Министерство образования Новосибирской области
ГБПОУ НСО «Новосибирский авиационный технический колледж им. Б.С. Галущака»

Обработка изменения, редактирования и удаления данных в WPF-приложениях

Методические указания к практическому занятию № 11

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

2025

В современных приложениях, взаимодействующих с базой данных присутствует выполнение основных операций пользователями. Все эти операции описываются аббревиатурой CRUD (Create-Read-Update-Delete).

На текущий момент используется два подхода к реализации CRUD-логики в приложениях:

1. использование ReST API контроллера для взаимодействия с объектами бизнес-логики;
2. использование ORM-фреймворков для работы с экземпляром класса базы данных.

В первом случае имеется некоторый виртуальный выделенный сервер (VDS), на котором запущен экземпляр класса базы данных и постоянно активен ReST API контроллер, позволяющий пользователям с других устройств взаимодействовать из приложения с базой данных, находящейся в некотором хранилище.

Второй вариант подхода похож на первый, единственное отличие заключается в том, что развёртывание экземпляра базы данных происходит локально вместе с приложением и через ORM-фреймворк настраивается взаимодействие.

В курсе МДК 01.01 чаще всего будет использоваться второй вариант взаимодействия и настройки CRUD-логики.

На прошлых занятиях были отработаны на практике навыки подключения модели данных из MS SQL Server в WPF-приложение. Если подключение было выполнено верно, то результат будет таковым:

