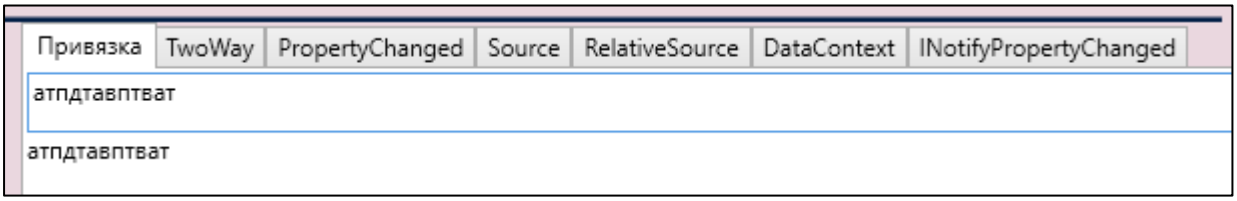
Лабораторная работа № 4.

Тема: «Работа с привязкой»

Задание 1. Проработать примеры из теоретического материала.

1. Обычная привязка

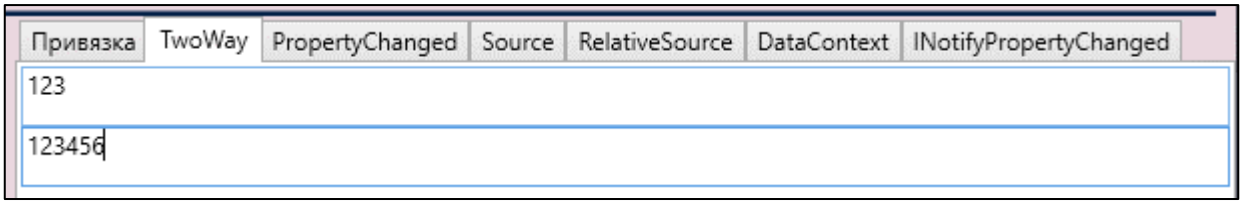
```
<TabItem Header="Привязка">
    <StackPanel>
        <TextBox x:Name="myTextBox" Height="30"/>
        <TextBlock x:Name="myTextBlock" Text="{Binding ElementName=myTextBox,Path=Text}" Height="30"/>
    </StackPanel>
</TabItem>
```



2. TwoWay

```
<TabItem Header="TwoWay">
    <StackPanel>
        <TextBox x:Name="textBox1" Height="30"/>
        <TextBox x:Name="textBox2" Height="30" Text="{Binding ElementName=textBox1, Path=Text, Mode=TwoWay}"/>
    </StackPanel>
</TabItem>
```

Двусторонняя привязка в WPF позволяет привязать свойство в пользовательском интерфейсе к свойству в вашей модели представления или коде позади, чтобы изменения, сделанные в одном месте, автоматически отражались в другом.

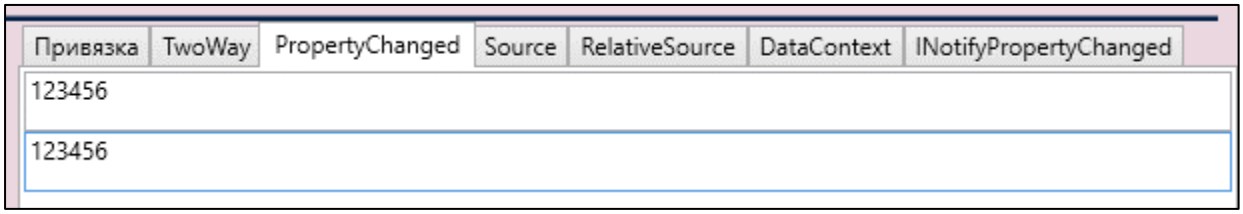


3. PropertyChanged

```
<TabItem Header="PropertyChanged">
    <StackPanel>
        <TextBox x:Name="textBox3" Height="30"/>
        <TextBox x:Name="textBox4" Height="30" Text="{Binding ElementName=textBox3, Path=Text, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}"/>
    </StackPanel>
</TabItem>
```

