

Le développement cross plateforme avec Xamarin

Module 3 – Découverte de Xamarin.Forms



Objectifs

- Comprendre la différence entre Xamarin Native et Xamarin.Forms
- Savoir quand utiliser Xamarin Native et quand préférer Xamarin.Forms
- Savoir créer un projet Xamarin.Forms
- Comprendre la structure et l'arborescence d'un projet Xamarin.Forms
- Savoir lancer un projet Xamarin.Forms sur Android et sur Windows

Présentation de Xamarin.Forms

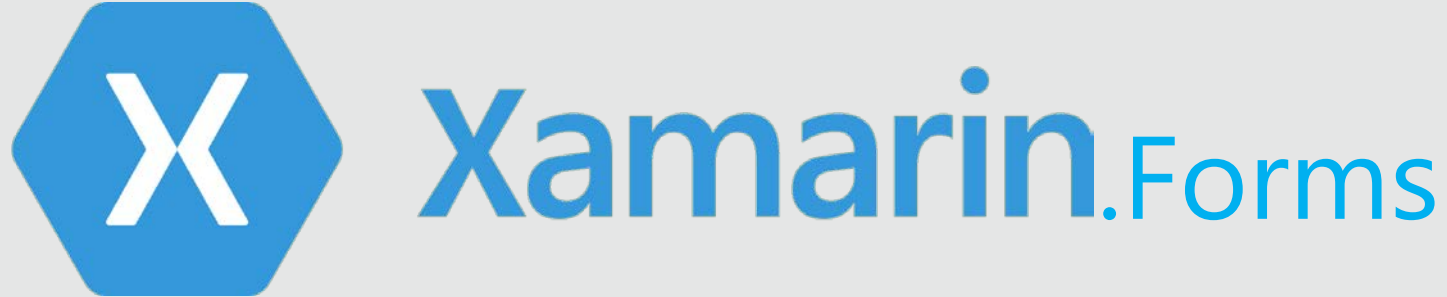
Xamarin permet de mutualiser toute la logique métier mais demande d'implémenter manuellement les vues pour chaque plateforme

Même en utilisant les bons patterns de conception et les bonnes pratiques de mutualisation, le développeur doit développer en double les interfaces utilisateurs

Découverte de Xamarin.Forms

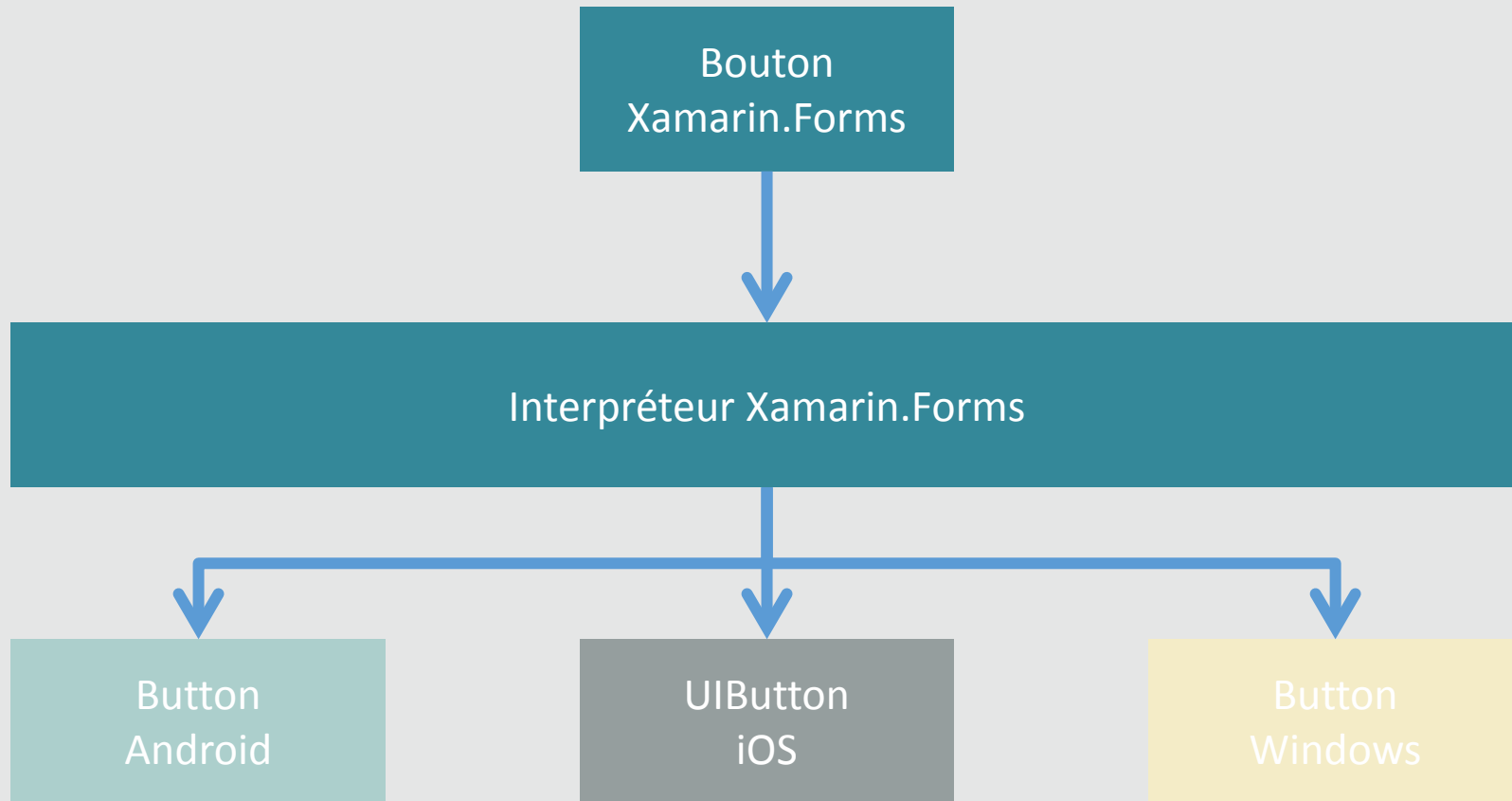
Présentation de Xamarin.Forms

Xamarin propose un nouvel outil qui permet de générer des vues simples pour toutes les plateformes.



