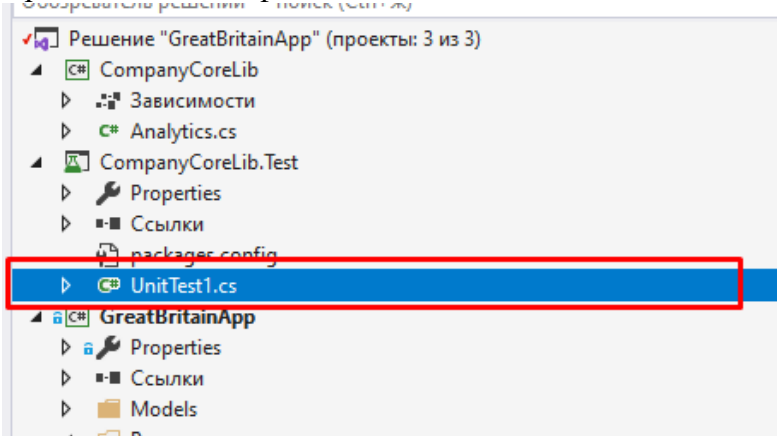
Выберите категорию Тестирование, далее Проект модульного теста(.NET Framework). Нажмите Далее.



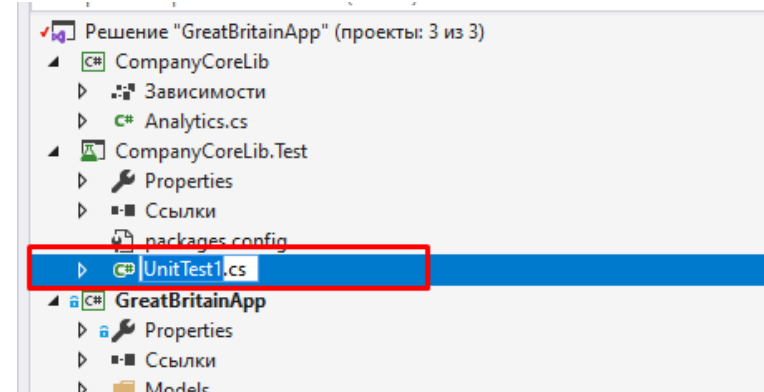
В поле имя проекта напишите **CompanyCoreLib.Test**



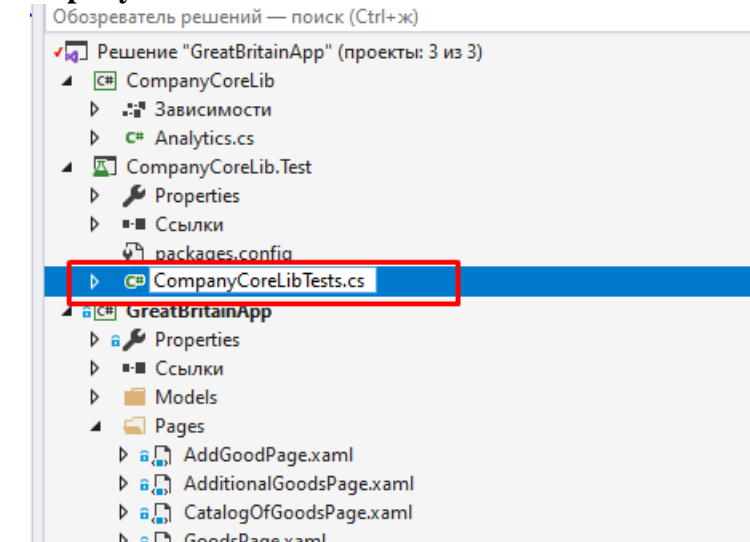
По умолчанию в проекте появится файл **UnitTest1.cs**.