					ККЭП 09.02.07 0053 Оm		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Бреус А.Д.			Лабораторная работа № 4.	Лит.	Лист 1 / Листов 10
Провер.		Шостак А.И..				у	
							Гр. 22-Д9-ЗИНС

“PropertyChanged” событие является частью “INotifyPropertyChanged” интерфейса и используется в WPF для уведомления пользовательского интерфейса об изменении значения свойства в модели представления или коде позади. Когда “PropertyChanged” возникает событие, WPF обновляет привязку, отражая изменения в пользовательском интерфейсе.



4. Source

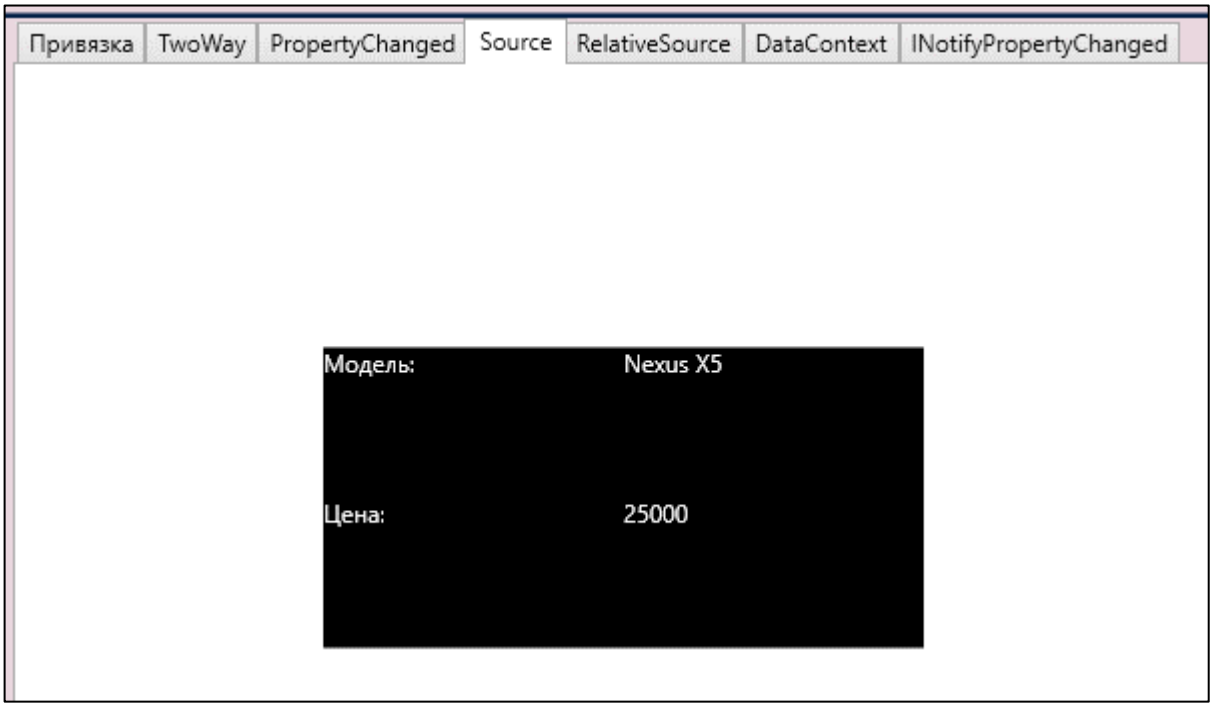
В WPF “StaticResource” привязка используется для привязки свойства элемента пользовательского интерфейса к статическому ресурсу, определенному в приложении. Привязка “StaticResource” часто используется для привязки свойства элемента пользовательского интерфейса к ресурсу, который используется во всем приложении и не изменяется в течение всего времени существования приложения.

```
<TabItem Header="Source">
    <Grid Background="Black" Height="150" Width="300">
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <TextBlock Text="Модель:" Foreground="White"/>
        <TextBlock x:Name="titleTextBlock" Text="{Binding Source={StaticResource nexusPhone}, Path=Title}" Foreground="White" Grid.Column="1"/>
        <TextBlock Text="Цена:" Foreground="White" Grid.Row="1"/>
        <TextBlock x:Name="priceTextBlock" Text="{Binding Source={StaticResource nexusPhone}, Path=Price}" Foreground="White" Grid.Column="1" Grid.Row="1"/>
    </Grid>
</TabItem>
```

