



# Développement d'application cross- platform avec Xamarin

Julien Mialon : [mialon.julien@gmail.com](mailto:mialon.julien@gmail.com)

Valentin Jubert : [valentin.jubert@outlook.fr](mailto:valentin.jubert@outlook.fr)

# Qui sommes-nous ?

## Julien Mialon

- Diplômé en 2014
- Dev Mobile depuis 2011 et Xamarin depuis 2014
- Banque Populaire, Voyages-SNCF, Alerto, Idelink
- Lead dev chez Genius Sport

## Valentin Jubert

- Diplômé en 2017
- Dev Xamarin depuis 2015
- Alerto, Idelink, Stibus, Amapez-Vous
- Dev Xamarin chez Ideine


# Au sommaire...

## Séance 1

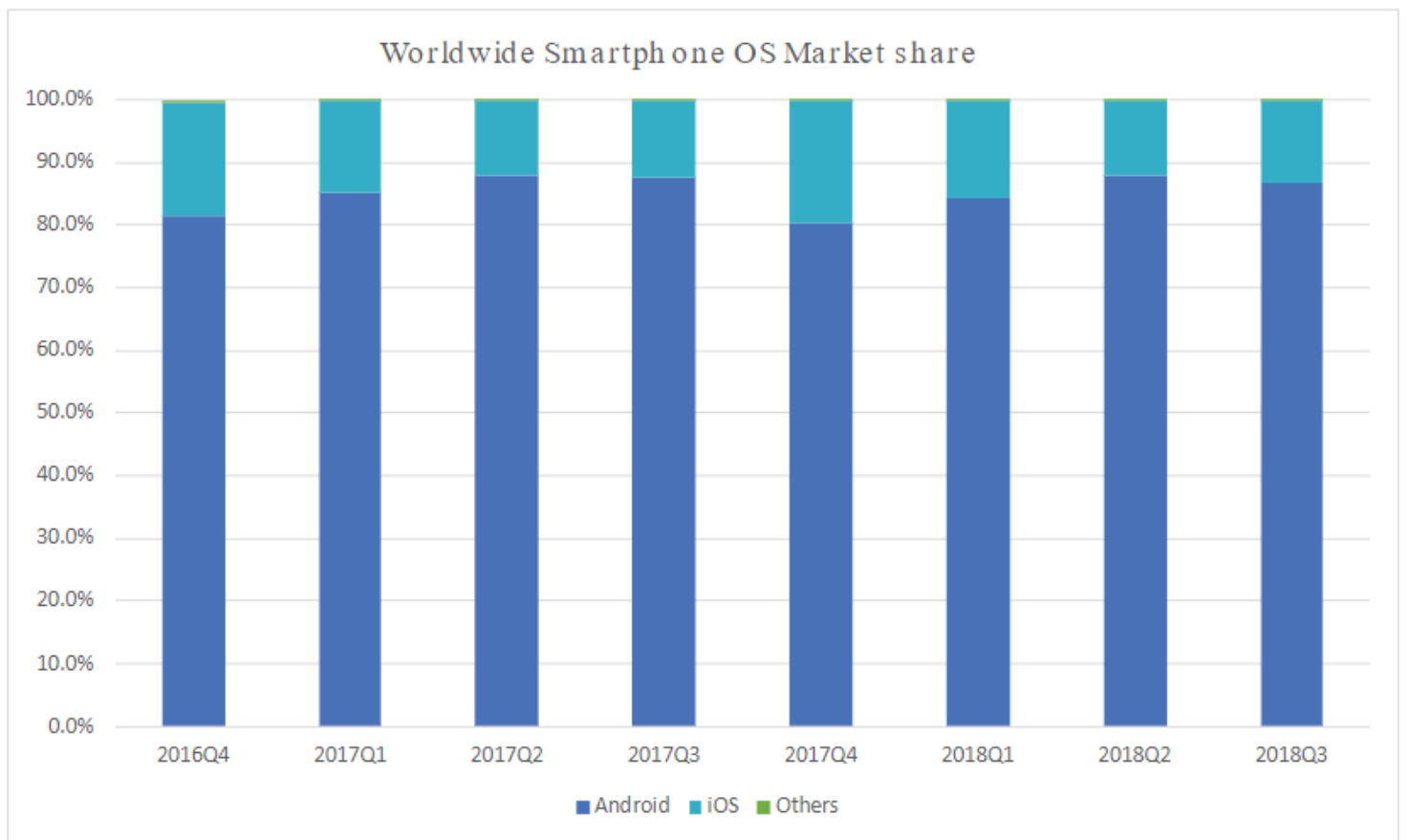
- Écosystème mobile
- Natif ou Cross-Platform?
- Xamarin et ses outils
- C# & .NET
- Xamarin.Forms

## Séance 2

- Xamarin.Forms avancé
- DevOps (CI / CD / CT) + AppCenter
- OAuth2 password flow
- Push notifications
- Cache
- Architectures
- Aperçu de Flutter et ReactNative
- Publication sur les stores

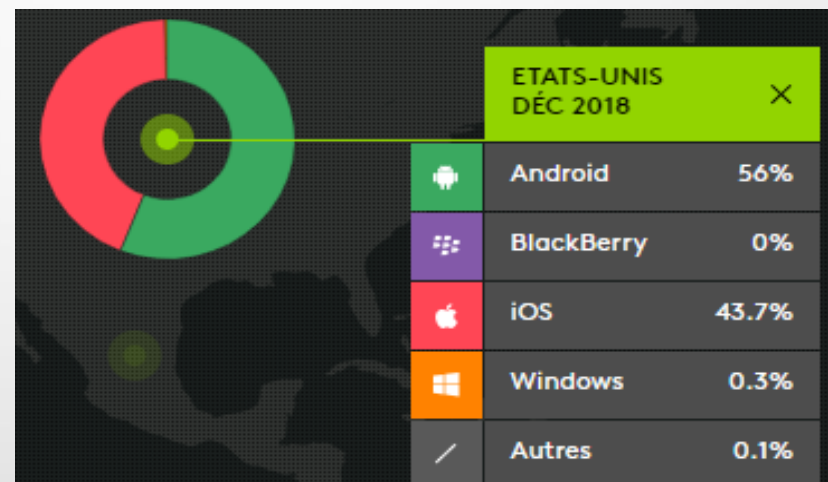
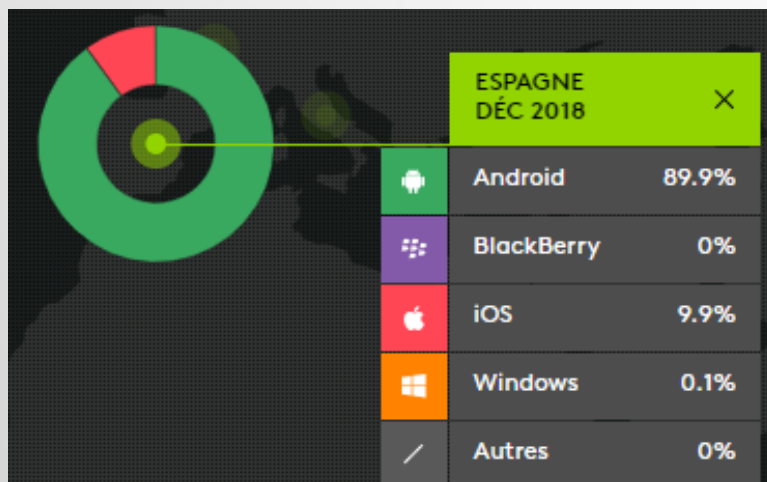
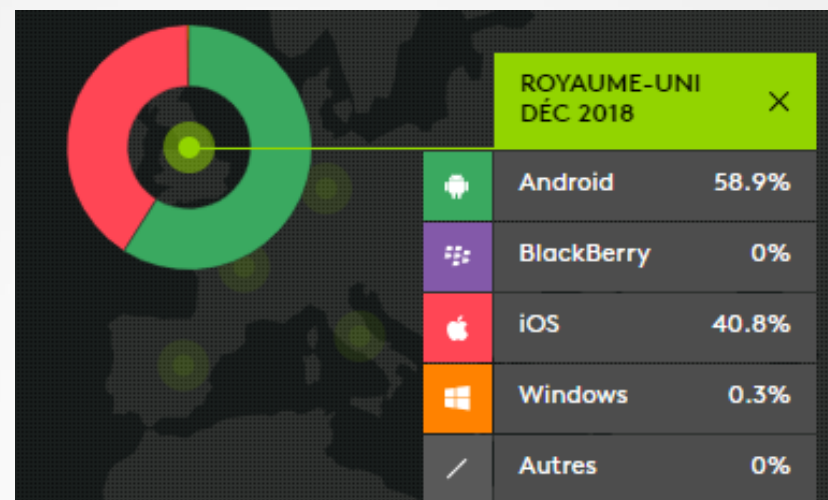
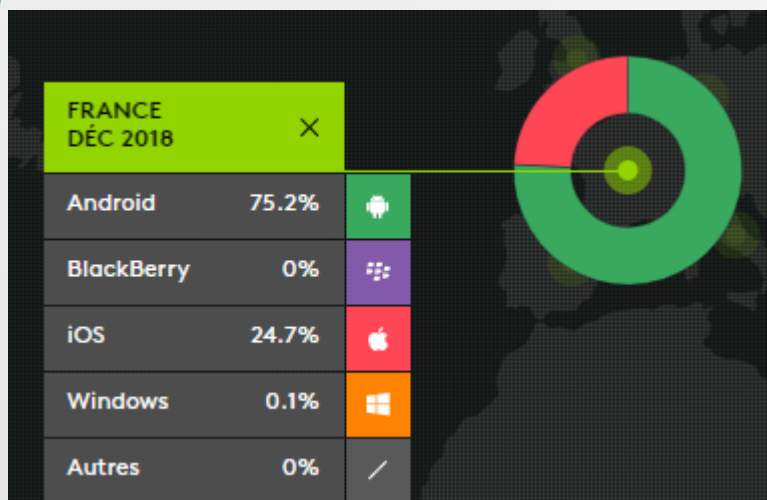