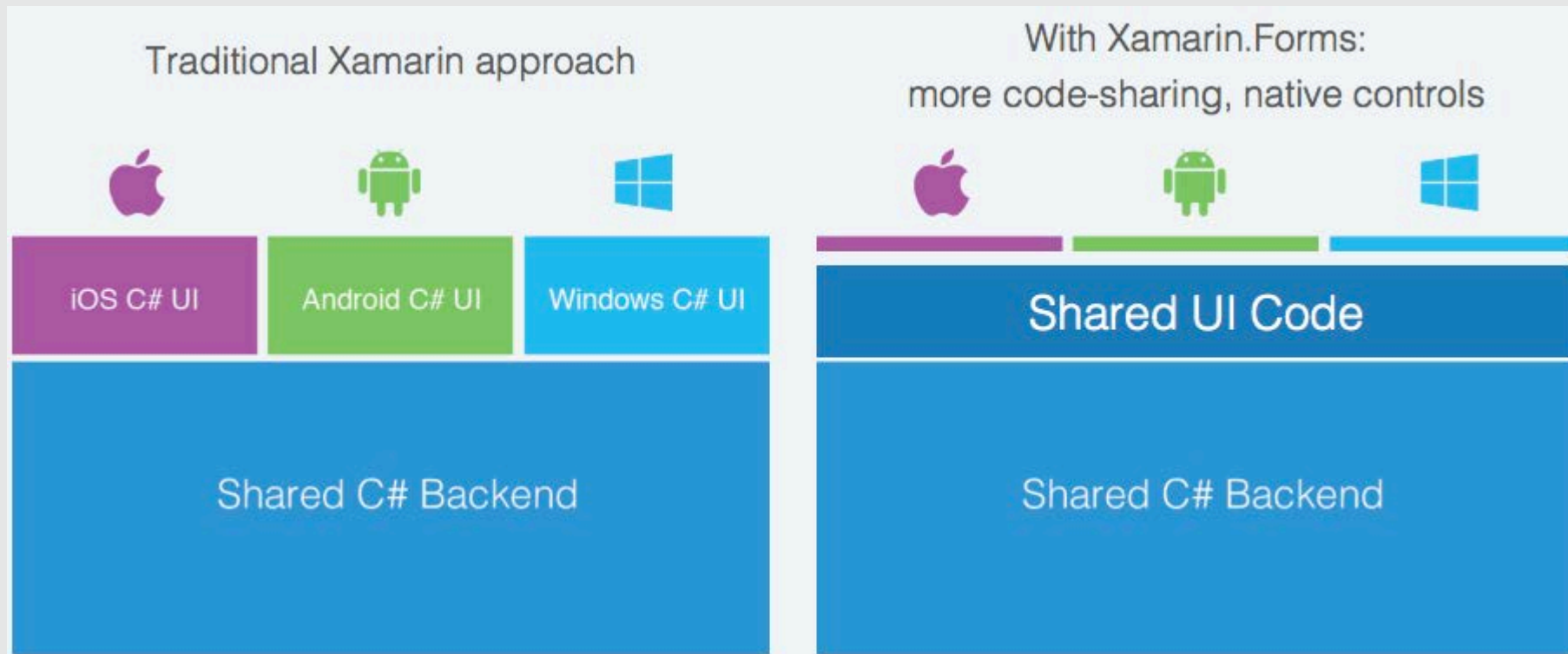
Tous les composants de base ont été traduits d'un métalangage vers des composants natifs.

Présentation de Xamarin.Forms



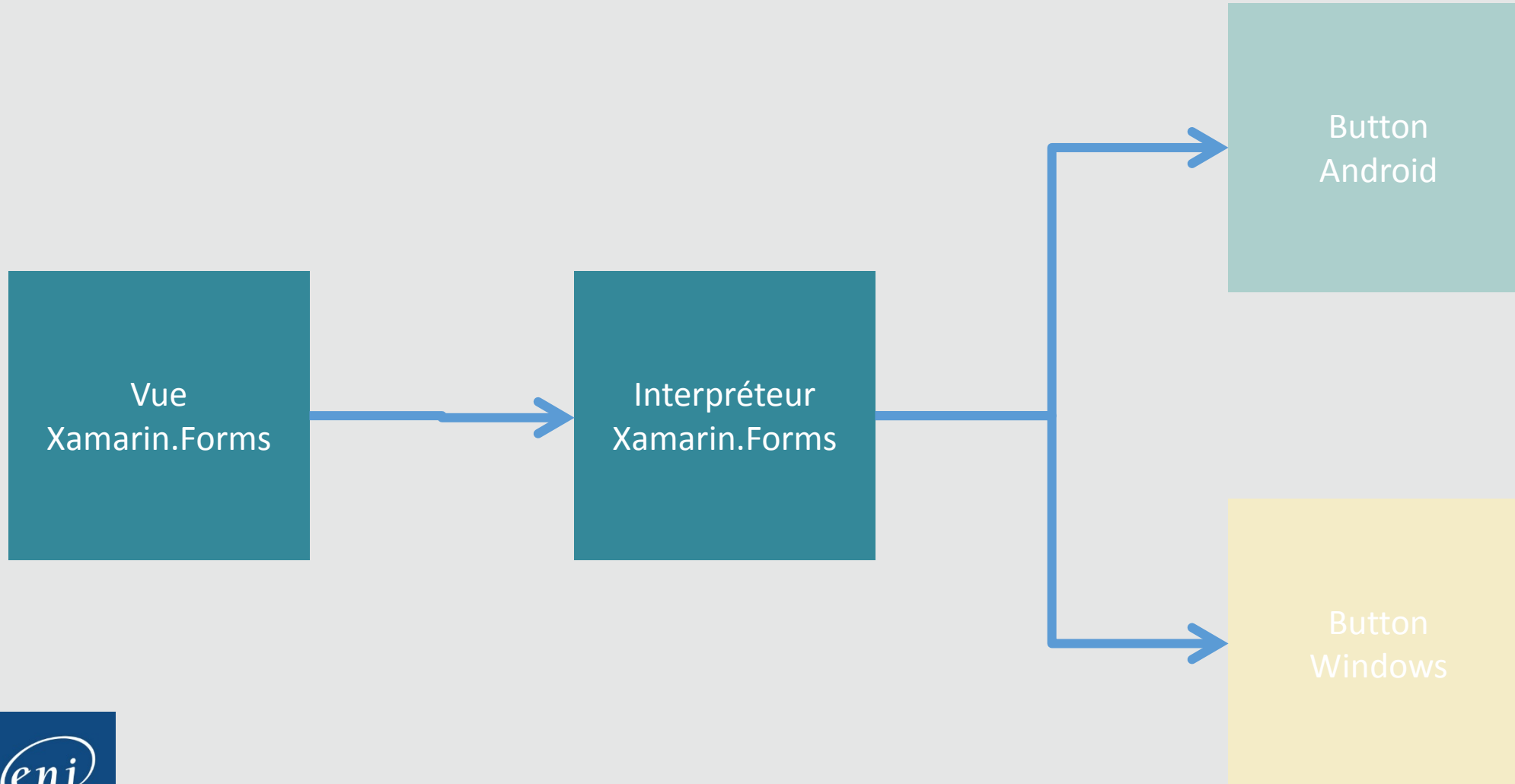
Présentation de Xamarin.Forms

Xamarin.Forms permet d'écrire du code 100% multiplateforme, tout en produisant une application native, utilisant uniquement des composants natifs



Découverte de Xamarin.Forms

Présentation de Xamarin.Forms



Quand utiliser Xamarin.Forms ?

Xamarin.Forms présente l'avantage de pouvoir mutualiser 100% du code source en offrant la possibilité de générer des vues natives à partir d'un métalangage

La limite de ce mode de fonctionnement est qu'il n'est possible d'utiliser que le dénominateur commun à toutes les plateformes !

Quand utiliser Xamarin.Forms ?

Exemple d'une zone de saisie de texte :

Sous Android, il est possible d'afficher une erreur directement sous le champ si l'utilisateur n'a pas effectué une saisie correcte

Sous iOS, ce comportement n'existe pas, il faut afficher une pop-up d'erreur

Donc, avec Xamarin.Forms, il n'est pas possible d'afficher un message d'erreur sous le texte car seul Android le prendrait en compte

Quand utiliser Xamarin.Forms ?

Xamarin.Forms permet de mutualiser le code de la vue mais ne tire parti que des composants et comportements présents sur toutes les plateformes

Les vues natives générées avec Xamarin.Forms sont donc très standards. Elles ne tirent pas autant parti du SDK natif que les vues Xamarin Native.

Quand utiliser Xamarin.Forms ?