Определяем ресурс в коде XAML.

```
<Page.Resources>
    <local:Phone x:Key="nexusPhone" Title="Nexus X5" Company="Google" Price="25000"/>
</Page.Resources>
```

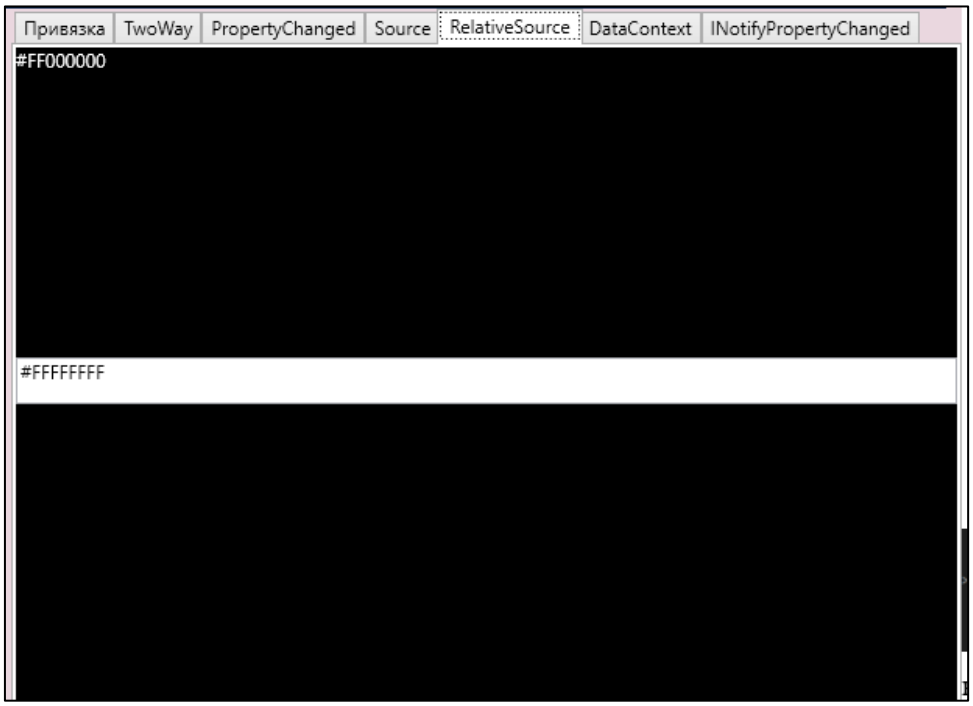
С этой настройкой при каждом “TextBlock” рендеринге будет отображаться значение “nexusPhone” ресурса. Привязка “StaticResource” гарантирует, что значение ресурса используется во всем приложении и что элемент пользовательского интерфейса всегда обновляется последним значением ресурса.



5. RelativeSource

```
<TabItem Header="RelativeSource">
    <Grid Background="Black">
        <TextBox Text="{Binding RelativeSource={RelativeSource Mode=Self}, Path=Background, Mode=TwoWay, UpdateSourceTrigger=PropertyChanged}" Height="30"/>
        <TextBlock Foreground="White" Text="{Binding RelativeSource={RelativeSource Mode=FindAncestor, AncestorType={x:Type Grid}}, Path=Background}"/>
    </Grid>
</TabItem>
```

“RelativeSource” привязка используется для привязки свойства элемента пользовательского интерфейса к источнику, который относится к положению элемента в визуальном дереве.