# Écosystème mobile



Quarter	2016Q4	2017Q1	2017Q2	2017Q3	2017Q4	2018Q1	2018Q2	2018Q3
Android	81,4%	85,0%	88,0%	87,6%	80,3%	84,3%	87,8%	86,8%
iOS	18,2%	14,7%	11,8%	12,4%	19,6%	15,7%	12,1%	13,2%

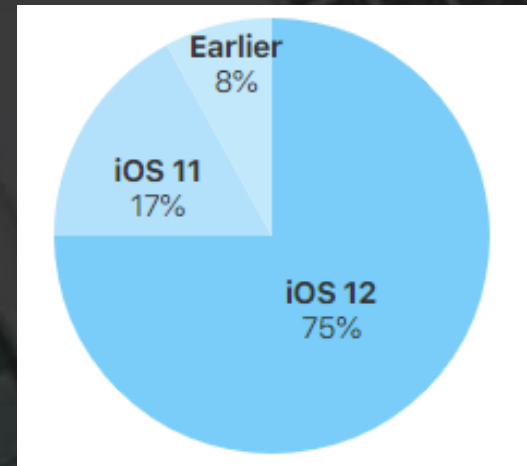
Source : IDC.com



Source : [kantardworldpanel.com](http://kantardworldpanel.com)

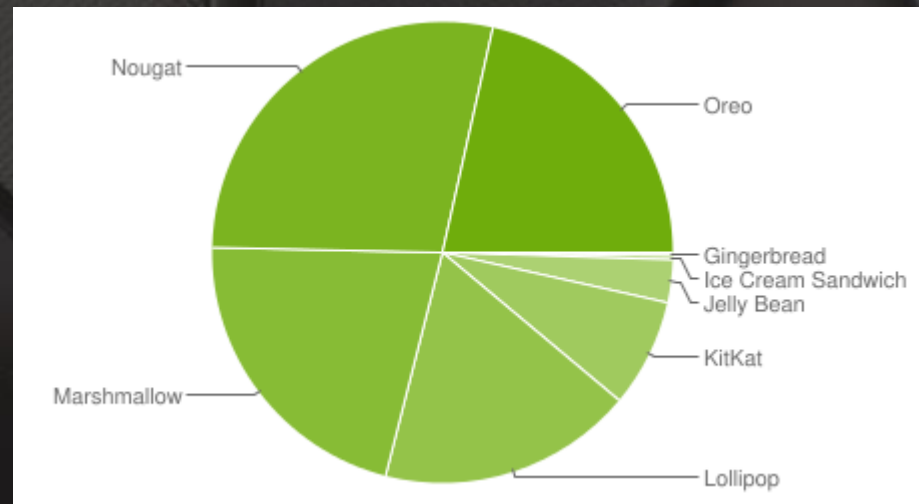
# iOS

- 21 iPhone + 6 iPod Touch
- 8 iPad + 4 iPad mini + 6 iPad Pro
- 5 Apple TV
- 5 Apple Watch



# Android

- +400 marques
- +4000 appareils







Natif ou Cross-Platform

# Développement

## iOS

- Un Mac
- XCode
- Swift, Objective-C ou C++
- Framework iOS

## Android

- Linux, Windows ou macOS
- Android Studio
- Kotlin, Java ou C++
- Framework Java + Android

# Ressources pour développer une application iOS + Android

- 1 développeur Android + 1 PC
- 1 développeur iOS + 1 Mac
- 2 plateformes avec 2 codes très ressemblant
  - 2x plus de bugs
  - 2x plus de maintenance
  - 2 développeurs qui ne se comprennent pas

# Le XPlatform à la rescousse

- Apache Cordova (anciennement PhoneGap) depuis 2008
- Appcelerator Titanium depuis 2010
- Progressive Web App depuis 2018
  
- Xamarin depuis 2009
- React Native depuis 2015
- Kotlin native (unstable)
  
- Flutter depuis décembre 2018
- Et des jeux vidéos avec Unity, Cocos2D, ...

# Cordova & Titanium (2010)

## Avantages

- JavaScript, HTML
- Un seul code pour iOS & Android
- Accès via plugin natif aux API natives de chaque plateforme

## Inconvénients

- JavaScript
- Rendu dans une webview
  - Incompatibilité entre Safari, Chrome et autres
- Performance
- Même design sur iOS & Android
- Plugin natif

# React Native (2015)

## Avantages

- TypeScript, Flow ou JavaScript
- Compilé vers du natif
- Un seul code
- Hot Reload Debugging
- OSS

## Inconvénients

- JavaScript
- Pas d'accès direct aux contrôles natif
- Plugin développé en natif (Java / Obj-C)
- Plusieurs retours négatifs
- Facebook

# Flutter (Décembre 2018)

## Avantages

- Compilé vers du natif
- Un seul code
- Hot Reload Debugging
- Supporté par Google
- OSS

## Inconvénients

- Dart
- Pas de contrôles natif
- Plugin développé en natif (Java / Obj-C)
- Pas d'unanimité chez Google

# Xamarin (2009)

## Avantages

- C# ou F# + .NET
- Environ 40/50% de code partagé
- Compilé vers du natif pour iOS / VM pour Android (CLR)
- Accès à toutes les APIs natives
- 100K librairies + binding iOS Android
- Supporté par Microsoft
- OSS

## Inconvénients

- Pas de Hot Reload Debugging
- Temps de compilation
- Nécessaire de connaître le développement natif



# Xamarin.Forms (2014)

## Avantages

- Environ 80/90% de code partagé
- Partage du code UI
- OSS

## Inconvénients

- Perte de performance

# Kotlin Native (unstable)

## Avantages

- Kotlin
- Interopérabilité avec Java / Swift
- Supporté par JetBrains
- OSS

## Inconvénients

- Nécessaire de connaître le développement natif
- Code UI iOS à écrire en Swift

# Unity

## Avantages

- C#
- Basé sur Mono & Xamarin
- Prévu pour du jeu vidéo
- Une très grande communauté
- Plein de plugins pour tout

## Inconvénients

- Pas prévu pour de l'appli (même si c'est possible)
- Un prix élevé





# Xamarin et ses outils

# Un peu d'histoire

- Mono : implémentation libre de .NET créé en 2001 par Miguel De Icaza et Nat Friedman
- MonoTouch : .NET pour iOS en 2009
- MonoDroid : .NET pour Android en 2011
- Xamarin fondé en 2011
- Microsoft rachète Xamarin en 2016 et le rend gratuit et open source

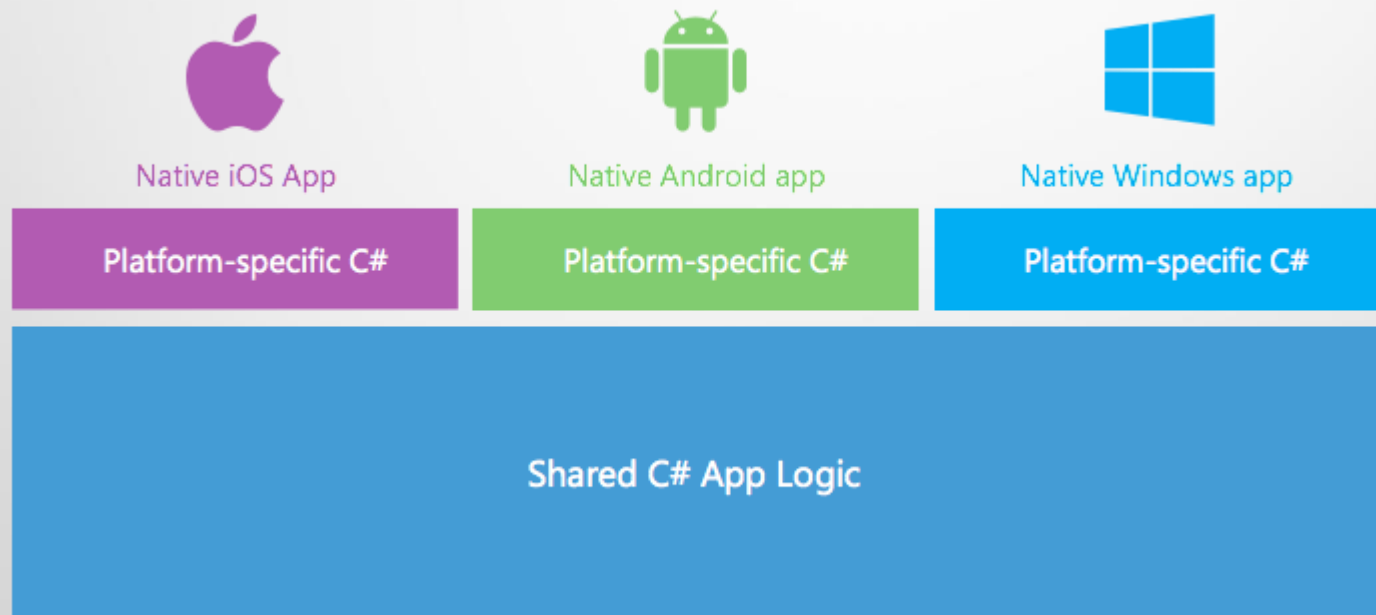
# Un peu d'histoire

- Miguel de Icaza
  - Fondateur du projet GNOME en 1997
  - Co-fondateur de Ximian en 1999
  - Fondateur du projet Mono en 2001
  - Co-fondateur de Xamarin en 2011
- Il annonce en mars 2016 avoir terminé l'entretien d'embauche le plus long de sa vie et rejoint Microsoft suite au rachat de Xamarin

