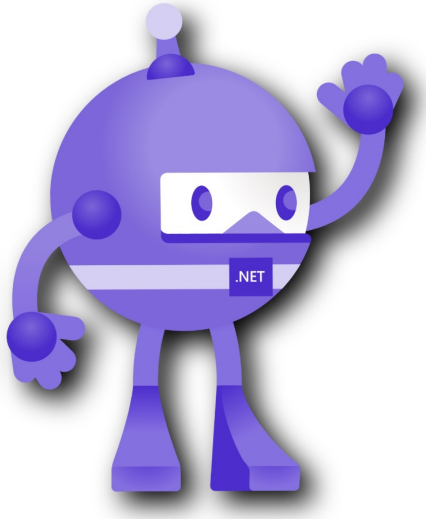
Xamarin / .NET MAUI



Institut Universitaire de Technologie du Limousin

Elise JOFFRE
elise@icohup.com

Université de Limoges

Xamarin, c’est quoi ?

Gratuit • Open source • Framework qui permet de créer des applications Android et iOS avec .NET et C#

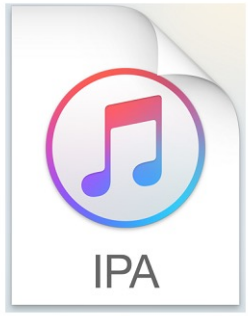
.NET est une plateforme de développement composée d'outils, de langages de programmation et de bibliothèques permettant de créer de nombreux types d'applications.

Xamarin étend la plateforme de développement .NET avec des outils et des bibliothèques spécifiquement pour créer des applications pour Android, iOS, tvOS, watchOS, macOS et Windows.

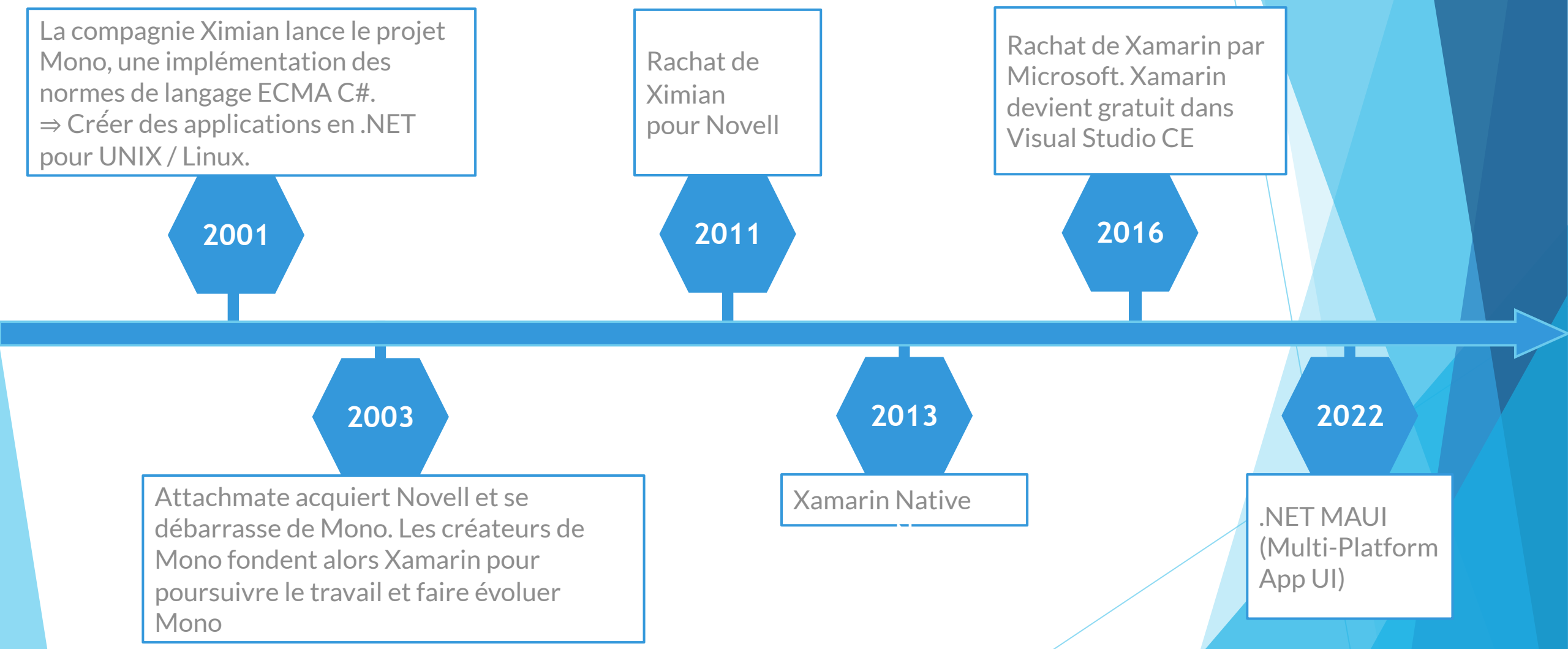
Xamarin, c'est quoi ?

Xamarin permet aux développeurs de **partager en moyenne 90 % de leur application entre les plateformes**. Ce modèle permet aux développeurs d'écrire toute leur logique métier dans un seul langage (ou de réutiliser du code d'application existant) tout en obtenant des performances et une apparence natives sur chaque plateforme.

Les applications Xamarin peuvent être écrites sur PC ou Mac et être compilées dans des paquets d'application natifs, par exemple un fichier **.apk** sur Android ou un fichier **.ipa** sur iOS.