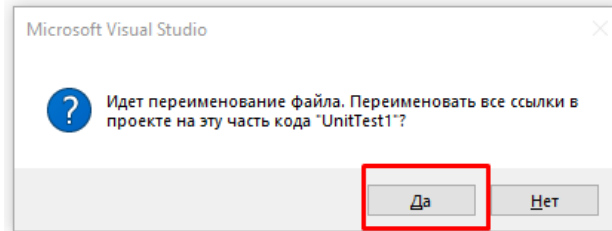
Выделите файл и нажмите F2.



Переименуйте в **CompanyCoreLibTests**.

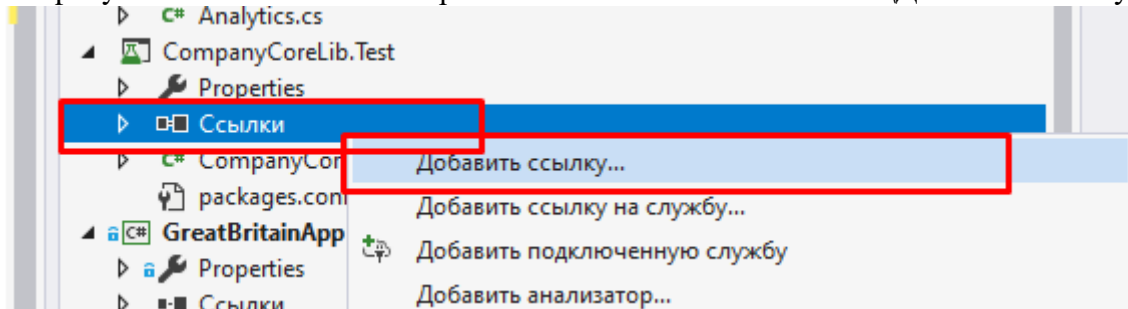


Подтвердите переименование файла.