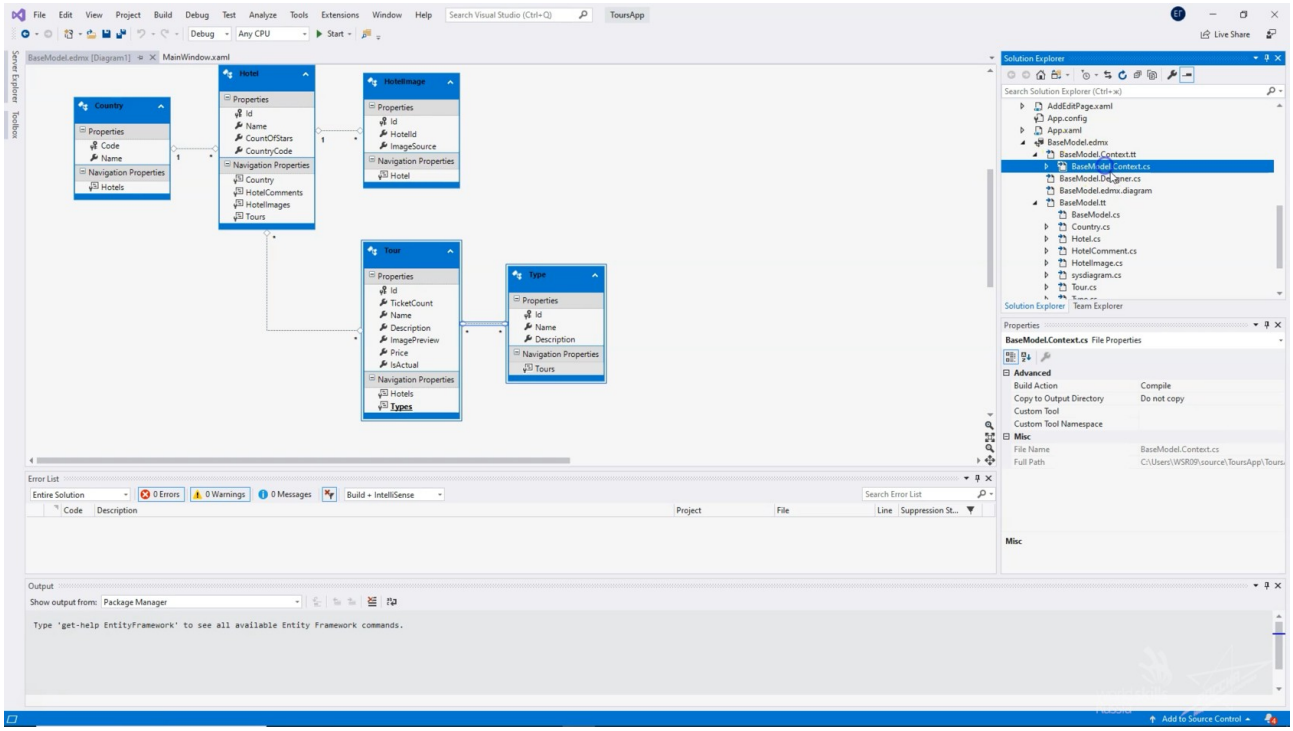

Рисунок 1 — Модель данных в WPF-приложении

Разработка CRUD-логики заключается в том, что необходимо настроить обращения к ADO.NET-модели.

Для этого необходимо знание, понимание и применение паттерна SingleTone (Одиночка).

SingleTone - порождающий паттерн, который гарантирует, что для определенного класса будет создан только один объект, а также предоставит к этому объекту точку доступа. Такой паттерн позволяет создать объект только тогда, когда это необходимо. В этом, кстати, и заключается отличие этого паттерна от глобальных переменных.

На практике применение этого паттерна происходит следующим образом. В сгенерированных файлах сущностей (как правило, каталог называют Entities) в файле BaseModel.Context.cs объявляется приватное статическое поле (переменная внутри класса), под которым понимается контекст.

Далее добавляется метод (функция внутри класса) получения экземпляра из этого контекста.

Пример кода:

```
namespace rul.Entities
{
    using System;
    using System.Data.Entity;
    using System.Data.Entity.Infrastructure;
    using rul.Pages;

    public partial class RulEntities : DbContext
    {
        private static RulEntities context;
        public RulEntities()
            : base("name=RulEntities")
        {
        }

        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            throw new UnintentionalCodeFirstException();
        }

        public virtual DbSet<Order> Order { get; set; }
        public virtual DbSet<PickupPoint> PickupPoint { get; set; }
        public virtual DbSet<Product> Product { get; set; }
        public virtual DbSet<Role> Role { get; set; }
        public virtual DbSet<sysdiagrams> sysdiagrams { get; set; }
        public virtual DbSet<User> User { get; set; }
        public virtual DbSet<OrderProduct> OrderProduct { get; set; }

        public static RulEntities GetContext()
        {
            if(context == null)
                context = new RulEntities();
            return context;
        }
    }
}
```

Применение метода получения контекста можно описать в виде вёрстки страницы и выдачи данных на элементы при помощи метода Binding. (См. Методические указания по работе с ListView).

Рассмотрим другой пример с добавлением данных. В вёрстке страницы необходимо добавить элементы, которые будут отвечать за выбор добавляемых данных (это может быть, например, ContextMenu) и интерактивный элемент, при нажатии на который будут добавлены данные (это может быть кнопка).

Пример добавления этих элементов можно описать частью кода разметки на Админ-Панели:

Вёрстка страницы:

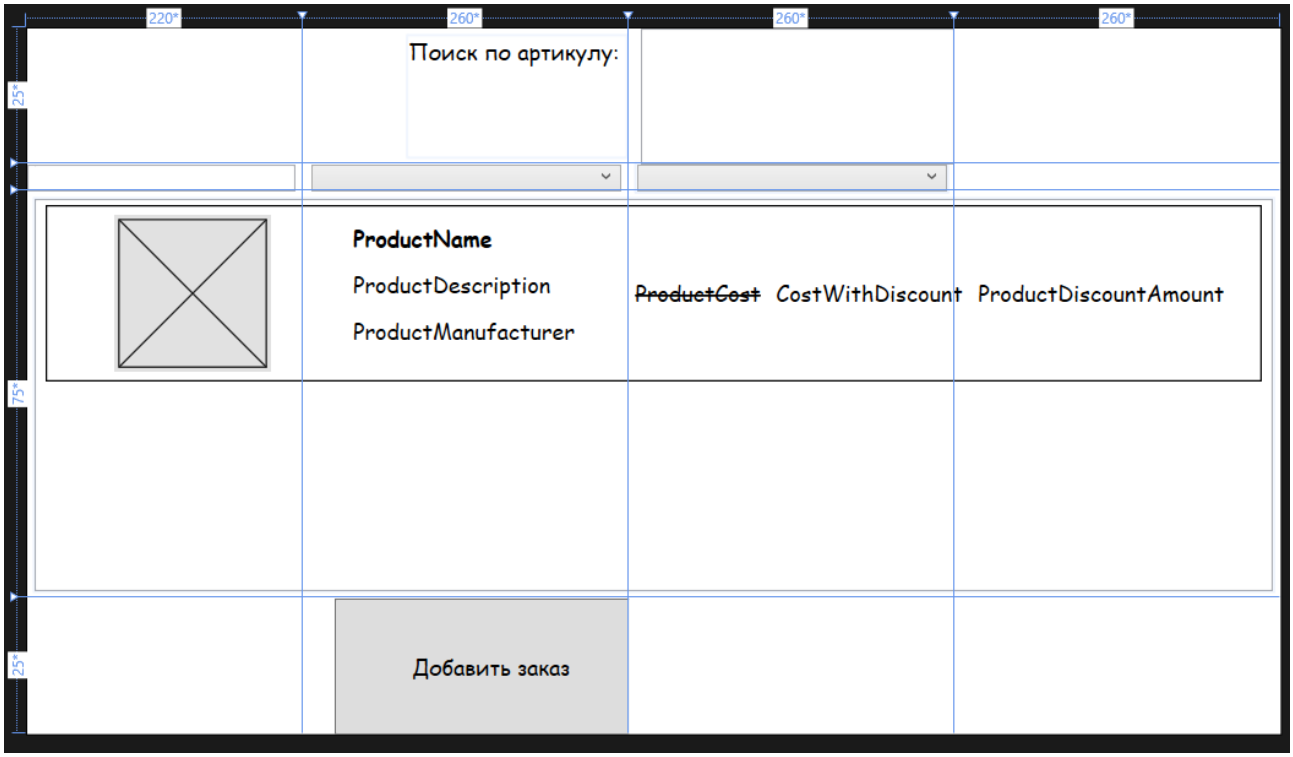

Рисунок 2 — Страница админа

```
<Page x:Class="rul.Pages.Admin"
      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="Admin" IsVisibleChanged="Page_IsVisibleChanged">
    <Page.Resources>
        <sys:String x:Key="defaultImage">
            pack://application:,,,/Resources/picture.png
        </sys:String>
    </Page.Resources>

    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="220*" />
            <ColumnDefinition Width="260*" />
            <ColumnDefinition Width="260*" />
            <ColumnDefinition Width="260*" />
        </Grid.ColumnDefinitions>

        <Grid.RowDefinitions>
            <RowDefinition Height="25*" />
            <RowDefinition Height="5*" />
            <RowDefinition Height="75*" />
            <RowDefinition Height="25*" />
        </Grid.RowDefinitions>

             <StackPanel  Orientation="Horizontal"  Grid.Row="0"  Grid.Column="0"
Grid.ColumnSpan="3" HorizontalAlignment="Right">
            <TextBlock Text="Поиск по артикулу: " />
```

```
            <TextBox x:Name="txtArticleNumber" Width="200" Margin="5,0,0,0" />
        </StackPanel>
                    <TextBox   Name   ="txtSearch"   Grid.Row="1"   Grid.Column="0"
VerticalAlignment="Center"          Margin="5,          0,          5,          0"
SelectionChanged="txtSearch_SelectionChanged"/>
                    <TextBox   Name="txtFullName"   Grid.Row="1"   Grid.Column="0"
SelectionChanged="txtSorting_SelectionChanged"        VerticalAlignment="Center"
Margin="0, 0, 5, 0" />
                    <ComboBox   Name="cmbSorting"   Grid.Row="1"   Grid.Column="1"
ItemsSource="{Binding                                          SortingList}"
SelectionChanged="cmbSorting_SelectionChanged"                 SelectedIndex="0"
VerticalAlignment="Center" Margin="5, 0, 5, 0" />
                     <ComboBox   Name="cmbFilter"   Grid.Row="1"   Grid.Column="2"
ItemsSource="{Binding FilterList}" SelectionChanged="cmbFilter_SelectionChanged"
SelectedIndex="0" Margin="5, 0, 5, 0" VerticalAlignment="Center" />
             <StackPanel  Orientation="Horizontal"  Grid.Row="1"  Grid.Column="3"
VerticalAlignment="Center" HorizontalAlignment="Center">
            <TextBlock x:Name="txtResultCount" />
            <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}"
MouseDoubleClick="lViewProduct_MouseDoubleClick">
            <ListView.ItemContainerStyle>
                <Style TargetType="ListViewItem">
                                <Setter  Property="HorizontalContentAlignment"
Value="Stretch" />
                </Style>
            </ListView.ItemContainerStyle>
            <ListView.ItemTemplate>
                <DataTemplate>
                                <Border  BorderBrush="Black"  BorderThickness="1"
Background="{Binding Background}">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="260*" />
                                <ColumnDefinition Width="260*" />
                                <ColumnDefinition Width="260*" />
                                <ColumnDefinition Width="260*" />
                            </Grid.ColumnDefinitions>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="Auto" />
                                <RowDefinition Height="Auto" />