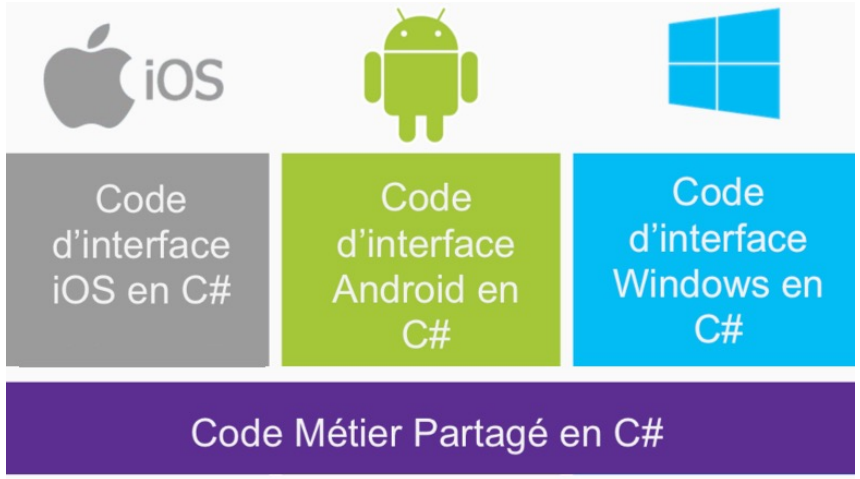


Petit retour en arrière



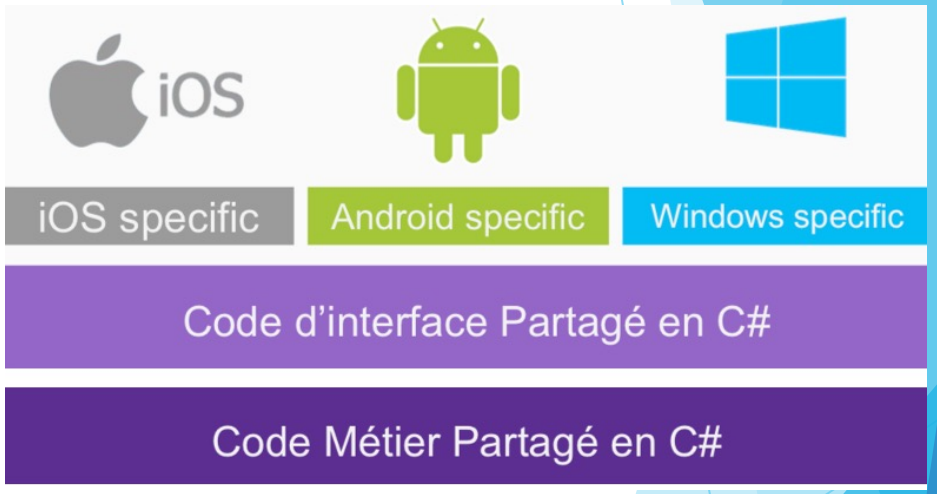
Plus simplement !

Xamarin



⇒ Utilisé pour des applications plus complexes, avec des spécificités / plateforme

Xamarin Forms



⇒ Utilisé pour des applications sans trop de particularités

Pourquoi alors Xamarin Forms ?

- Un seul code et une seule UI
- Une compétence C# et XAML à réutiliser
- Réutilisation des outils (Visual Studio, Unit testing..)
- Réutilisation des méthodes

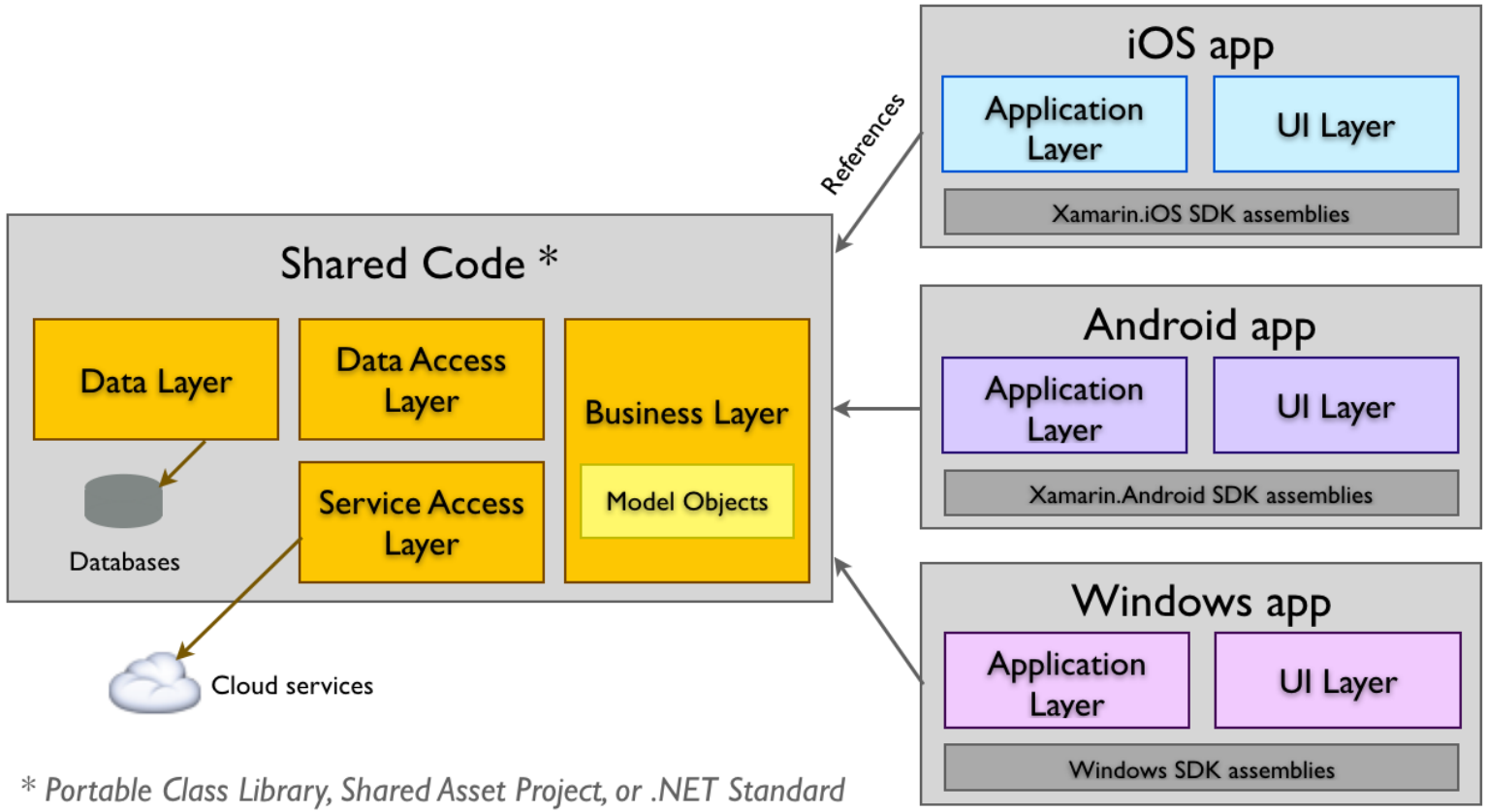
Gain de temps et d'argent !

