Министерство образования Новосибирской области

ГБПОУ НСО «Новосибирский авиационный технический колледж им. Б.С. Галущака»

|  |  |  |
| --- | --- | --- |
| УТВЕРЖДАЮ:  Председатель цикловой комиссии УГС 09.00.00  \_\_/ А.А. Терехова  Протокол № \_\_ от «\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г. | УТВЕРЖДАЮ:  Председатель цикловой комиссии УГС 09.00.00  \_\_/ А.А. Терехова  Протокол № \_\_ от «\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г. | УТВЕРЖДАЮ:  Председатель цикловой комиссии УГС 09.00.00  \_\_/ А.А. Терехова  Протокол № \_\_ от «\_\_» \_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |

**List View. Обработка данных для приложений с графическим пользовательским интерфейсом**

Методические указания к практическому занятию № \_\_

Профессиональный модуль: ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

Междисциплинарный курс: МДК 01.01 Разработка программных модулей

Специальность: 09.02.07 Информационные системы и программирование

Разработал:

А.С. Иванов

2024

**Содержание**

[1 Теоретический раздел 3](#_Toc185022931)

[1.1 Основные понятия и термины 3](#_Toc185022932)

[1.2 Вёрстка элемента ListView 5](#_Toc185022933)

[2 Практический раздел 9](#_Toc185022934)

[2.1 Использование ListView в программных решениях для бизнеса 9](#_Toc185022935)

[2.2 Задание 19](#_Toc185022936)

# **1 Теоретический раздел**

## **1.1 Основные понятия и термины**

Класс ListView — это специализированный класс спискового типа, предназначенный для отображения различных представлений одних и тех же данных. Он особенно удобен, когда требуется создать представление, состоящее из множества столбцов и отображающее о каждом элементе данных несколько различных фрагментов информации.

Класс ListView унаследован от ListBox и дополняет его одной единственной деталью: свойством View. Свойство View представляет собой еще одну точку для создания многофункциональных списковых представлений. Если это свойство не установлено, элемент ListView ведет себя просто точно так же, как и его менее мощный предок — класс ListBox. Однако он становится гораздо более интересным, когда разработчик предоставляет объект представления, указывающий, каким должно быть форматирование и стили у элементов данных.

На рисунке 1 представлены графические представления ListBox и ListView

|  |  |
| --- | --- |
| Picture background | ListView на основе сетки |
| Рисунок 1 – Графические представления элементов ListBox и ListView | |

Формально свойство View указывает на экземпляр любого класса, унаследованного от ViewBase (который представляет собой абстрактный класс). Класс ViewBase удивительно прост: в действительности это не более чем оболочка, объединяющая вместе два стиля. Один из этих стилей применяется к элементу управления ListView (и указывается в свойстве DefaultStyleKey), в то время как другой применяется элементам внутри ListView (и указывается в свойстве ItemContainerDefaultStyleKey). Свойства DefaultStyleKey и ItemContainerDefaultStyleKey фактически стиль не предоставляют, а вместо этого они просто возвращают указывающий на него объект ResourceKey.

Также, свойство View является полезной абстракцией, предоставляющая разработчику следующие преимущества:

* представления, пригодные для многократного применения. ListView выделяет все касающиеся представления детали в один объект. Это упрощает создание представлений, не зависящих от данных и пригодных для применения с более чем одним списком;
* множественные представления. Отделение элемента управления ListView от объектов View также упрощает переключение между множеством различных представлений в одном и том же списке. (Например, именно такая технология применяется в проводнике Windows для получения различных представлений файлов и папок пользователя.) Точно такую же функциональную возможность можно создать и за счет динамического изменения шаблонов и стилей, но построить единственный объект, инкапсулирующий все детали представления, все-таки гораздо легче;
* более удобная организация. Объект представления упаковывает два стиля: один для корневого элемента управления ListView и еще один — для отдельных элементов в списке. Поскольку эти стили упаковываются вместе, очевидно, что эти два фрагмента связаны между собой и могут разделять определенные детали и взаимозависимости. Например, для основанного на столбцах элемента управления ListView это имеет большой смысл, поскольку ему необходимо, чтобы размещение заголовков столбцов совпадало с размещением предназначенных для этих столбцов данных.\

Применение этой модели открывает замечательные возможности для создания ряда полезных готовых представлений, которыми бы могли пользоваться все разработчики. К сожалению, в настоящее время в состав WPF входит пока только один объект представления — объект GridView. Хотя его можно и удобно использовать для построения списков с множеством столбцов, при наличии каких-то других потребностей придется создавать собственное специальное представление. В последующих разделах будет показано, как делать то и другое.