# Un peu d'histoire

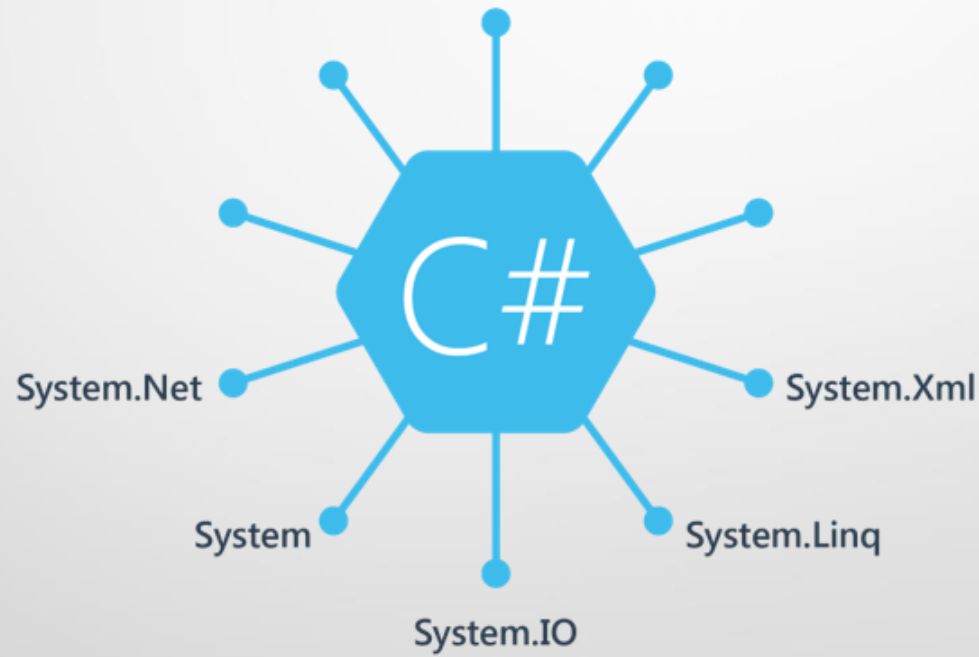
- Nat Friedman
  - Co-fondateur de Ximian en 1999
  - Co-fondateur de Xamarin en 2011
  - CEO de Github depuis octobre 2018

# Xamarin : comment ça marche ?

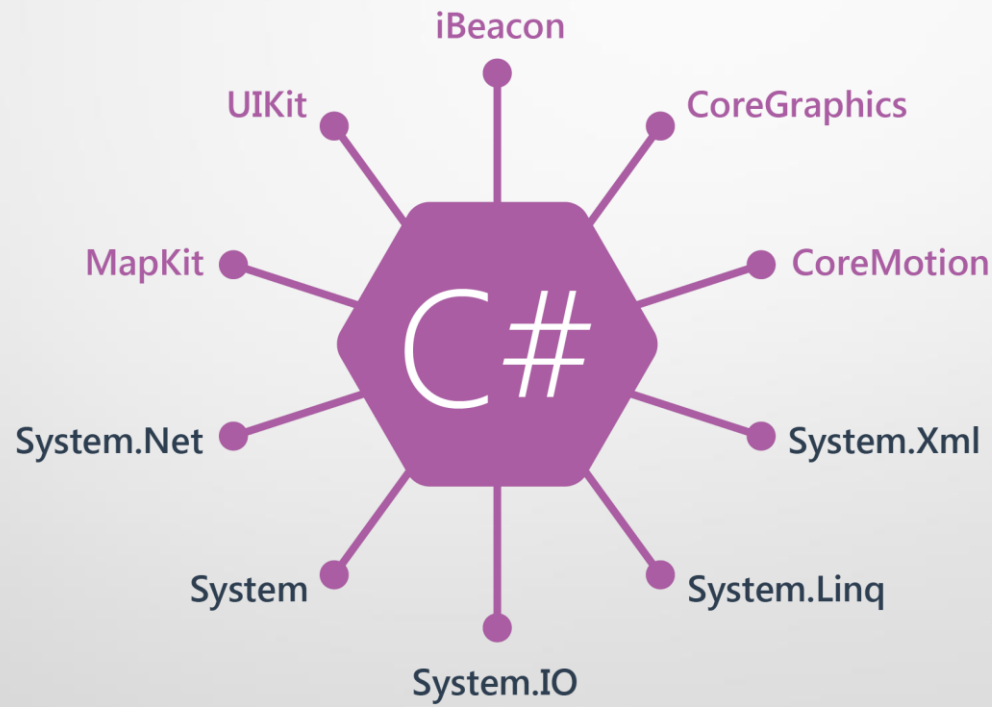




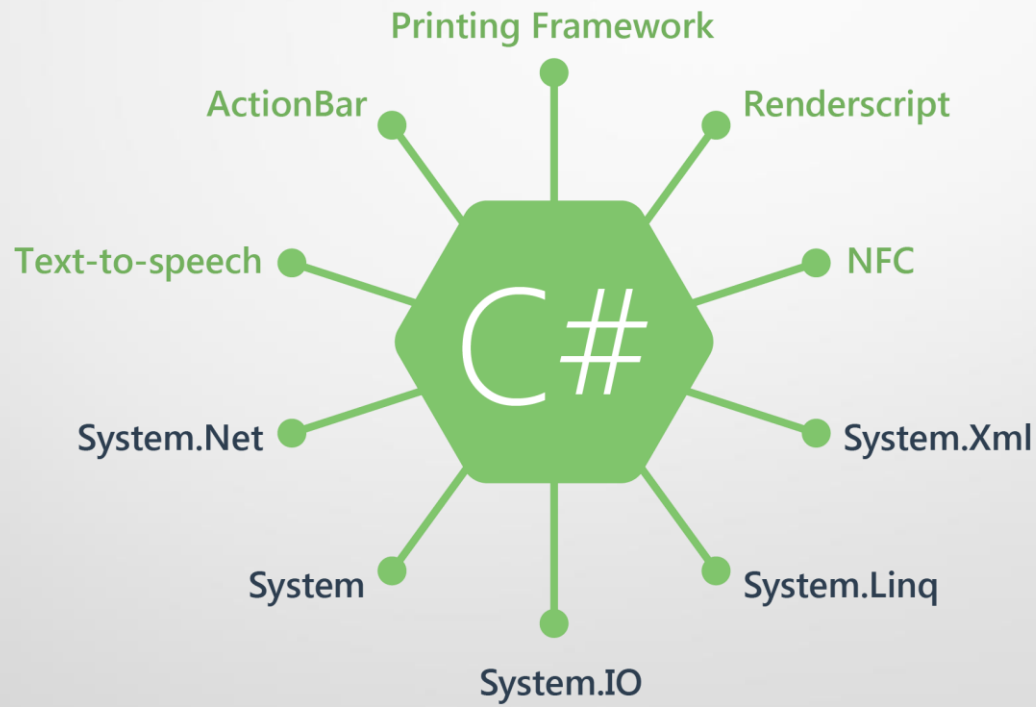
# Xamarin : comment ça marche ?



# Xamarin : comment ça marche ?

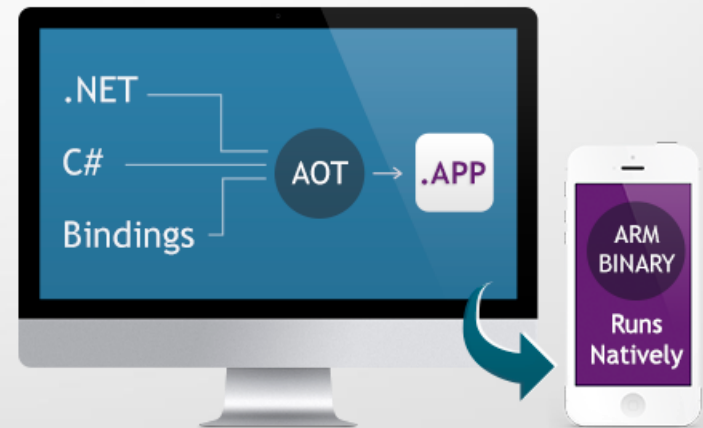


# Xamarin : comment ça marche ?



# Compilation pour iOS

- Compilation du code C# vers du natif
- Linking avec les API natives (framework ou binding)
- Création d'un binaire ARM pour le store



# Compilation pour Android

- Compilation du code C# en IL
- Inclusion de la machine virtuelle CLR dans le package de l'application
- Exécution du code IL via le JIT
- Fun fact : meilleure performance qu'une appli qui s'exécute sur la JVM



# Un seul IDE



Visual  
Studio

- Disponible pour Windows
- Mais aussi pour macOS

# Une seule source pour les libs



- [nuget.org](https://nuget.org)
- Quelques noms :
  - Xamarin.Essentials
  - Xamarin.Forms
  - Newtonsoft.Json
  - Xamarin.Android.Support.\*