Besoin	Xamarin.Forms	Xamarin Native
Prototype, Proof of Concept	Oui	Seulement si le cœur de métier repose sur un élément d'ergonomie ou visuel propre aux plateformes natives
Ecrans de type formulaire, de connexion, préférences utilisateurs	Oui	Seulement si des éléments visuels ou d'ergonomie propres aux plateformes natives
Ecran avec une UI personnalisée (avec composants non standards ou une ergonomie adhérente à la plateforme)	Non	Oui

Découverte de Xamarin.Forms

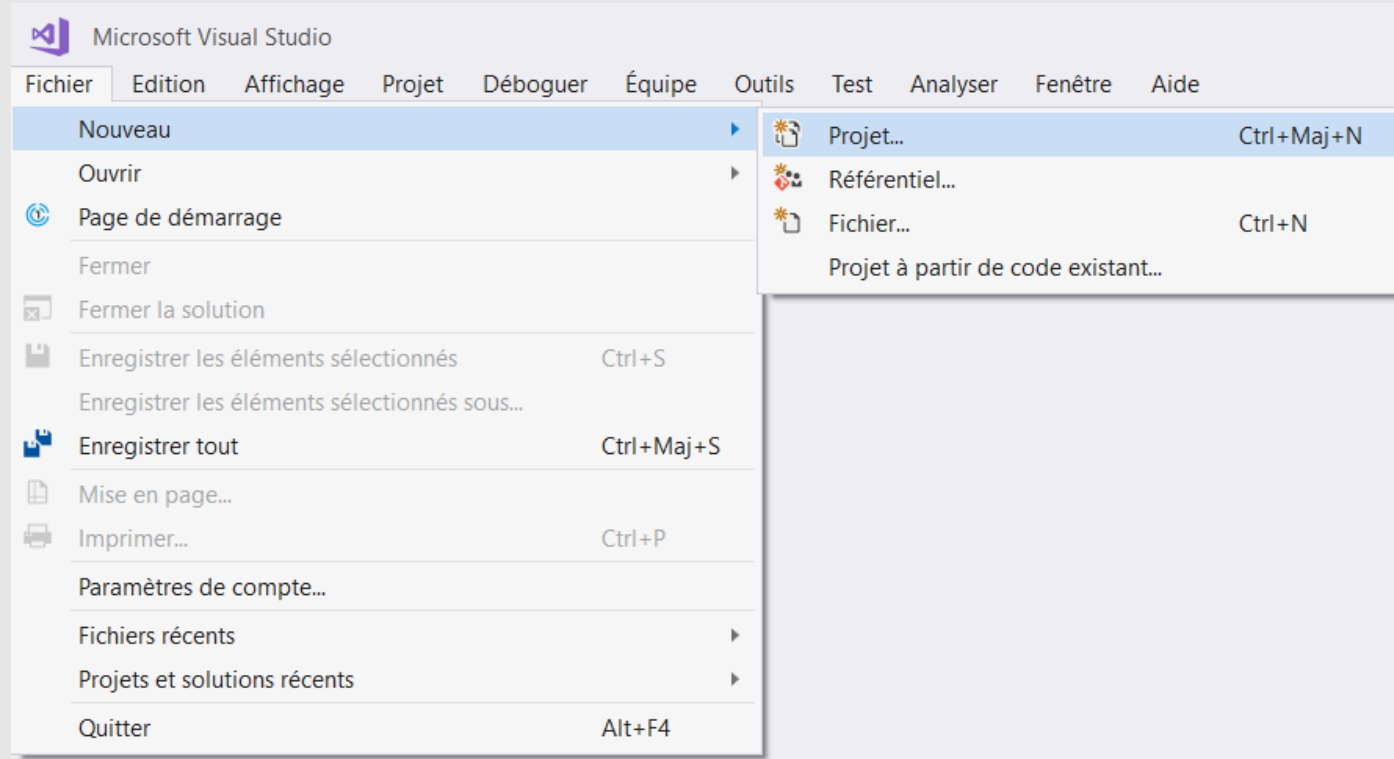
Créer un projet Xamarin.Forms

La création d'un projet Xamarin.Forms s'effectue en un clic à partir de l'IDE
Visual Studio

Visual Studio génère le projet commun à toutes les plateformes et un projet par
plateforme native

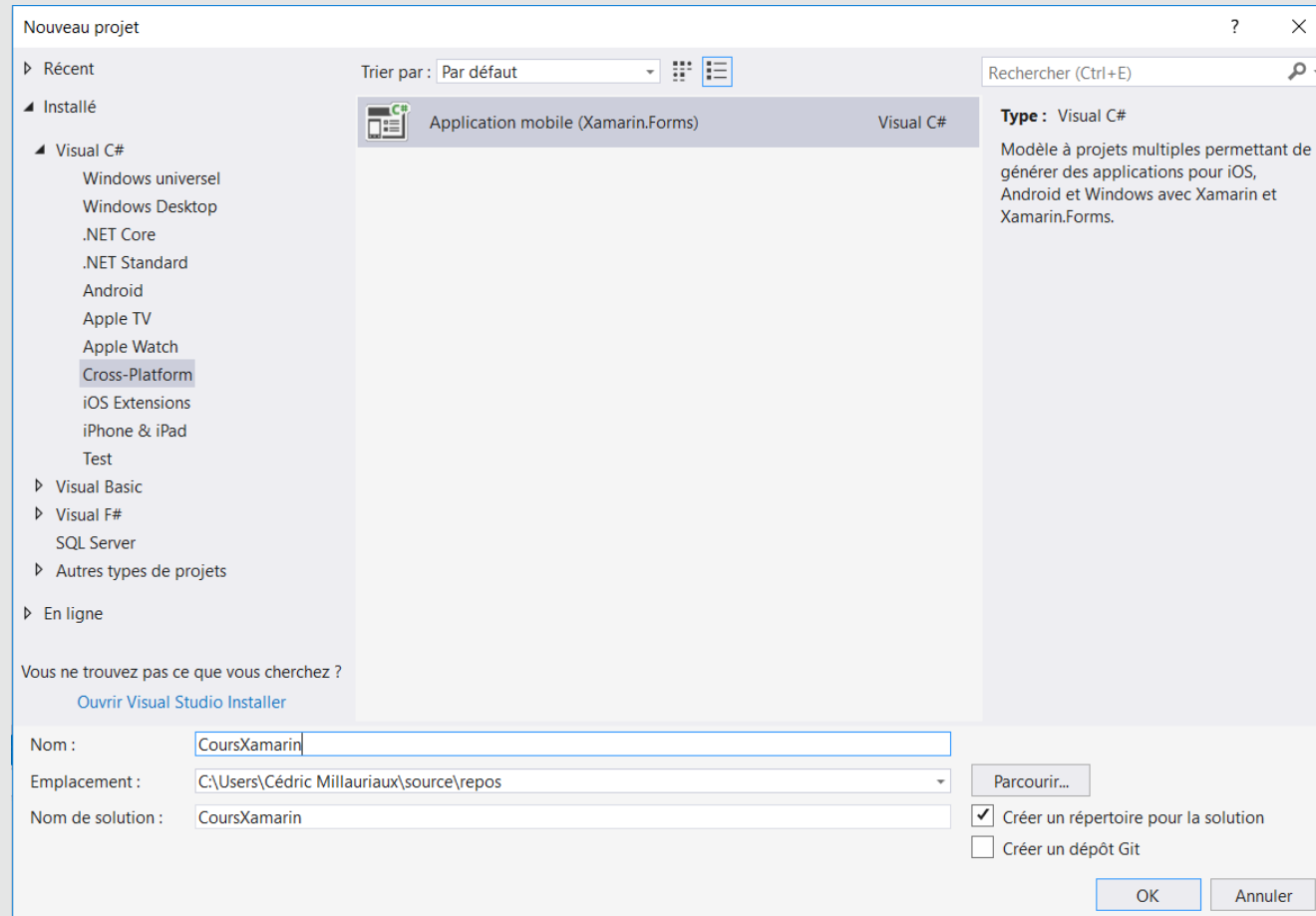


Créer un projet Xamarin.Forms



Découverte de Xamarin.Forms

Créer un projet Xamarin.Forms





Découverte de Xamarin.Forms


Créer un projet Xamarin.Forms

New Cross Platform App - VideoXamarin

Sélectionnez un modèle :

 Blank

 Master-Detail

 Tabbed

A project template for a new Xamarin.Forms app that has no extra sample pages or sample data.