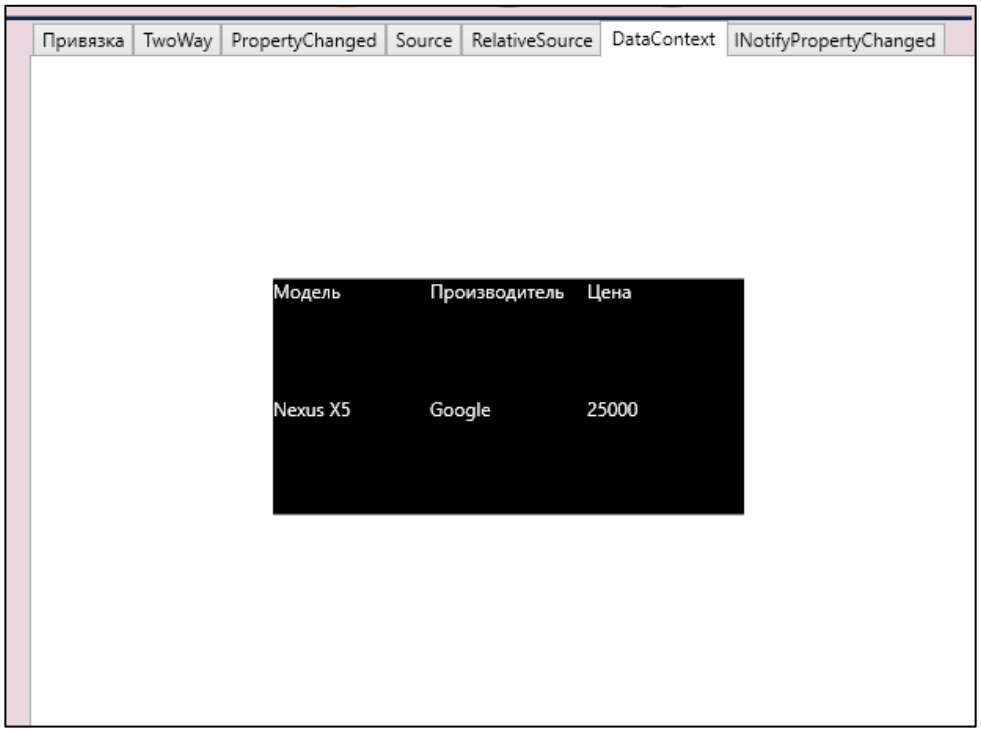
6. DataContext

```
<TabItem Header="DataContext">
    <Grid Background="Black" DataContext="{StaticResource nexusPhone}" TextBlock.Foreground="White" Height="150" Width="300">
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
        <TextBlock Text="Модель"/>
        <TextBlock Text="{Binding Title}" Grid.Row="1"/>
        <TextBlock Text="Производитель" Grid.Column="1"/>
        <TextBlock Text="{Binding Company}" Grid.Column="1" Grid.Row="1"/>
        <TextBlock Text="Цена" Grid.Column="2"/>
        <TextBlock Text="{Binding Price}" Grid.Column="2" Grid.Row="1"/>
    </Grid>
</TabItem>
```

