



Développement d'application cross- platform avec Xamarin

Julien Mialon : mialon.julien@gmail.com

Valentin Jubert : valentin.jubert@outlook.fr

Qui sommes-nous ?

Julien Mialon

- Diplômé en 2014
- Dev Mobile depuis 2011 et Xamarin depuis 2014
- Banque Populaire, Voyages-SNCF, Idelink, OPAP, Good Angel
- Lead dev chez Good Angel

Valentin Jubert

- Diplômé en 2017
- Dev Xamarin depuis 2015
- 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
- DevOps (CI / CD / CT) + AppCenter


Séance 2

- Xamarin.Forms avancé
- OAuth2 password flow
- Push notifications
- Cache
- Architectures
- Publication sur les stores

Au sommaire...

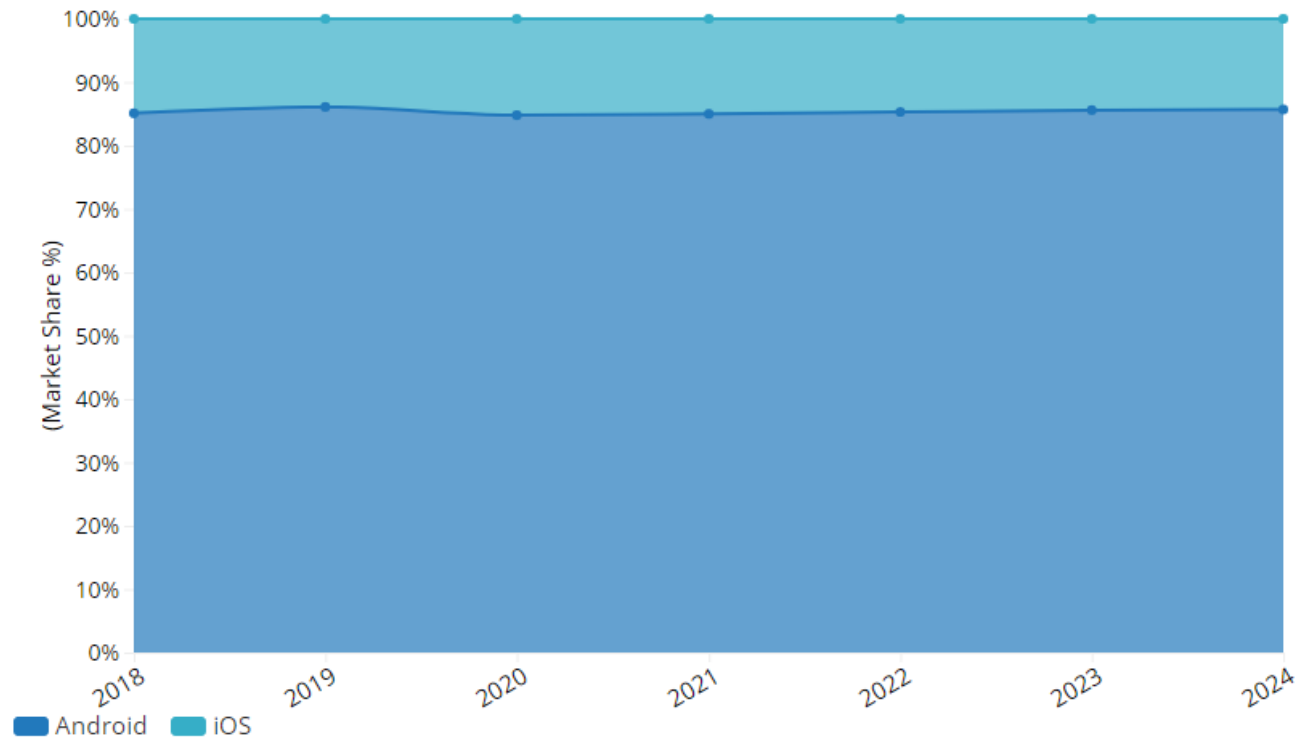
QCM le 13/03 (environ 30 minutes)