Il y a quand même des limites..

- Connaitre le C#
- Des composants complexes à redévelopper par plateforme
- Des connaissances nécessaires en iOS
- Des connaissances nécessaires en Android

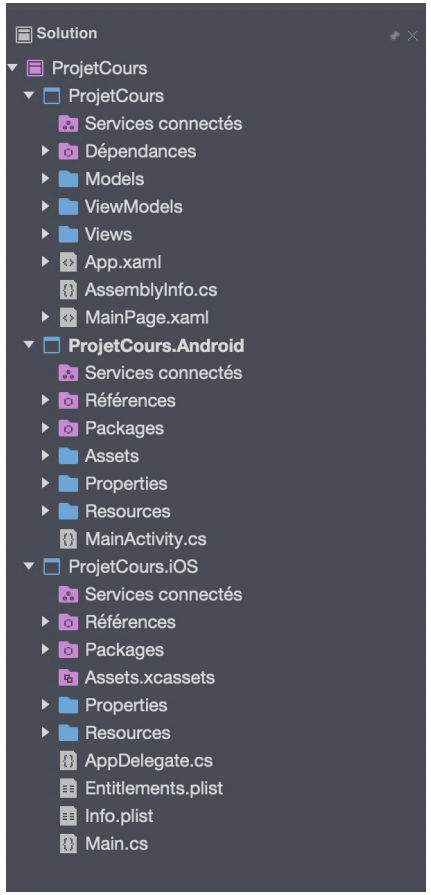
Bein réfléchir en début de projet pour utiliser la bonne techno !

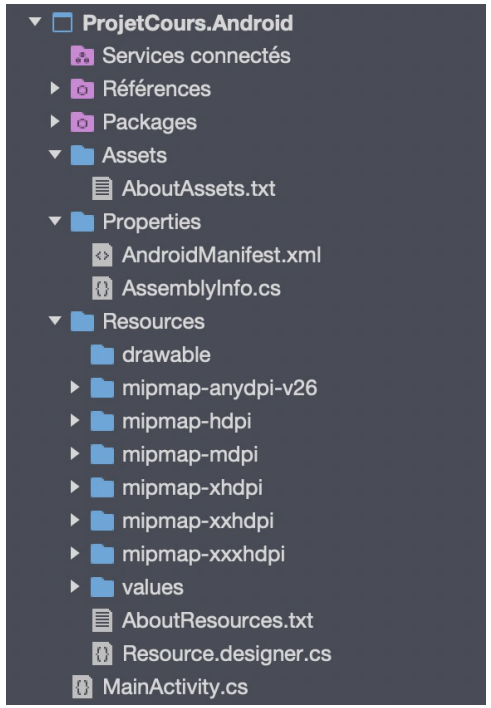
Un projet en Xamarin

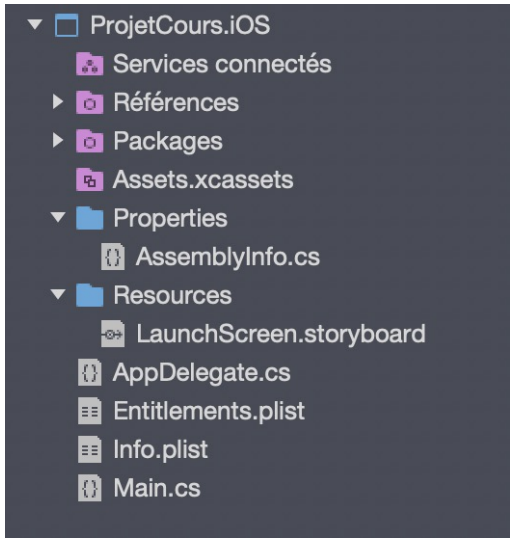


* *Portable Class Library, Shared Asset Project, or .NET Standard*

Un projet en Xamarin







Visual Studio ou Rider

Structure du projet

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

    void Handle_Clicked(object sender, System.EventArgs e)
    {
        boxview.BackgroundColor = Color.Red;
    }
}
```