                            </Grid.RowDefinitions>
                                        <Image Width="100" Height="100" Margin="5"
Stretch="Uniform"    Source="{Binding    ImgPath,    FallbackValue={StaticResource
defaultImage}}" />
                            <StackPanel Grid.Column="1" Margin="5">
                                        <TextBlock Text="{Binding ProductName}"
FontWeight="Bold" />
                                                <TextBlock  Text="{Binding
ProductDescription}" />
                                                <TextBlock  Text="{Binding
ProductManufacturer}" />
                            </StackPanel>
                            <StackPanel Orientation="Horizontal" Grid.Column="2"
VerticalAlignment="Center">
                                        <TextBlock Text="{Binding ProductCost}"
TextDecorations="Strikethrough" />
                                    <TextBlock Text="{Binding CostWithDiscount,
StringFormat={}(0)}" />
                            </StackPanel>
```

```
                                                    <StackPanel  Grid.Column="3"
VerticalAlignment="Center">
                                        <TextBlock Grid.Column="2" Text="{Binding
ProductDiscountAmount, StringFormat={}(0) %}" VerticalAlignment="Center" />
                            </StackPanel>
                        </Grid>
                    </Border>
                </DataTemplate>
            </ListView.ItemTemplate>
            <ListView.ContextMenu>
                <ContextMenu x:Name="contextMenu">
                          <MenuItem Name="btnAddProduct" Header="Добавить заказ"
Click="btnAddProduct_Click" />
                </ContextMenu>
            </ListView.ContextMenu>
        </ListView>

          <Button x:Name="btnOrder" Grid.Row="3" Grid.Column="0" Content="Заказ"
Visibility="Collapsed" HorizontalAlignment="Right" Width="200" Margin="20 0 0 0"
Click="btnOrder_Click" />
                <Button x:Name="btnAddNewProduct" Grid.Row="3" Grid.Column="1"
Content="Добавить заказ" HorizontalAlignment="Left" Width="200" Margin="20 0 0
0" Click="btnAddNewProduct_Click" />
    </Grid>
</Page>
```

Опишем взаимодействие с этими элементами:

```
public partial class Admin : Page
    {
public Admin(User currentUser)
        {
            InitializeComponent();

            var product = RulEntities.GetContext().Product.ToList();
            lViewProduct.ItemsSource = product;
            DataContext = this;

            txtAllAmount.Text = product.Count().ToString();

            user = currentUser;

            UpdateData();
            User();
        }
private                 void                lViewProduct_MouseDoubleClick(object
sender,System.Windows.Input.MouseButtonEventArgs e)
        {
                                                NavigationService.Navigate(new
AddEditProductPage(lViewProduct.SelectedItems as Product));
        }

        private void btnAddNewProduct_Click(object sender, RoutedEventArgs e)
        {
            NavigationService.Navigate(new AddEditProductPage(null));
        }
///<summary>
/// остальной код в зависимости от других обработчиков
/// </summary>
}
```

Далее рассмотрим изменение и добавление данных при переходе редактирования и удаления продукта.