# Une chose à retenir

*Tout ce que vous pouvez faire en Objective-C, Swift, Java ou Kotlin peut être fait en C# avec Xamarin et Visual Studio.*





C# & .NET

# C#

- Créé par Anders Hejlsberg (TurboPascal, Delphi, TypeScript)
- Sorti par Microsoft en 2002
- Basé sur C++ et certains concept de Java
- Actuellement en version 7.3
- 100% open-source

# .NET

- Framework créé par Microsoft pour VB, C# et F#
- Contient toutes les API de bases
  - Collections génériques
  - Appel HTTP
  - Connexion à une DB
  - Cryptographie

# À l'aide, je ne sais pas coder en C# !

```
using System;

namespace DemoCours
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World !");
        }
    }
}
```

À l'aide, je ne sais pas coder en C# !

```
struct MyStruct  
{  
    public int MyField;  
}
```

# À l'aide, je ne sais pas coder en C# !

```
// Overloading '+' operator:  
public static ComplexNumber operator+(ComplexNumber a, ComplexNumber b)  
{  
    return new ComplexNumber(a.real + b.real, a.imaginary + b.imaginary);  
}
```

À l'aide, je ne sais pas coder en C# !

```
// Declare the generic class.  
public class GenericList<T>  
{  
    void Add(T input) { }  
}
```

À l'aide, je ne sais pas coder en C# !

```
class Program
{
    static void Main(string[] args)
    {
        Run();
    }

    static void Run()
    {
        throw new NotImplementedException();
    }
}
```



À l'aide, je ne sais pas coder en C# !

```
#warning A warning  
#error An error  
  
#if DEBUG  
    // Code Debug  
#else  
    // Code Release  
#endif
```

À l'aide, je ne sais pas coder en C# !

```
string x = "Hello world !";  
string y = "Bonjour !";  
  
if(x == y)  
{  
    Console.WriteLine(x + y);  
}
```

# À l'aide, je ne sais pas coder en C# !

```
class User
{
    private string _firstName;

    public string FirstName
    {
        get => _firstName;
        set
        {
            if(_firstName != value)
            {
                _firstName = value;
            }
        }
    }

    public string LastName { get; set; }

    public string Name => FirstName + " " + LastName;
}
```

À l'aide, je ne sais pas coder en C# !

```
(x, y) => x + y;
```

```
(x, y) =>
```

```
{
```

```
    x++;
```

```
    return x + y;
```

```
};
```

À l'aide, je ne sais pas coder en C# !

```
class NumberTab
{
    private int[] _tab = new int[10];

    public int this[int index]
    {
        get => _tab[index];
        set => _tab[index] = value;
    }
}
```

```
NumberTab tab = new NumberTab();
tab[0] = 42;
```

# À l'aide, je ne sais pas coder en C# !

```
public event EventHandler<string> OnPropertyChanged;

public void NotifyPropertyChanged(string propertyName)
    => OnPropertyChanged ?.Invoke(this, propertyName);

public void PropertyChangedCallback(object sender, string propertyName)
{
    //TODO: handle property update
}

public void Main()
{
    OnPropertyChanged += PropertyChangedCallback;
}
```

# À l'aide, je ne sais pas coder en C# !

```
public void TryDoSomethingWith1(object obj)
{
    if(obj is User)
    {
        User user = (User)obj;
        Console.WriteLine(user.Name);
    }
}
```

```
public void TryDoSomethingWith2(object obj)
{
    User user = obj as User;
    if (user != null)
    {
        Console.WriteLine(user.Name);
    }
}
```

# À l'aide, je ne sais pas coder en C# !

```
public void TryDoSomethingWith3(object obj)
{
    if (obj is User user)
    {
        Console.WriteLine(user.Name);
    }
}
```

```
public void TryDoSomethingWith4(object obj)
{
    switch(obj)
    {
        case User user when user.FirstName == "Brian":
            Console.WriteLine("Where is Brian ?");
            break;
        case User user:
            Console.WriteLine(user.Name);
            break;
    }
}
```



# À l'aide, je ne sais pas coder en C# !

```
public int Parse(string input)
{
    if(int.TryParse(input, out int result))
    {
        return result;
    }

    return -1;
}

public void Increment(ref int x)
{
    x++;
}

public void PromisJeTouchePas(in User user)
{
    user = new User();
}
```

À l'aide, je ne sais pas coder en C# !

```
public (int diviseur, int reste) DivisionEuclidienne(int x, int y)
{
    return (
        diviseur: x / y,
        reste: x % y
    );
}
```

À l'aide, je ne sais pas coder en C# !

```
public User LePlusJeuneBrian(List<User> users)
{
    return users.Where(x => x.FirstName == "Brian")
        .OrderBy(x => x.Age)
        .FirstOrDefault();
}
```

# Async / Await

```
public async Task<string> GetHtml(string url)
{
    HttpClient client = new HttpClient();

    var response = await client.GetAsync(url);

    if(response.IsSuccessStatusCode)
    {
        return await response.Content.ReadAsStringAsync();
    }

    return null;
}
```

```
public Task<HttpResponseMessage> GetAsync(string requestUri);
```

# Async / Await

```
public async void ButtonClicked()
{
    string result = await GetHtml("perdu.com");
    // OU
    Task<string> resultTask = GetHtml("perdu.com");
    string realResult = await resultTask;
}
```



Aa Bb Cc Dd

$2 + 2 = \dots$

Questions ?

# Xamarin



Native iOS App

Platform-specific C#



Native Android app

Platform-specific C#

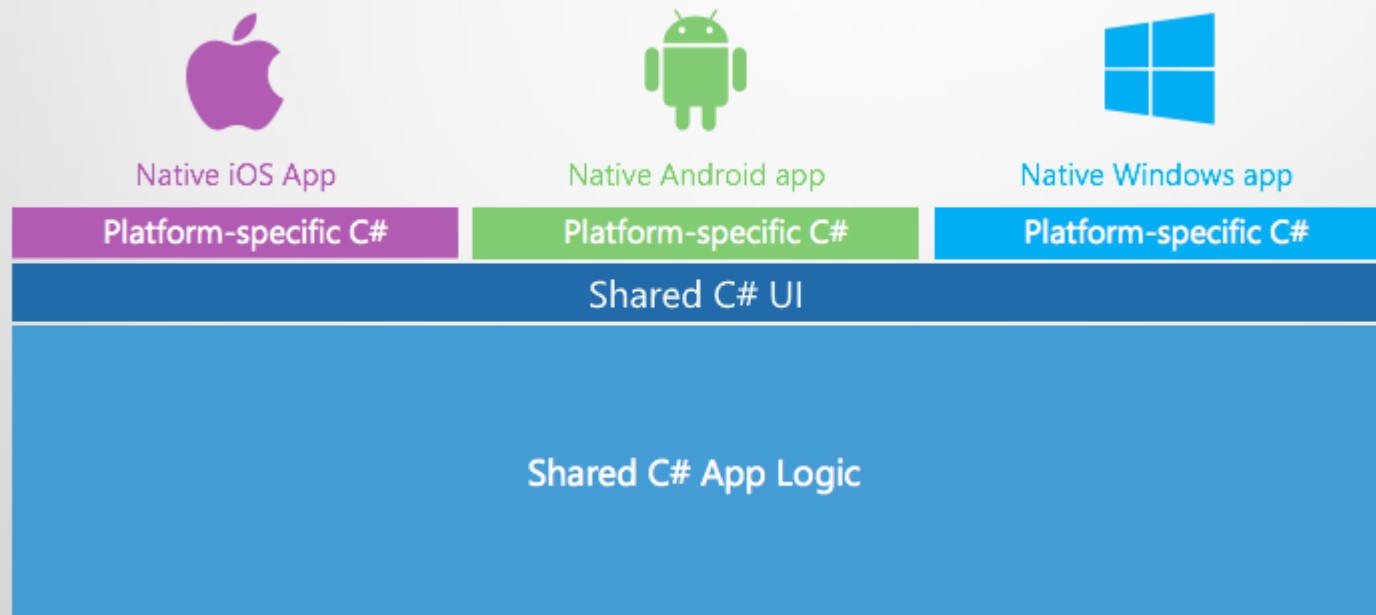


Native Windows app

Platform-specific C#

Shared C# App Logic

# Xamarin.Forms





# Vue Xamarin.Forms

```
<?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:App1"
              x:Class="App1.Views.HomePage">

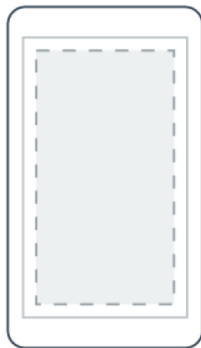
    <StackLayout Orientation="Vertical">

        <Label Text="Welcome to Xamarin.Forms!"
              HorizontalOptions="Center"
              VerticalOptions="CenterAndExpand"
              />

    </StackLayout>
</ContentPage>
```

Welcome to Xamarin.Forms!