```
<StackLayout HeightRequest="150"
             HorizontalOptions="Center"
             VerticalOptions="Center">

    <Label Text="Welcome to Xamarin.Forms!"
           TextColor="Purple"
           VerticalOptions="CenterAndExpand" />

    <Button Text="Cliquez moi !"
            WidthRequest="150"
            BackgroundColor="Teal"
            TextColor="White"
            Clicked="Handle_Clicked"
            VerticalOptions="Center"
            HorizontalOptions="CenterAndExpand"/>

    <BoxView WidthRequest="200"
             x:Name="boxview"
             BackgroundColor="Purple"
             VerticalOptions="Center"
             HorizontalOptions="CenterAndExpand"
             HeightRequest="100"/>

</StackLayout>
```

Du côté du C# …

C# est un langage introduit par Microsoft en 2000. C'est un langage objet avec un typage statique fort, une syntaxe héritée du C/C++ et une philosophie très proche de Java. C# est un langage phare du framework .net qui se popularise pour la conception de sites Web (ASP), d'ERP (Sharepoint), de scripting et d'applications lourdes.

La syntaxe du C# est très proche du C et de Java.

https://docs.microsoft.com/fr-fr/dotnet/csharp/programming-guide/

Les variables

```
var nomPersonne = "Dupond"; // type String implicite
String nomPersonne; // type String explicite
Vehicule porscheVehicule = new Voiture(250); // type parent explicite
```

Les conditionnelles

```
If (condition1)  {

}
else if (condition2){

}else{

}
```

```
switch (legume)  {
      case  « Carottes »:

            break;
      default:

            break;
}
```

Les boucles

```
var legumes = new []{"carottes", "haricots"};

for (var i = 0; i<legumes.Count(); i++) {
    var legume = legumes[i];
}

// OU

foreach (var legume in legumes) {
}
// OU

var i = 0;
while (i < legumes.Count()) {
    var legume = legumes[i];
    i++;
}
```

Les fonctions

```
// Fonction qui prend un paramètre (obligatoire) et qui ne retourne rien
void DisplayToday(bool displayTime) {
	// TODO
}


DisplayToday(true);


// Fonction qui prend 2 paramètres (dont 1 avec une valeur par défaut), et retourne un entier
int GetDay(DateTime uneDate, bool dayOfYear = false) {
	return 0;
}

var dayOfMonth = GetDay(DateTime.Today);
var dayOfYear = GetDay(DateTime.Today, true)
```

Les classes

```
public class Voiture {

        // Propriété public en lecture, protected en écriture
        // Commence toujours par une majuscule !
        public Double VitesseMax {
                get; protected set;
        }