Plateforme

☒ Android

☒ iOS

☒ Windows (UWP)

Stratégie de partage de code ?

☒ .NET Standard

☐ Projet partagé

OK Annuler

Créer un projet Xamarin.Forms

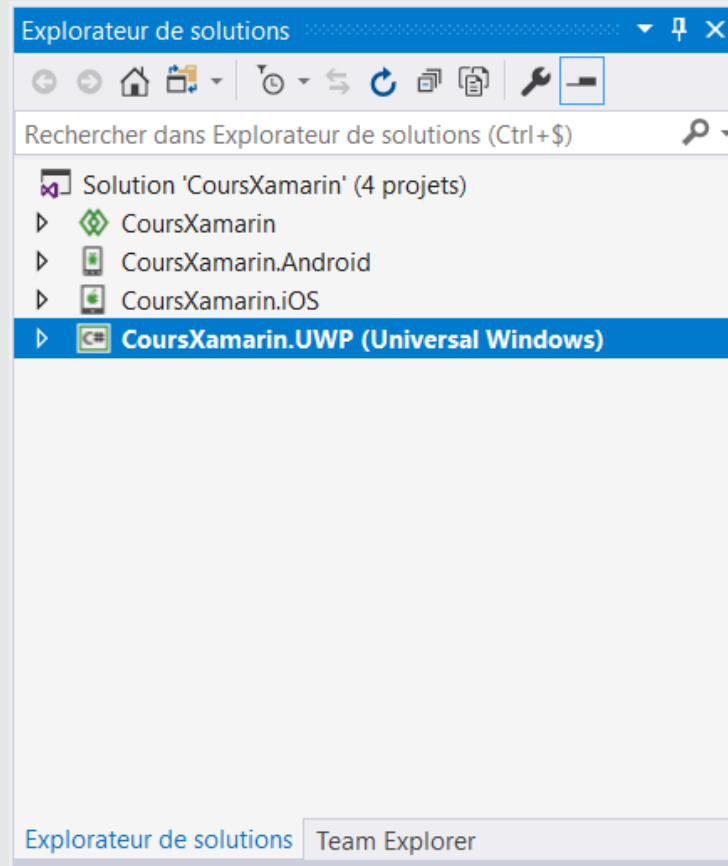
- Xamarin peut être utilisé avec plusieurs stratégies de génération de code
 - La stratégie **.NET Standard**
 - Un seul projet contenant le code commun est développé à l'aide du framework .NET Standard.
 - Ce framework est implémenté sous iOS, Android et Windows
 - Ce même code peut s'exécuter indifféremment sur ces plateformes
 - Attention : le framework est plus restreint que le .NET framework classique
 - La stratégie **PCL** (dépréciée)
 - Stratégie similaire au .NET Standard mais ne suivant pas une norme standardisée
 - La stratégie **projet partagé**
 - Un seul projet contenant du code spécifique à iOS, Android ou Windows
 - Des directives de compilation permettent d'isoler le code spécifique à une plateforme
 - Au moment du packaging, seul le code de la plateforme est embarqué

Créer un projet Xamarin.Forms

- Xamarin peut être utilisé avec plusieurs stratégies de génération de code
 - La stratégie **.NET Standard** permet d'écrire une seule fois le code mais impose d'utiliser le framework .NET de manière restreinte et de ne pas appeler de fonctionnalités du SDK natif d'une plateforme en particulier
 - La stratégie de **projet partagé** nécessite d'écrire une partie du code plusieurs fois mais permet de tirer parti des fonctionnalités du SDK natif de la plateforme
- C'est le plus souvent la stratégie **.NET Standard** qui est utilisée car elle permet de mieux isoler le code multiplateforme du code natif