TD/mini-projet à rendre pour le 28/03



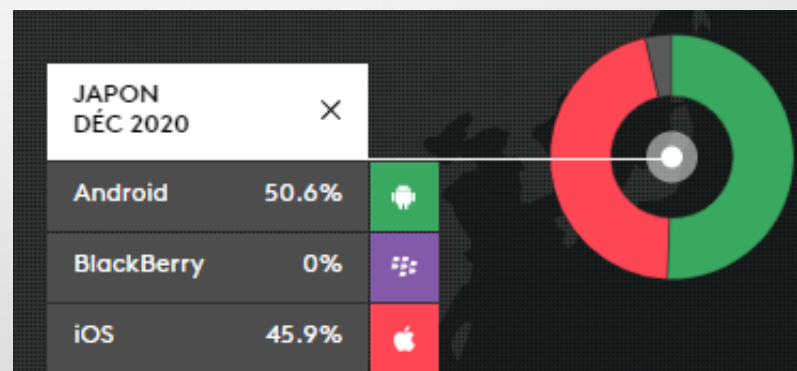
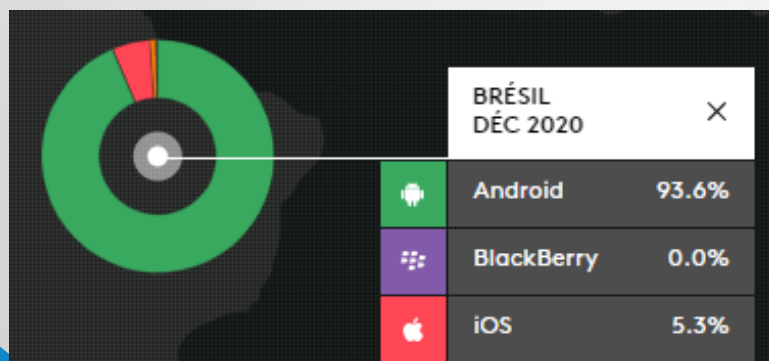
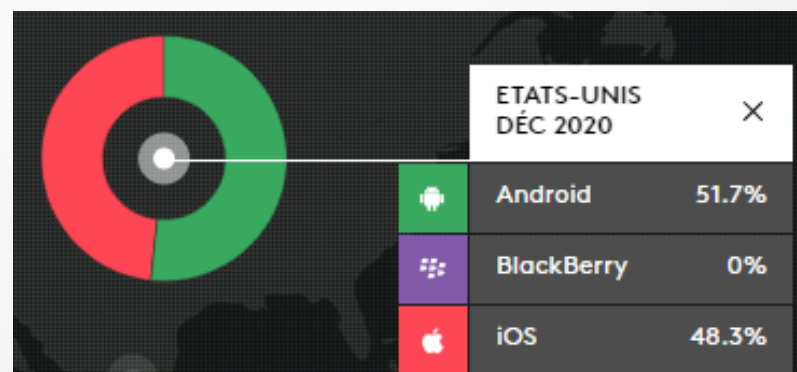
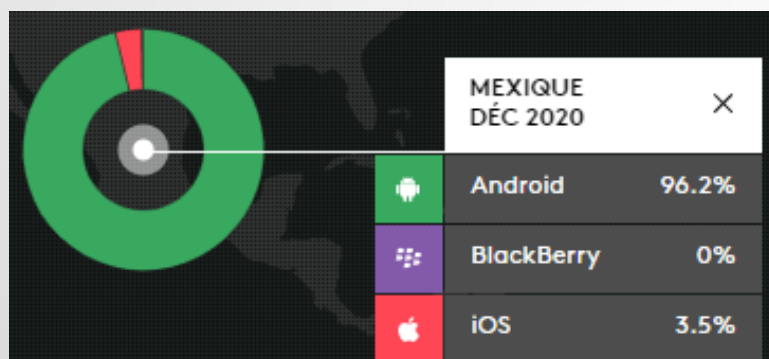
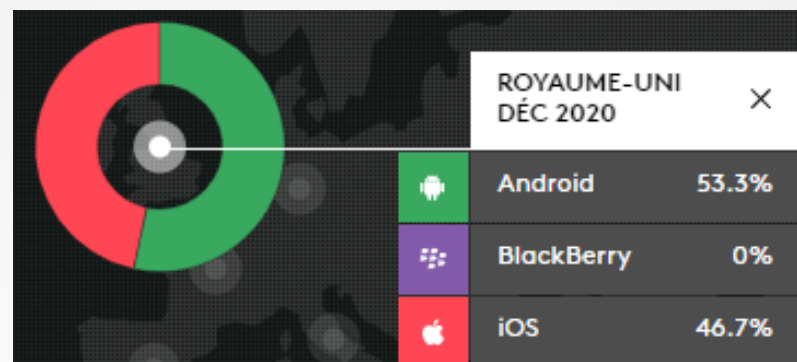
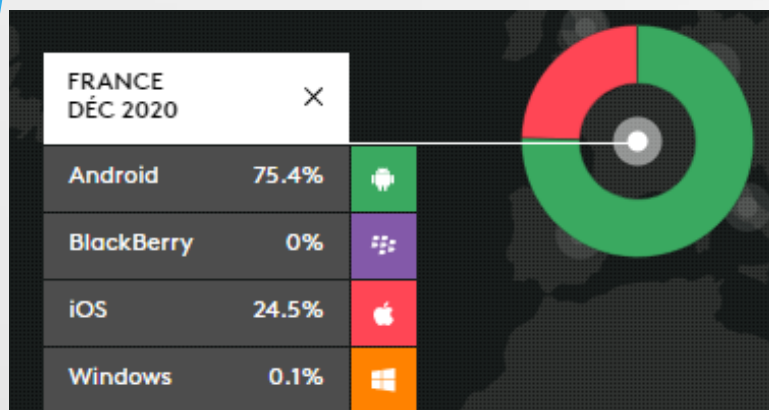
Écosystème mobile

Worldwide Smartphone Shipment OS Market Share Forecast



Year	2018	2019	2020	2021	2022	2023	2024
Android	85.1%	86.1%	84.8%	85.0%	85.3%	85.6%	85.7%
iOS	14.9%	13.9%	15.2%	15.0%	14.7%	14.4%	14.3%
Others	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%