        // Constructeur avec valeur par défaut pour le paramètre
        public Voiture (Double vitesseMax = 0 ){
                this.VitesseMax = vitesseMax;
        }
}
```

Les fichiers

.cs Code logique

.xaml Interface graphique

.xaml.cs Code logique lié à un fichier xaml

Shell

L'interpréteur de commandes réduit la complexité du développement d'applications mobiles en fournissant les fonctionnalités fondamentales nécessaires à la plupart des applications mobiles, notamment :

- Un emplacement unique pour décrire la hiérarchie visuelle d'une application.
- Une expérience utilisateur de navigation commune.
- Un schéma de navigation basée sur des URI permettant la navigation vers n'importe quelle page dans l'application.
- Un gestionnaire de recherche intégré.

En outre, les applications Shell bénéficient d'une vitesse de rendu accrue et d'une consommation de mémoire réduite.

Hiérarchie visuelle de l'app

Dans une Xamarin.Forms application d'interpréteur de commandes, la hiérarchie visuelle de l'application est décrite dans une classe qui sous-classe la Shell classe. Cette classe peut être composée de trois objets hiérarchiques principaux :

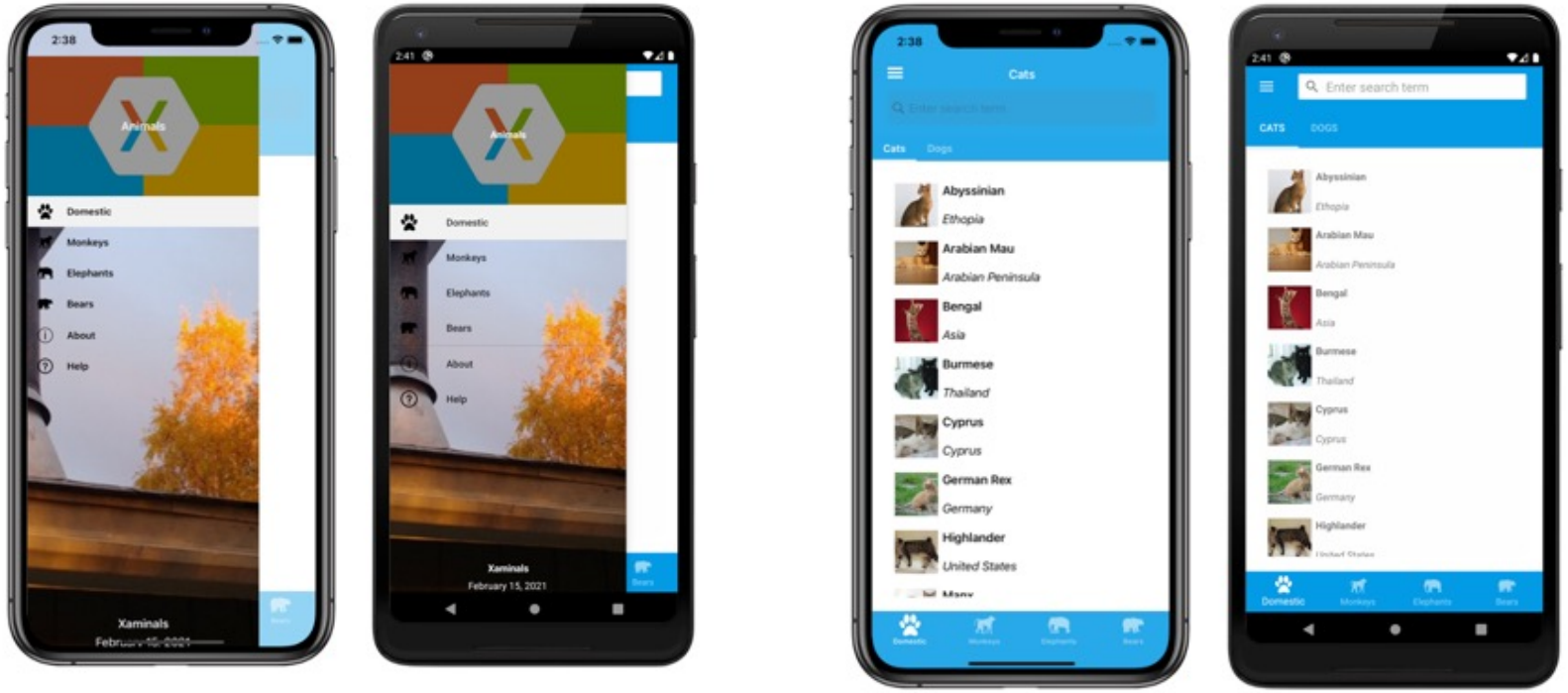
FlyoutItem. Un FlyoutItem représente un ou plusieurs éléments dans le menu volant et doit être utilisé lorsque le modèle de navigation de l'application requiert un menu volant.