Разметка страницы выглядит следующим образом:

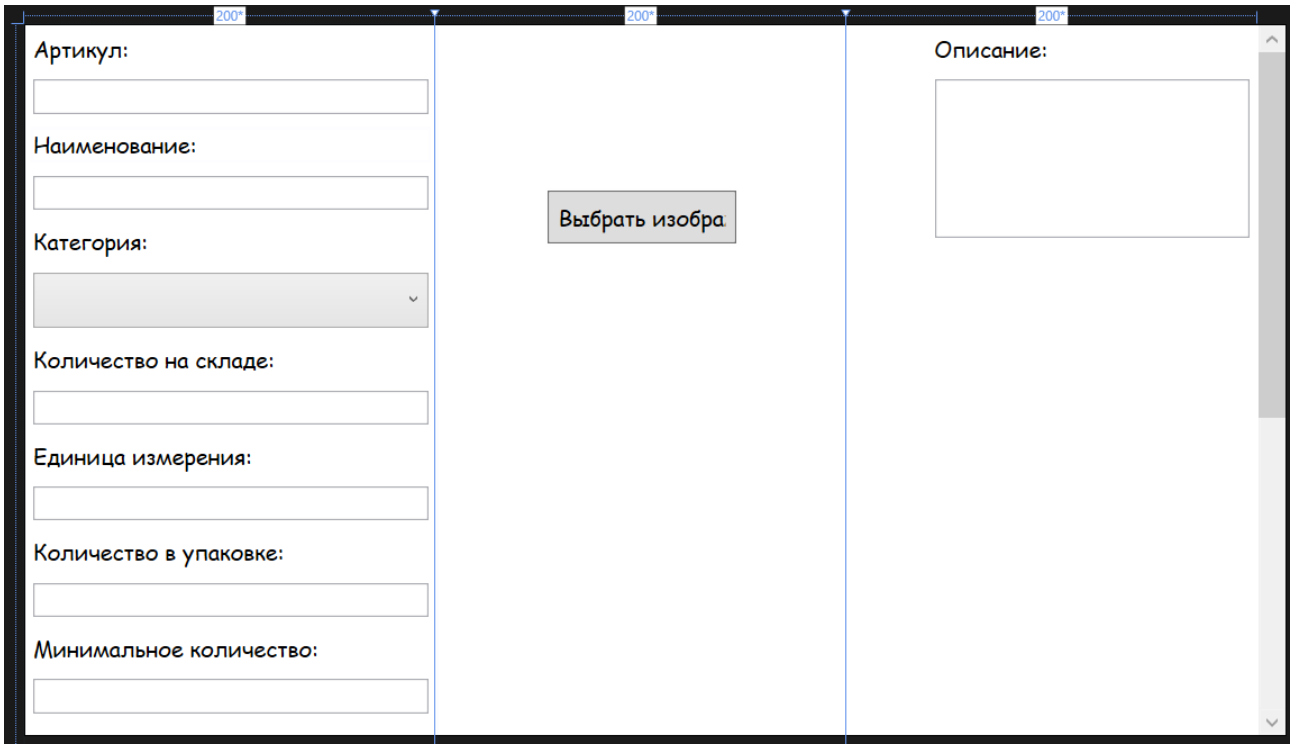

Рисунок 3 — Страница добавления и редактирования товаров

```
<Page x:Class="rul.Pages.AddEditProductPage"
      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="AddEditProductPage">

    <Page.Resources>
        <sys:String x:Key="defaultImage">../Resources/picture.png</sys:String>
    </Page.Resources>

    <ScrollViewer>
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="200*" />
                <ColumnDefinition Width="200*" />
                <ColumnDefinition Width="200*" />
            </Grid.ColumnDefinitions>

            <StackPanel Grid.Column="0">
                <TextBlock Text="Артикул: " />
                                    <TextBox  x:Name="txtArticle"  Text="{Binding
ProductArticleNumber}" />
                <TextBlock Text="Наименование: " />
                <TextBox x:Name="txtTitle" Text="{Binding ProductName}" />
                <TextBlock Text="Категория: " />
```

```
                        <ComboBox x:Name="cmbCategory" SelectedItem="{Binding
ProductCategory}" />
                <TextBlock Text="Количество на складе: " />
                            <TextBox x:Name="txtCountInStock" Text="{Binding
ProductQuantityInStock}" />
                <TextBlock Text="Единица измерения: " />
                <TextBox x:Name="txtUnit" Text="{Binding Unit}" />
                <TextBlock Text="Количество в упаковке: " />
                <TextBox x:Name="txtCountInPack" Text="{Binding CountInPack}" />
                <TextBlock Text="Минимальное количество: " />
                <TextBox x:Name="txtMinCount" Text="{Binding MinCount}" />
                <TextBlock Text="Поставщик: " />
                <TextBox x:Name="txtSupplier" Text="{Binding Supplier}" />
                <TextBlock Text="Максимальная скидка (количество): " />
                             <TextBox x:Name="txtMaxDiscount" Text="{Binding
MaxDiscountAmount}" />
                <TextBlock Text="Максимальная скидка (процент): " />
                          <TextBox x:Name="txtMaxDiscountPercent" Text="{Binding
MaxDiscountPercent}" />