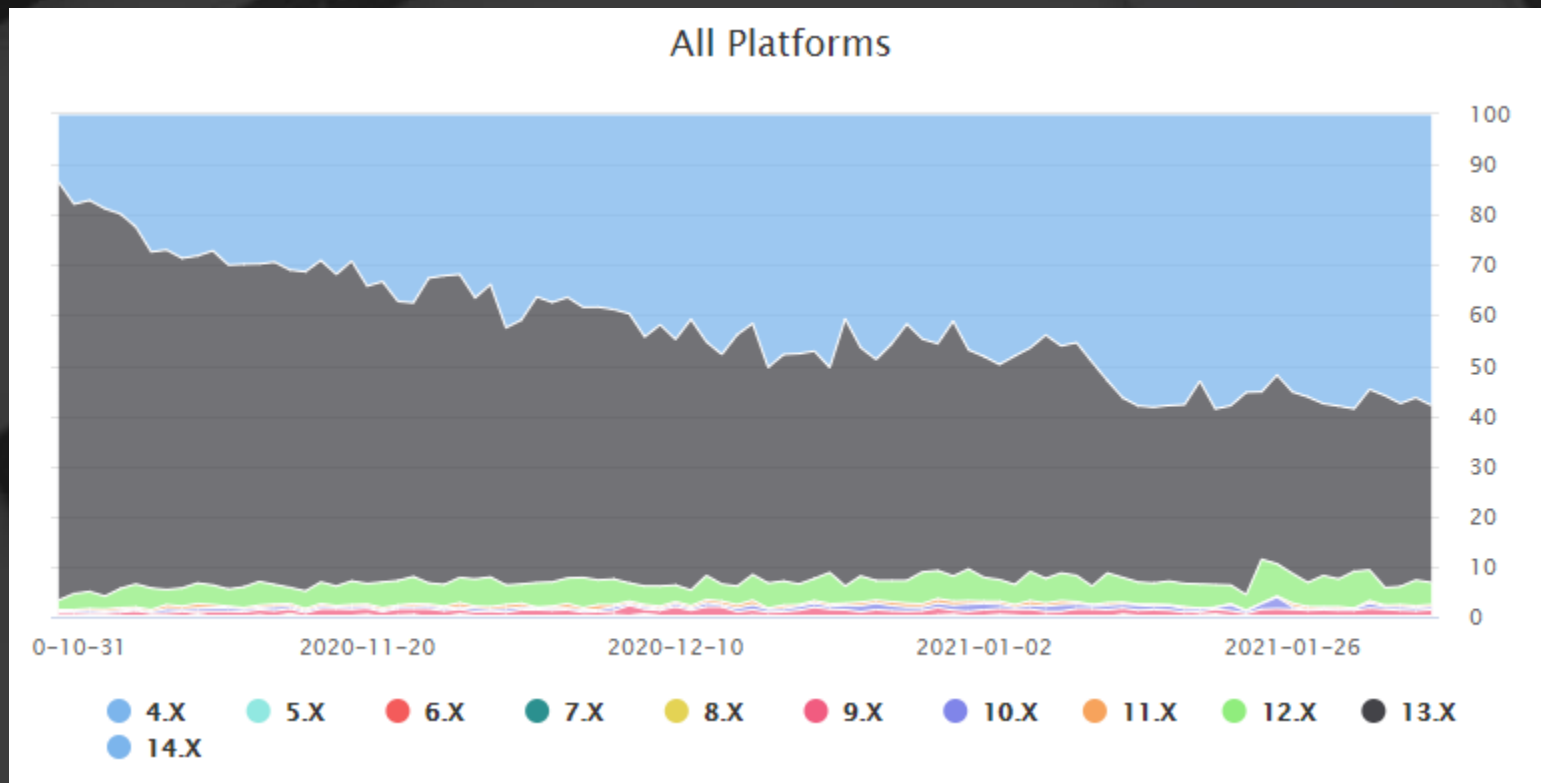
Source : IDC.com



Source : kantarworldpanel.com

iOS

- 29 iPhone + 7 iPod Touch
- 17 iPad + 8 iPad Pro
- 5 Apple TV
- 7 Apple Watch





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 ressemblants
 - 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
- Hot Reload Debugging

