

Développement d'application Nomade

MVVM & XAMARIN.FORMS

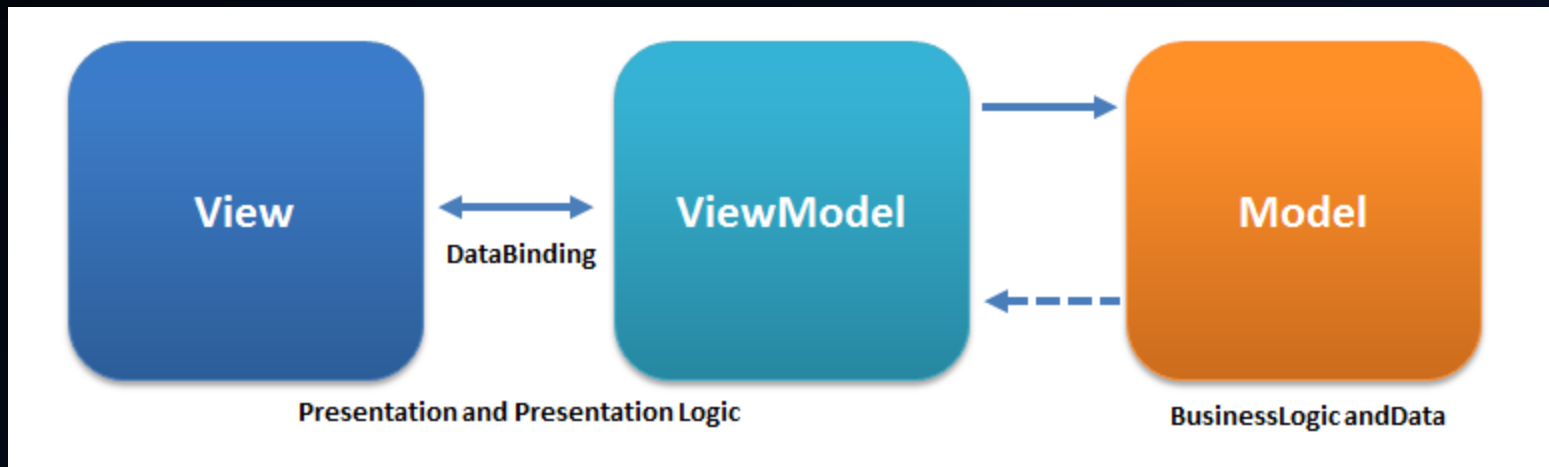




MVVM

Qu'est ce que c'est ?

- Model-View-ViewModel

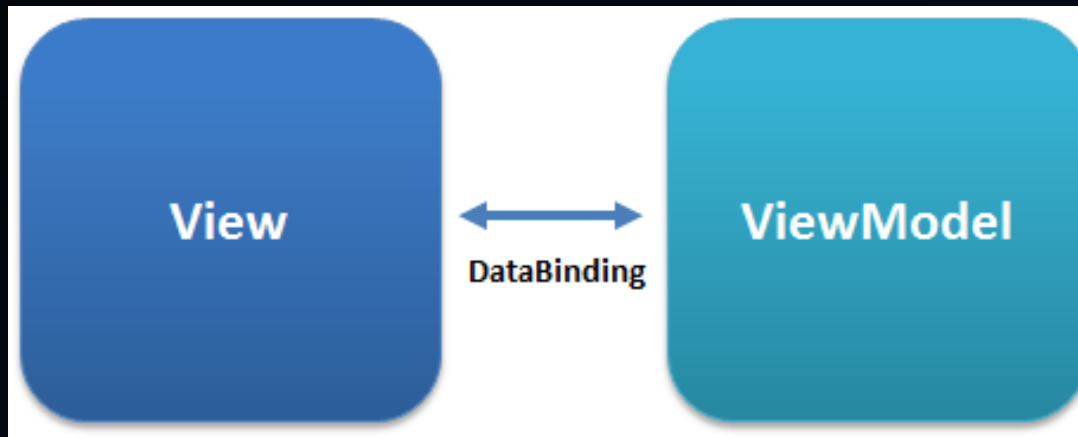


- Model/ViewModel => C#
- View => Xaml

Quel intérêt ?