Tab, qui représente le contenu regroupé, accessible via les onglets inférieurs.

ShellContent, qui représente les **ContentPage** objets de chaque onglet.

Ces objets ne représentent pas une interface utilisateur, mais plutôt l'organisation de la hiérarchie visuelle de l'application. Shell sélectionnera ces objets et générera l'interface utilisateur de navigation pour le contenu.

Visuellement ..



Les groupes principaux

Pages

- Représente les différents écrans
- iOS : `UIViewController` | Android : `ViewGroup`

Layouts

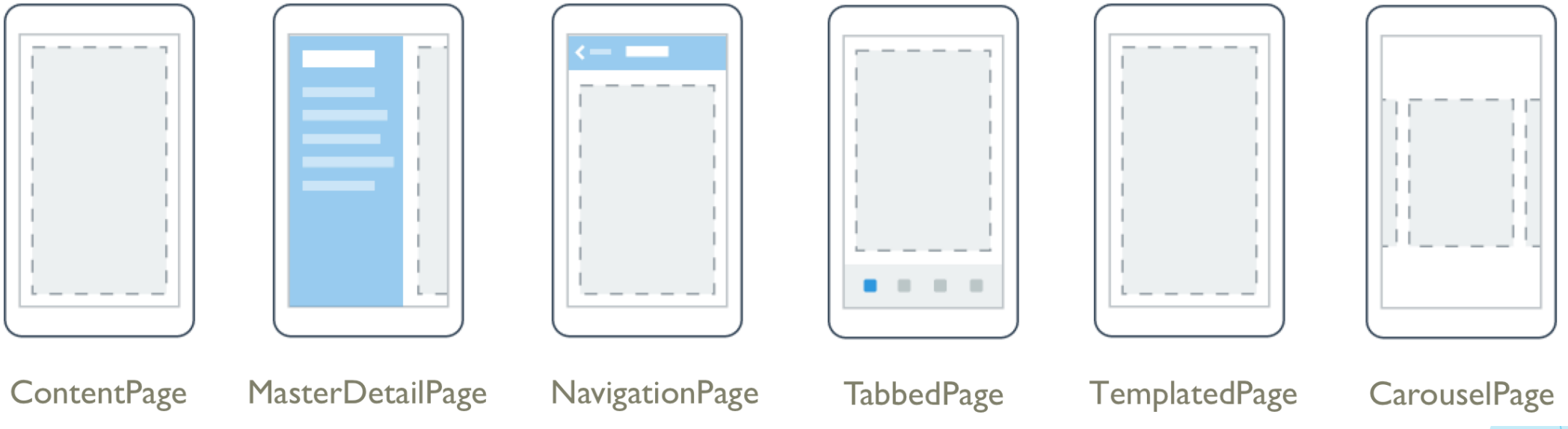
- Représente les conteneurs
- iOS : `UIView` | Android : `FrameLayout, LinearLayout`

Views

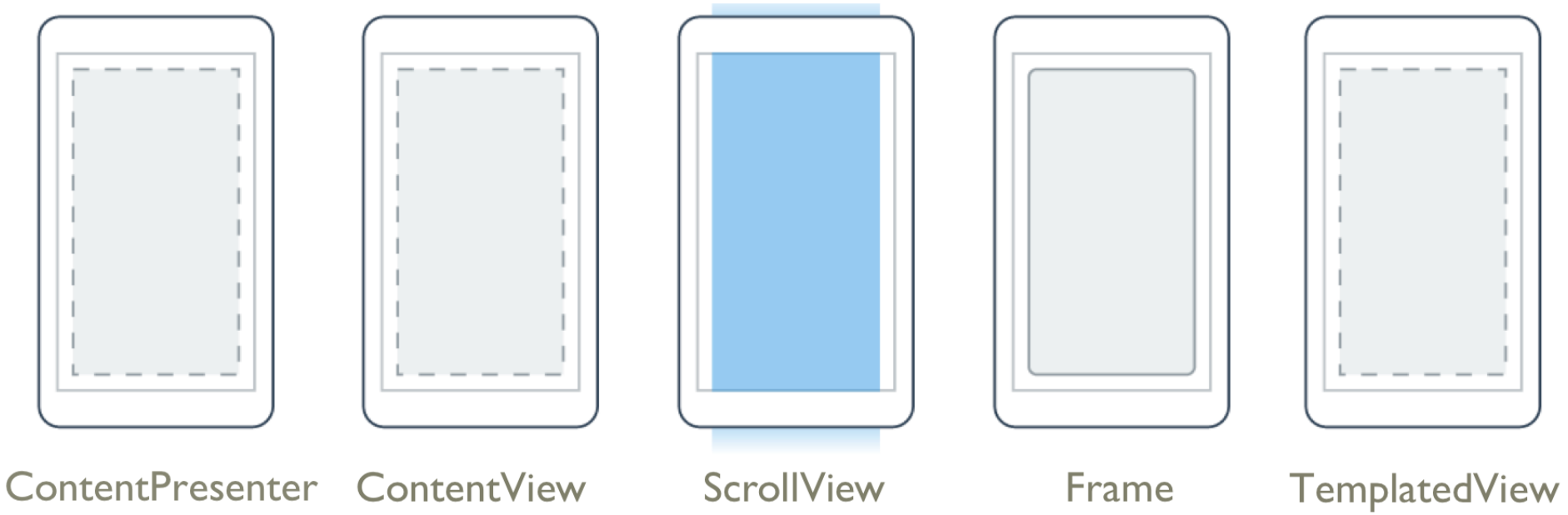
- Représente les éléments d'interface (Boutons, Label, …)

Cells

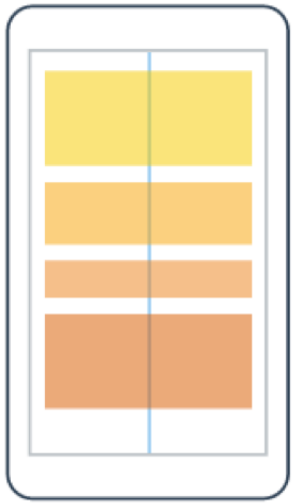
Les pages


ContentPage
MasterDetailPage
NavigationPage
TabbedPage
TemplatedPage
CarouselPage

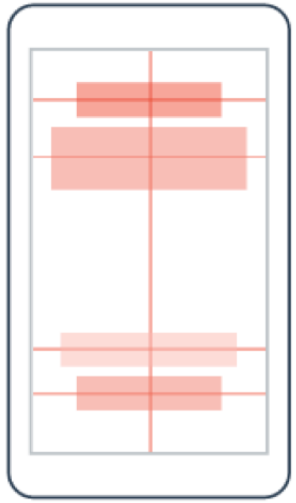
Les views


ContentPresenter
ContentView
ScrollView
Frame
TemplatedView

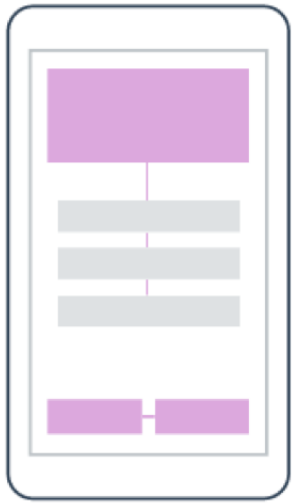
Les layouts



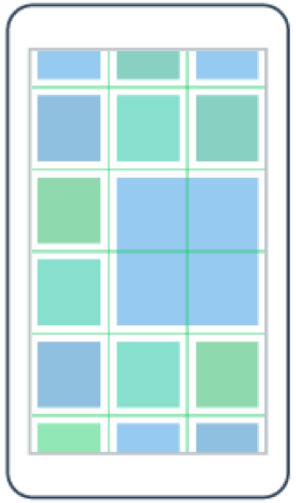
StackLayout



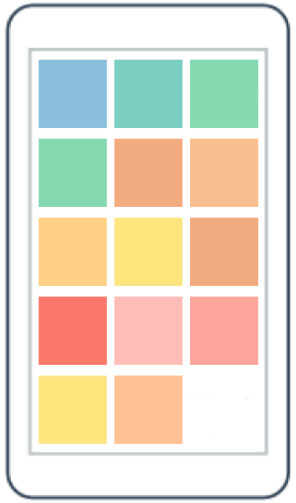
AbsoluteLayout



RelativeLayout



Grid