Элемент GridView будет замечательным выбором, если нужно, чтобы отображаемые данные можно было конфигурировать, а представление в стиле сетки было одной из доступных для пользователя опций. Но если требуется, чтобы в сетке дополнительно поддерживались расширенная стилизация, выбор или редактирование, придется перейти к использованию полнофункционального элемента управления DataGrid.

## **1.2 Вёрстка элемента ListView**

GridView — это класс, который наследуется от ViewBase и предоставляет списковое представление с множеством столбцов. Определяются эти столбцы путем добавления в коллекцию GridViewColumns объектов GridViewColumn.

GridView и GridViewColumn предлагают небольшой набор полезных методов, которые разработчик может использовать для настройки внешнего вида своего списка. Чтобы создать самый простой прямолинейный список (вроде Details (Список) в проводнике Windows), потребуется установить для каждого объекта GridViewColumn всего лишь два свойства: Header и DisplayMemberBinding. Свойство Header отвечает за размещаемый в верхней части столбца текст, а свойство DisplayMemberBinding содержит привязку, извлекающую из каждого элемента данных подлежащий отображению фрагмент информации.

Ниже приведен код разметки, необходимый для определения трех используемых в этом примере столбцов:

|  |
| --- |
| <ListView Name="lstw" Margin="5">  <ListView.View>  <GridView>  <GridView.Columns>  <GridViewColumn Header="Имя" DisplayMemberBinding="{Binding Path=ModelName}"></GridViewColumn>  <GridViewColumn Header="Модель" DisplayMemberBinding="{Binding Path=ModelNumber}"></GridViewColumn>  <GridViewColumn Header="Цена" DisplayMemberBinding="{Binding Path=UnitCost, StringFormat={}{0:C}}"></GridViewColumn>  </GridView.Columns>  </GridView>  </ListView.View>  </ListView> |

В результате имеем следующей список с данными на рисунке 2.



Рисунок 2 – Вид ListView

Замечание: данные, представленные в этом элементе, не указаны в коде разметки. Разметке включается в себя обращение к элементам наполнения.

Действительно, данные, служащие наполнителем интерактивных элементов, берутся из базы данных. В классических оконных WPF-приложениях база данных импортируется в решение как модель при помощи ORM-решения EntityFramework.

Как правило, создается некоторый класс данных, принимающий данные из модели. Пример такого класса представлен ниже:

|  |
| --- |
| public class Phone  {      public string Title { get; set; }      public string Company { get; set; }      public int Price { get; set; }  } |

В данном случае мы определяем класс Phone, который хранит данные о заголовке, производителе и цене телефона. Это минимальный набор для карточки товара, встречающихся, например, в приложении магазина по продаже компьютерной техники.

Теперь, при вёрстке интерактивных элементов, можно обратиться к данным посредством привязки.

Ниже приведён пример такой разметки.

|  |
| --- |
| <Window x:Class="ControlsApp.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:ControlsApp"          xmlns:col="clr-namespace:System.Collections;assembly=mscorlib"          mc:Ignorable="d"          Title="ListView" Height="220" Width="300">      <Grid Background="Lavender">          <ListView Name="phonesList" ItemsSource="{DynamicResource ResourceKey=phones}" >              <ListView.View>                  <GridView>                      <GridViewColumn DisplayMemberBinding="{Binding Path=Title}">Модель</GridViewColumn>                      <GridViewColumn DisplayMemberBinding="{Binding Path=Company}" Width="100">Компания</GridViewColumn>                      <GridViewColumn DisplayMemberBinding="{Binding Path=Price}">Цена</GridViewColumn>                  </GridView>              </ListView.View>              <ListView.Resources>                  <col:ArrayList x:Key="phones">                      <local:Phone Title="iPhone 6S" Company="Apple" Price="54990" />                      <local:Phone Title="Lumia 950" Company="Microsoft" Price="39990" />                      <local:Phone Title="Nexus 5X" Company="Google" Price="29990" />                      <local:Phone Title = "Galaxy Edge" Company = "Samsung" Price = "45670" />                  </col:ArrayList>              </ListView.Resources>          </ListView>      </Grid>  </Window> |

В результате получим некоторый список данных, как на рисунке 3.