                <TextBlock Text="Скидка (количество): " />
                               <TextBox x:Name="txtDiscount" Text="{Binding
ProductDiscountAmount}" />
                <TextBlock Text="Скидка (процент): " />
                            <TextBox x:Name="txtDiscountPercent" Text="{Binding
ProductDiscountPercent}" />
                <TextBlock Text="Цена: " />
                <TextBox x:Name="txtCost" Text="{Binding ProductCost}" />
            </StackPanel>

            <StackPanel Grid.Column="1">
                  <Image x:Name="img" Width="100" Height="100" Source="{Binding
ImgPath, FallbackValue={StaticResource defaultImage}}" />
                                        <Button Content="Выбрать изображение"
Click="btnSelectImage_Click" />
            </StackPanel>

            <StackPanel Grid.Column="2" Margin="50 0 0 0">
                <TextBlock Text="Описание: " />
                             <TextBox x:Name="txtDescription" Text="{Binding
ProductDescription}" Height="100" Width="200" TextWrapping="Wrap" />
            </StackPanel>

               <StackPanel HorizontalAlignment="Center" Orientation="Horizontal"
Grid.ColumnSpan="3" Margin="0 20 0 0" Visibility="Collapsed">
                 <Button x:Name="btnSaveProduct" Content="Сохранить" Width="200"
Click="btnSaveProduct_Click" />
                 <Button x:Name="btnDeleteProduct" Content="Удалить" Width="200"
Click="btnDeleteProduct_Click" />
            </StackPanel>
        </Grid>
    </ScrollViewer>
</Page>
```

Добавление данных в данном случае описывается интерактивнфми элементами, в которые мы можем заносить данные: данные о товаре, изображение, описание товара. Для таких элементов использованы логические построения StackPanel и Binding на элементах.

В разметке необходимо определить текстовые поля, которые будут соответствовать данным для заполнения информации при добавлении/изменении товара. В них с помощью Binding задается названия соответствующего им поля в классе объекта бизнес-логики (дата-класс сущности базы данных).

Для разметки страницы, представленной выше, в классе AddEditProductPage опишем основные элементы взаимодействия.