Inconvénients

- 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

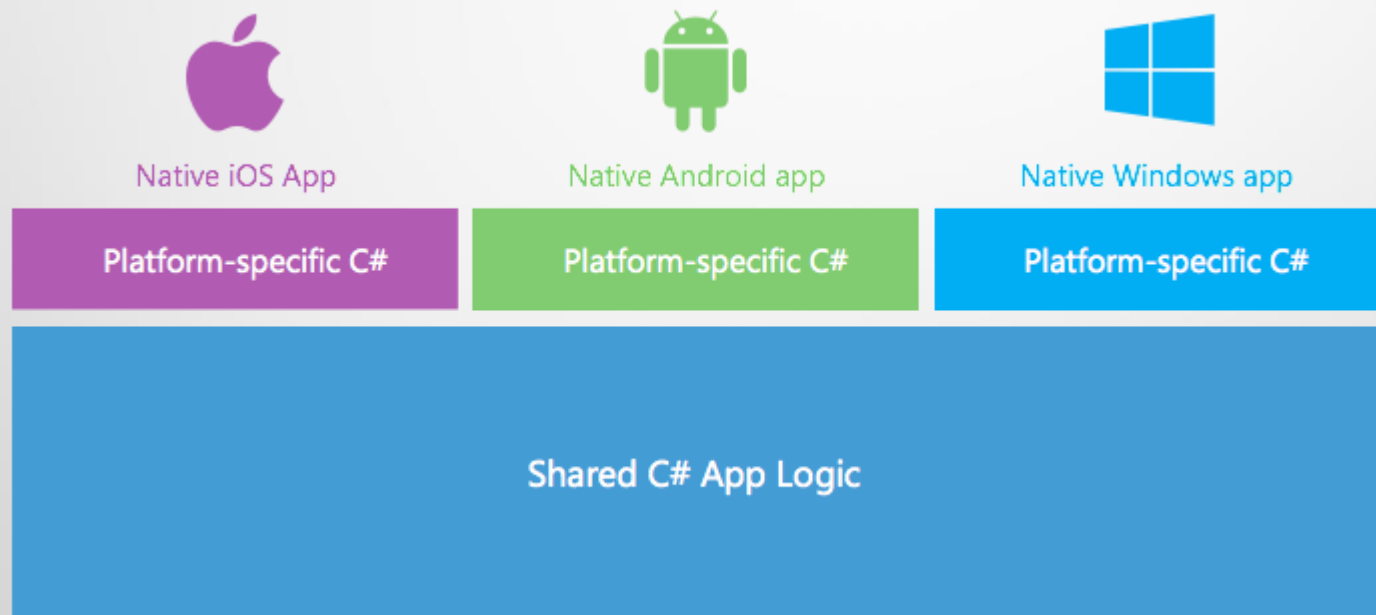
Un peu d'histoire

- 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

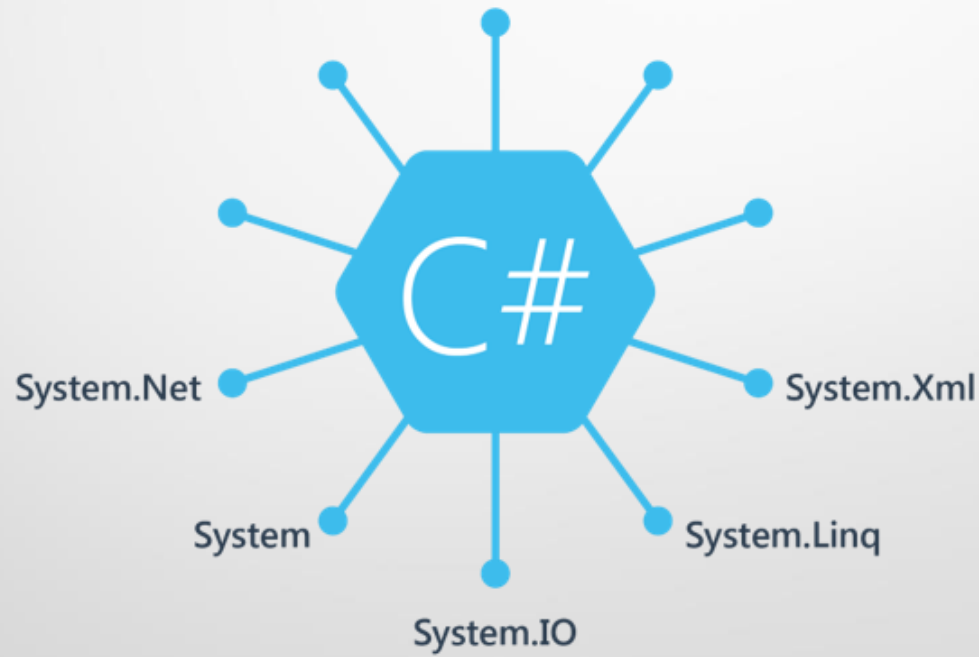
Un peu d'histoire

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

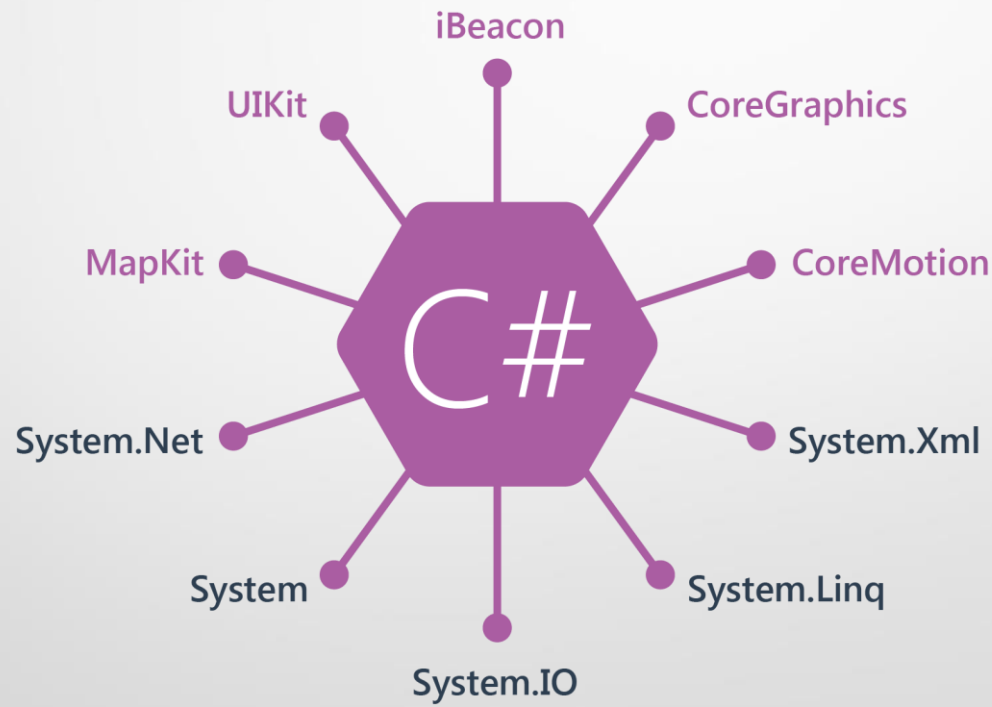
Xamarin : comment ça marche ?