Découverte de Xamarin.Forms

Créer un projet Xamarin.Forms



Démonstration

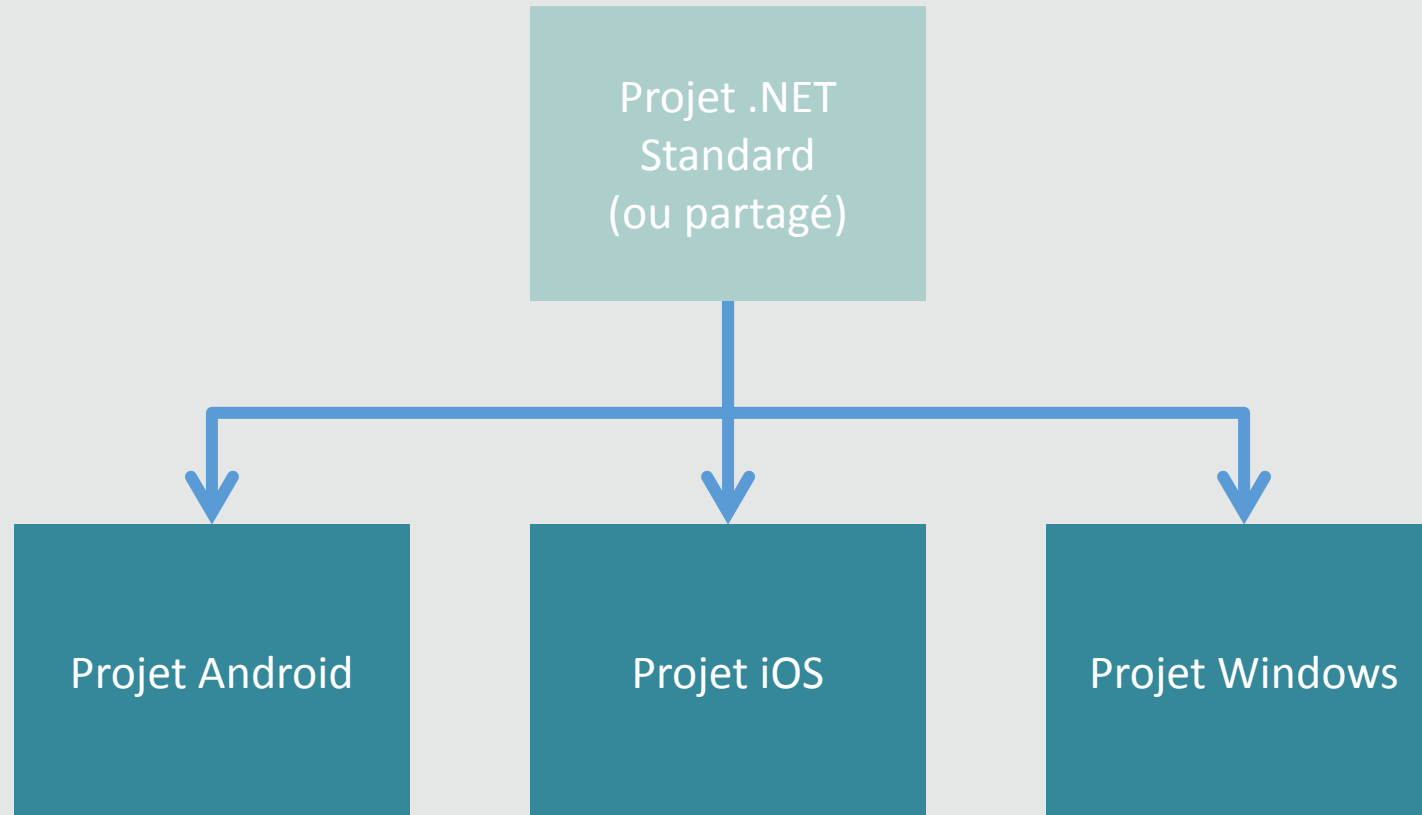


Créer un projet Xamarin.Forms

- Créer un projet de base Xamarin.Forms utilisant un projet .NET Standard pour la mutualisation du code
 - Pour iOS
 - Pour Android
 - Pour Windows

Structure de la solution Xamarin.Forms

Une solution Xamarin.Forms est constituée de 4 projets

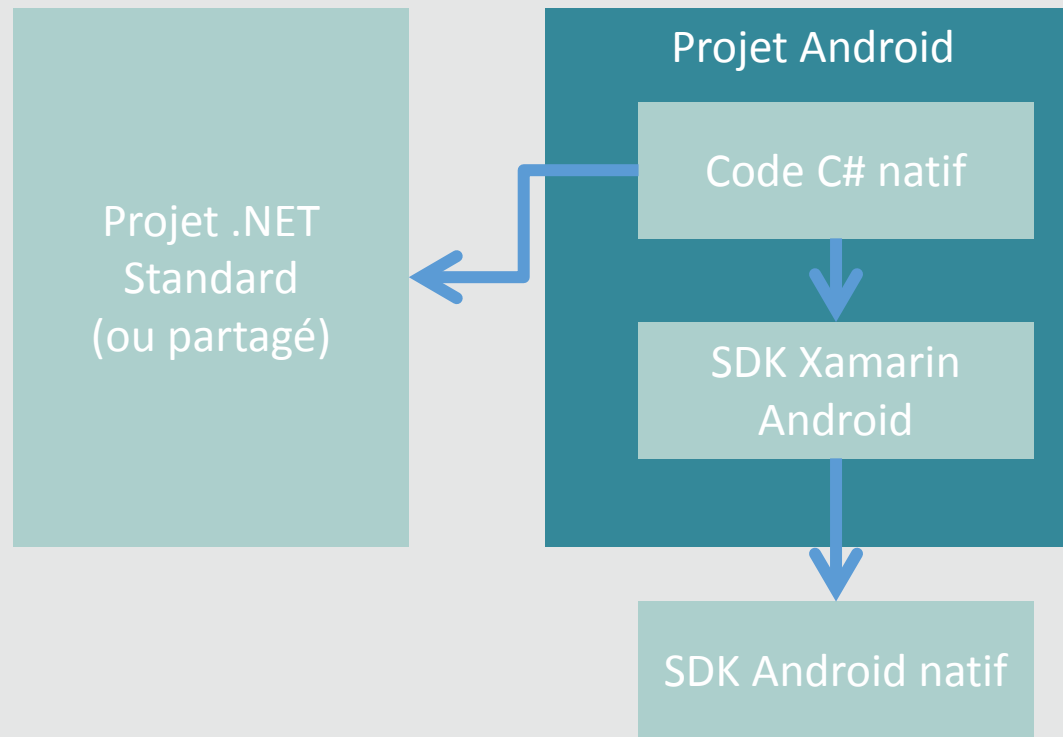


Structure de la solution Xamarin.Forms

- Chaque projet natif contient du code source permettant de compiler l'application pour la plateforme concernée
- Le projet .NET Standard (ou partagé) est importé dans chaque projet natif afin de pouvoir utiliser le code multiplateforme

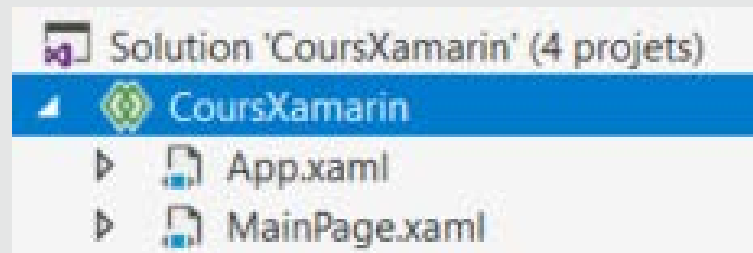
Structure de la solution Xamarin.Forms

- Chaque projet natif repose lui-même sur le SDK Xamarin qui lui est consacré et qui permet d'utiliser l'API native du système



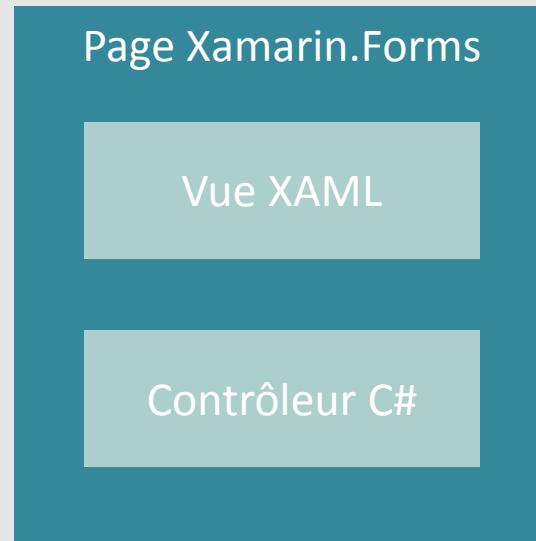
Revue du projet partagé

- Le projet partagé contient, dans le cas des applications Xamarin natives, uniquement le code source métier qui peut être exécuté sur toutes les plateformes
- Il s'agit donc de classes C# tirant parti du framework .NET Standard
- Dans le cas d'un projet Xamarin.Forms, figurent également les pages de l'application



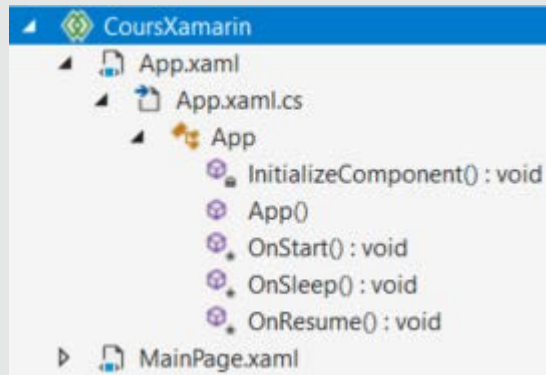
Revue du projet partagé

- Une page Xamarin.Forms est constituée de deux fichiers sources
 - Une page XAML dans laquelle la vue est représentée au format XML
 - Un fichier C# qui fait office de contrôleur pour cette vue



Revue du projet partagé

- La page **App.xaml** est le point d'entrée de l'application
- Elle est la page dans laquelle seront injectées les autres pages
- Des méthodes sont exposées pour gérer le cycle de vie de l'application :
instanciation, démarrage, mise en veille, reprise