FlexLayout

Les views pour la **présentation**

BoxView	Ellipse	Label
Line	Image	Map
OpenGLView	Path	Polygon
Polyline	Rectangle	WebView

Les views **lançant des commandes**

- Button
- ImageButton
- RadioButton
- RefreshView
- SearchBar
- SwipeView

Les views pour les valeurs des paramètres

CheckBox

Slider

Stepper

Switch

DatePicker

TimePicker

Les views pour la modification de texte

Entry

Editor

Les views pour indiquer une activité

ActivityIndicator

ProgressBar

Les views **qui montrent des collections**

- CarouselView
- CollectionView
- IndicatorView
- ListView
- Picker
- TableView

Les cells

TextCell

ImageCell

EntryCell

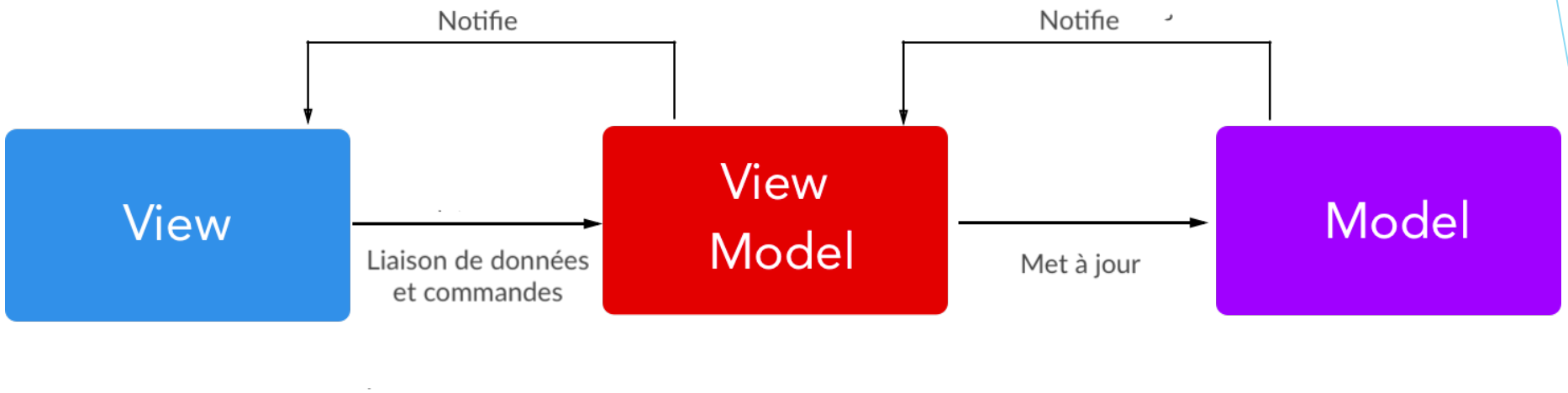
SwitchCell

Interfaces


Performance Dashboard
Tasks performance
Last week
244
Open Tasks
129
Day Services
190
Active Users
Team Member
Open
Closed
Score
Regina Joplin
Donald Davies
Employee Performance
Checkout
Social Card
Clare Japi
Monterrey
Shipping Details
Complete
Open

Bien ranger son code !

Pattern MVVM



Pattern MVVM : Composants

Model :

- Contient les données

ViewModel

- Lien de connexion entre le model et la view
- Convertit les objets de données du modèle de sorte que les objets soient facilement gérés et présentés.

View :

- Représente les composants de l'interface graphique

Model