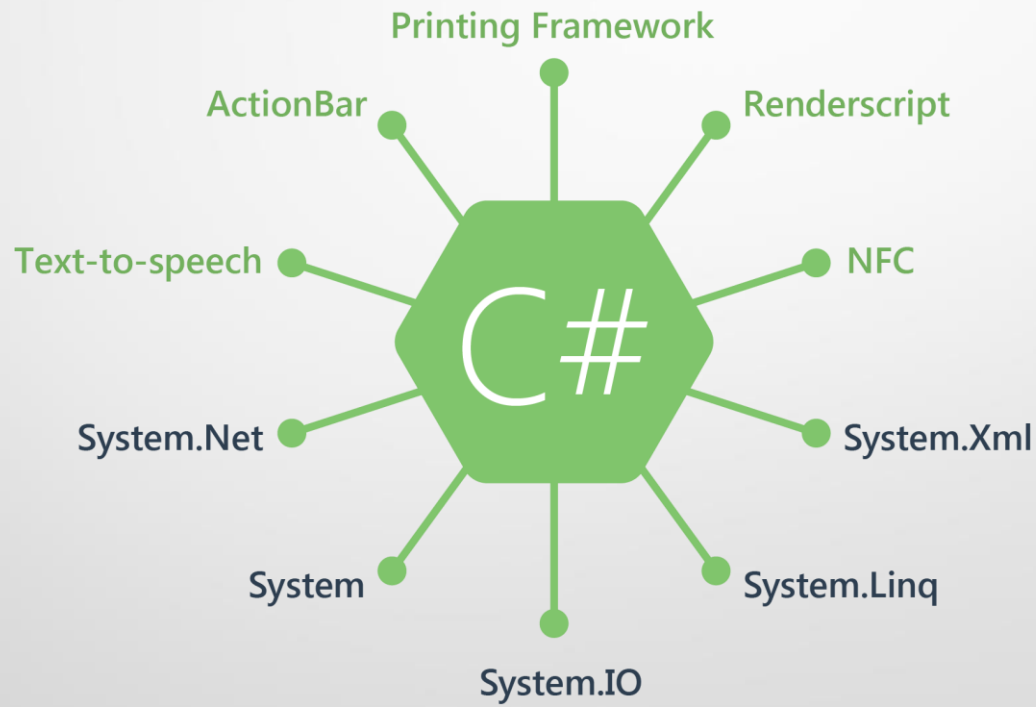
Xamarin : comment ça marche ?



Xamarin : comment ça marche ?

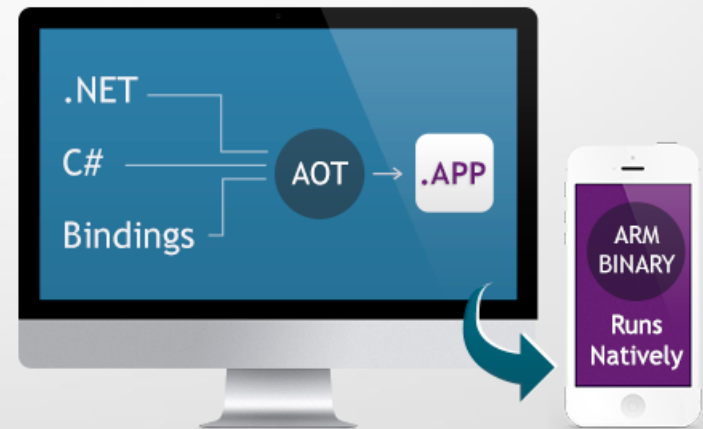


Xamarin : comment ça marche ?



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
- Quelques noms :
 - Xamarin.Essentials
 - Xamarin.Forms
 - Newtonsoft.Json
 - Xamarin.AndroidX.*

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 8.0 (pour Xamarin)
- 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 ?



DÉMO

— live





Aa Bb Cc Dd

$2 + 2 = \dots$

Questions ?

Xamarin



Native iOS App

Platform-specific C#



Native Android app

Platform-specific C#

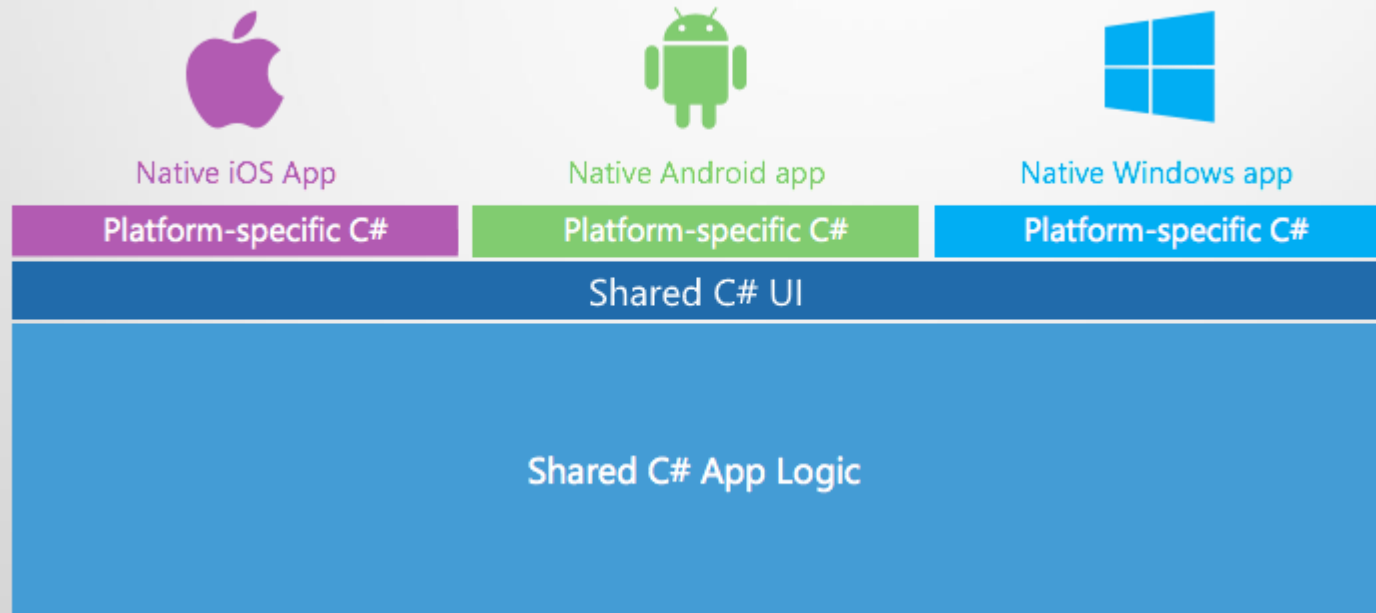


Native Windows app

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!