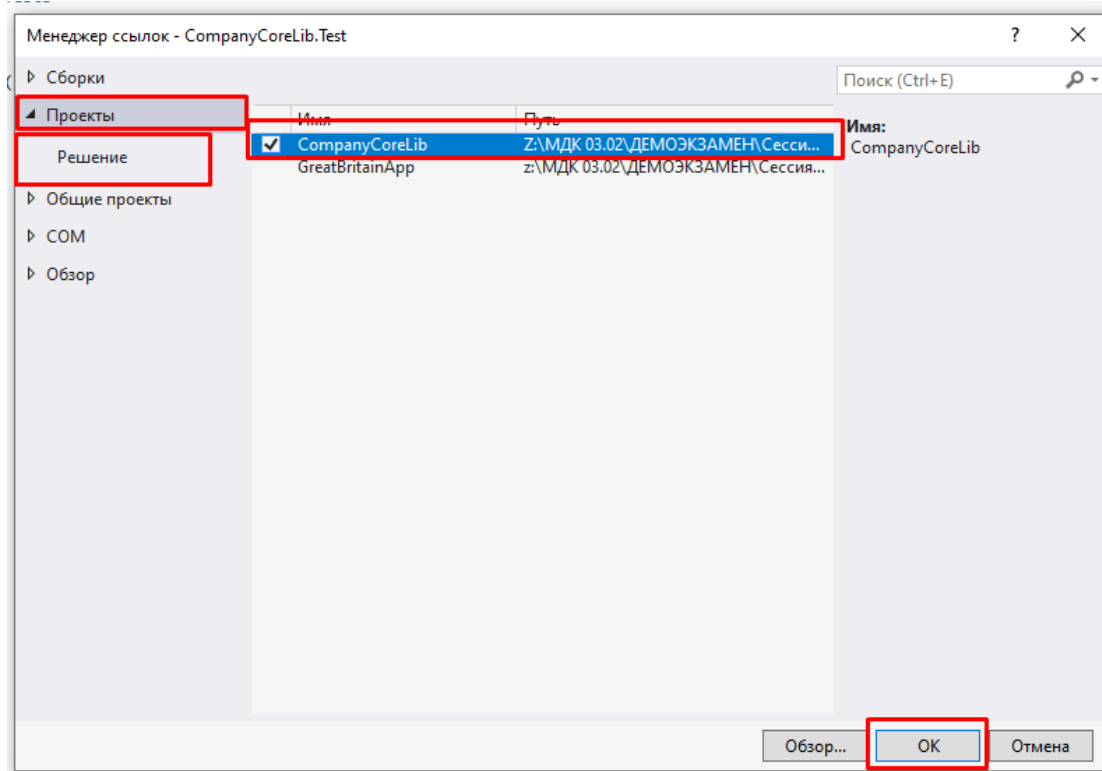


Добавление ссылки на тестируемую библиотеку

В проекте CompanyCoreLib.Test нажмите правой кнопкой мыши по Ссылки\Добавить ссылку...



В появившемся окне слева выберите Проект\Решение, поставьте галочку возле CompanyCoreLib и нажмите ОК.



Необходимо разработать модульные тесты, которые на основании исходных данных можно условно разделить на 2 группы следующим образом: 10 методов низкой сложности и 5 методов высокой сложности.

Программный код CompanyCoreLibTests.cs

Ниже представлены примеры трех тестовых методов

```
using System;
using System.Collections.Generic;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace CompanyCoreLib.Test
{
    [TestClass]
    public class CompanyCoreLibTests
    {
        //перед методом всегда надо ставить [TestMethod]
        [TestMethod]
        // тест на передачу пустого списка в метод PopularMonth
        public void PopularMonths_NullList()
        {
            // данные для передачи
            List<DateTime> dates = new List<DateTime>();
            // ожидаемый ответ
            // новый пустой список
            // количество его элементов равно 0
            int expected = 0;

            // создание экземпляра класса
            Analytics analytics = new Analytics();
            // возвращение результата работы метода
            int actual = analytics.PopularMonth(dates).Count;
            // проверка
            // если ожидаемый и актуальный ответ одинаковы
            // то тест будет пройден
            Assert.AreEqual(expected, actual);
        }
        [TestMethod]
        // тест низкой сложности 1
        public void PopularMonths_EasyOne()
        {
            // данные для передачи
            List<DateTime> dates = new List<DateTime>();
            dates.Add(new DateTime(2019, 12, 12, 14, 43, 0));
            dates.Add(new DateTime(2019, 12, 01, 15, 05, 0));
            dates.Add(new DateTime(2019, 11, 04, 9, 1, 0));

            // ожидаемый список результата
            List<DateTime> returnedDates = new List<DateTime>();
            returnedDates.Add(new DateTime(2019, 12, 01));
            returnedDates.Add(new DateTime(2019, 11, 01));
            // создание экземпляра класса
            Analytics analytics = new Analytics();
            // возвращение результата работы метода
            List<DateTime> actual = analytics.PopularMonth(dates);
            // для проверки коллекций используется класс
            // CollectionAssert
            CollectionAssert.AreEqual(returnedDates, actual);
        }
    }
}
```

```

[TestMethod]
// тест высокой сложности 1
public void PopularMonths_HardOne()
{
    // данные для передачи
    List<DateTime> dates = new List<DateTime>();
    dates.Add(new DateTime(2019, 12, 12, 14, 43, 0));
    dates.Add(new DateTime(2019, 12, 01, 15, 05, 0));
    dates.Add(new DateTime(2019, 11, 04, 9, 1, 0));
    dates.Add(new DateTime(2019, 11, 06, 9, 1, 0));
    dates.Add(new DateTime(2019, 11, 04, 9, 1, 0));
    dates.Add(new DateTime(2019, 11, 06, 9, 1, 0));
    dates.Add(new DateTime(2019, 11, 04, 9, 1, 0));
    dates.Add(new DateTime(2019, 11, 06, 9, 1, 0));
    dates.Add(new DateTime(2019, 10, 12, 14, 43, 0));
    dates.Add(new DateTime(2019, 10, 01, 15, 05, 0));
    dates.Add(new DateTime(2019, 9, 12, 14, 43, 0));
    dates.Add(new DateTime(2019, 9, 01, 15, 05, 0));
    // ожидаемый список результата
    List<DateTime> returnedDates = new List<DateTime>();
    returnedDates.Add(new DateTime(2019, 11, 01));
    returnedDates.Add(new DateTime(2019, 9, 01));
    returnedDates.Add(new DateTime(2019, 10, 01));
    returnedDates.Add(new DateTime(2019, 12, 01));
    // создание экземпляра класса
    Analytics analytics = new Analytics();
    // возвращение результата работы метода
    List<DateTime> actual = analytics.PopularMonth(dates);
    // для проверки коллекций используется класс
    // CollectionAssert
    CollectionAssert.AreEqual(returnedDates, actual);
}
}

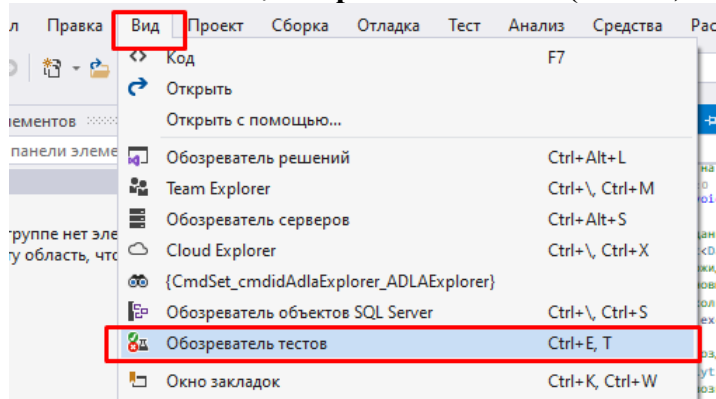
```

