Xamarin

Martin Bojnanský
Microsoft Student Partner & SW Engineer @ Riganti

@martinbojnansky
martin.bojnansky@outlook.com
http://bojnansky.com

SW Architektúra

Analógia v architektúre budov

Dobrá SW Architektúra

Tvorí základ projektu

Pomáha zvládať komplexnosť

Zjednocuje tím

Zrýchľuje a šetrí náklady pre vývoj a hľadanie chýb

Znižuje závislosti a zjednodušuje rozšírenie

...

SOLID

5 pravidiel pre lepšiu testovateľnosť kódu

SOLID

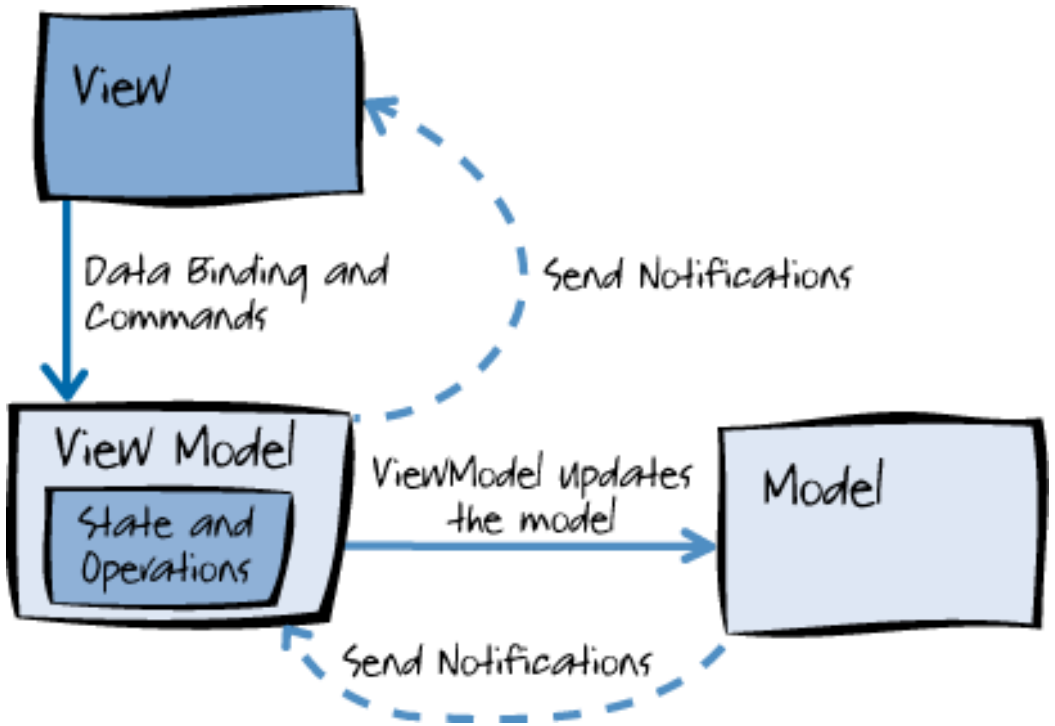
1. Single Responsibility – trieda má jeden účel
2. Open/Closed – otvorená rozšíreniu / uzavretá zmenám
3. Liskov substation – správne použitie dedičnosti
4. Interface segregation – viac jednoúčelových rozhraní namiesto komplexného
5. Dependency Inversion – závislosť na abstrakciách

MVVM

MVVM

Model-View-ViewModel

Návrhový vzor



Výhody MVVM

Rieši komplexnosť

Znižuje závislosť na UI vrstve = vyššia prenositeľnosť kódu

Odstraňuje množstvo kódu pomocou DataBinding

Jednoduchá testovateľnosť ViewModel

Nevýhoda MVVM

Väčšia náročnosť na pamäť

Nutnosť implementácie *INotifyPropertyChanged*

View

Definuje používateľské rozhranie

Previazaný s hodnotami a akciami ViewModel

```
public partial class CategoriesView : ContentPage
{
    1 reference
    public CategoriesViewModel ViewModel
    {
        get
        {
            return BindingContext as CategoriesViewModel;
        }
        set
        {
            BindingContext = value;
        }
    }
```

```
<controls:NewItemControl
    Grid.Row="1"
    Text="{Binding NewTodoItemText}"
    AddCommand="{Binding AddTodoCommand}" />
```