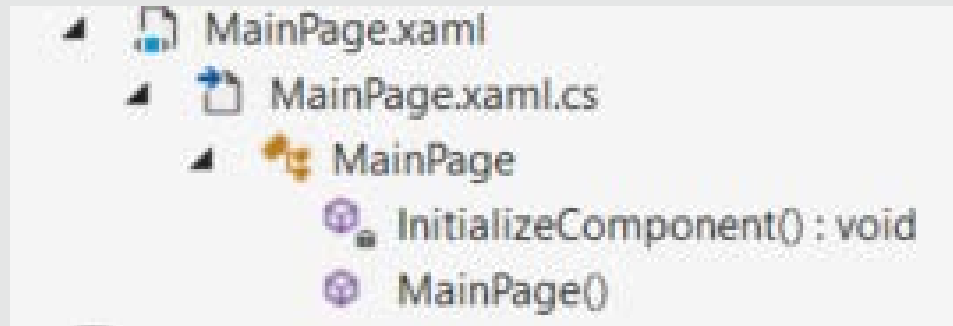
Revue du projet partagé

Aperçu du fichier App.xaml.cs

```
namespace CoursXamarin
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();
            MainPage = new MainPage();
        }
        protected override void OnStart()
        {}
        protected override void OnSleep()
        {}
        protected override void OnResume()
        {}
    }
}
```

Revue du projet partagé

- La page **MainPage.xaml** est créée par l'assistant de Visual Studio et affiche un message de bienvenue à l'utilisateur
- Elle est également constituée d'une vue et d'un contrôleur



Revue du projet partagé

Aperçu du fichier MainPage.xaml

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              xmlns:local="clr-namespace:CoursXamarin"
              x:Class="CoursXamarin.MainPage">

    <StackLayout>
        <!-- Place new controls here -->
        <Label Text="Welcome to Xamarin.Forms!"
              HorizontalOptions="Center"
              VerticalOptions="CenterAndExpand" />
    </StackLayout>

</ContentPage>
```

Revue du projet partagé

Aperçu du fichier MainPage.xaml.cs

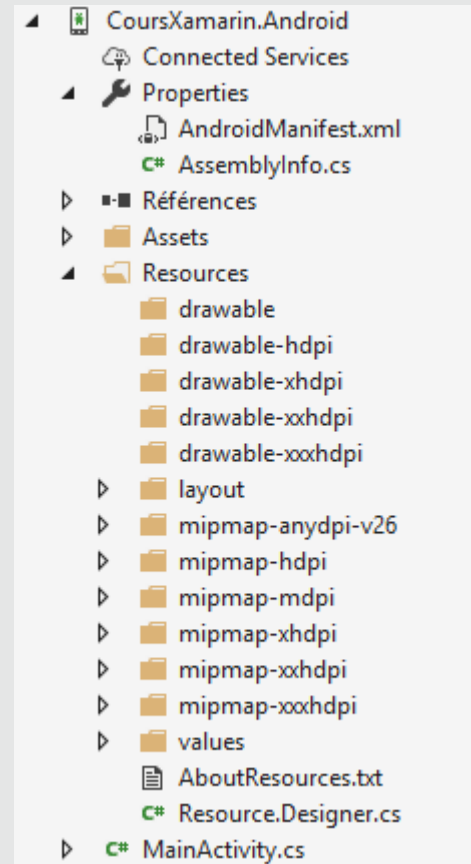
```
namespace CoursXamarin
{
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

Revue du projet Android

- Le projet Android possède la même arborescence qu'un projet natif :
 - Un fichier **AndroidManifest.xml** qui permet de déclarer l'identité, la liste des composants et les droits nécessaires à l'exécution de l'application
 - Un répertoire **Resources** dans lequel figurent tous les médias, les fichiers de configuration, les polices de l'application ainsi que les vues au format XML
 - Un ensemble d'activités qui représentent les contrôleurs des pages

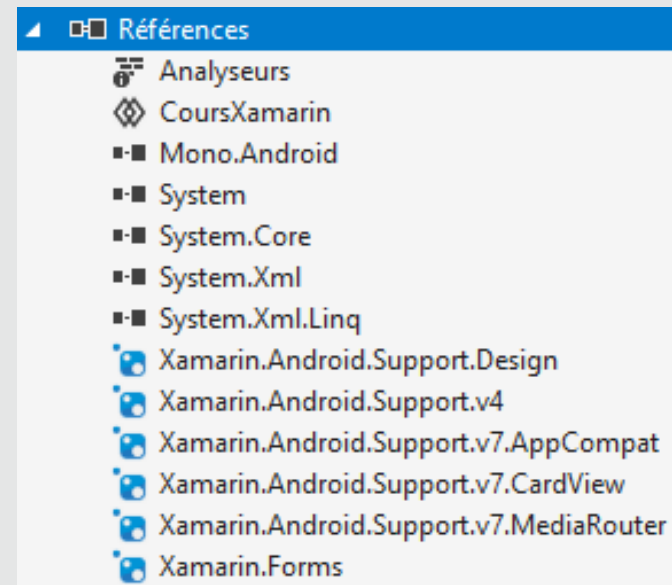
Revue du projet Android

Aperçu de l'arborescence du projet Android



Revue du projet Android

- Plusieurs références (ou dépendances) permettent de dialoguer avec le SDK natif Android, Xamarin.Forms ou le framework .NET
 - Mono.Android
 - .NET Standard
 - Xamarin.Android.Support
 - Xamarin.Forms



Revue du projet Android

Aperçu du fichier MainActivity.cs

```
namespace CoursXamarin.Droid
{
    [Activity(Label = "CoursXamarin", Icon = "@mipmap/icon", Theme = "@style/MainTheme", MainLauncher = true, ConfigurationChanges =
    ConfigChanges.ScreenSize | ConfigChanges.Orientation)]
    public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
    {
        protected override void onCreate(Bundle savedInstanceState)
        {
            TabLayoutResource = Resource.Layout.Tabbar;
            ToolbarResource = Resource.Layout.Toolbar;

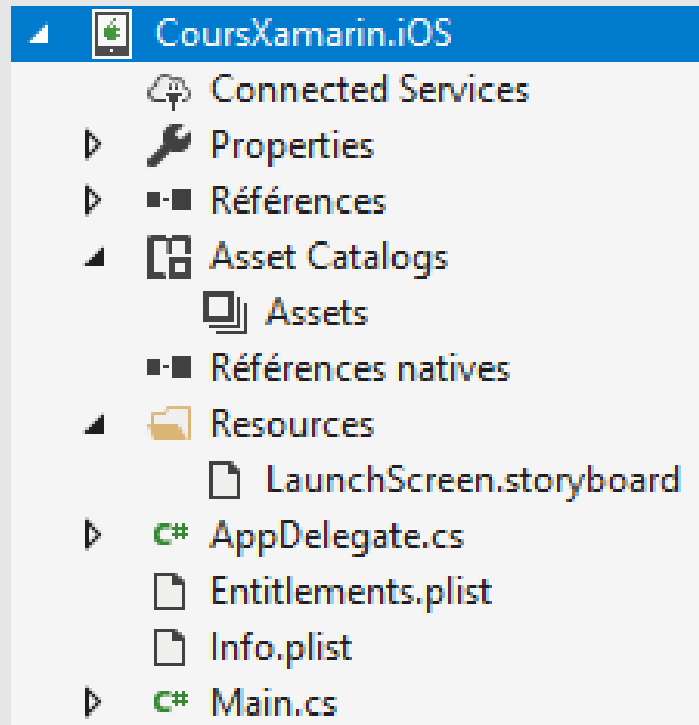
            base.OnCreate(savedInstanceState);
            global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
            LoadApplication(new App());
        }
    }
}
```