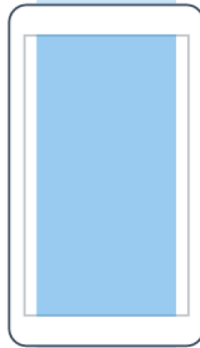
Les layouts Xamarin.Forms



ContentPresenter



ContentView



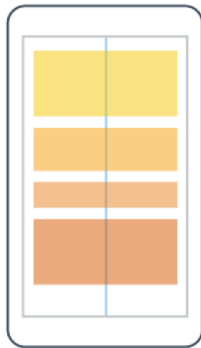
ScrollView



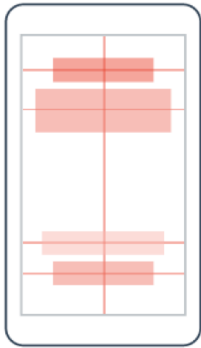
Frame



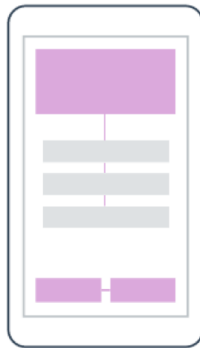
TemplatedView



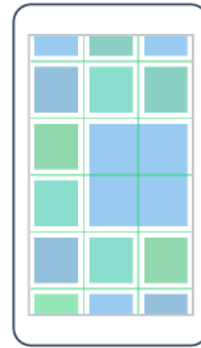
StackLayout



AbsoluteLayout



RelativeLayout



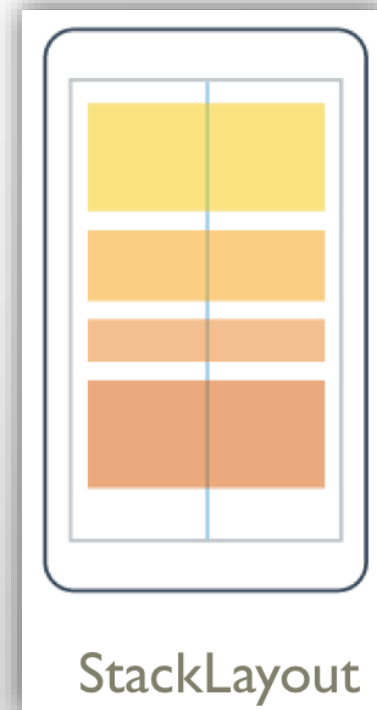
Grid



FlexLayout

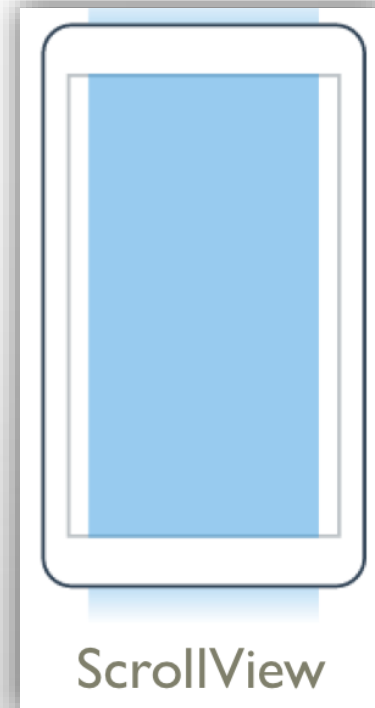
Les layouts Xamarin.Forms

```
<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>
```

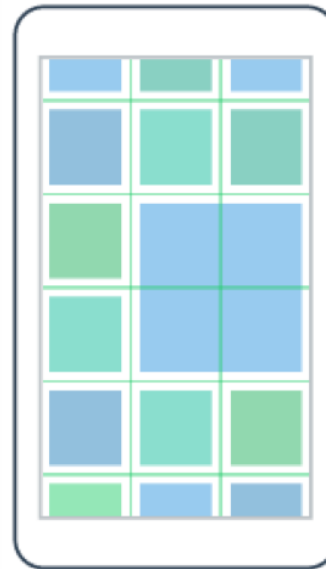


Les layouts Xamarin.Forms

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

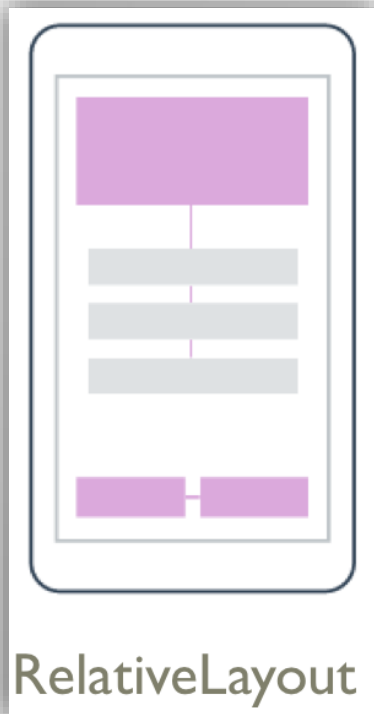
  <Grid.ColumnDefinitions>
    <ColumnDefinition Width="Auto" />
    <ColumnDefinition Width="*" />
  </Grid.ColumnDefinitions>