По аналогии добавьте еще 8 тестов низкой сложности и 4 высокой сложности.

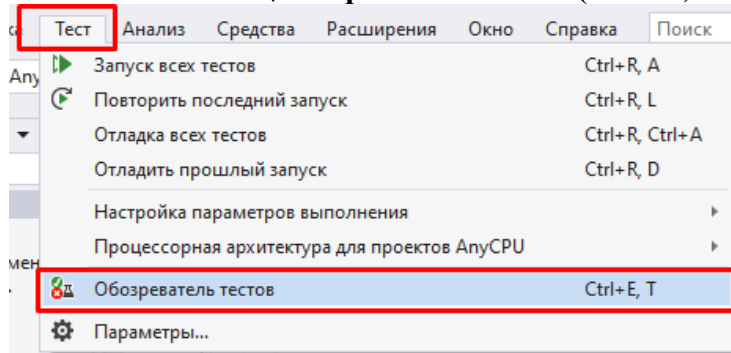
Запуск тестов

Для выполнения тестов нужно открыть окно Обзоратель тестов.

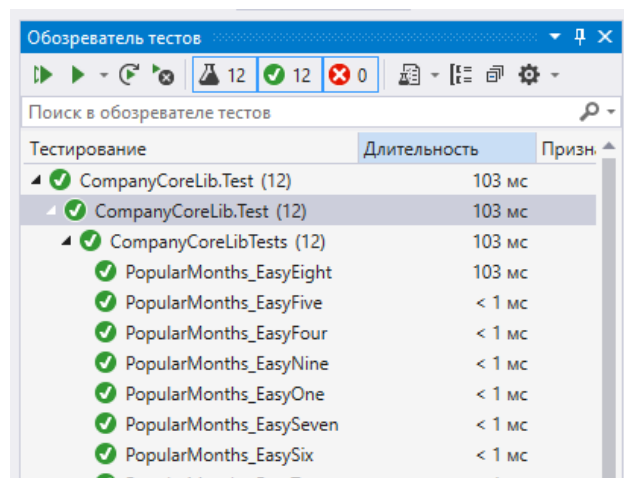
А. Меню Вид\Обзоратель тестов(Ctrl+E)