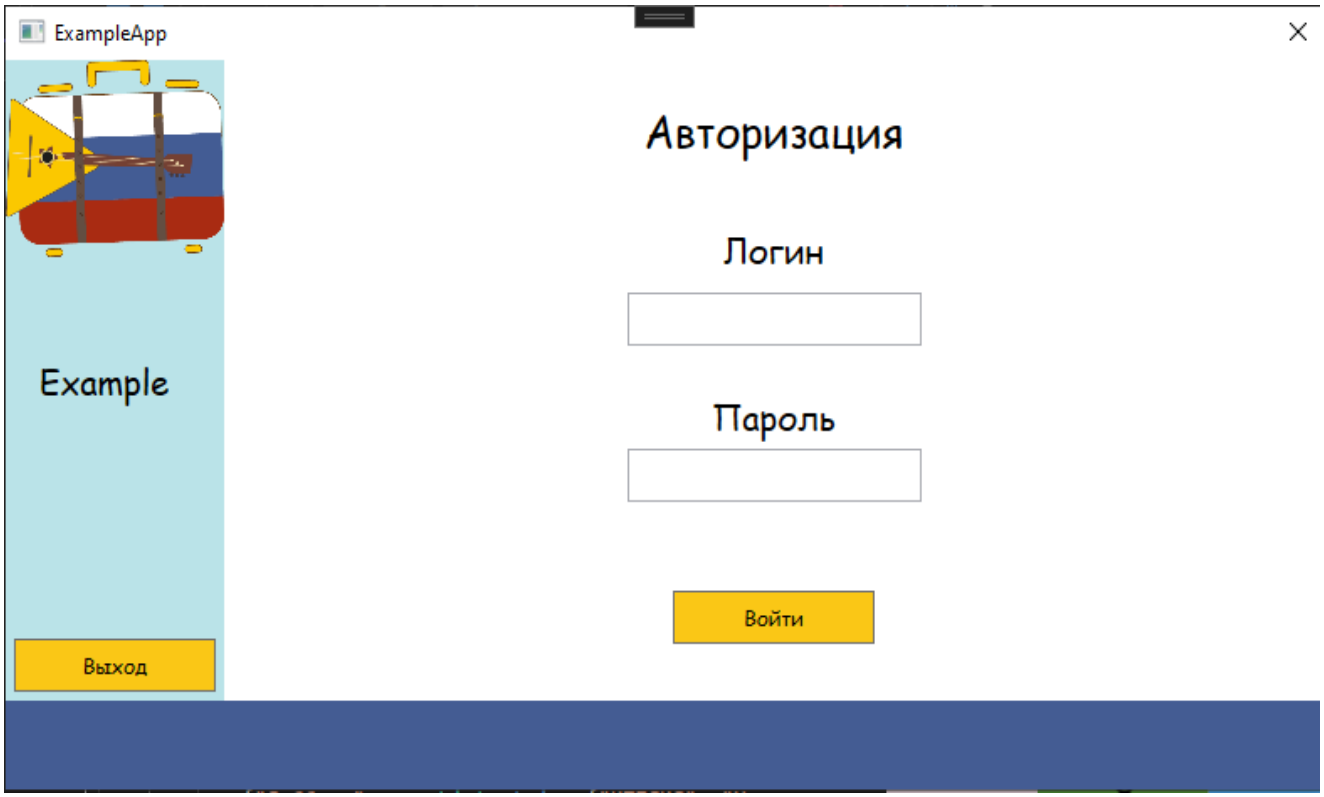
“DataContext” привязка используется для указания источника данных для привязки данных в элементе пользовательского интерфейса.

```
<Grid Background="Black" DataContext="{StaticResource nexusPhone}" TextBlock.Foreground="White" Height="150" Width="300">
```

“DataContext” для параметра “StackPanel” устанавливается “nexusPhone” ресурс, использующий “StaticResource” привязку “Text”.



Задание 2. Проработать каркас приложения из файла.