```
public class Car
{
    public int CarID { get; set; }
    public string Make { get; set; }
    public int YearOfModel { get; set; }
}
```

ViewModel

```
public class CarsViewModel : INotifyPropertyChanged
{

    private ObservableCollection<Car> items;

    public event PropertyChangedEventHandler PropertyChanged;

    public ObservableCollection<Car> Items
    {
        get { return items; }
        set
        {

            items = value;
        }
    }
```

ViewModel

```
public CarsViewModel()
{
    Items = new ObservableCollection<Car>() {
        new Car()
        {
            CarID = 1,
            Make = "Tesla Model S",
            YearOfModel = 2015
        },
        new Car()
        {
            CarID = 2,
            Make = "Audi R8",
            YearOfModel = 2012
        },

    };

}
```

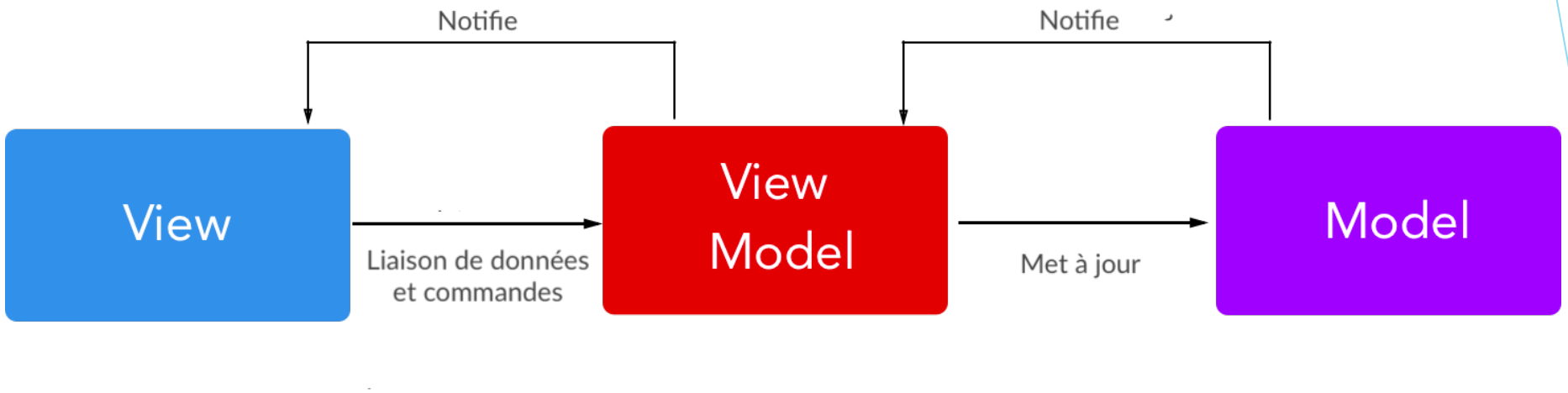
View

```
public CarsView()
{
    InitializeComponent();
    BindingContext = new CarsViewModel();
}
```

View

```
<ListView
    CachingStrategy="RecycleElement"
    ItemsSource="{Binding Items}"
    RowHeight="70">
    <ListView.ItemTemplate>
        <DataTemplate>
            <ViewCell>
                <StackLayout Margin="8">
                    <Label
                        FontAttributes="Bold"
                        FontSize="Subtitle"
                        Text="{Binding Make}" />
                    <Label
                        FontAttributes="None"
                        FontSize="Body"
                        Text="{Binding YearOfModel}" />
                    <Line Stroke="Gray" />
                </StackLayout>
            </ViewCell>
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Pattern MVVM



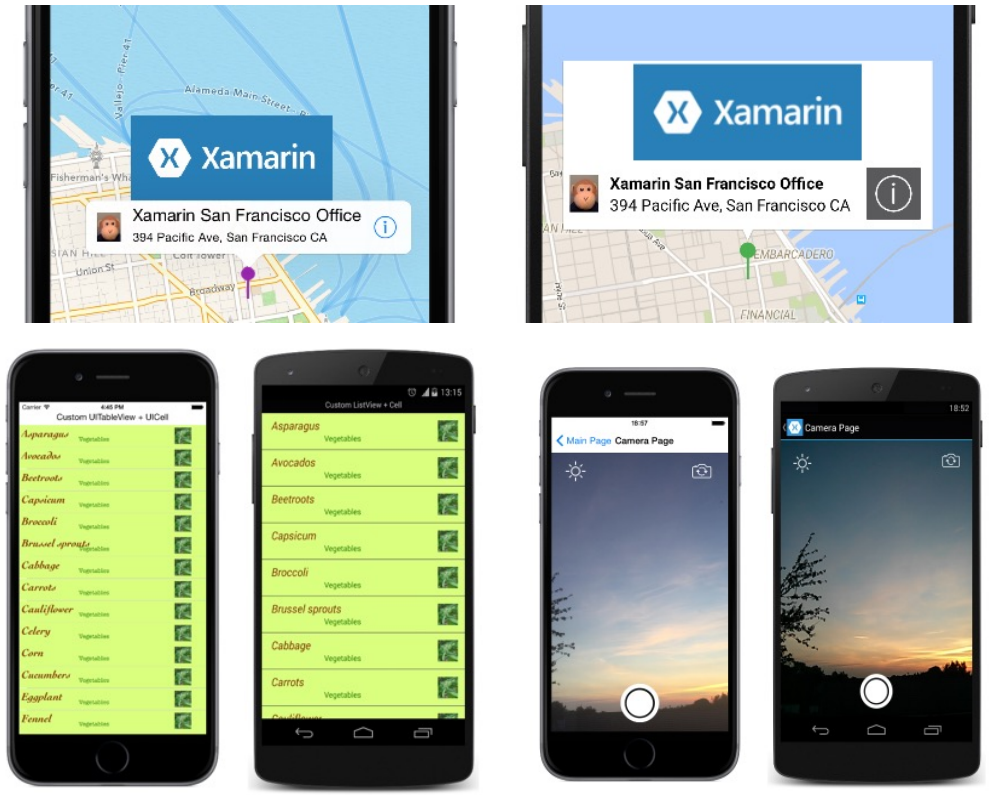
Nugets

Plusieurs bibliothèques open source qui vont permettre de réaliser plusieurs fonctionnalités:

- Cartes
- Graphes
- Bluetooth
- Usb
- MySQL
- Camera
- Json
- Api
- Etc ….

Renderers

Chaque Page, Layout ou Views peut être rendu différemment sur chaque plateforme, à l'aide d'une Renderer classe qui crée à son tour un contrôle natif (correspondant à la Xamarin.Forms représentation), le réorganise à l'écran et ajoute le comportement spécifié dans le code partagé



Evolution .NET MAUI

- Intégré avec .NET 6
- Structure du projet plus partagée
- Customisation plus simple
- Les ressources sont uniques dans le projet
- Rechargement à chaud pour toutes les plateformes
- Plus d'API pour les graphismes
- Ouverture des fenêtres de rendu sur toutes les plateformes en même temps

Webographie

- *https://www.inflexsys.com/application-native-ou-hybride/*
- *https://www-pixelcrayons-com.cdn.ampproject.org/c/s/www.pixelcrayons.com/blog/revealed-top-10-reasons-why-xamarin-is-the-best-platform-for-mobile-app-development-in-2021/amp/*
- *https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/app-fundamentals/shell/introduction*
- *https://www.nuget.org/profiles/Xamarin*
- *https://www.syncfusion.com/blogs/post/xamarin-versus-net-maui.aspx*