В. Меню Тест\Обзоратель тестов(Ctrl+E)

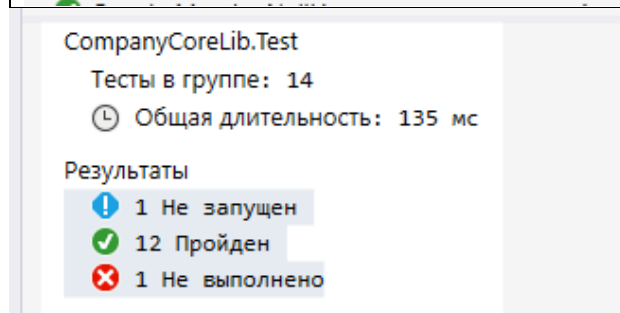


Окно обзорателя тестов



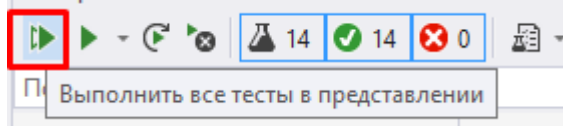
Результаты тестов

✓ PopularMonths_HardOne	< 1 мс	Белая галочка в зеленом – тест пройден успешно
! PopularMonths_HardThree		Восклицательный знак в синем – тест создан, но еще ни разу не запускался
✗ PopularMonths_HardTwo	76 мс	Белый крестик в красном – тест не пройден



Можно выполнить все тесты в решении, все тесты в группе или выбранный набор тестов. Выполните одно из следующих действий.

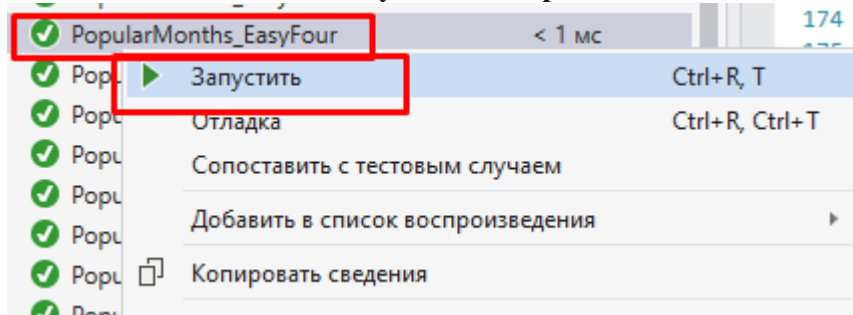
- Чтобы выполнить все тесты в решении, выберите значок **Выполнить все**.



- Чтобы выполнить все тесты в группе по умолчанию, выберите значок **Запуск**, а затем группу в меню.



- Выберите отдельные тесты, которые требуется запустить, откройте контекстное меню для выбранного теста и щелкните **Запустить выбранные тесты**.



Обзор элементов управления

Категория элементов управления WPF	Примеры членов	Назначение
Основные пользовательские элементы управления	Button, RadioButton, ComboBox, CheckBox, Calendar, DatePicker, Expander, DataGrid, ListBox, ListView, Slider, ToggleButton, TreeView, ContextMenu, ScrollBar, TabControl, TextBlock, TextBox, RepeatButton, RichTextBox, Label	WPF предлагает полное семейство элементов управления, которые можно использовать для построения пользовательских интерфейсов
Элементы украшения окон и элементов управления	Menu,ToolBar, StatusBar, ToolTip, ProgressBar	Эти элементы пользовательского интерфейса служат для декорирования рамки объекта Window компонентами для ввода (наподобие Menu) и элементами информирования пользователя (StatusBar, ToolTip и т.п.)
Элементы мультимедиа	Image, MediaElement, SoundPlayerAction	Эти элементы управления предоставляют поддержку воспроизведения аудио/видео и визуализации изображений
Элементы управления компоновкой	Border, Canvas, DockPanel, Grid, GridView, GridSplitter, GroupBox, Panel, TabControl, StackPanel, Viewbox, WrapPanel	WPF предлагает множество элементов управления, которые позволяют группировать и организовывать другие элементы для управления компоновкой

Свойство Name

Наверное важнейшее свойство. По установленному имени впоследствии можно будет обращаться к элементу, как в коде, так и в xaml разметке.

