Le développement cross plateforme avec Xamarin

Module 7 – Développement de services Xamarin

Objectifs

- Connaître le rôle des services métiers et des services natifs dans une application Xamarin
- Savoir mettre en place un service métier
- Savoir mettre en place un service natif pour chaque plateforme
- Savoir utiliser l'injection de dépendances pour disposer du service adapté à la plateforme au lancement de l'application

Rôle du service dans une application Xamarin

Les **services** ont un rôle fondamental dans une application Xamarin : ils traitent les actions métier, mais c'est surtout le code qui peut être mutualisé entre toutes les plateformes

Les services Xamarin ont plus de responsabilités que des services classiques

Par exemple : la vérification du format d'un mail à l'aide d'une expression régulière, d'ordinaire développé dans le contrôleur, doit être déportée dans le service à des fins de mutualisation

Rôle du service dans une application Xamarin

- Les services Xamarin ont a minima les responsabilités suivantes :
  - Contrôles des données (expressions régulières, présence obligatoire, longueur des champs)
  - Décisions sur le workflow (navigation, erreurs, etc.)
  - Calculs métiers
  - Communication avec une API
  - Requêtes en base de données locale
  - Lecture/enregistrement de données locales

Mise en place d'un service métier

Les services métier, partagés entre toutes les plateformes sans code spécifique, sont de simples classes C# publiques.

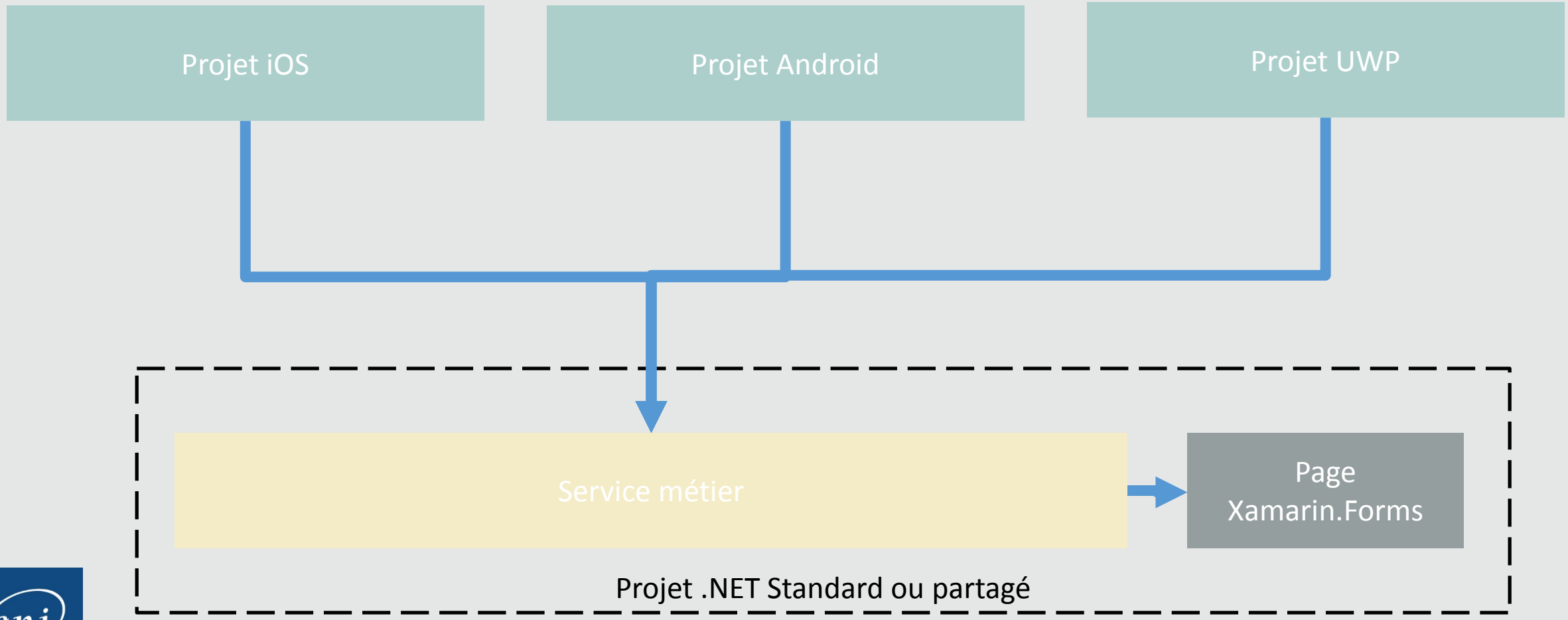
Ces services exposent des méthodes publiques, de préférence sans états, et reposent sur des méthodes privées qui participent à la bonne lisibilité et maintenabilité du code.