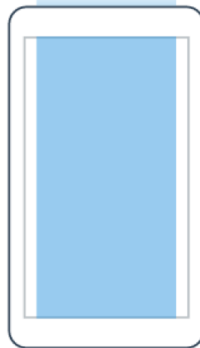
# Les layouts Xamarin.Forms



ContentPresenter



ContentView



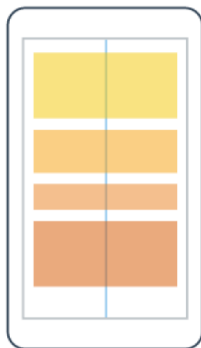
ScrollView



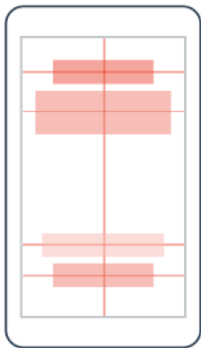
Frame



TemplatedView



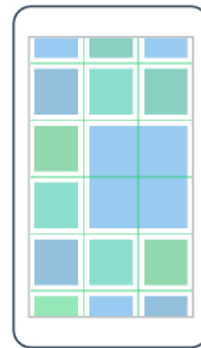
StackLayout



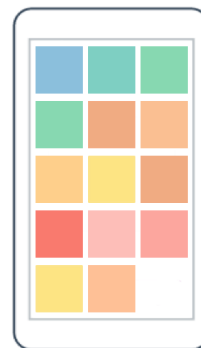
AbsoluteLayout



RelativeLayout



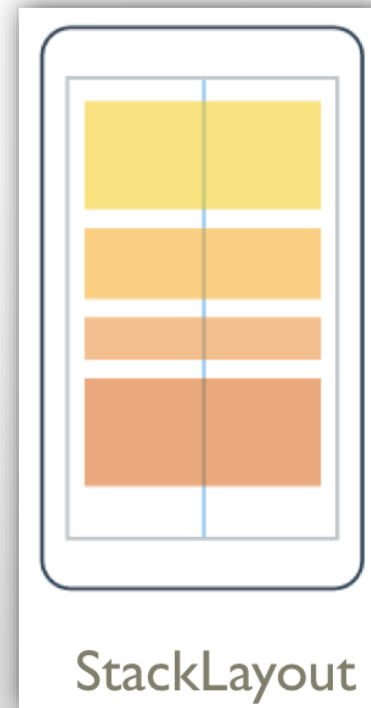
Grid



FlexLayout

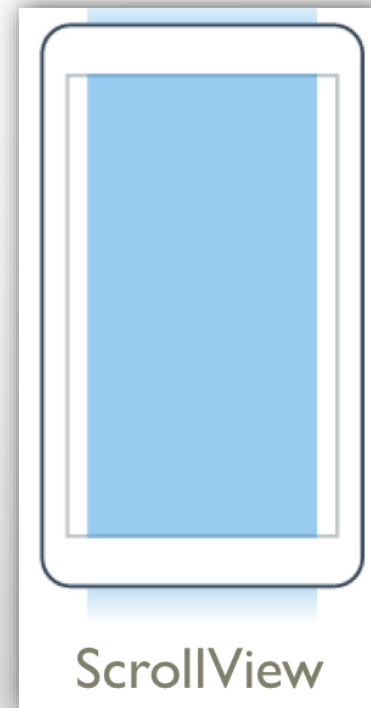
# Les layouts Xamarin.Forms

```
<StackLayout Orientation="Vertical">  
  <Label Text="Label1" />  
  <Label Text="Label2" />  
</StackLayout>  
  
<StackLayout Orientation="Horizontal">  
  <Label Text="Label1" />  
  <Label Text="Label2" />  
</StackLayout>
```



# Les layouts Xamarin.Forms

```
<ScrollView Orientation="Both">  
...  
</ScrollView>  
  
<ScrollView Orientation="Vertical">  
...  
</ScrollView>  
  
<ScrollView Orientation="Horizontal">  
...  
</ScrollView>
```

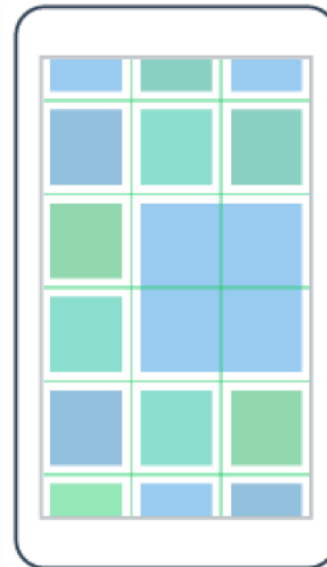


# Les layouts Xamarin.Forms

```
<Grid>
  <Grid.RowDefinitions>
    <RowDefinition Height="100" />
    <RowDefinition Height="*" />
  </Grid.RowDefinitions>

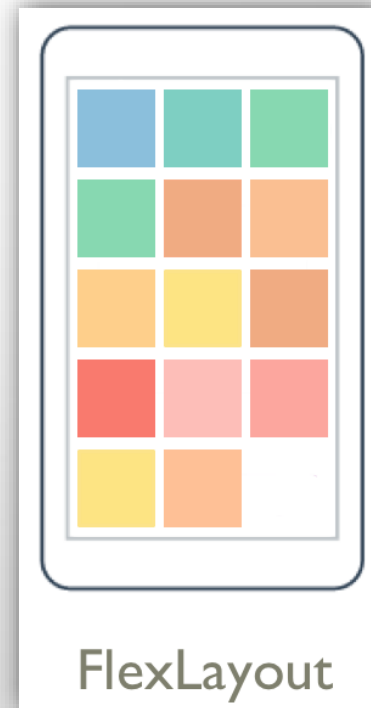
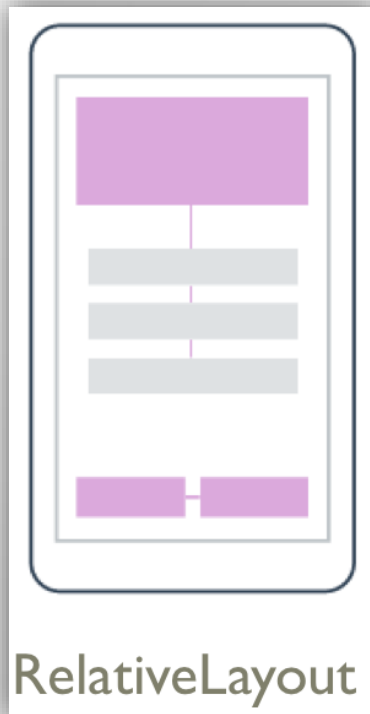
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>

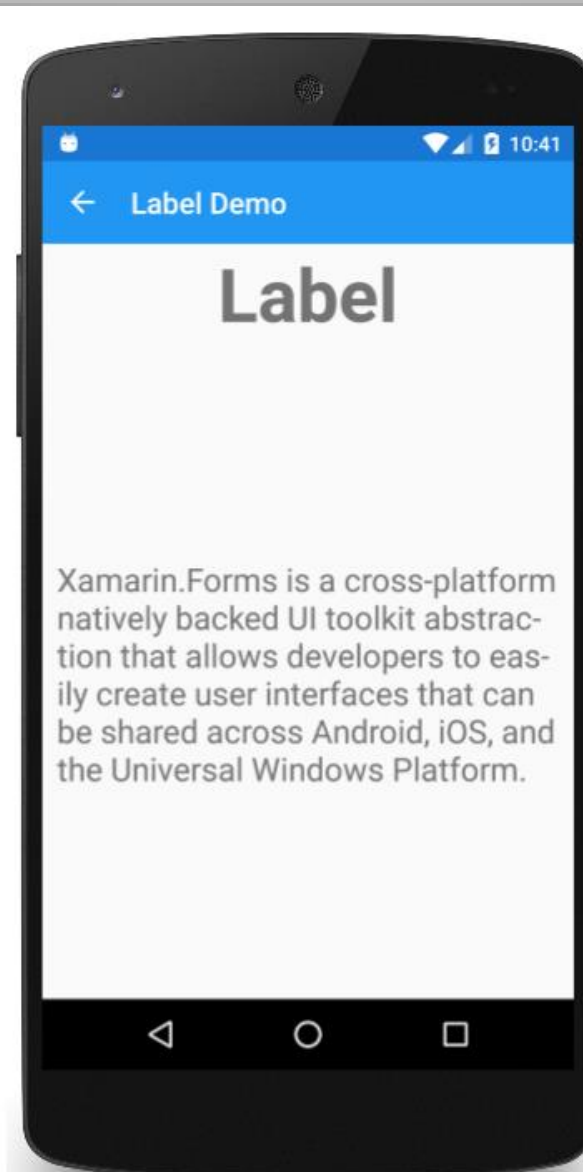
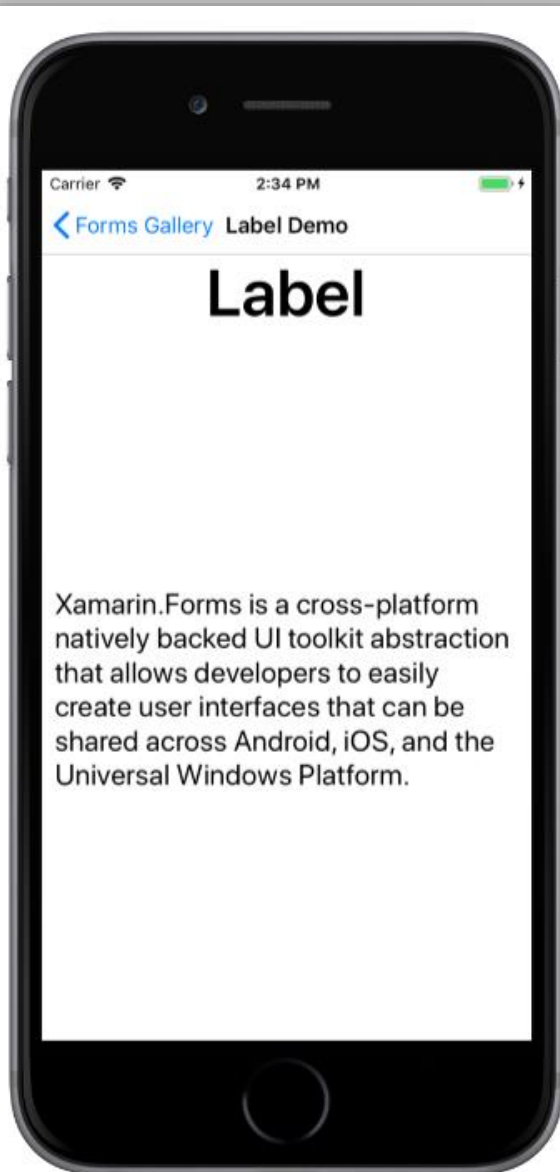
  <Label Text="Top Left" Grid.Row="0" Grid.Column="0" />
  <Label Text="Top Right" Grid.Row="0" Grid.Column="1" />
  <Label Text="Bottom Left" Grid.Row="1" Grid.Column="0" />
  <Label Text="Bottom Right" Grid.Row="1" Grid.Column="1" />
</Grid>
```