Revue du projet iOS

- Le projet iOS possède la même arborescence qu'un projet natif
 - Un fichier **Info.plist** qui permet de déclarer l'identité, la liste des composants et les droits nécessaires à l'exécution de l'application
 - Un répertoire **Resources** dans lequel figurent les vues au format storyboard
 - Un **catalogue d'assets** pour recevoir les médias

Revue du projet iOS

Aperçu de l'arborescence du projet Android



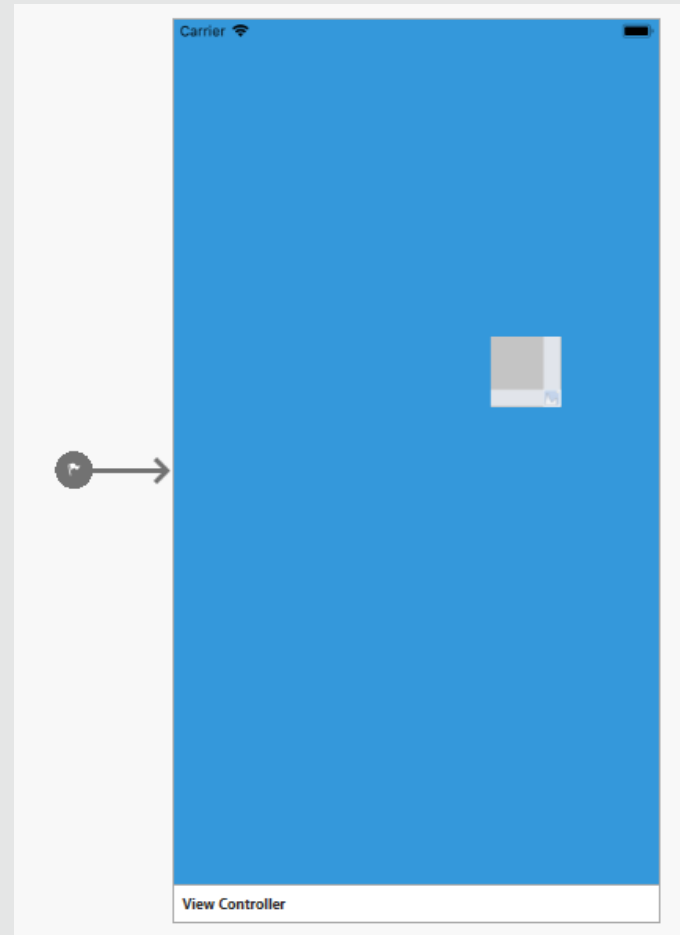
Revue du projet iOS

- Plusieurs références (ou dépendances) permettent de dialoguer avec le SDK natif Android, Xamarin.Forms ou le framework .NET
 - Xamarin.iOS
 - .NET Standard
 - Xamarin.Forms

Découverte de Xamarin.Forms

Revue du projet iOS

Aperçu du storyboard



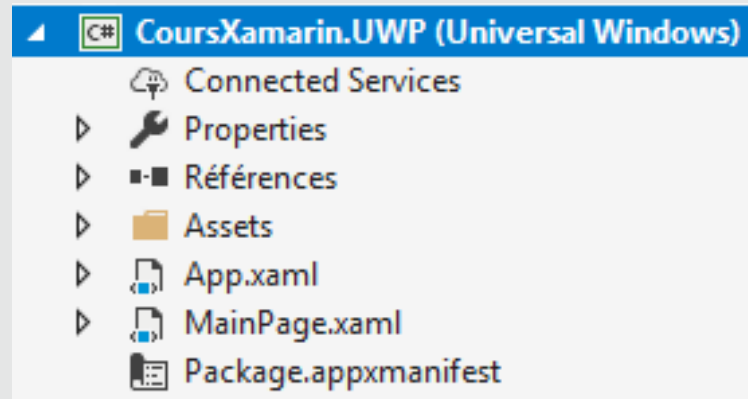
Revue du projet Windows

- Le projet Universal Windows Platform possède la même arborescence qu'un projet WPF
 - Un fichier **Package.appxmanifest** qui permet de déclarer l'identité, la liste des composants et les droits nécessaires à l'exécution de l'application
 - Un répertoire **Assets** dans lequel figurent les médias
 - Un ensemble de pages (XAML/CS)

Découverte de Xamarin.Forms

Revue du projet Windows

Aperçu de l'arborescence du projet Universal Windows Platform



Revue du projet Windows

- Plusieurs références (ou dépendances) permettent de dialoguer avec le SDK natif Android, Xamarin.Forms ou le framework .NET
 - Universal Windows
 - .NET Standard
 - Xamarin.Forms

Lancement du projet Android

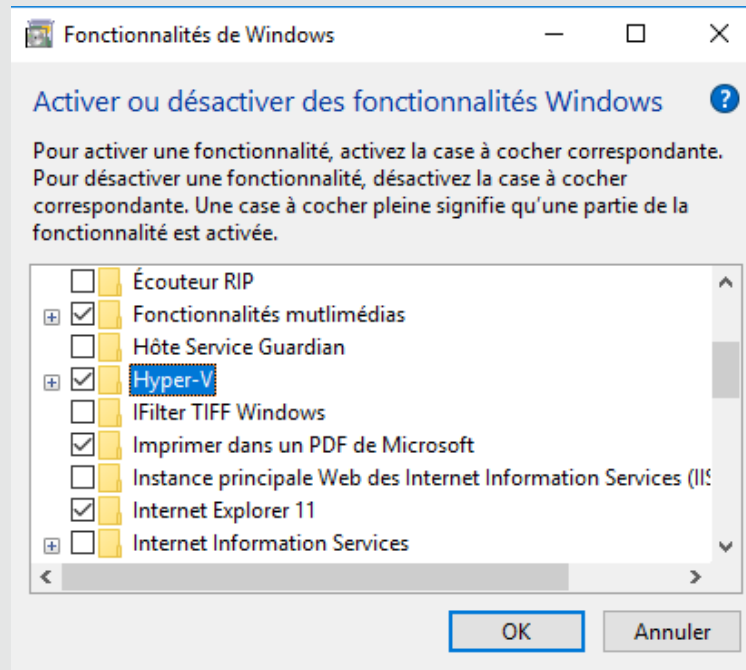
- Il existe deux moyens de lancer un projet sur Android
 - Via l'émulateur
 - En connectant un téléphone Android en mode développeur

Lancement du projet Android

- Au moment de l'installation de Xamarin, Visual Studio installe un émulateur Android ainsi que l'ensemble des outils nécessaires pour compiler et déployer un exécutable Android
- Il est nécessaire d'activer Hyper-V pour pouvoir utiliser cet émulateur optimisé pour Visual Studio