XAML код главной страницы.

```
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="*"/>
        <RowDefinition Height="50"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="130"/>
        <ColumnDefinition Width="*"/>
    </Grid.ColumnDefinitions>
    <Grid Grid.Row="0" Grid.Column="0" Background="#bae3e8"/>
    <Grid Grid.Row="1" Grid.Column="0" Background="#445c93"/>
    <Grid Grid.Row="1" Grid.Column="1" Background="#445c93"/>
    <Frame Name="mainFrame" Grid.Row="0" Grid.Column="1" NavigationUIVisibility="Hidden" ContentRendered="mainFrame_ContentRendered"></Frame>
    <Image Source="/Resources/logo.png" VerticalAlignment="Top"/>
    <TextBlock Text="Example" Width="90" HorizontalAlignment="Center" VerticalAlignment="Center" FontSize="20" Grid.Row="0" Grid.Column="0"/>
    <StackPanel VerticalAlignment="Bottom">
        <Button Name="btnBack" VerticalAlignment="Bottom" Content="Назад" Click="btnBack_Click"/>
        <Button Name="btnExit" VerticalAlignment="Bottom" Content="Выход" Click="btnExit_Click"/>
    </StackPanel>
</Grid>
```

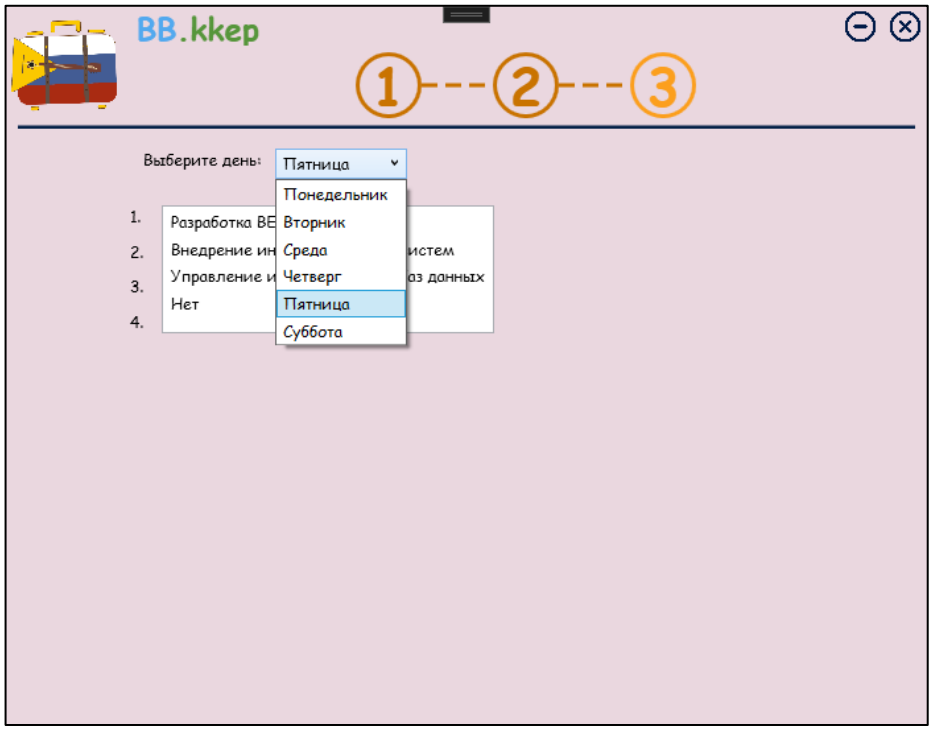
XAML код страницы авторизации.