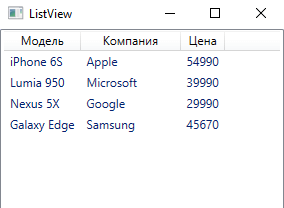


Рисунок 3 – ListView, наполненный переданными в него данными при помощи привязки

# **2 Практический раздел**

## **2.1 Использование ListView в программных решениях для бизнеса**

Предположим, что разрабатывается программное решение для автосервиса. Пользователь, при входе в приложение, должен понимать, какие товары предоставляет для покупки автосервис. В базе данных есть таблица, связанная с товарами. Поля таблицы представлены на рисунке 4.

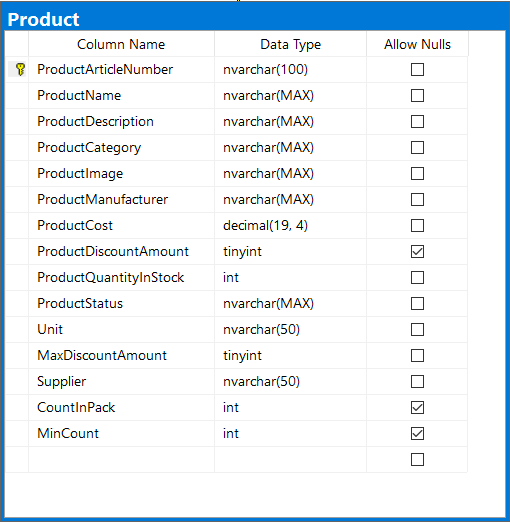


Рисунок 4 – Поля таблицы товаров автосервиса

В классическое WPF-решение мигрирована таблица в комплекте со всей базой даных через EntityFramework. См. рисунок 5.

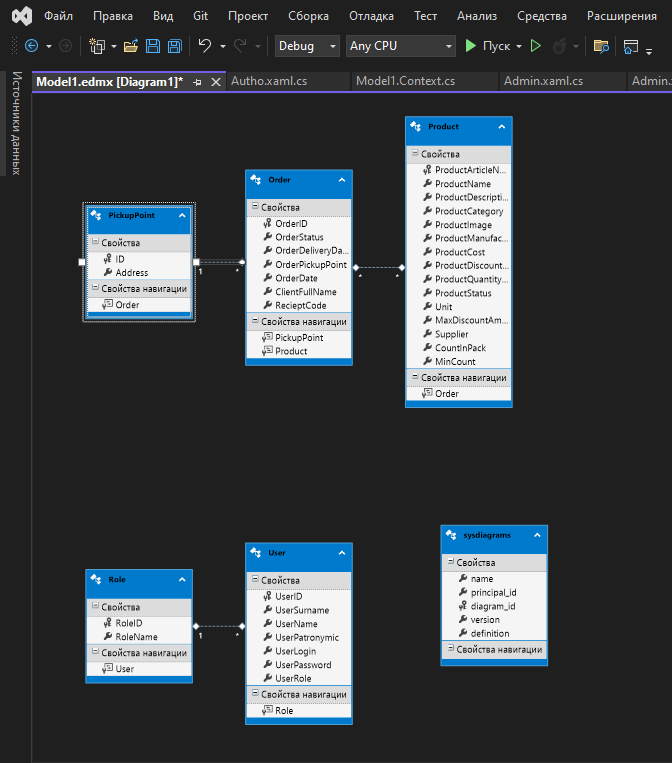


Рисунок 5 – EDM-модель базы данных

Соответственно, при миграции данных, EntityFramework определяет классы каждой сущности, находящихся в модели. Класс товаров выглядит следующим образом:

|  |
| --- |
| //------------------------------------------------------------------------------  // <auto-generated>  // Этот код создан по шаблону.  //  // Изменения, вносимые в этот файл вручную, могут привести к непредвиденной работе приложения.  // Изменения, вносимые в этот файл вручную, будут перезаписаны при повторном создании кода.  // </auto-generated>  //------------------------------------------------------------------------------  namespace rul.Entities  {  using System;  using System.Collections.Generic;    public partial class Product  {  [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]  public Product()  {  this.Order = new HashSet<Order>();  }    public string ProductArticleNumber { get; set; }  public string ProductName { get; set; }  public string ProductDescription { get; set; }  public string ProductCategory { get; set; }  public string ProductImage { get; set; }  public string ProductManufacturer { get; set; }  public decimal ProductCost { get; set; }  public Nullable<byte> ProductDiscountAmount { get; set; }  public int ProductQuantityInStock { get; set; }  public string ProductStatus { get; set; }  public string Unit { get; set; }  public byte MaxDiscountAmount { get; set; }  public string Supplier { get; set; }  public Nullable<int> CountInPack { get; set; }  public Nullable<int> MinCount { get; set; }    [System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]  public virtual ICollection<Order> Order { get; set; }  }  } |

Для отображения товаров пользователям приложения необходима страница клиента. Выглядит она следующим образом:

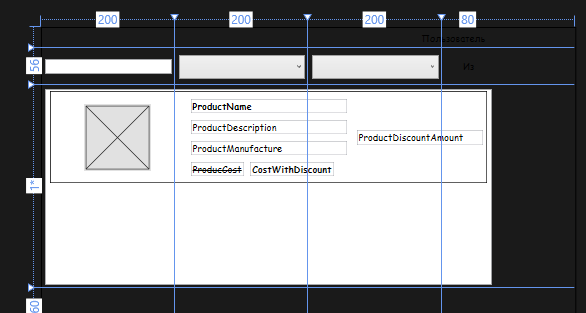


Рисунок 6 – Страница клиента

Основное наполнение страницы будет осуществляться с помощью элемента ListView и настройки скроллинга страницы.

Вёрстка страницы:

|  |
| --- |
| <Page x:Class="rul.Pages.Client"  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:rul.Pages" xmlns:sys="clr-namespace:System;assembly=mscorlib"  mc:Ignorable="d"  d:DesignHeight="450" d:DesignWidth="800"  Title="Client">  <Page.Resources>  <sys:String x:Key="defaultImage">  pack://application:,,,/Resources/picture.png  </sys:String>  </Page.Resources>  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="200"/>  <ColumnDefinition Width="200"/>  <ColumnDefinition Width="200"/>  <ColumnDefinition Width="80"/>  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="30"/>  <RowDefinition Height="56"/>  <RowDefinition />  <RowDefinition Height="60"/>  </Grid.RowDefinitions>  <StackPanel Orientation="Horizontal" Grid.Row="0" Grid.Column="2" Grid.ColumnSpan="3" HorizontalAlignment="Right">  <TextBlock Text="Пользователь"/>  <TextBlock x:Name="txtFullName"/>  </StackPanel>  <TextBox Name ="txtSearch" Grid.Row="1" Grid.Column="0" VerticalAlignment="Center" Margin="5, 0, 5, 0" SelectionChanged="txtSearch\_SelectionChanged"/>  <ComboBox Name ="cmbSorting" Grid.Row="1" Grid.Column="1" SelectedIndex="0" VerticalAlignment="Center" Margin="5, 0, 5, 0" ItemsSource="{Binding SortingList}" SelectionChanged="cmbSorting\_SelectionChanged"/>  <ComboBox Name ="cmbFilter" Grid.Row="1" Grid.Column="2" SelectedIndex="0" VerticalAlignment="Center" Margin="5, 0, 5, 0" ItemsSource="{Binding FilterList}" SelectionChanged="cmbFilter\_SelectionChanged"/>  <StackPanel Orientation="Horizontal" Grid.Row="1" Grid.Column="3" VerticalAlignment="Center" HorizontalAlignment="Center">  <TextBlock x:Name="txtResultAmount"/>  <TextBlock Text="Из"/>  <TextBlock x:Name="txtAllAmount"/>  </StackPanel>  <ListView Name="LViewProduct" Grid.Column="0" Grid.ColumnSpan="4" Grid.Row="2" Margin="5,5,5,5" d:ItemsSource="{d:SampleData ItemCount=1}">  <ListView.ItemContainerStyle>  <Style TargetType="ListViewItem">  <Setter Property="HorizontalContentAlignment" Value="Stretch"/>  </Style>  </ListView.ItemContainerStyle>  <ListView.ItemTemplate>  <DataTemplate>  <Border BorderBrush="Black" BorderThickness="1" Background="{Binding Backgroud}">  <Grid>  <Grid.ColumnDefinitions>  <ColumnDefinition Width="200"/>  <ColumnDefinition Width="\*"/>  <ColumnDefinition Width="200"/>  </Grid.ColumnDefinitions>  <Image Width="100" Height="100" Margin="5" Source="{Binding ImgPath, FallbackValue={StaticResource defaultImage}}"/>  <StackPanel Grid.Column="1" Margin="5">  <TextBox Text="{Binding ProductName}" FontWeight="Bold"/>  <TextBox Text="{Binding ProductDescription}"/>  <TextBox Text="{Binding ProductManufacture}"/>  <StackPanel Orientation="Horizontal">  <TextBox Text="{Binding ProducCost}" TextDecorations="Strikethrough"/>  <TextBox Text="{Binding CostWithDiscount, StringFormat='\0 рублей'}" FontWeight="Bold"/>  </StackPanel>  </StackPanel>  <TextBox Grid.Column="2" Text="{Binding ProductDiscountAmount, StringFormat='\0 %'}" VerticalAlignment="Center"/>  </Grid>  </Border>  </DataTemplate>  </ListView.ItemTemplate>  <ListView.ContextMenu>  <ContextMenu Name="contextMenu">  <MenuItem Name="btnAddProduct" Header="Добавить к заказу" Click="btnAddProduct\_Click"/>  </ContextMenu>  </ListView.ContextMenu>  </ListView>  <Button x:Name="btnOrder" Grid.Row="3" Grid.Column="0" Content="Заказ" Visibility="Collapsed" HorizontalAlignment="Left" Width="200" Margin="20, 0, 0, 0" Click="bthOrder\_Click"/>  </Grid>  </Page> |

Пояснения:

Для отображения изображений товаров имеется настройка <Image Width="100" Height="100" Margin="5" Source="{Binding ImgPath, FallbackValue={StaticResource defaultImage}}"/> , получающая информацию из каталога ресурсов. При отсутствии изображения товара будет отображаться крестик, означающий отсутствие изображения, созданный с помощью надстройки вёрстки <Page.Resources>

<sys:String x:Key="defaultImage">

pack://application:,,,/Resources/picture.png

</sys:String>

</Page.Resources>

Далее идет основной компонент наполнения – ListView:

<ListView Name="LViewProduct" Grid.Column="0" Grid.ColumnSpan="4" Grid.Row="2" Margin="5,5,5,5" d:ItemsSource="{d:SampleData ItemCount=1}">

<ListView.ItemContainerStyle>

<Style TargetType="ListViewItem">

<Setter Property="HorizontalContentAlignment" Value="Stretch"/>

</Style>

</ListView.ItemContainerStyle>

<ListView.ItemTemplate>

<DataTemplate>

<Border BorderBrush="Black" BorderThickness="1" Background="{Binding Backgroud}">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="200"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="200"/>

</Grid.ColumnDefinitions>

<Image Width="100" Height="100" Margin="5" Source="{Binding ImgPath, FallbackValue={StaticResource defaultImage}}"/>

<StackPanel Grid.Column="1" Margin="5">

<TextBox Text="{Binding ProductName}" FontWeight="Bold"/>

<TextBox Text="{Binding ProductDescription}"/>

<TextBox Text="{Binding ProductManufacture}"/>

<StackPanel Orientation="Horizontal">

<TextBox Text="{Binding ProducCost}" TextDecorations="Strikethrough"/>

<TextBox Text="{Binding CostWithDiscount, StringFormat='\0 рублей'}" FontWeight="Bold"/>