Mise en place d'un service métier

Attention : les services métier doivent être intégrés uniquement dans le projet partagé ou .NET Standard

De cette manière, les pages Xamarin.Forms peuvent accéder à ces services, ainsi que les pages natives iOS, Android et Windows.

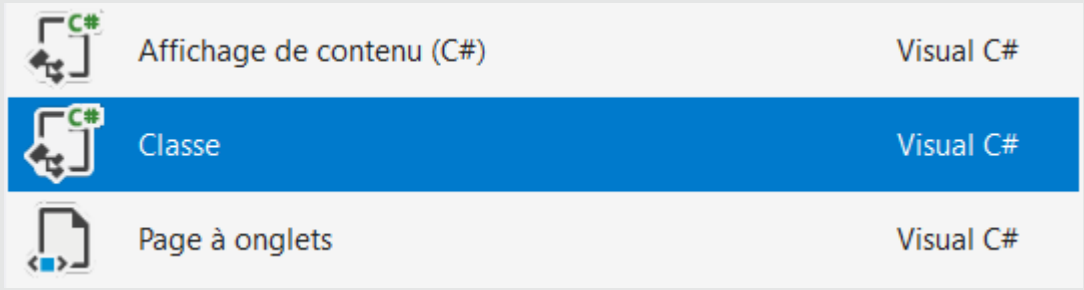
Mise en place d'un service métier



Mise en place d'un service métier

Pour créer un service métier, il suffit faire un clic droit sur le projet .NET Standard ou partagé et de sélectionner **Ajouter** puis **Nouvel Elément**.

Dans la fenêtre de création d'un nouvel élément, sélectionner **Classe Visual C#**



Mise en place d'un service métier

```
namespace CoursXamarin
{
    public class InscriptionSvc
    {
        public bool isMailCorrect(string mail)
        {…}

        public bool isPasswordCorrect(string password)
        {…}

        public bool register(string mail, string password)
        {…}
    }
}
```

Services nécessitant une implémentation native

L'architecture des services métiers ne permet pas de répondre à la contrainte de l'implémentation native

Dans le cas de l'implémentation native, le service est réécrit pour chaque plateforme afin de tirer parti du SDK natif

Exemple :

Pour interroger les capteurs du téléphone ou pour utiliser une bibliothèque dont l'implémentation est différente sur chaque plateforme (Firebase)