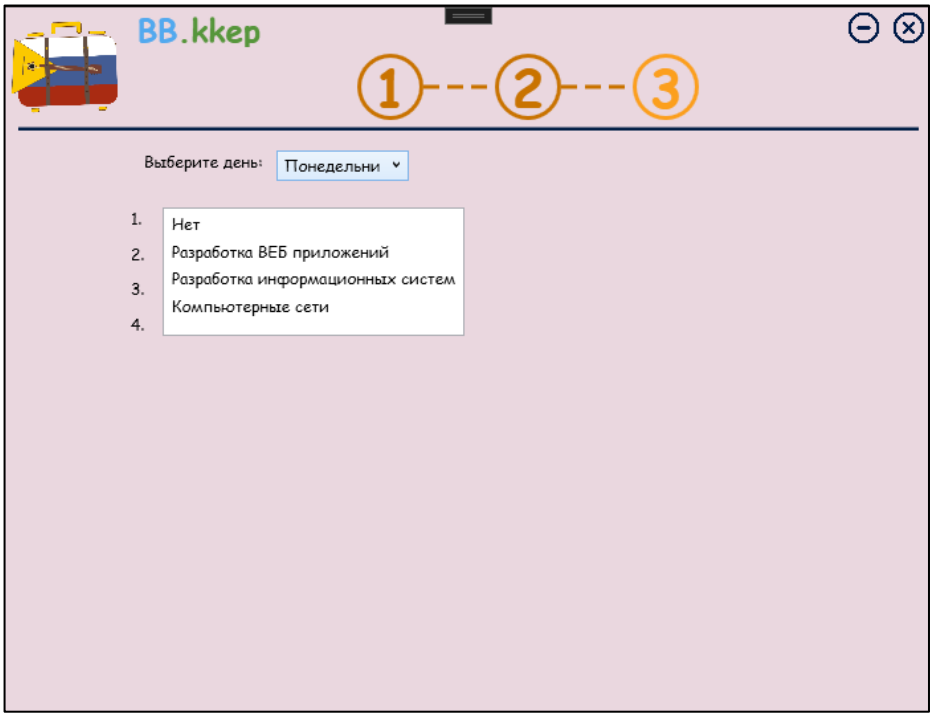
```
  Title="AuthorizationPage">
<Grid>
    <Grid.RowDefinitions>
        <RowDefinition Height="80"/>
        <RowDefinition/>
        <RowDefinition/>
        <RowDefinition/>
    </Grid.RowDefinitions>
    <TextBlock Text="Авторизация" FontSize="25" HorizontalAlignment="Center" VerticalAlignment="Center"/>
    <StackPanel HorizontalAlignment="Center" Grid.Row="1" Orientation="Vertical" VerticalAlignment="Center">
        <TextBlock Text="Логин" FontSize="20" HorizontalAlignment="Center" Margin="5"/>
        <TextBox Name="txtLogin"/>
    </StackPanel>
    <StackPanel HorizontalAlignment="Center" Grid.Row="2" Orientation="Vertical" VerticalAlignment="Center">
        <TextBlock Text="Пароль" FontSize="20" HorizontalAlignment="Center"/>
        <TextBox Name="txtBoxPassword" Visibility="Collapsed"/>
        <PasswordBox Name="txtPassword" Width="175" Height="30" Margin="5"/>
    </StackPanel>
    <Button Content="Войти" Grid.Row="4" Name="btnEnter" Click="btnEnter_Click"/>
</Grid>
```

C# код.

```
public ExampleApp()
{
    InitializeComponent();
    mainFrame.Navigate(new AuthorizationPage());
    Manager.mainFrame = mainFrame;
}

Ссылок: 1
private void btnExit_Click(object sender, RoutedEventArgs e)
{
    this.Close();
}

Ссылок: 1
private void btnBack_Click(object sender, RoutedEventArgs e)
{
    Manager.mainFrame.GoBack();
}

Ссылок: 1
private void mainFrame_ContentRendered(object sender, EventArgs e)
{
    if (Manager.mainFrame.CanGoBack)
    {
        this.ResizeMode = ResizeMode.CanResize;
        btnBack.Visibility = Visibility.Visible;
    }
    else
    {
        this.ResizeMode=ResizeMode.NoResize;
        btnBack.Visibility=Visibility.Collapsed;
    }
}
```

Задание 3. Система вуза «Расписание на неделю»