</StackPanel>

</StackPanel>

<TextBox Grid.Column="2" Text="{Binding ProductDiscountAmount, StringFormat='\0 %'}" VerticalAlignment="Center"/>

</Grid>

</Border>

</DataTemplate>

</ListView.ItemTemplate>

<ListView.ContextMenu>

<ContextMenu Name="contextMenu">

<MenuItem Name="btnAddProduct" Header="Добавить к заказу" Click="btnAddProduct\_Click"/>

</ContextMenu>

</ListView.ContextMenu>

</ListView>

Вёрстка разбита на несколько частей.

* ListView:

Name="LViewProduct": Имя элемента, которое можно использовать для доступа к нему в коде.

Grid.Column="0" Grid.ColumnSpan="4" Grid.Row="2" Margin="5,5,5,5": Определяет положение и отступы элемента в сетке.

d:ItemsSource="{d:SampleData ItemCount=1}": Используется для отображения примеров данных в режиме дизайна.

* ListView.ItemContainerStyle:

Style TargetType="ListViewItem": Определяет стиль для элементов списка.

Setter Property="HorizontalContentAlignment" Value="Stretch": Устанавливает горизонтальное выравнивание содержимого элементов списка на "Stretch", что позволяет содержимому заполнять всю доступную ширину.

* ListView.ItemTemplate:

DataTemplate: Определяет шаблон для отображения каждого элемента списка.

Border: Определяет границу и фон для каждого элемента списка.

BorderBrush="Black" BorderThickness="1" Background="{Binding Backgroud}": Устанавливает чёрную границу толщиной 1 и привязывает фон к свойству Backgroud данных.

* Grid:

Grid.ColumnDefinitions: Определяет три столбца для сетки.

Первый столбец шириной 200.

Второй столбец занимает оставшееся пространство.

Третий столбец шириной 200.

* Image:

Width="100" Height="100" Margin="5" Source="{Binding ImgPath, FallbackValue={StaticResource defaultImage}}": Отображает изображение продукта. Источник изображения привязывается к свойству ImgPath, а если значение отсутствует, используется ресурс defaultImage.

* StackPanel:

Grid.Column="1" Margin="5": Определяет панель для отображения текстовых полей во втором столбце.

TextBox: Отображает различные свойства продукта:

ProductName: Название продукта.

ProductDescription: Описание продукта.

ProductManufacture: Производитель продукта.

ProducCost: Стоимость продукта с зачёркиванием.

CostWithDiscount: Стоимость продукта со скидкой, форматированная как "0 рублей".

* TextBox:

Grid.Column="2" Text="{Binding ProductDiscountAmount, StringFormat='\0 %'}" VerticalAlignment="Center": Отображает размер скидки в третьем столбце, форматированный как "0 %".

* ListView.ContextMenu:

ContextMenu Name="contextMenu": Определяет контекстное меню для элементов списка.

MenuItem Name="btnAddProduct" Header="Добавить к заказу" Click="btnAddProduct\_Click": Определяет элемент контекстного меню для добавления продукта к заказу. При нажатии вызывается обработчик события btnAddProduct\_Click.

Для получения окончательного результата применим реализацию выборки данных из базы:

|  |
| --- |
| public partial class MainWindow : Window  {  public MainWindow()  {  InitializeComponent();  LoadProducts();  }  private void LoadProducts()  {  var products = GetProductsFromDatabase();  LViewProduct.ItemsSource = products;  }  private void btnAddProduct\_Click(object sender, RoutedEventArgs e)  {  // Обработчик события для добавления продукта к заказу  if (LViewProduct.SelectedItem is Product selectedProduct)  {  // Логика добавления продукта к заказу  MessageBox.Show($"Добавлен продукт: {selectedProduct.ProductName}");  }  }  } |

Результат:

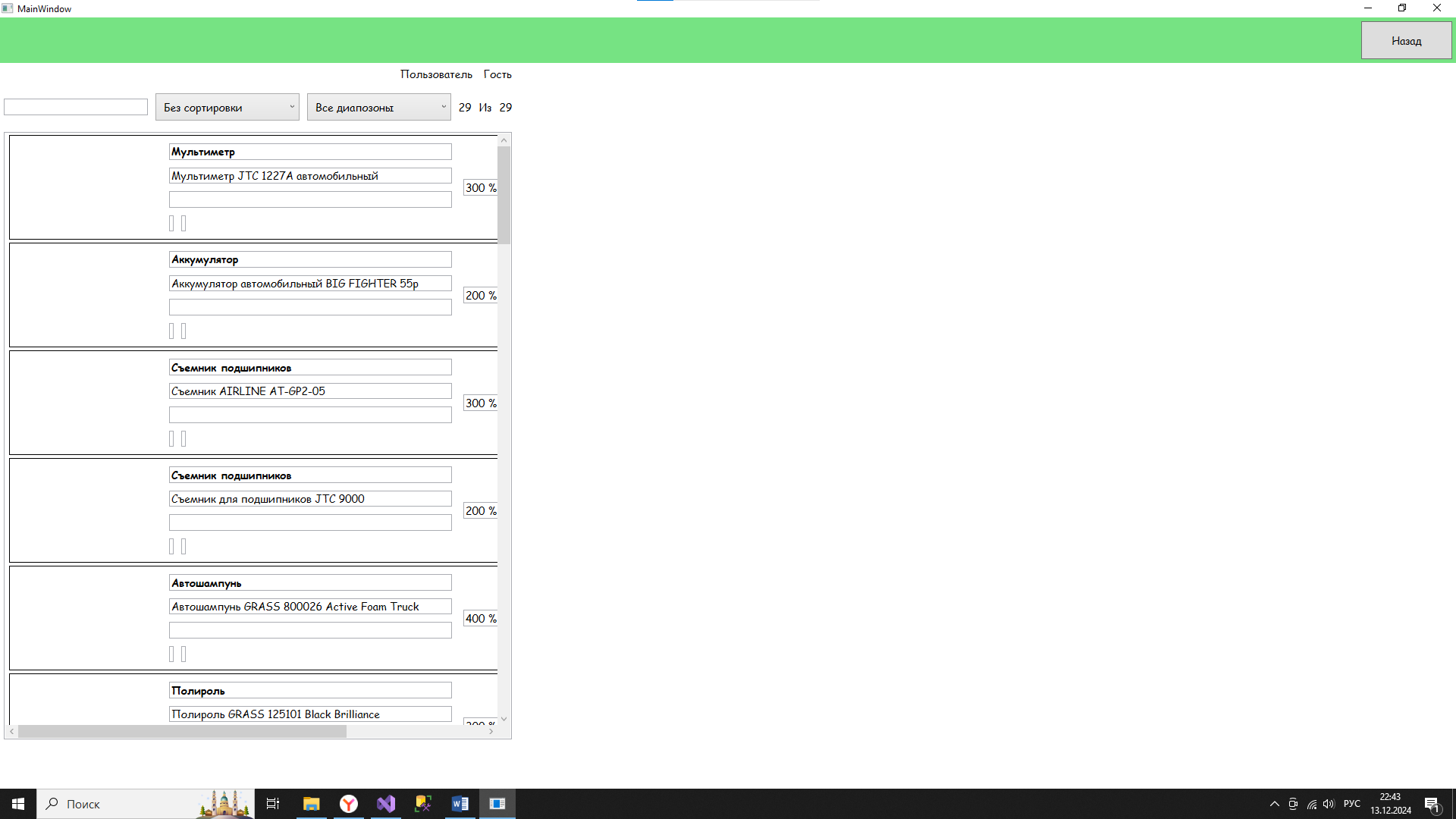


Рисунок 7 – Результат отображения данных

Данные, предназначенные для наполнения, находятся в базе данных

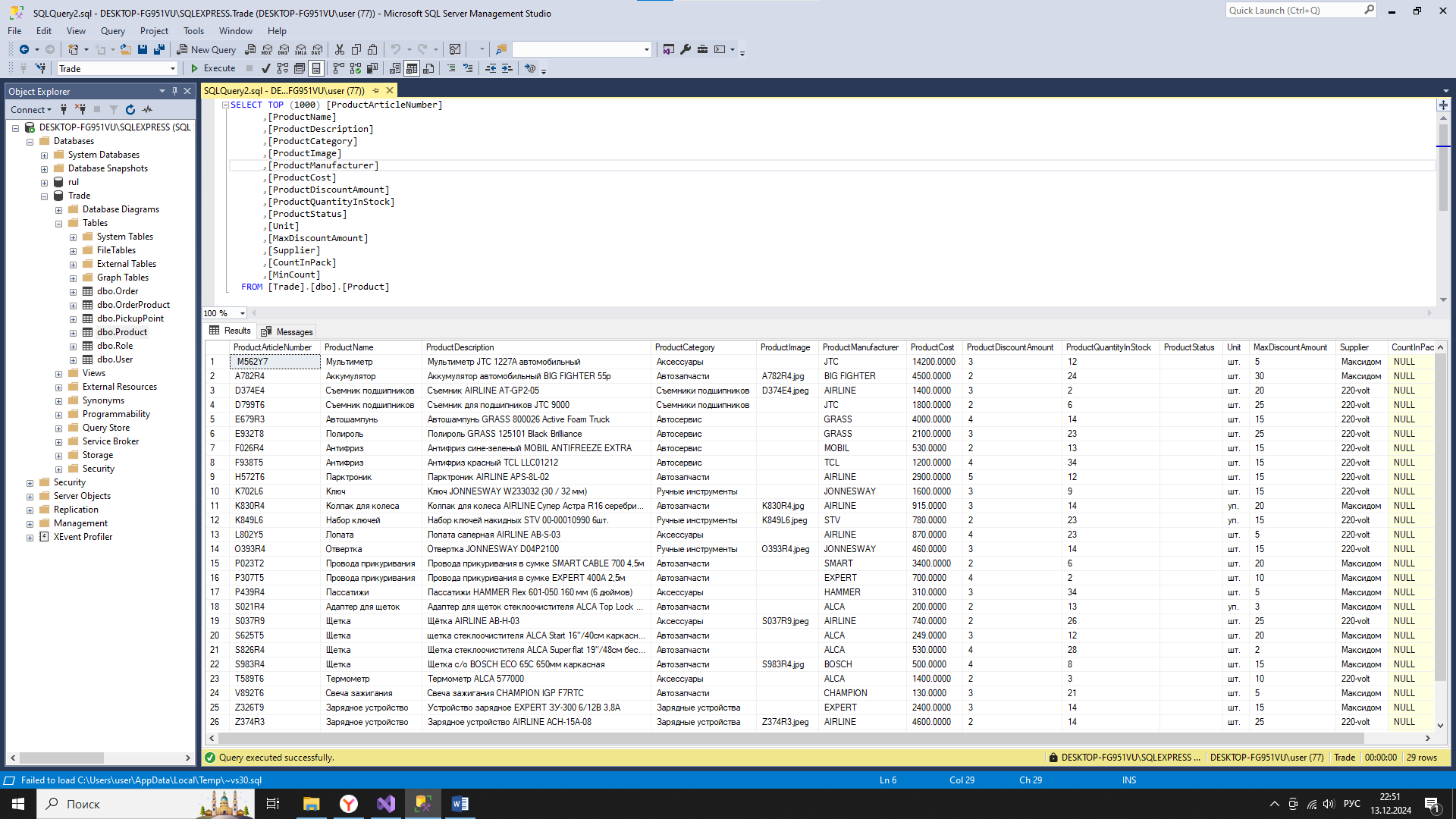


Рисунок 8 – Наполнитель данных

## **2.2 Задание**

В качестве задания, для отработки изученного материала, предлагается добавить возможность вывода товаров в индивидуальном проекте (информационной системе), варианты для которого были выданы на вводном (первом занятии). Предполагается, что к моменту выполнения практической работы, имеется готовая, заполненная база данных и импортированная в WPF-решение через EntityFramework.

Задание является выполненным только в том случае, когда оно реализует поставленную задачу и защищено преподавателю в очной форме.