Например, в xaml-коде у нас определена следующая кнопка:

```
<Button x:Name="button1" Width="60" Height="30" Content="Текст" Click="button1_Click" />
```

Задание цветов в XAML

Свойство	Описание	Пример
Background	задании цвета фона или переднего плана в XAML	Background="#FFFF0000"
Foreground	Задание цвета текста в элементе	Foreground="#FFFF0000"
BorderBrush	Задание цвета рамки компонента	BorderBrush="#FFFF0000"
BorderThickness	Задание толщины рамки компонента	BorderThickness = "3"

Использование шрифтов

Свойство	Описание	Пример
FontFamily	Имя шрифта	FontFamily="Calibri"
FontSize	Размер шрифта в независимых от устройства единицах (по 1/96 дюйма)	FontSize = "14"
FontStyle	Наклон текста, представленный объектом FontStyle. Возможные значения свойства FontStyle содержатся в	FontStyle = "Italic"

	статических свойствах класса FontStyles, который включает написание символов Normal, Italic или Oblique. (Oblique — это искусственный способ создания курсивного текста на компьютере, в котором нет необходимого курсивного шрифта. Буквы берутся из обычного шрифта и скашиваются с помощью специального преобразования. Как правило, результат получается неважным.)	
FontWeight	Плотность текста, представленная объектом FontWeight. Вначале свойство FontWeight устанавливается из статических свойств класса FontWeights. Наиболее популярен вариант Bold (жирный), хотя в некоторых гарнитурах имеются и другие, такие как Heavy, Light, ExtraBold и т.д	FontWeight="Bold"
FontStretch	Коэффициент растяжения или сжатия текста, представленный объектом FontStretch. Возможные значения свойства FontStretch содержатся в статических свойствах класса FontStretches. Например, UltraCondensed сжимает текст до 50% от обычной ширины, а UltraExpanded растягивает его до 200%. Растяжение шрифта является особенностью OpenType, которая не поддерживается многими гарнитурами. (Чтобы поэкспериментировать с этим свойством, попробуйте шрифт Rockwell, который поддерживает его.)	FontStretch="UltraCondensed"

Ширина и высота

общая рекомендация состоит в том, что желательно избегать жестко закодированных в коде ширины и высоты.

Свойство	Описание	Пример
Width	Ширина элемента	Width="100"
Height	Высота элемента	Height="100"
MinWidth/MaxWidth	возможный диапазон ширины. при растяжении или сжатии контейнеров элементы с данными заданными свойствами не будут выходить за пределы установленных значений	MinWidth="300" MaxWidth="800"
MinHeight/MaxHeight	возможный диапазон высоты. при растяжении или сжатии контейнеров элементы с данными заданными свойствами не будут выходить за пределы установленных значений	MinHeight="300" MaxHeight="800"

