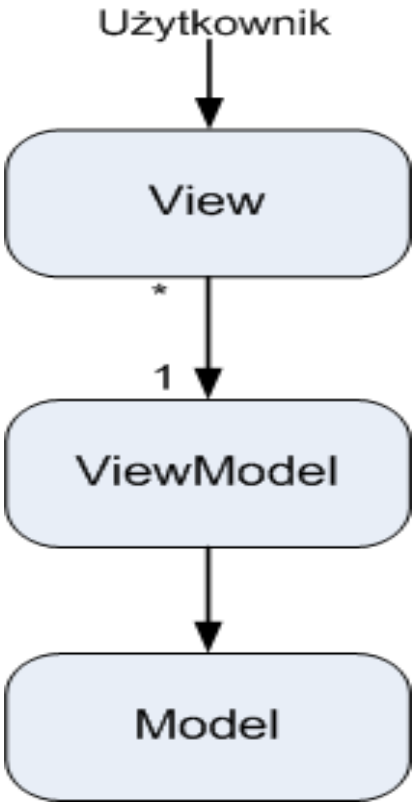
Implementacja wzorca MVVM w PRISM

Wzorzec MVVM (Model-View-ViewModel) umożliwia oddzielenie warstwy prezentacji od warstwy logiki aplikacji. Dzięki temu można w łatwy sposób stworzyć aplikację która jest testowalna, prosta w rozbudowie i pozwala na uzyskanie wysokiej reużywalności kodu w przyszłych projektach.

Zasada działania wzorca.



1. W pierwszej kolejności tworzymy katalog o nazwie „Models” i umieszczamy w nim klasę o nazwie „Person” oraz dodajemy jej własności.

```
public class Person
{
    public string Name { get; set; }
    public int Year { get; set; }
    public int Age { get; set; }
    public string FavouriteColor { get; set; }

    public int CalculateAge()
    {
        return System.DateTime.Now.Year - Year;
    }
}
```

2. Tworzymy nową klasę o nazwie „WidokViewModel” i umieszczamy ją w katalogu „ViewModels”. Ta klasa posłuży nam jako łącznik naszego modelu z widokiem. Jeżeli chcemy obsłużyć zdarzenia ViewModel musi dziedziczyć po klasie BindableBase, która implementuje INotifyPropertyChanged.

```
public class WidokViewModel : BindableBase
    {
        private int _year;
        private int _age;
        private string _color;
        private string _name;

        Person person = new Person();

         //Zdarzenia
        public DelegateCommand CalculateCommand { get; set; }
        public DelegateCommand ResetCommand { get; set; }
        public DelegateCommand AddCommand { get; set; }

        public string[] AllColors { get; set; }

        public WidokViewModel()
        {
            CalculateCommand = new DelegateCommand(CalcAge);
            ResetCommand = new DelegateCommand(Reset);
            AddCommand = new DelegateCommand(AddPerson);

            this.AllColors = new[] { "czerwony", "żółty", "zielony" };
        }
```

```
public void AddPerson()
{
   person.Year = _year;
   person.FavouriteColor = _color;
   person.Name = _name;

   Reset();
}

public void Reset()
{
   Name = "";
   Color = "";
   Age = 0;
   Year = 0;
}

public int Year
{
   get
   {
      return _year;
   }
   set
   {
      SetProperty<int>(ref _year, value);
   }
}

public int Age
{
   get
   {
      return _age;
   }
   set
   {
      SetProperty<int>(ref _age, value);
   }
}

public void CalcAge()
{
   person.Year = _year;
   Age = person.CalculateAge();
}
```

```
        public string Name
        {
            get
            {
                return _name;
            }
            set
            {
                SetProperty(ref _name, value);
            }
        }

        public string Color
        {
            get
            {
                return _color;
            }
            set
            {
                SetProperty(ref _color, value);
            }
        }
    }
```

3. Zakładamy katalog o nazwie „Views” i umieszczamy w nim nasz nowy widok i za pomocą databindingu łączymy widok z ViewModelem.

```
<Window x:Class="prismMVVM.Views.Widok"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     xmlns:prism="clr-
namespace:Microsoft.Practices.Prism.Mvvm;assembly=Microsoft.Practices.Prism.Mvvm.Desktop"
        prism:ViewModelLocator.AutoWireViewModel="True"
        Title="Widok" Height="300" Width="300">
    <Grid>
        <Grid.ColumnDefinitions>
            <ColumnDefinition/>
            <ColumnDefinition/>
            <ColumnDefinition/>
        </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
            <RowDefinition/>
        </Grid.RowDefinitions>
```

```
        <TextBlock Text="Imię" Grid.Column="0" Grid.Row="0" HorizontalAlignment="Center"
        VerticalAlignment="Center"/>

        <TextBlock Text="Rok urodzenia" Grid.Column="0" Grid.Row="1"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>

        <TextBlock Text="Ulubiony kolor" Grid.Column="0" Grid.Row="2"
        HorizontalAlignment="Center" VerticalAlignment="Center"/>

        <TextBlock Text="{Binding Age}" Grid.Column="1" Grid.Row="1" Margin="59,11,10,26"/>


      <TextBox Text="{Binding Name}" Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="0"
    Margin="10,10,10,10"/>


        <TextBox Text="{Binding Year}" Grid.Column="1" Grid.Row="1" Margin="10,11,52,26" />

        <ListView Grid.Column="1" Grid.Row="2" Grid.ColumnSpan="2" ItemsSource="{Binding
        AllColors}" SelectedItem="{Binding Color}"/>

        <Button Content="Reset" Grid.Column="1" Grid.Row="3" Margin="10,10,10,10"
        Command="{Binding ResetCommand}"/>

        <Button Content="Dodaj" Grid.Column="2" Grid.Row="3" Margin="10,10,10,10"
        Command="{Binding AddCommand}"/>

        <Button Content="Oblicz wiek" Grid.Column="2" Grid.Row="1" Margin="10,10,10,10"
        Command="{Binding CalculateCommand}"/>


        <TextBlock Text="{Binding ShowName}" Grid.Column="0" Grid.Row="4" />
        <TextBlock Text="{Binding ShowAge}" Grid.Column="1" Grid.Row="4" />
        <TextBlock Text="{Binding ShowColor}" Grid.Column="2" Grid.Row="4" />
      </Grid>
    </Window>
```

4. W „Widok.xaml.cs” musimy zaimplementować IView
   public partial class Widok : Window, IView