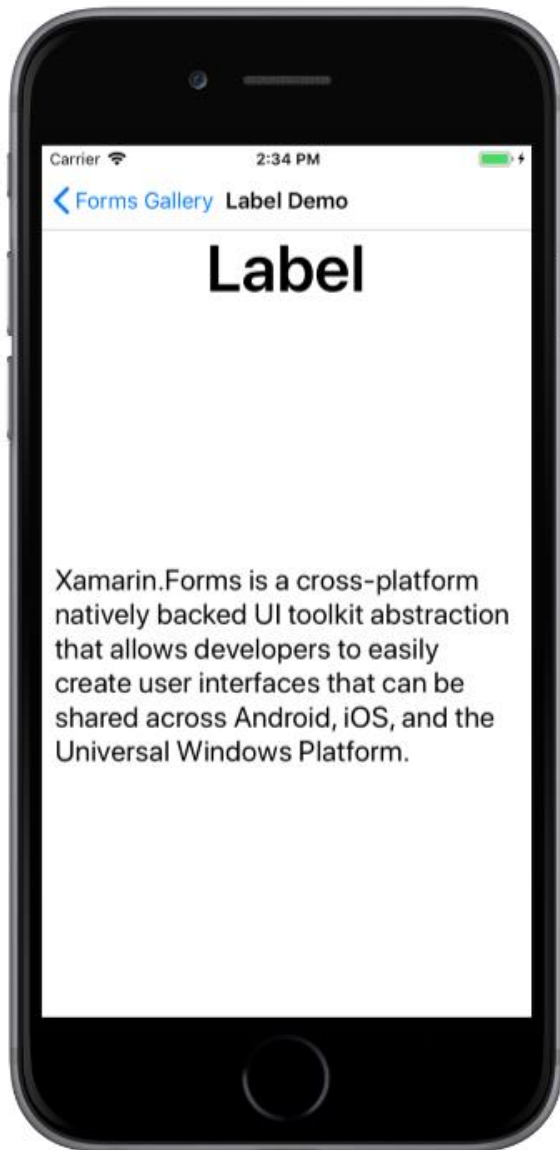
  <Label Text="Top Left" Grid.Row="0" Grid.Column="0" />
  <Label Text="Top Right" Grid.Row="0" Grid.Column="1" />
  <Label Text="Bottom Left" Grid.Row="1" Grid.Column="0" />