- Couplage faible entre Vue et ViewModel
- Développement dissocié
 - => Plus simple de gérer le développement en équipe
- Le designer développe la Vue,
- Le développeur le reste 😊

Comment communiquer ?



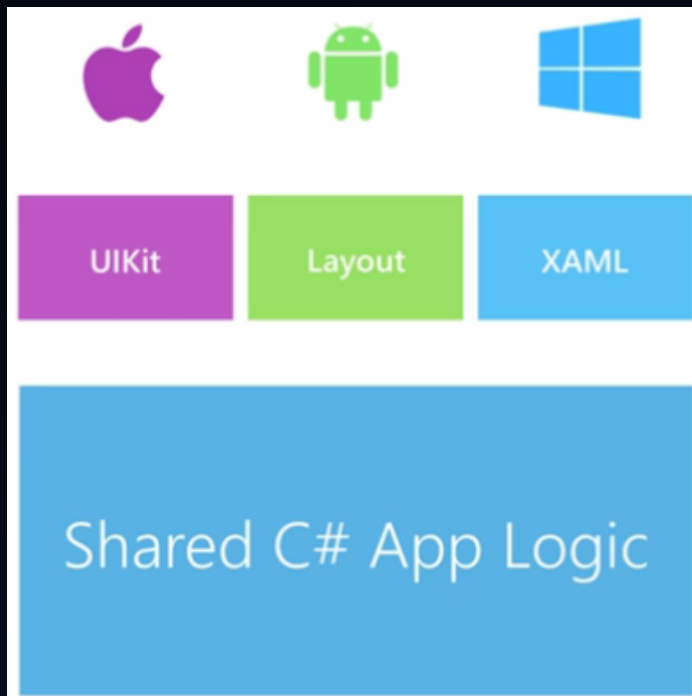
- Moteur de binding
- La vue « demande » qu'on lie certaines propriétés du ViewModel
 - Propriété « simple » => Récupération de valeur pour l'affichage
 - Command => Lien pour que le ViewModel réagisse sur les actions de la Vue

Xamarin.Forms

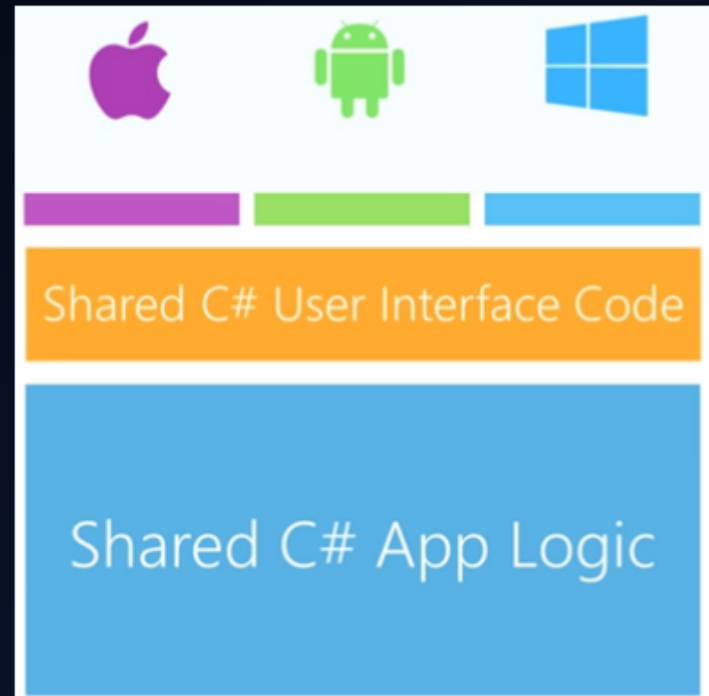


Quelle différence avec Xamarin ?

