

Министерство науки и высшего образования Российской Федерации
Севастопольский государственный университет
Кафедра ИС

Отчет
по лабораторной работе №3
«Исследование разработки GUI. Создание sdі-приложений. Обработка событий»
по дисциплине
«ПЛАТФОРМА .NET»

Выполнил студент группы ИС/б-17-2-о

Горбенко К. Н.

Проверил

Забаштанский А.К.

Севастополь
2019

1 ЦЕЛЬ РАБОТЫ

1. Изучить принципы разработки графического интерфейса приложений для ОС Windows в Visual Studio .Net;
2. Освоить использование элементов графического интерфейса для управления работой приложения;
3. Освоить принципы построения иерархических меню, создания диалоговых окон;
4. Изучить модель обработки событий в языке C#.

2 ЗАДАНИЕ НА РАБОТУ

1. Создать SDI-приложение (Single Document Interface, однодокументный интерфейс) с элементами ввода и отображения полей класса из задания к лабораторной работе 2. Для этого используйте различные элементы управления: текстовые поля, списки, независимые и радиокнопки, а также панели и менеджеры компоновки.
2. Ввод новых данных осуществлять через дополнительную диалоговую форму.
3. При изменении данных запрашивать подтверждение через окно диалога. В случае неполных данных сообщать об ошибке.
4. Объекты сохранять в коллекции.
5. Реализовать просмотр всей коллекции объектов через список. Для редактирования выбранного объекта создать дополнительную форму модального диалога.
6. Добавить на форму меню, позволяющее работать с пунктами: добавить, просмотреть, удалить, редактировать, справка.
7. Дублировать основные операции панелью инструментов.

Для варианта № 3 задана следующая схема данных: **ПЕЧАТНОЕ ИЗДАНИЕ: название, ФИО автора, стоимость**

3 ХОД РАБОТЫ

Создадим WPF проект на .NET Core 3.0, создадим главную страницу приложения. Точкой входа приложения является окно App.xaml. Разработку будем

осуществлять с помощью паттерна MVVM, поддерживаемого WPF. Зададим в нем ссылку на главную страницу:

```
1 <Application x:Class="PrintedEditionSdiApp.App"
2           xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3           xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4           StartupUri="Views/MainWindow.xaml">
5 </Application>
```

3.1 Главная страница

Создадим главную страницу MainWindow.xaml. В ней зададим представление главной страницы - PrintedEditionControl, размеры окна. Свойство ResizeMode в данном случае используется для запрета возможности изменения размеров окна.

```
1 <Window x:Class="PrintedEditionSdiApp.Views.MainWindow"
2       ...
3       xmlns:views="clr-namespace:PrintedEditionSdiApp.Views"
4       Title="PrintedEditionMainWindow"
5       ResizeMode="NoResize"
6       Width="400" Height="500">
7
8 <views:PrintedEditionControl/>
9 </Window>
```