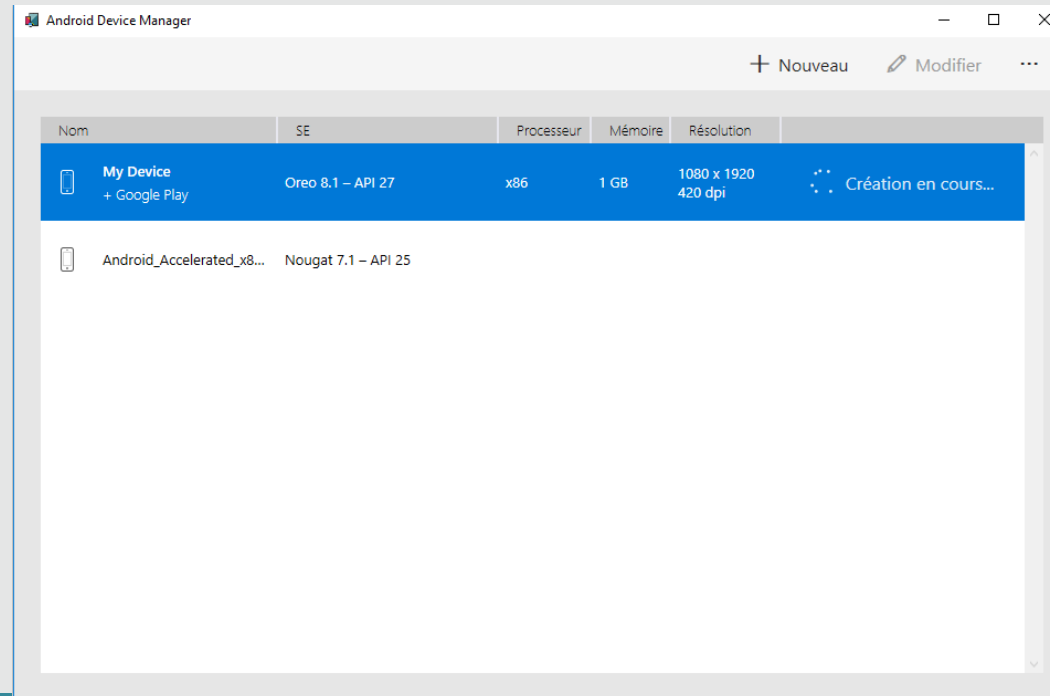
Lancement du projet Android

- Pour activer Hyper-V, effectuer une recherche Windows du programme **Activer ou désactiver des fonctionnalités Windows**, puis cocher **Hyper-V** ainsi que **Plateforme de l'hyperviseur Windows**



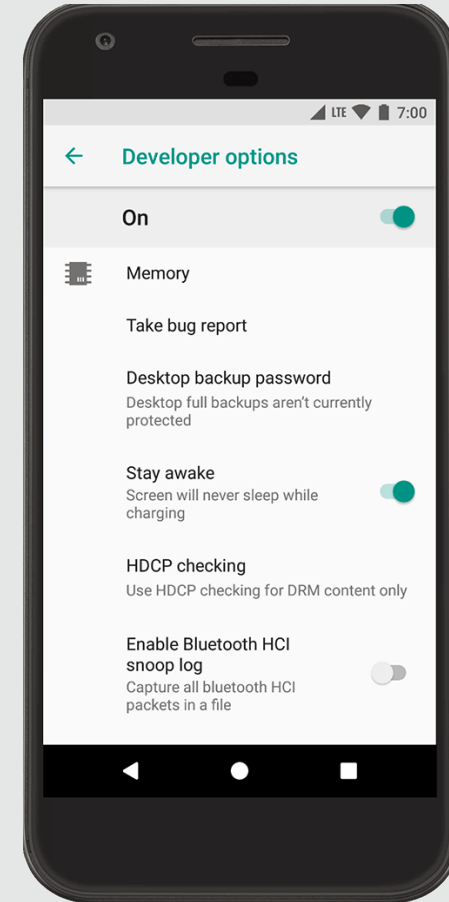
Lancement du projet Android

- Pour ajouter, modifier ou supprimer un émulateur, il suffit de se rendre dans le menu **Outils - Android** de Visual Studio et de cliquer sur l'entrée **Android Device Manager**



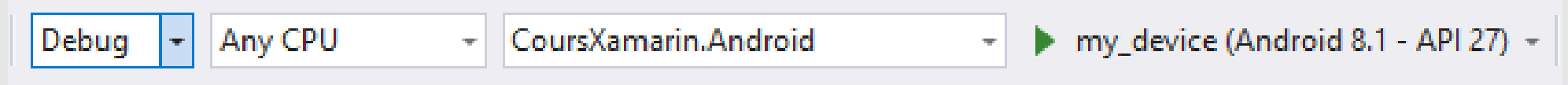
Lancement du projet Android

- Pour activer le mode développeur sur un terminal Android, dans les paramètres du téléphone, sélectionner **Système - À propos du téléphone** puis appuyer 7 fois sur l'entrée **Numéro de build**
- Une fois l'opération terminée, il suffit de brancher le téléphone au PC ou au Mac via un câble USB



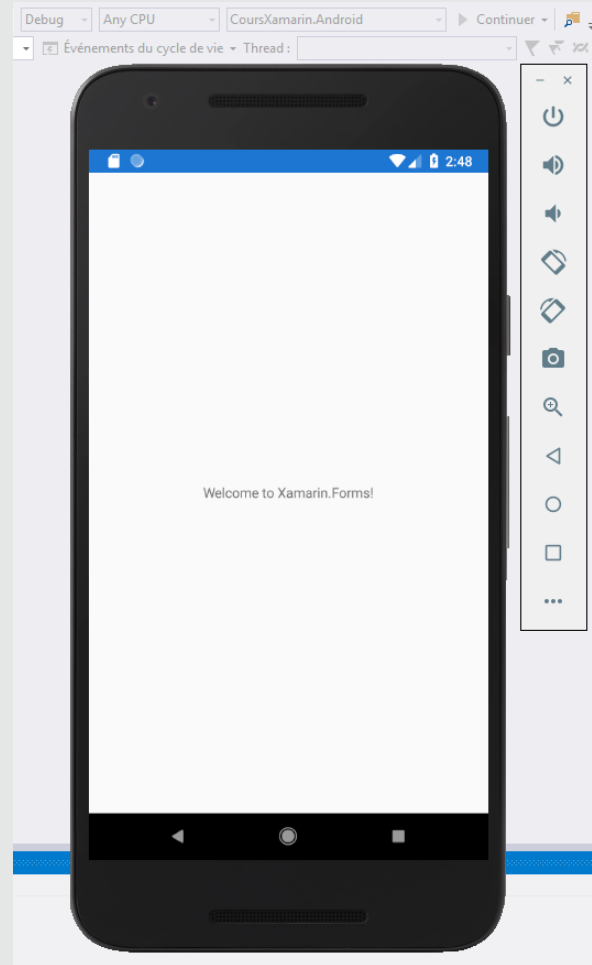
Lancement du projet Android

- Une fois l'émulateur installé ou un smartphone Android connecté, il suffit de sélectionner le projet Android comme cible et d'appuyer sur le bouton **Démarrer** de Visual Studio



Découverte de Xamarin.Forms

Lancement du projet Android



Lancement du projet Windows

- Si le projet Xamarin.Forms a été créé en supportant la plateforme Windows, il suffit de sélectionner le projet Windows comme cible et d'appuyer sur le bouton **Ordinateur local** de Visual Studio



Découverte de Xamarin.Forms

Lancement du projet Windows



Démonstration



Découverte de Xamarin.Forms

Lancer un projet Xamarin.Forms sur Android et Windows

- Lancer le projet Xamarin.Forms de base sur Android
- Lancer le projet Xamarin.Forms de base sur Windows



Découverte de Xamarin.Forms

Installer son environnement de développement

TP