XAMARIN

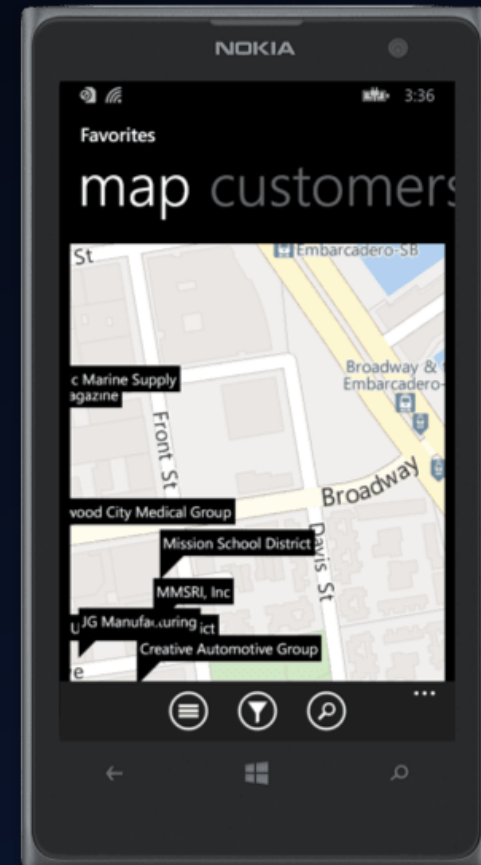
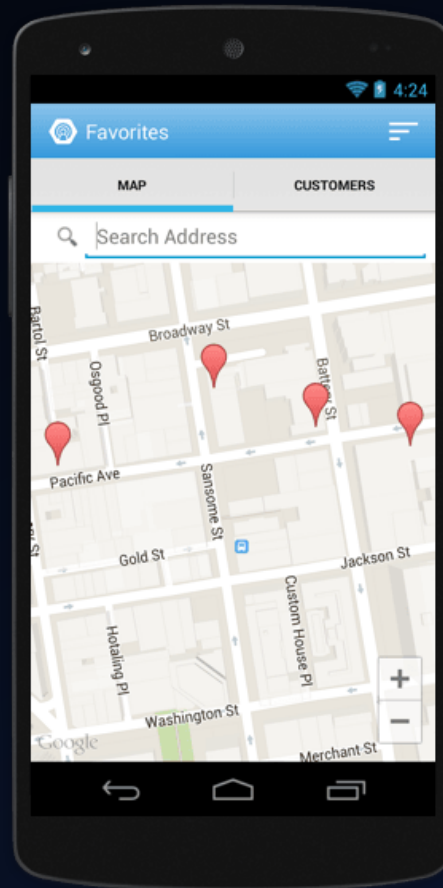
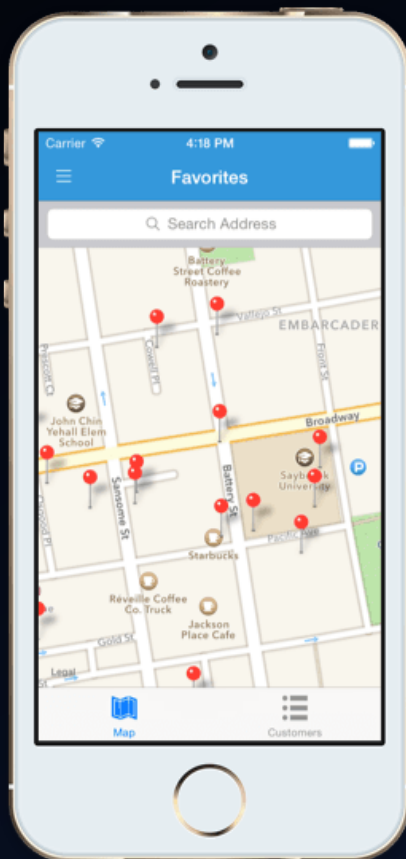


XAMARIN.FORMS



Xamarin.Forms ?