Опишем представление PrintedEditionControl. Оно включает StackPanel. StackPanel, в свою очередь, включает ListView со списком доступных печатных изданий. Под списком находится три текстовых поля, предоставляющих информацию о выбранном печатном издании. Поля заблокированы с помощью свойства IsEnabled т.к. редактирование должно осуществляться в отдельном окне. Список изданий привязан к коллекции view модели, соответствующей этой странице. Выбранное издание привязано к свойству view модели. Все привязки осуществляются с помощью биндингов MVVM.

```
1 <Page x:Class="PrintedEditionSdiApp.Views.PrintedEditionControl"
2       ...
3       mc:Ignorable="d"
4       Title="PrintedEditionMainWindow"
5       d:DataContext="{d:DesignInstance viewModels:PrintedEditionViewModel}">
6
7 <StackPanel>
8     <StackPanel>
9         <Label HorizontalAlignment="Center">Printed Editions</Label>
10        <ListView ItemsSource="{Binding PrintedEditions}" SelectedItem="{
            Binding SelectedPrintedEdition}">
```

```

11         ScrollViewer.VerticalScrollBarVisibility="Visible"
12         Height="250" Margin="5">
13     <ListView.Resources>
14         <Style TargetType="ListViewItem">
15             <Style.Triggers>
16                 <Trigger Property="IsKeyboardFocusWithin" Value="
17                     True">
18                     <Setter Property="IsSelected" Value="True"/>
19                 </Trigger>
20             </Style.Triggers>
21         </Style>
22     </ListView.Resources>
23     <ListView.ItemTemplate>
24         <DataTemplate>
25             <Grid>
26                 <Grid.ColumnDefinitions>
27                     <ColumnDefinition Width="240"/>
28                     <ColumnDefinition Width="50" />
29                     <ColumnDefinition Width="50" />
30                 </Grid.ColumnDefinitions>
31                 <StackPanel Grid.Column="0" Margin="5">
32                     <TextBlock Text="{Binding Name}" />
33                     <TextBlock Text="{Binding Author}" />
34                     <TextBlock Text="{Binding Price}" />
35                 </StackPanel>
36                 <Button Grid.Column="1"
37                     Click="EditPrintedEditionButtonClick"
38                     Margin="5" Content="Edit"/>
39                 <Button Grid.Column="2"
40                     Command="{Binding Path=DataContext.
41                         DeleteCommand,
42                         RelativeSource={RelativeSource
43                             FindAncestor,
44                             AncestorType={x:Type Page}}}"
45                     Margin="5" Content="Delete"/>
46             </Grid>
47         </DataTemplate>
48     </ListView.ItemTemplate>
49 </ListView>
50 <Button Click="AddPrintedEditionButtonClick" Margin="5">Add</Button
51 >
52 </StackPanel>
53 <StackPanel IsEnabled="False" DataContext="{Binding
    SelectedPrintedEdition}" Margin="5">
54     <Label HorizontalAlignment="Center">Selected Edition</Label>
55     <TextBlock Text="Name" />
56     <TextBox Text="{Binding Name, UpdateSourceTrigger=PropertyChanged}"

```

```

        />
54     <TextBlock Text="Author" />
55     <TextBox Text="{Binding Author, UpdateSourceTrigger=PropertyChanged
        }" />
56     <TextBlock Text="Price" />
57     <TextBox Text="{Binding Price, UpdateSourceTrigger=PropertyChanged
        }" />
58     </StackPanel>
59 </StackPanel>
60 </Page>

```

Вид главной страницы:

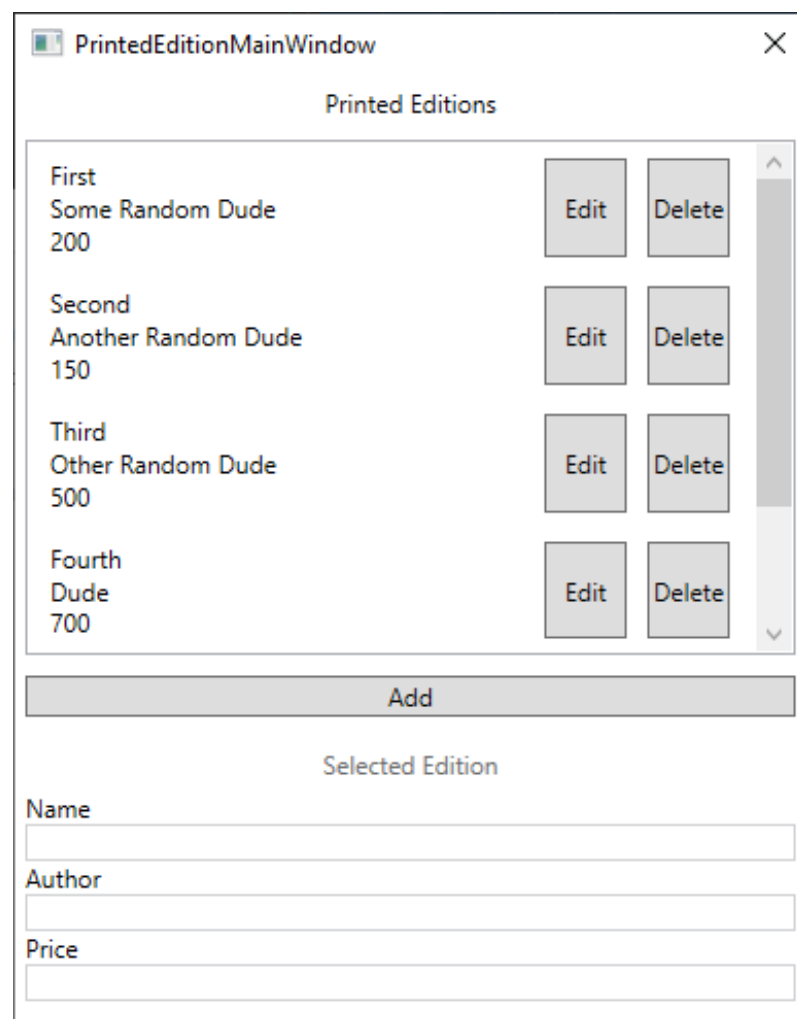


Рисунок 1 – Вид главного окна приложения

Создадим соответствующую этой странице вью модель. Она содержит коллекцию `PrintedEditions`, свойство `SelectedPrintedEdition`, методы для добавления, изменения и удаления печатных изданий. Любое изменение любого свойства вью модели приведет к триггеру события `PropertyChanged`. Это событие

оповестит представление о том, что какое-то свойство изменилось, что приведет к обновлению представления. Методы редактирования и добавления печатного издания используется другими вью моделями.

```

1 public class PrintedEditionViewModel : INotifyPropertyChanged
2 {
3     private static PrintedEditionViewModel instance;
4     private PrintedEdition selectedPrintedEdition;
5     private ICommand deleteCommand;
6
7     public event PropertyChangedEventHandler PropertyChanged;
8     public ObservableCollection<PrintedEdition> PrintedEditions { get; set; }
9
10    public PrintedEdition SelectedPrintedEdition
11    {
12        get => selectedPrintedEdition;
13        set
14        {
15            selectedPrintedEdition = value;
16            OnPropertyChanged(nameof(SelectedPrintedEdition));
17        }
18    }
19
20    public ICommand DeleteCommand =>
21        deleteCommand ??= new RelayCommand(
22            obj => RemovePrintedEdition(selectedPrintedEdition),
23            obj => true);
24
25    [NotifyPropertyChangedInvocator]
26    protected virtual void OnPropertyChanged([CallerMemberName] string
        propertyName = null)
27    {
28        PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
29    }
30
31    public static PrintedEditionViewModel GetInstance()
32    {
33        return instance ??= new PrintedEditionViewModel();
34    }
35
36    private PrintedEditionViewModel()
37    {
38        PrintedEditions = new ObservableCollection<PrintedEdition>
39        {
40            new PrintedEdition { Id = 1, Name = "First", Author = "Some Random
                Dude", Price = 200},
41            new PrintedEdition { Id = 2, Name = "Second", Author = "Another

```

```

        Random Dude", Price = 150},
42     new PrintedEdition { Id = 3, Name = "Third", Author = "Other Random
        Dude", Price = 500},
43     new PrintedEdition { Id = 4, Name = "Fourth", Author = "Dude",
        Price = 700}
44 };
45 }
46
47 public void AddPrintedEdition(PrintedEdition printedEdition)
48 {
49     if (printedEdition == null)
50     {
51         throw new ArgumentNullException(nameof(printedEdition));
52     }
53
54     if (printedEdition.Id == 0)
55     {
56         printedEdition.Id = GetUniqueId();
57     }
58     PrintedEditions.Add(printedEdition);
59 }
60
61 public void RemovePrintedEdition(PrintedEdition printedEdition)
62 {
63     if (printedEdition == null)
64     {
65         throw new ArgumentNullException(nameof(printedEdition));
66     }
67
68     if (PrintedEditions.Contains(printedEdition))
69     {
70         PrintedEditions.Remove(printedEdition);
71     }
72 }
73
74 public void ReplacePrintedEdition(PrintedEdition printedEdition)
75 {
76     if (printedEdition == null)
77     {
78         throw new ArgumentNullException(nameof(printedEdition));
79     }
80
81     var oldPrintedEdition = PrintedEditions.FirstOrDefault(x => x.Id ==
        printedEdition.Id);
82     if (oldPrintedEdition != null)
83     {
84         oldPrintedEdition.Name = printedEdition.Name;
85         oldPrintedEdition.Author = printedEdition.Author;