Выравнивание

Свойство	Описание	Пример
HorizontalAlignment	выравнивает элемент по горизонтали относительно правой или левой стороны контейнера и соответственно может принимать значения Left, Right, Center (положение по центру), Stretch (растяжение по всей ширине).	HorizontalAlignment="Left" HorizontalAlignment="Center" HorizontalAlignment="Right" HorizontalAlignment="Stretch"
VerticalAlignment	выравнивает элемент по вертикали, которое принимает следующие значения: Top (положение вверху контейнера), Bottom (положение внизу), Center (положение по центру), Stretch (растяжение по всей высоте)	VerticalAlignment="Bottom" VerticalAlignment="Center" VerticalAlignment="Top" VerticalAlignment="Stretch"
HorizontalContentAlignment	Выравнивание содержимого внутри элемента по горизонтали. принимает значения Left, Right, Center (положение по центру), Stretch (растяжение по всей ширине)	
VerticalContentAlignment	Выравнивание содержимого внутри элемента по вертикали. принимает значения Top (положение вверху), Bottom (положение внизу), Center (положение по центру), Stretch (растяжение по всей высоте)	

Отступы

Свойство	Описание	Пример
Margin	устанавливает отступы вокруг элемента	Margin="5" Margin="5 10" Margin="5 10 15 20"
Padding	отступ содержимого элемента	Padding="5" Padding="5 10" Padding="5 10 15 20"

Класс Window

Класс Window привносит ряд свойств, которые позволяют настроить окно приложения:

- **AllowsTransparency:** при значении true позволяет установить прозрачный фон окна
- **Icon:** представляет иконку, которая отображается в левом верхнем углу окна и в панели задач. Если иконка не установлена, то система будет использовать стандартную иконку по умолчанию.
- **Top:** устанавливает отступ окна приложения от верхней границы экрана
- **Left:** устанавливает отступ окна приложения от левой границы экрана
- **ResizeMode:** задает режим изменения размеров окна. Может принимать следующие значения:
 - CanMinimize: окно можно только свернуть
 - NoResize: у окна нельзя изменить начальные размеры
 - CanResize: у окна можно изменять размеры
 - CanResizeWithGrip: в правом нижнем углу окна появляется визуализация того, что у окна можно изменять размеры
- **RestoreBounds:** возвращает границы окна
- **ShowInTaskbar:** при значении true иконка окна отображается на панели задач
- **SizeToContent:** позволяет автоматически масштабировать размеры окна в зависимости от содержимого. Может принимать следующие значения:
 - Width: автоматически масштабируется только ширина
 - Height: автоматически масштабируется только высота
 - WidthAndHeight: автоматически масштабируются высота и ширина
 - Manual: автоматическое масштабирование отсутствует
- **Title:** заголовок окна
- **Topmost:** при значении true окно устанавливается поверх других окон приложения
- **WindowStartupLocation:** устанавливает стартовую позицию окна. Может принимать следующие значения:
 - CenterOwner: если данное окно было запущено другим окном, то данное окно позиционируется относительно центра запустившего его окна
 - CenterScreen: окно помещается в центре экрана
 - Manual: позиция устанавливается вручную с помощью свойств Top и Left
- **WindowState:** состояние окна. Возможные значения:
 - Maximized: раскрыто на весь экран
 - Minimized: свернуто
 - Normal: стандартное состояние

Жизненный цикл

В процессе работы окно в WPF проходит ряд этапов жизненного цикла, которые доступны нам через обработку событий класса Window:

1. **Initialized:** это событие возникает при инициализации окна, когда у него устанавливаются все свойства, но до применения к нему стилей и привязки данных. Это общее событие для всех элементов управления в WPF, поэтому следует учитывать, что сначала возникают события вложенных элементов, а затем их контейнеров. То есть событие Initialized окна приложения генерируется только после того, как отработает событие Initialized для всех вложенных элементов.
2. **Loaded:** возникает после полной инициализации окна и применения к нему стилей и привязки данных. После генерации этого события происходит визуализация элемента, и окно отображается на экране и становится видимым для пользователя
3. **Closing:** возникает при закрытии окна
4. **Closed:** возникает, когда окно становится закрытым
5. **Unloaded:** возникает после закрытия окна при выгрузке всех связанных ресурсов из памяти

Соответственно, если нам надо выполнить некоторые действия при загрузке или при закрытии окна, мы можем обработать события Loaded и Closing/Closed.