- Un code, trois plateformes, trois applications



Xamarin.Forms & MVVM

Comment communiquer ?

- ViewModel.cs

```
public class ViewModel : BaseViewModel
{
    private string _name;
    private string _lastName;

    public string LastName {
        get { return _lastName; }
        set { SetProperty(ref _lastName, value); }
    }
    public string Name {
        get { return _name; }
        set { SetProperty(ref _name, value); }
    }
    public ICommand ButtonCommand { get { return new Command(ButtonAction); } }

    private void ButtonAction() {
        Debug.WriteLine("Name = " + Name + ", LastName = " + LastName);
    }
}
```

Comment communiquer ?

- View.xaml

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:vm="clr-namespace:XamarinTDLiveCoding.ViewModels;assembly=XamarinTDLiveCoding"
              x:Class="XamarinTDLiveCoding.Views.View">
  <ContentPage.BindingContext>
    <vm:ViewModel />
  </ContentPage.BindingContext>

  <StackLayout Orientation="Vertical">
    <Entry Placeholder="Name..."
           Text="{Binding Name, Mode=TwoWay}"
           />
    <Entry Placeholder="Last name..."
           Text="{Binding LastName, Mode=TwoWay}"
           />
    <Button Text="Click me !"
           Command="{Binding ButtonCommand}"
           />
  </StackLayout>
</ContentPage>
```

Comment communiquer ?

- Association de la Vue et du ViewModel

```
xmlns:vm="clr-namespace:XamarinTDLiveCoding.ViewModels;assembly=XamarinTDLiveCoding"
```


```
<ContentPage.BindingContext>  
    <vm:ViewModel />  
</ContentPage.BindingContext>
```

- xmlns:vm => déclare un namespace xml (ici nommé vm)
- clr-namespace: => définit le namespace C# associé
- assembly= => définit le nom de l'assembly qui le contient (dll / nom du projet)

Comment communiquer ?

- Utilisation des bindings pour récupérer les valeurs du ViewModel

```
<Entry Placeholder="Name..."
      Text="{Binding Name, Mode=TwoWay}"
/>
```



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

Comment utiliser les Bindings ?

- {Binding NomPropriété [, attribut=valeur] }
- *Les attributs*
- Mode :
 - OneWay : met à jour du ViewModel vers la Vue
 - OneWayToSource : met à jour de la Vue vers le ViewModel
 - TwoWay : met à jour dans les deux sens
 - Default : met le mode par défaut (dépendant des propriétés)
- StringFormat : format pour afficher la valeur (@see string.Format)
 - Exemple : "{Binding Age, StringFormat='Vous avez {0} ans'}«

Comment utiliser les Bindings ?

- Converter : permet un prétraitement des données avant de les afficher
- Un converter est une classe C# qui implémente `IValueConverter`
 - `Convert` (ViewModel -> Vue)
 - `ConvertBack` (Vue -> ViewModel)
- Exemple : `StringToUpperConverter.cs`

```
public class StringToUpperConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        string val = value as string;
        return val != null ? val.ToUpper() : value;
    }

    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        throw new NotImplementedException();
    }
}
```

Comment utiliser les Bindings ?

- ConverterParameter : permet de passer un paramètre à votre Converter
- Utilisation des Converters

```
<StackLayout.Resources>
  <ResourceDictionary>
    <converters:StringToUpperConverter x:Key="StringToUpperConverter"/>
  </ResourceDictionary>
</StackLayout.Resources>
<Label Text="{Binding Name, Converter={StaticResource StringToUpperConverter}}"
  />
```

- Remarque : {StaticResource ResourceKey} vous permet d'accéder à ce que vous avez défini en resources
- /!\ Toutes les resources doivent avoir une clé ! (x:Key)

Questions ?