```

```

86         oldPrintedEdition.Price = printedEdition.Price;
87     }
88 }
89
90 private int GetUniqueId()
91 {
92     var generator = new Random();
93     var uniqueId = 0;
94
95     while (PrintedEditions.FirstOrDefault(x => x.Id == uniqueId) != null)
96     {
97         uniqueId = generator.Next();
98     }
99
100     return uniqueId;
101 }
102 }

```

3.2 Окно создания

Создадим окно создания печатного издания и его вью модель:

```

1 <Window x:Class="PrintedEditionSdiApp.Views.AddPrintedEditionWindow"
2     ...
3     xmlns:viewModels="clr-namespace:PrintedEditionSdiApp.ViewModels"
4     Title="PrintedEditionMainWindow"
5     Width="400" Height="210"
6     d:DataContext="{d:DesignInstance viewModels:AddPrintedEditionViewModel}">
7
8     <StackPanel>
9         <StackPanel DataContext="{Binding PrintedEditionToAdd}" Margin="5">
10             <Label HorizontalAlignment="Center">Enter printed edition details</
11                 Label>
12             <TextBlock Text="Name" />
13             <TextBox Text="{Binding Name}" />
14             <TextBlock Text="Author" />
15             <TextBox Text="{Binding Author}" />
16             <TextBlock Text="Price" />
17             <TextBox Text="{Binding Price}" />
18         </StackPanel>
19         <Button Command="{Binding AddPrintedEditionCommand}" Click="
20             AddButtonClick" Margin="5" Content="Add">
21 </Button>
22 </StackPanel>
23 </Window>

```

```

1 public class AddPrintedEditionViewModel : INotifyPropertyChanged
2 {

```



```

3     private PrintedEdition printedEditionToAdd;
4     private readonly PrintedEditionViewModel printedEditionViewModel;
5     private ICommand addPrintedEditionCommand;
6
7     public event PropertyChangedEventHandler PropertyChanged;
8     public PrintedEdition PrintedEditionToAdd
9     {
10         get => printedEditionToAdd;
11         set
12         {
13             printedEditionToAdd = value;
14             OnPropertyChanged(nameof(PrintedEditionToAdd));
15         }
16     }
17
18     public ICommand AddPrintedEditionCommand =>
19         addPrintedEditionCommand ??=
20         new RelayCommand(
21             obj => AddPrintedEdition(printedEditionToAdd),
22             obj => true);
23
24     [NotifyPropertyChangedInvocator]
25     protected virtual void OnPropertyChanged([CallerMemberName] string
        propertyName = null)
26     {
27         PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
28     }
29
30     public AddPrintedEditionViewModel()
31     {
32         printedEditionToAdd = new PrintedEdition();
33         printedEditionViewModel = PrintedEditionViewModel.GetInstance();
34     }
35
36     private void AddPrintedEdition(PrintedEdition printedEdition)
37     {
38         if (printedEdition == null)
39         {
40             throw new ArgumentNullException(nameof(printedEdition));
41         }
42
43         printedEditionViewModel.AddPrintedEdition(printedEdition);
44     }
45 }

```

Вид диалога добавления печатного издания:

Рисунок 2 – Вид диалога добавления печатного издания

3.3 Диалог редактирования

Создадим диалог редактирования и его вью модель:

```

1 <Window x:Class="PrintedEditionSdiApp.Views.EditPrintedEditionWindow"
2     ...
3     xmlns:viewModels="clr-namespace:PrintedEditionSdiApp.ViewModels"
4     Title="PrintedEditionEditFormWindow" Width="400" Height="210"
5     d:DataContext="{d:DesignInstance viewModels:EditPrintedEditionViewModel}">
6
7     <StackPanel>
8         <StackPanel DataContext="{Binding PrintedEditionToEdit}" Margin="5">
9             <Label HorizontalAlignment="Center">Enter printed edition details</
10                Label>
11             <TextBlock Text="Name" />
12             <TextBox Text="{Binding Name}" />
13             <TextBlock Text="Author" />
14             <TextBox Text="{Binding Author}" />
15             <TextBlock Text="Price" />
16             <TextBox Text="{Binding Price}" />
17         </StackPanel>
18         <Button Command="{Binding EditPrintedEdition}" Click="EditButtonClick"
19             Margin="5">Edit</Button>
20     </StackPanel>
21 </Window>

```

```

1 public class EditPrintedEditionViewModel : INotifyPropertyChanged
2 {
3     private PrintedEdition printedEditionToEdit;
4     private readonly PrintedEditionViewModel printedEditionViewModel;
5     private ICommand editPrintedEdition;
6
7     public event PropertyChangedEventHandler PropertyChanged;
8
9     public PrintedEdition PrintedEditionToEdit

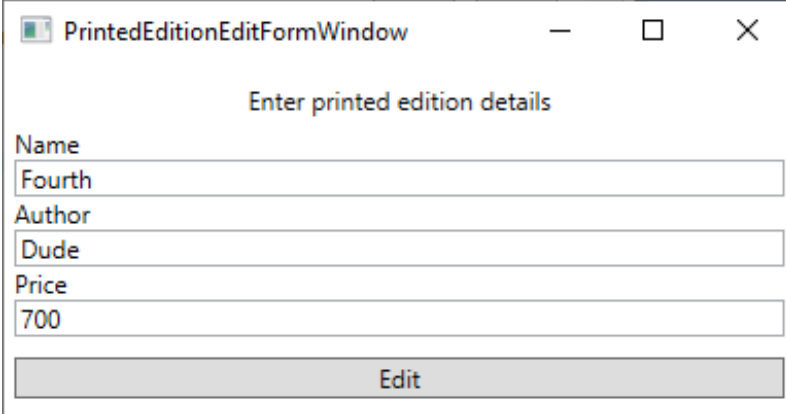
```

```

10     {
11         get => printedEditionToEdit;
12         set
13         {
14             printedEditionToEdit = value;
15             OnPropertyChanged(nameof(PrintedEditionToEdit));
16         }
17     }
18
19     public ICommand EditPrintedEdition =>
20         editPrintedEdition ??= new RelayCommand(
21             obj => ReplacePrintedEdition(printedEditionToEdit),
22             obj => true);
23
24     public EditPrintedEditionViewModel(PrintedEdition printedEdition)
25     {
26         printedEditionToEdit = printedEdition ?? throw new
27             ArgumentNullException(nameof(printedEdition));
28         printedEditionViewModel = PrintedEditionViewModel.GetInstance();
29     }
30     [NotifyPropertyChangedInvocator]
31     protected virtual void OnPropertyChanged([CallerMemberName] string
32         propertyName = null)
33     {
34         PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
35     }
36
37     private void ReplacePrintedEdition(PrintedEdition printedEdition)
38     {
39         if (printedEdition == null)
40         {
41             throw new ArgumentNullException(nameof(printedEdition));
42         }
43         printedEditionViewModel.ReplacePrintedEdition(printedEdition);
44     }
45 }

```

Вид диалога редактирования печатного издания:



PrintedEditionEditFormWindow

Enter printed edition details

Name
Fourth

Author
Dude

Price
700

Edit

Рисунок 3 – Вид диалога редактирования печатного издания

ВЫВОДЫ

В ходе лабораторной работы были изучены основы создания десктопных приложений под Windows на платформе .NET. Для создания приложений на WPF используется паттерн MVVM, основанный на разделении представления приложения от его логики. Реализовывается паттерн за счет событий платформы .NET.

Для создания представления используются классы разметки с расширением `xaml`. Каждый класс может содержать только один элемент. Для того, чтобы поместить на странице несколько элементов, их необходимо объединить в один с помощью высокоуровневых контролов: `StackPanel`, `Grid`, `WrapPanel` и других.