XAML код задания 3:

```
  Title="Exercise3">

<Grid Background="Transparent">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
    </Grid.RowDefinitions>
    <StackPanel Orientation="Horizontal" Margin="10">
        <Label Content="Выберите день:"/>
        <ComboBox x:Name="DayComboBox" Width="100" Margin="5" SelectionChanged="DayComboBox_SelectionChanged"/>
    </StackPanel>
    <StackPanel Orientation="Horizontal" Grid.Row="1">
        <StackPanel Orientation="Vertical">
            <Label Content="1. "/>
            <Label Content="2. "/>
            <Label Content="3. "/>
            <Label Content="4. "/>
        </StackPanel>
        <ListBox x:Name="ScheduleListBox" Margin="5"/>
    </StackPanel>
</Grid>
```

C# Код:

```
public partial class Exercise3 : Page
{
    private Dictionary<string, List<string>> schedule = new Dictionary<string, List<string>>()
    {
        {"Понедельник", new List<string>{"Нет", "Разработка ВЕБ приложений", "Разработка информационных систем", "Компьютерные сети"}},
        {"Вторник", new List<string>{ "Ин.язык в проф.деятельности", "Внедрение информационных систем", "Разработка ВЕБ приложений", "Нет"}},
        {"Среда", new List<string>{ "Разработка ВЕБ приложений", "Разработка информационных систем", "Тестирование информационных систем", "Нет"}},
        {"Четверг", new List<string>{ "Интеллектуальные системы и технологии", "Управление и автоматизация баз данных", "Разработка ВЕБ приложений", "Нет"}},
        {"Пятница", new List<string>{ "Разработка ВЕБ приложений", "Внедрение информационных систем", "Управление и автоматизация баз данных", "Нет"}},
        {"Суббота", new List<string>{"ИТПСИС", "Управление и автоматизация баз данных", "Интеллектуальные системы и технологии", "Нет"}}
    };
    Ссылок: 1
    public Exercise3()
    {
        InitializeComponent();
        PopulateDayComboBox();
    }
    Ссылок: 1
    private void PopulateDayComboBox()
    {
        foreach (string day in schedule.Keys)
        {
            DayComboBox.Items.Add(day);
        }
        DayComboBox.SelectedIndex = 0;
    }

    Ссылок: 1
    private void DayComboBox_SelectionChanged(object sender, RoutedEventArgs e)
    {
        string selectedDay = DayComboBox.SelectedItem.ToString();
        ScheduleListBox.ItemsSource = schedule[selectedDay];
    }
}
```

Программа будет состоять из базы данных предметов, преподавателей и аудиторий. Она позволит администраторам составлять расписание на каждую неделю, распределяя предметы по определенным аудиториям и временным интервалам, а также назначая преподавателей для преподавания каждого курса. Программа также сможет отображать расписание на неделю в удобном для студентов и преподавателей формате. Расписание будет редактируемым, что позволит вносить изменения по мере необходимости в течение семестра. В программу также будет включена функция создания отчетов о расписании курсов, наличии аудиторий и назначении преподавателей.

Общие требования

При создании приложения руководствуйтесь требованиями, описанными в документе
«Требования_и_рекомендации_Каркас_Приложения_КЗ_ИндВариант.pdf».

Не допускайте орфографические и грамматические ошибки.

Использование логотипа

Все экранные формы пользовательского интерфейса должны иметь заголовок с логотипом (в ресурсах). Не искажайте логотип (не изменяйте изображение, его пропорции, цвет).

Также для приложений должна быть установлена иконка.

Шрифт

Используйте шрифт Comic Sans MS.

Цветовая схема

В качестве основного фона используется бледно розовый цвет: RGB (234, 215, 223), HEX (#FFEAD7DF); в качестве дополнительного: RGB (82, 169, 239), HEX (#FF52A9EF).

Для акцентирования внимания пользователя на целевое действие интерфейса используйте цвет RGB (86, 150, 59), NEX (#FF56963B).

Основной фон	Дополнительный фон	Акцентирование внимания
RGB(234, 215, 223) HEX (#FFEAD7DF)	RGB (82, 169, 239) HEX (#FF52A9EF)	RGB (86, 150, 59) NEX (#FF56963B)