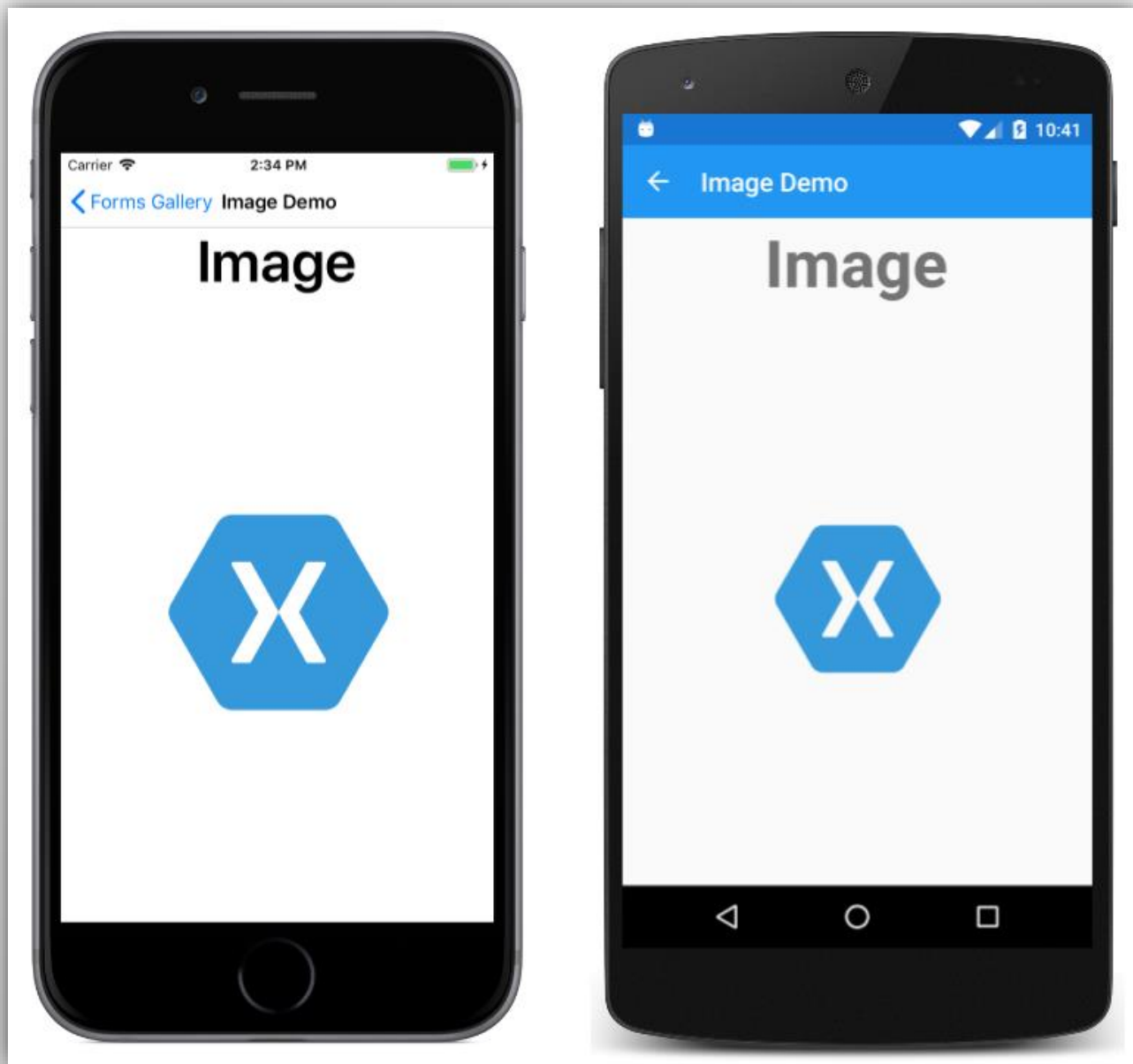


Grid

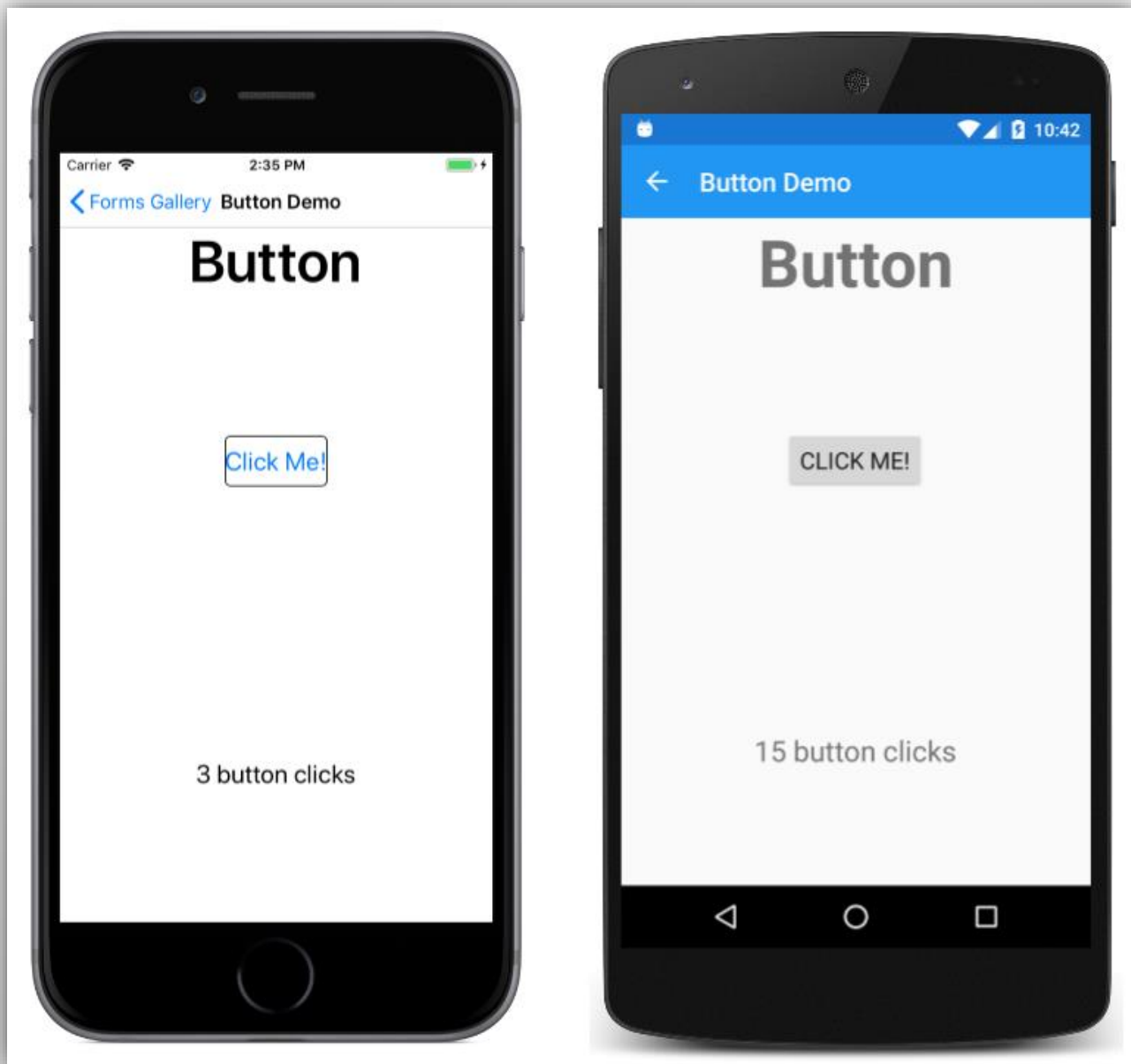
# Les layouts Xamarin.Forms



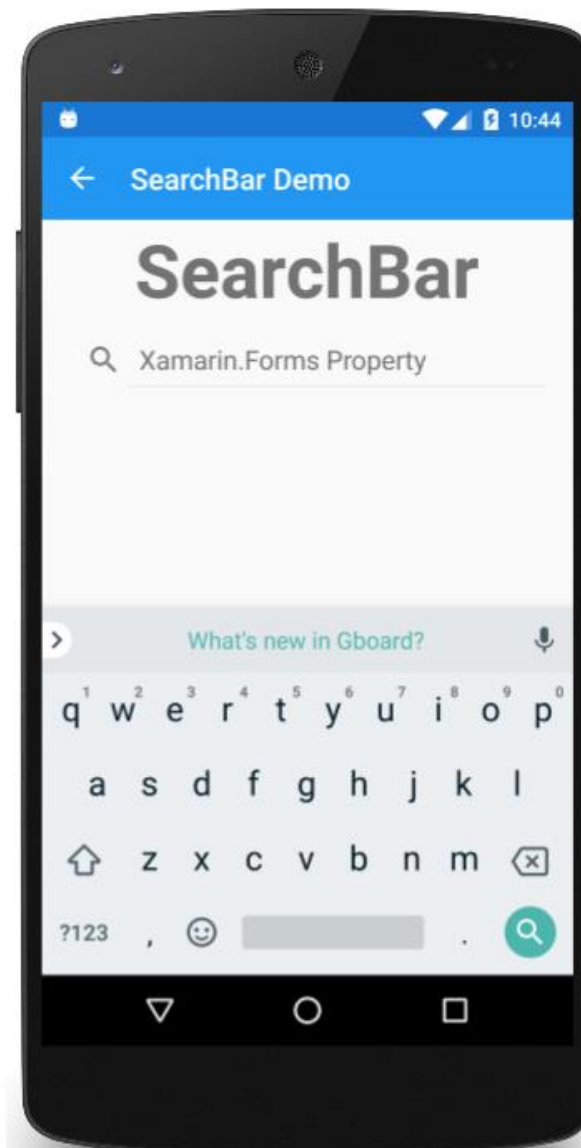
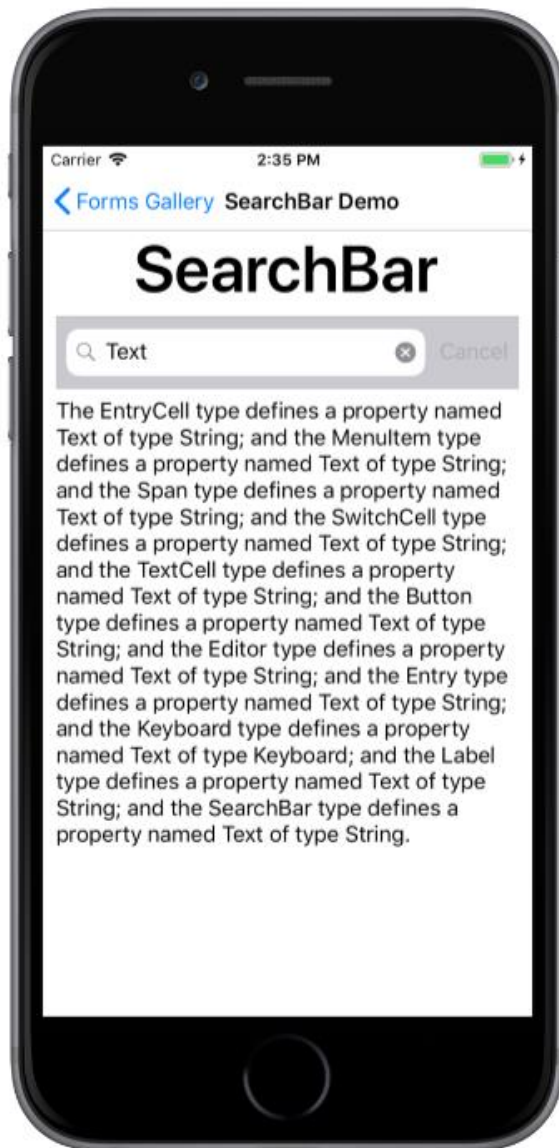