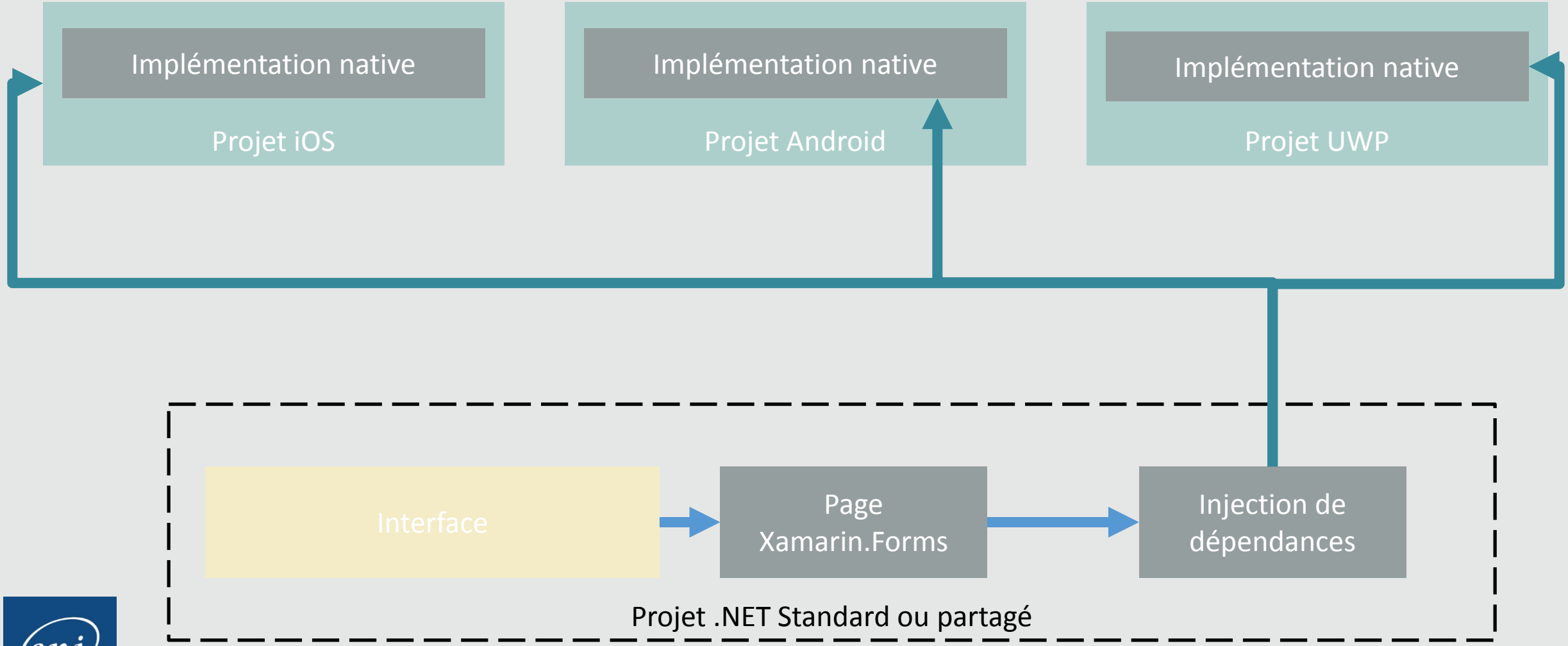
Services nécessitant une implémentation native

La base des services natifs consiste à implémenter une interface dans le projet partagé ou .NET Standard

Dans chaque projet natif, une classe implémente cette interface en tirant parti du SDK natif

Un mécanisme d'injection de dépendances permet de sélectionner et d'instancier dynamiquement la bonne classe au démarrage de l'application