Код класса:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using Microsoft.Win32;
using rul.Entities;

namespace rul.Pages
{
    /// <summary>
    /// Логика взаимодействия для AddEditProductPage.xaml
    /// </summary>
    public partial class AddEditProductPage : Page
    {
        Product product = new Product();

        public AddEditProductPage(Product currentProduct)
        {
            InitializeComponent();

            if(currentProduct != null)
            {
                product = currentProduct;

                btnDeleteProduct.Visibility = Visibility.Visible;
                txtArticle.IsEnabled = false;
            }
            DataContext = product;
            cmbCategory.ItemsSource = CategoryList;
        }

        public string[] CategoryList =
        {
            "Акксесуары",
            "Автозапчасти",
            "Автосервис",
            "Съёмники подшипников",
```

```
            "Ручные инструменты",
            "Зарядные устройства",
        };

        private void btnEnterImage_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog GetImageDialog = new OpenFileDialog();

            GetImageDialog.Filter = "Файлы изображений: (*.png, *.jpeg, *.jpg), 
*.png, *.jpeg, *.jpg";
            GetImageDialog.InitialDirectory = "D:\\igor\\rul\\rul\\Resources\\";

            if(GetImageDialog.ShowDialog() == true)
            {
                product.ProductImage = GetImageDialog.SafeFileName;
            }
        }

        private void btnDeleteProduct_Click(object sender, RoutedEventArgs e)
        {
            if (MessageBox.Show($"Вы действительно хотите удалить 
{product.ProductName}?", "Внимание", MessageBoxButton.YesNo, 
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                try
                {
                    RulEntities.GetContext().Product.Remove(product);
                    RulEntities.GetContext().SaveChanges();
                    MessageBox.Show("Запись удалена", "Информация", 
MessageBoxButton.OK, MessageBoxImage.Information);
                    NavigationService.GoBack();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message.ToString(), "Ошибка", 
MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
        }

        private void btnSaveProduct_Click(object sender, RoutedEventArgs e)
        {
            StringBuilder errors = new StringBuilder();

            if (product.ProductCost < 0)
                errors.AppendLine("Стоимость не может быть отрицательной!");
            if (product.MinCount < 0)
                errors.AppendLine("Минимальное количество не может быть 
отрицательным!");
            if (product.ProductDiscountAmount > product.MaxDiscountAmount)
                errors.AppendLine("Действующая скидка на товар не может быть 
больше максимальной скидки!");

            if (errors.Length > 0)
            {
                MessageBox.Show(errors.ToString(), "Вывод ошибки");
                return;
            }

            if (product.ProductArticleNumber == null)
                RulEntities.GetContext().Product.Add(product); // добавление 
объекта в БД
            try
            {
```

```
                RulEntities.GetContext().SaveChanges();
                MessageBox.Show("Информация сохранена", "Информация",
MessageBoxButton.OK, MessageBoxImage.Information);
                NavigationService.GoBack(); // переход на предыдущую страницу
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    }
}
```

Создав разделяемый класс логики взаимодействия на страницы, объявляем поле товара, получаемого из объекта бизнес-логики Product.

```
Product product = new Product();
```

В конструкторе класса передаем параметр выбранного товара currentProduct. В конструкторе класса объявляем методы инициализации компонента, условие (если товар не выбран, затмить кнопку удаления товара). Также в конструкторе объявляется ссылка на контекст данных (необходимо для корректной работы Binding-методов) и элементы выпадающего списка категорий товара.

```
public AddEditProductPage(Product currentProduct)
  {
      InitializeComponent();

      if(currentProduct != null)
      {
          product = currentProduct;

          btnDeleteProduct.Visibility = Visibility.Visible;
          txtArticle.IsEnabled = false;
      }
      DataContext = product;
      cmbCategory.ItemsSource = CategoryList;
  }
```

Далее создаем поле CategoryList, в котором будут содержаться элементы списка. (Также можно это описать при помощи обращения к записям о категориях товаров в базе данных, вместо создания списка)

```
public string[] CategoryList =
  {
      "Аксессуары",
      "Автозапчасти",
      "Автосервис",
      "Съёмники подшипников",
```

```
            "Ручные инструменты",
            "Зарядные устройства",
        };
```

Далее описываем методы взаимодействия с элементами вёрстки.

Метод btnEnterImage_Click необходим для выбора изображения. Здесь мы показываем представление диалогового окна с сохранёнными файлами, и вызываем метод получения изображения с фильтрацией расширений файлов изображений и путём к директории, в которой будут храниться изображения.