ViewModel

Reprezentuje stav UI

Obsahuje akcie, ktoré je možné volať z UI

```
private string _newTodoItemText = "";

3 references
public string NewTodoItemText
{
    get => _newTodoItemText;
    set
    {
        _newTodoItemText = value;
        OnPropertyChanged();
    }
}
```

```
private Command _addTodoCommand;
0 references
public Command AddTodoCommand => _addTodoCommand ?? (_addTodoCommand = new Command(AddTodo));

1 reference
private void AddTodo()
{
```

Model

Dátové modely

Business / Validation logika

DTOs, Repositories, …

Naming

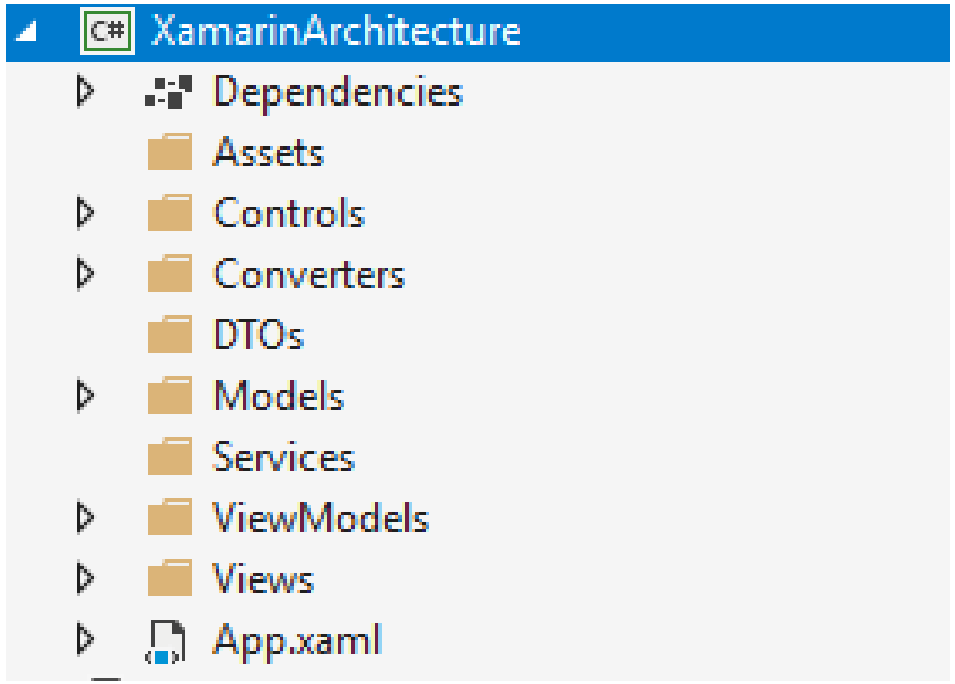
https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines

Folders

Folders

Podľa funkcie

Zároveň definujú **namespace**



User Controls

User Controls

Vytváranie kontroliek / šablón pre znovu-použiteľnosť

Naplnenie hodnotami

```
<controls:NewItemControl
    Text="{Binding NewTodoItemText}"
    AddCommand="{Binding AddTodoCommand}" />
```

```
public static readonly BindableProperty TextProperty =
    BindableProperty.Create(nameof(Text), typeof(string), typeof(NewItemControl), "", BindingMode.TwoWay);

1 reference
public string Text
{
    get { return (string)GetValue(TextProperty); }
    set { SetValue(TextProperty, value); }
}
```

Class Libraries

Class Libraries

Oddelenie logických častí projektu (infraštruktúra, API klient, ...)

Zdieľanie medzi rôznymi projektmi / nuget (infraštruktúra,)

Kompilovaný balík .dll

Refactoring