Services nécessitant une implémentation native

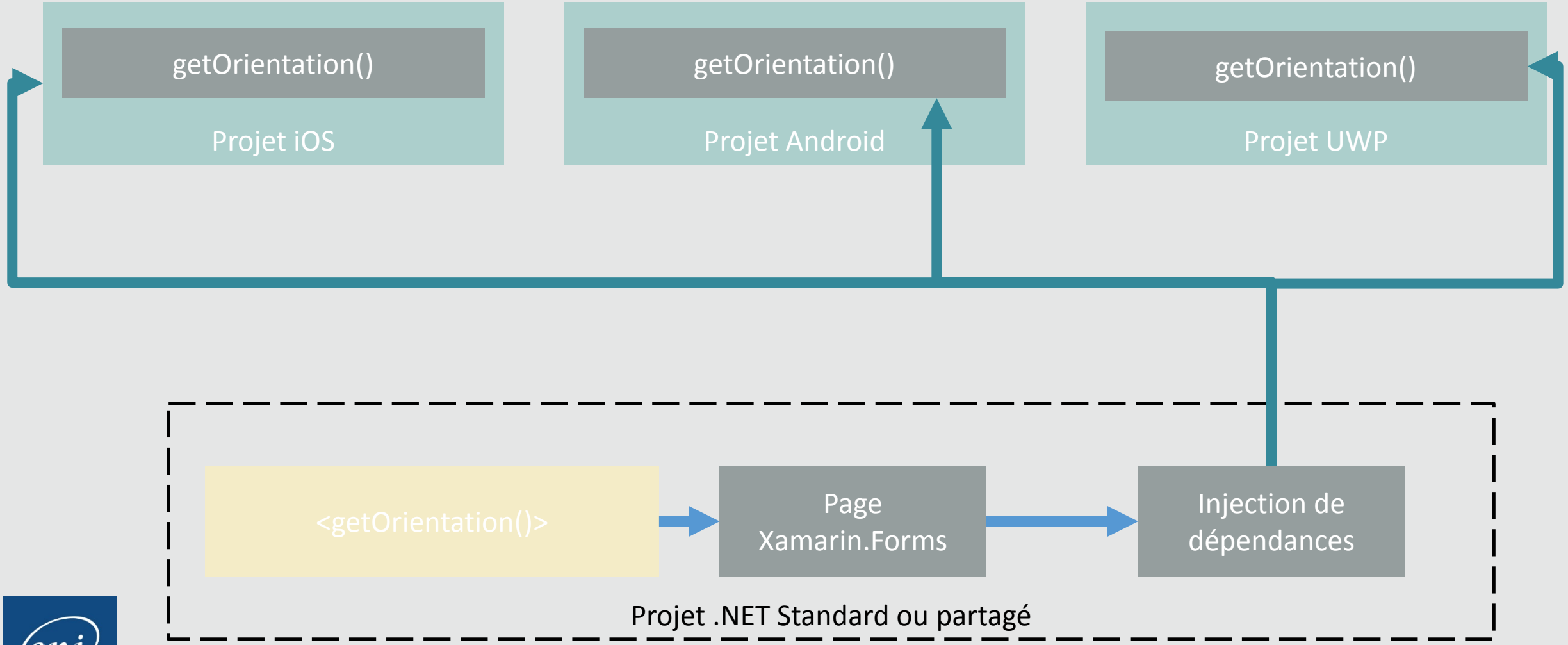


Exemple de service natif

L'objectif est de développer un service natif capable de nous renseigner sur l'orientation actuelle du téléphone (paysage ou portrait)

Cela ne peut être réalisé avec un service métier présent dans le projet .NET Standard, car il faut interroger le SDK natif de la plateforme afin de le déterminer

Exemple de service natif



Exemple de service natif

L'interface est créée dans le projet .NET Standard ou partagé

```
namespace CoursXamarin
{
    public enum DeviceOrientations
    {
        Undefined,
        Landscape,
        Portrait
    }

    public interface IDeviceOrientation
    {
        DeviceOrientations GetOrientation();
    }
}
```

Exemple de service natif

Dans le contrôleur qui souhaite interroger le service natif, il est nécessaire de faire appel au gestionnaire de dépendances afin de récupérer l'implémentation native courante.

```
var orientation = DependencyService.Get<IDeviceOrientation>().GetOrientation();

switch(orientation){…}
```

Exemple de service Android

L'implémentation Android se doit d'implémenter l'interface précédemment créée.

Elle interroge ensuite, en C#, le SDK Android natif et récupère une instance d'un service système WindowService qui donne des informations sur l'écran et l'orientation de celui-ci.