```
    private void btnEnterImage_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog GetImageDialog = new OpenFileDialog();

            GetImageDialog.Filter = "Файлы изображений: (*.png, *.jpeg, *.jpg), 
*.png, *.jpeg, *.jpg";
            GetImageDialog.InitialDirectory = "D:\\igor\\rul\\rul\\Resources\\";

            if(GetImageDialog.ShowDialog() == true)
            {
                product.ProductImage = GetImageDialog.SafeFileName;
            }
        }
```

Метод btnDeleteProduct_Click необходим для удаления товара (кнопка, которая будет скрыта, пока не выбран продукт). В данном случае описывается конструкция использования предупреждения в MessageBox, а также конструкция обработки исключений Try-Catch-Finally. В конструкции обработки исключений при положительном исходе удаляется элемент, сохраняется состояние и происходит возвращение к исходной странице AddEditProductPage. В противном случае будет выведена ошибка обработки действий по нажатию.

```
    private void btnDeleteProduct_Click(object sender, RoutedEventArgs e)
        {
            if (MessageBox.Show($"Вы действительно хотите удалить 
{product.ProductName}?", "Внимание", MessageBoxButton.YesNo, 
MessageBoxImage.Warning) == MessageBoxResult.Yes)
            {
                try
                {
                    RulEntities.GetContext().Product.Remove(product);
                    RulEntities.GetContext().SaveChanges();
                    MessageBox.Show("Запись удалена", "Информация", 
MessageBoxButton.OK, MessageBoxImage.Information);
                    NavigationService.GoBack();
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message.ToString(), "Ошибка",
```

```
MessageBoxButton.OK, MessageBoxImage.Error);
                }
            }
        }
```

Метод btnSaveProduct_Click необходим для сохранения данных о товаре. В данном случае будет работать добавление данных в базу через поле контекста.

```
      private void btnSaveProduct_Click(object sender, RoutedEventArgs e)
        {
            StringBuilder errors = new StringBuilder();

            if (product.ProductCost < 0)
                errors.AppendLine("Стоимость не может быть отрицательной!");
            if (product.MinCount < 0)
                errors.AppendLine("Минимальное количество не может быть
отрицательным!");
            if (product.ProductDiscountAmount > product.MaxDiscountAmount)
                errors.AppendLine("Действующая скидка на товар не может быть
больше максимальной скидки!");

            if (errors.Length > 0)
            {
                MessageBox.Show(errors.ToString(), "Вывод ошибки");
                return;
            }

            if (product.ProductArticleNumber == null)
                RulEntities.GetContext().Product.Add(product); // добавление
объекта в БД
            try
            {
                RulEntities.GetContext().SaveChanges();
                MessageBox.Show("Информация сохранена", "Информация",
MessageBoxButton.OK, MessageBoxImage.Information);
                NavigationService.GoBack(); // переход на предыдущую страницу
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message.ToString(), "Ошибка",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
```

После обновления данных необходимо применить это самое обновление. Реализация такого метода может быть выполнена следующим образом:

```
private void Page_IsVisibleChanged(object sender,
DependencyPropertyChangedEventArgs e)
        {
            if(Visibility == Visibility.Visible)
            {
            // логика получения (обновления) данных в ListView
 RulEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p =>
p.Reload());
```

```
            // задаем коллекцию для создания содержимого в ListView
            lViewProduct.ItemsSource = RulEntities.GetContext().Product.ToList();
            }
        }
```

В данном случае используется обработчик события IsVisibleChanged. На вёрстке это выглядит так:

```
<Page x:Class="rul.Pages.Admin"
      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="Admin" IsVisibleChanged="Page_IsVisibleChanged">
// остальная вёрстка страницы
```