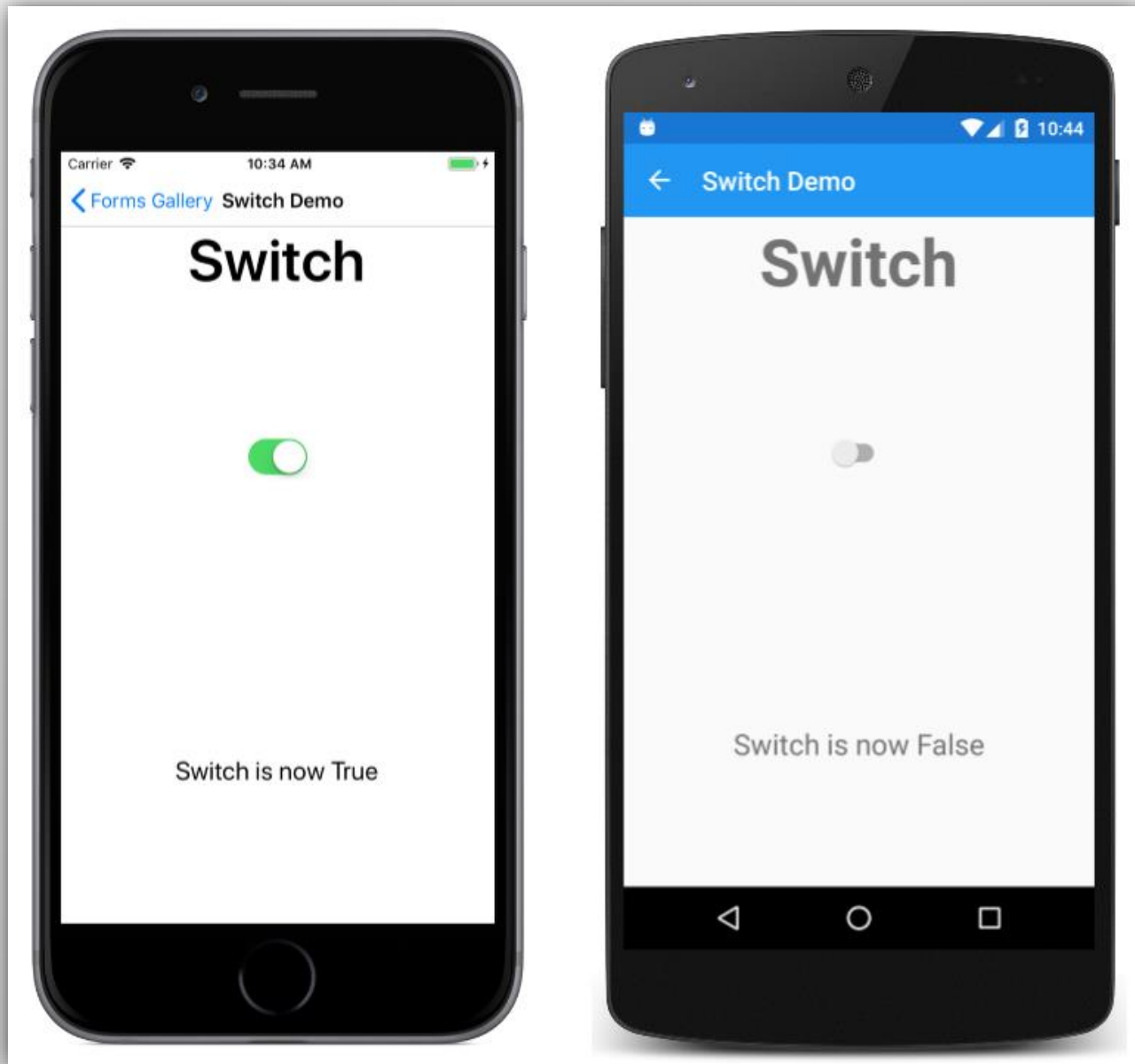


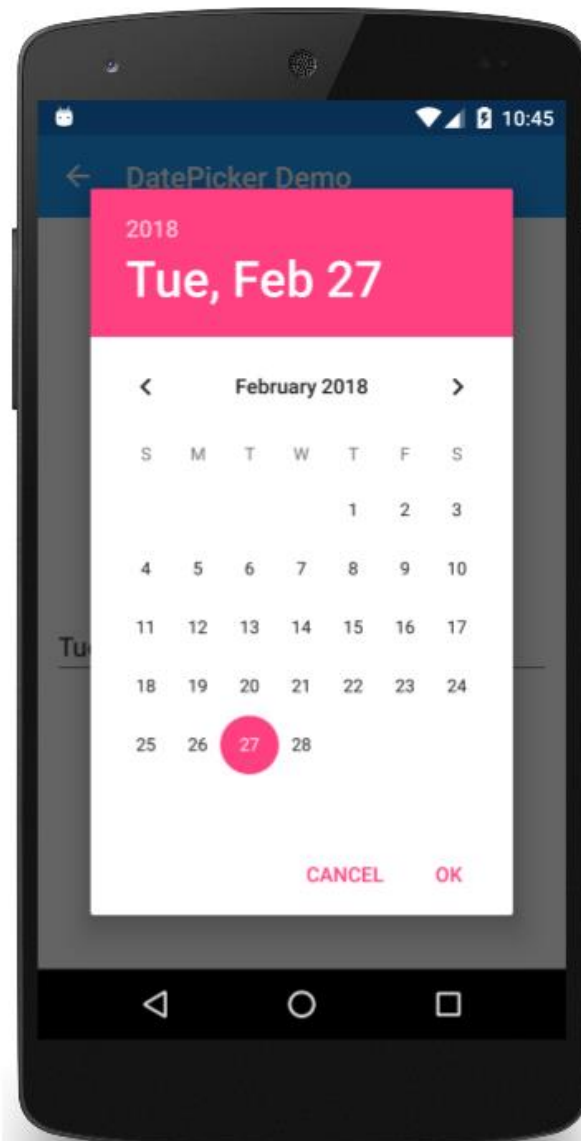


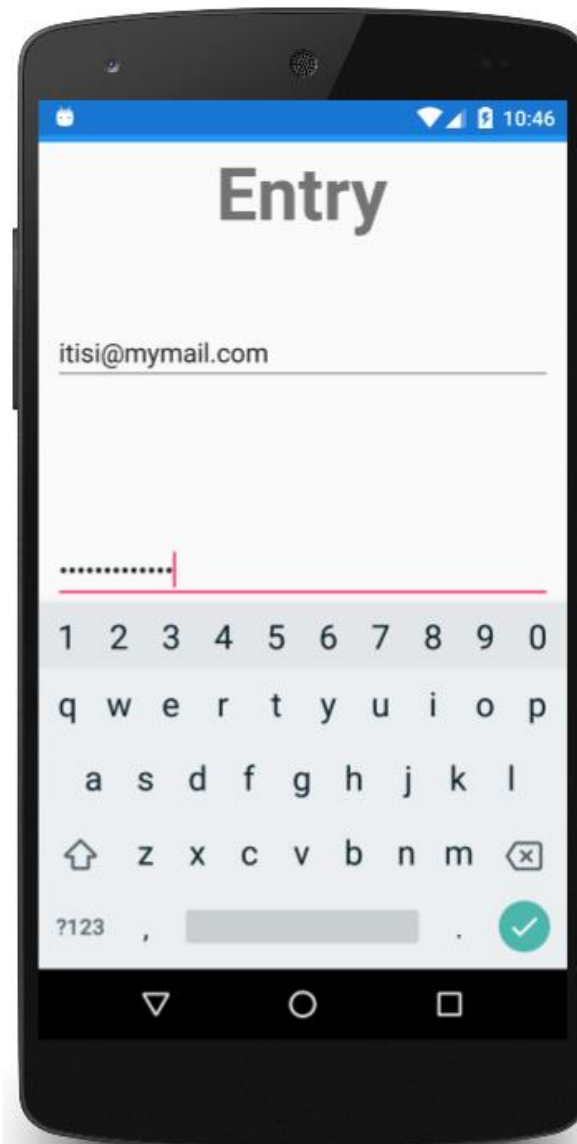


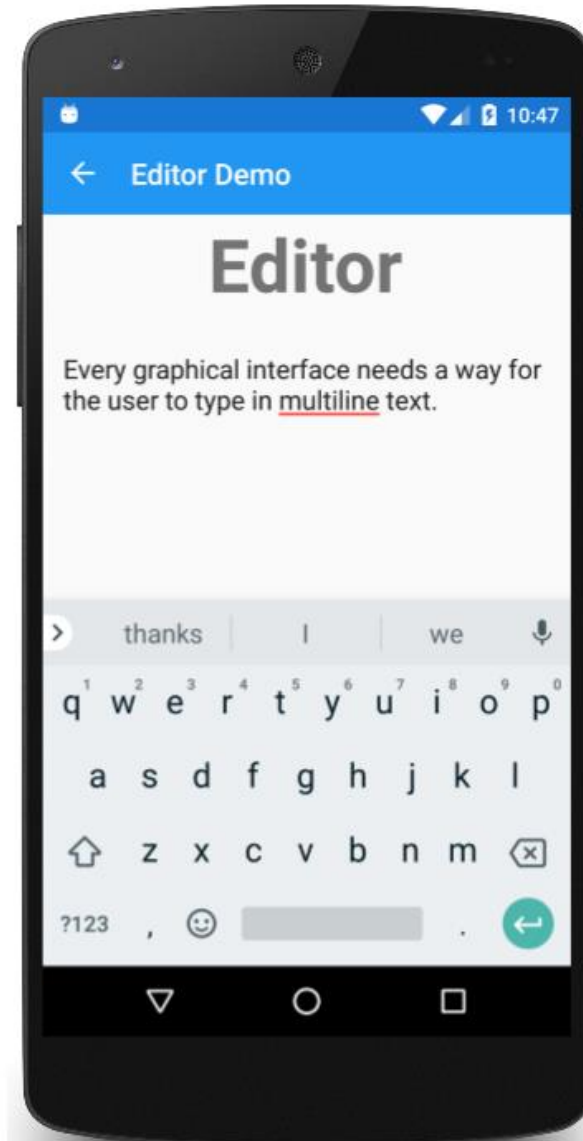


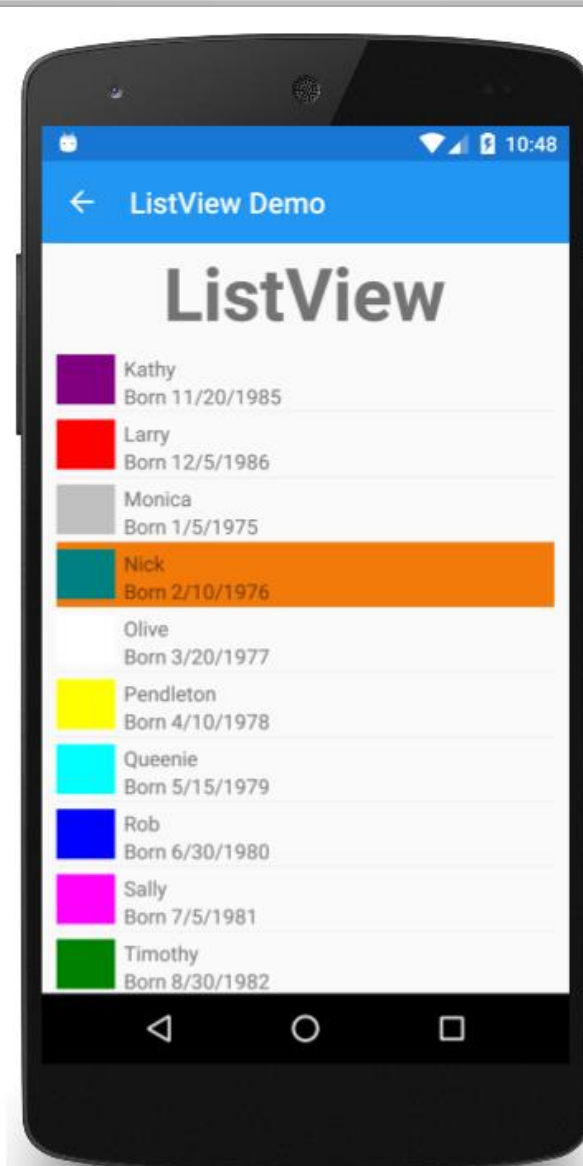
















DÉMO

— live



# Ressources

- Pour l'application complète
- Pour une page
- Pour un layout
- Mutualise à un seul endroit vos couleurs, text, taille de texte...

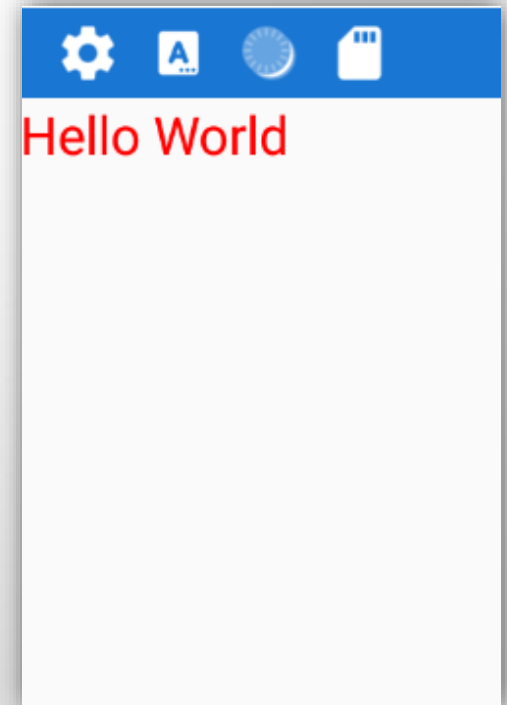
```
<Application.Resources>  
  <x:Double x:Key="LabelSize">14</x:Double>  
</Application.Resources>
```

```
<ContentPage.Resources>  
  <x:String x:Key="MyText">Hello World</x:String>  
</ContentPage.Resources>
```

```
<StackLayout.Resources>  
  <Color x:Key="Red">#FF0000</Color>  
</StackLayout.Resources>
```

# Ressources

```
<Label Text="{StaticResource MyText}"  
      TextColor="{StaticResource Red}"  
      FontSize="{StaticResource LabelSize}"  
/>
```



# MVVM



# MVVM

## Principe

- Model: vos données brut
- ViewModel:
  - agrège et transforme les données
  - Réagit aux événements
- View: se charge d'afficher l'écran

## Avantages

- Découplage View / ViewModel
- Développement séparé
  - Plus simple pour une équipe
- Dans un monde idéal, le designer pourrait faire la View



DÉMO

— live



# Références

- Liste des vues : <https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/user-interface/controls/views>
- Liste des layout : <https://docs.microsoft.com/fr-fr/xamarin/xamarin-forms/user-interface/controls/layouts>



Aa Bb Cc Dd

$2 + 2 = \dots$

Questions ?



# TD

- <https://github.com/Julien-Mialon/Cours-Xamarin-2019>