Exemple de service Android

```
namespace CoursXamarin.Droid
{
    public class DeviceOrientationImplementation : IDeviceOrientation
    {
        public DeviceOrientationImplementation() { }
        public static void Init() { }
        public DeviceOrientations GetOrientation()
        {
            IWindowManager windowManager =
       Android.App.Application.Context.GetSystemService(Context.WindowService)
              .JavaCast<IWindowManager>();
            var rotation = windowManager.DefaultDisplay.Rotation;
            bool isLandscape = rotation == SurfaceOrientation.Rotation90
               || rotation == SurfaceOrientation.Rotation270;
            return isLandscape ? DeviceOrientations.Landscape : DeviceOrientations.Portrait;
        }}}
```

Exemple de service Android

Il est indispensable d'ajouter une annotation à la classe nouvellement créée, **DeviceOrientationImpl**, pour qu'elle soit enregistrée par l'injecteur de dépendances comme étant une implémentation de l'interface **IDeviceOrientation**.

```
[assembly: Xamarin.Forms.Dependency (typeof (DeviceOrientationImplementation))]
namespace CoursXamarin.Droid {…}
```

Exemple de service iOS

L'implémentation iOS doit également implémenter l'interface précédemment créée.

Elle interroge ensuite, en C#, le SDK iOS natif et récupère directement l'orientation de l'écran à travers un singleton mis à disposition par le système : **UIApplication.SharedApplication**.

Exemple de service iOS

```
 namespace CoursXamarin.iOS
{
    public class DeviceOrientationImplementation : IDeviceOrientation
    {
        public DeviceOrientationImplementation() { }
        public DeviceOrientations GetOrientation()
        {
            var currentOrientation = UIApplication.SharedApplication.StatusBarOrientation;
            bool isPortrait = currentOrientation == UIInterfaceOrientation.Portrait
                || currentOrientation == UIInterfaceOrientation.PortraitUpsideDown;

            return isPortrait ? DeviceOrientations.Portrait : DeviceOrientations.Landscape;
        }
    }
}
```

Exemple de service iOS

Il est indispensable, comme pour chaque plateforme native, d'ajouter l'annotation permettant l'enregistrement auprès de l'injecteur de dépendances

```
[assembly: Xamarin.Forms.Dependency (typeof (DeviceOrientationImplementation))]
namespace CoursXamarin.iOS {…}
```

Exemple de service Windows

L'implémentation Windows doit implémenter l'interface précédemment créée.

Elle interroge ensuite, en C#, le SDK Windows natif et, de la même manière qu'iOS, récupère une instance d'un service système via un singleton pour récupérer l'orientation.

Exemple de service Windows

```
namespace DependencyServiceSample.WindowsPhone
{
    public class DeviceOrientationImplementation : IDeviceOrientation
    {
        public DeviceOrientationImplementation() { }
        public DeviceOrientations GetOrientation()
        {
            var orientation =
Windows.UI.ViewManagement.ApplicationView.GetForCurrentView().Orientation;
            if (orientation == Windows.UI.ViewManagement.ApplicationViewOrientation.Landscape)
            { return DeviceOrientations.Landscape; }
            else
            { return DeviceOrientations.Portrait; }
        }}}
```

Exemple de service Windows

Il est indispensable d'ajouter une annotation à la classe nouvellement créée, **DeviceOrientationImpl**, pour qu'elle soit enregistrée par l'injecteur de dépendances comme étant une implémentation de l'interface **IDeviceOrientation**.

```
[assembly: Dependency(typeof(DeviceOrientationImplementation))]
namespace CoursXamarin.WindowsPhone {…}
```

Démonstration

Développement de services Xamarin

Créer un service natif de synthèse vocale

- Créer une interface dans le projet .NET Standard
- Créer une implémentation native pour iOS
- Créer une implémentation native pour Android
- Créer une implémentation native pour Windows
- Lancer la synthèse vocale depuis une page Xamarin.Forms

Développement de services Xamarin

Installer son environnement de développement

TP