  <Label Text="Bottom Right" Grid.Row="1" Grid.Column="1" />
</Grid>
```

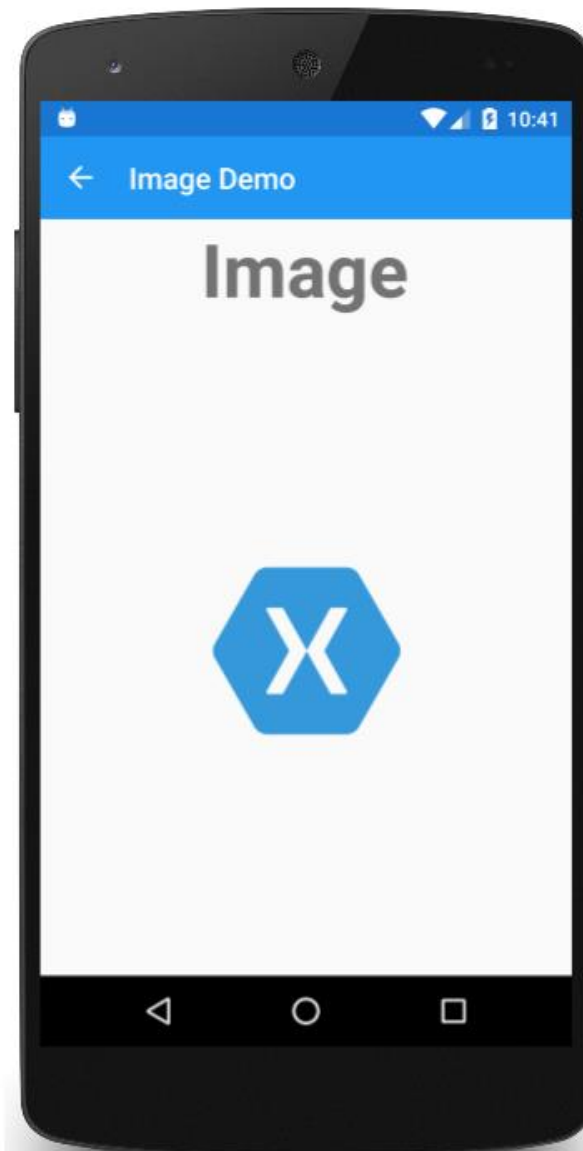


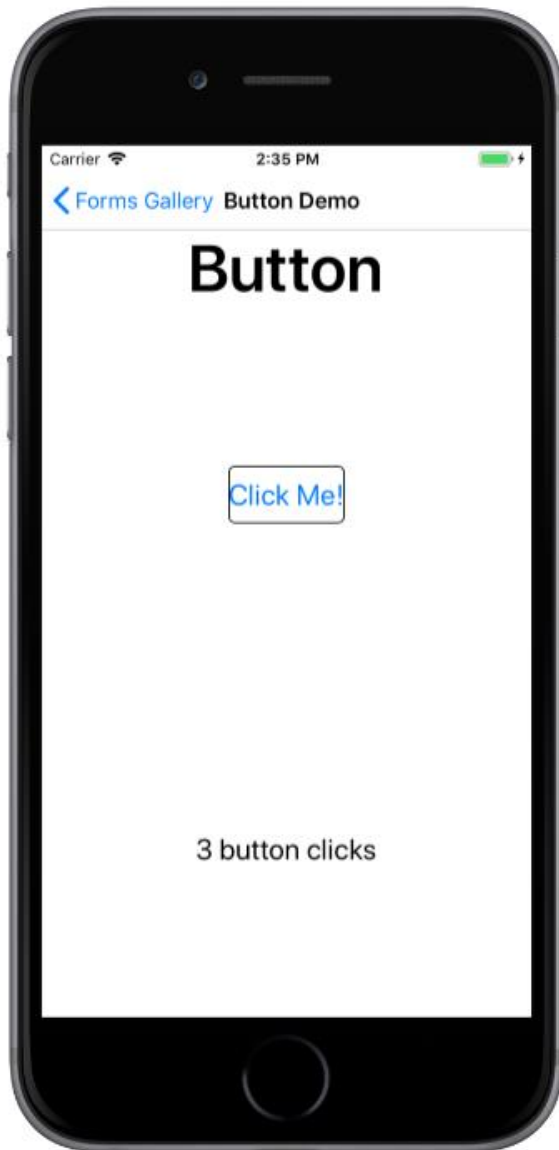
Grid

Les layouts Xamarin.Forms

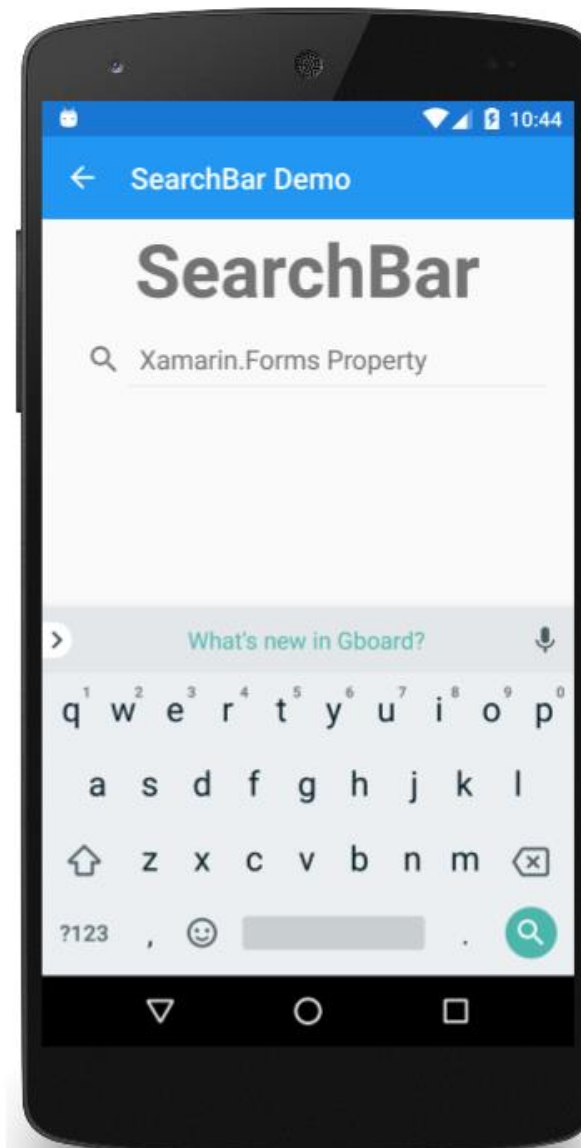
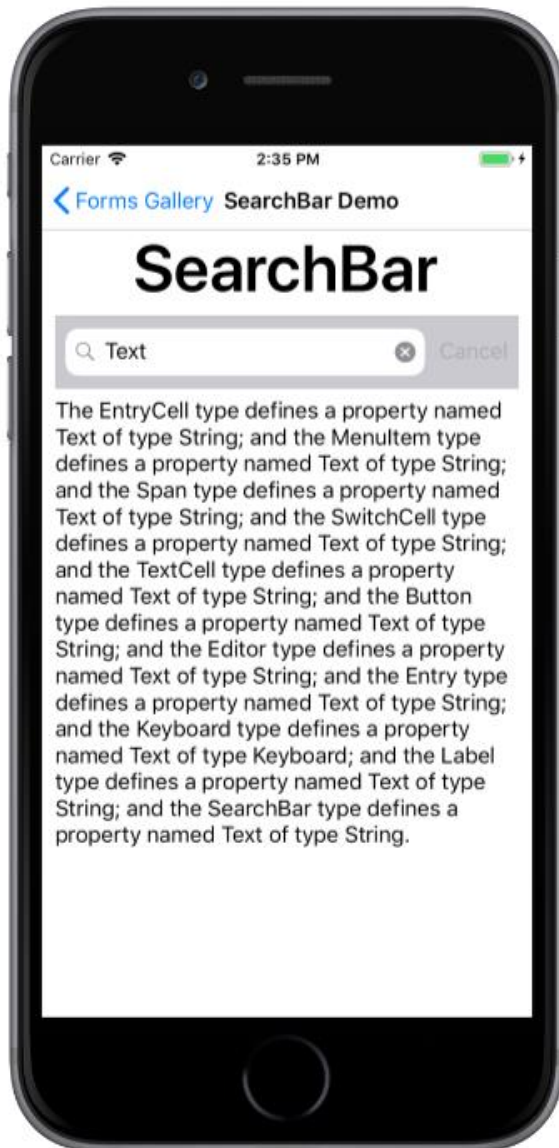




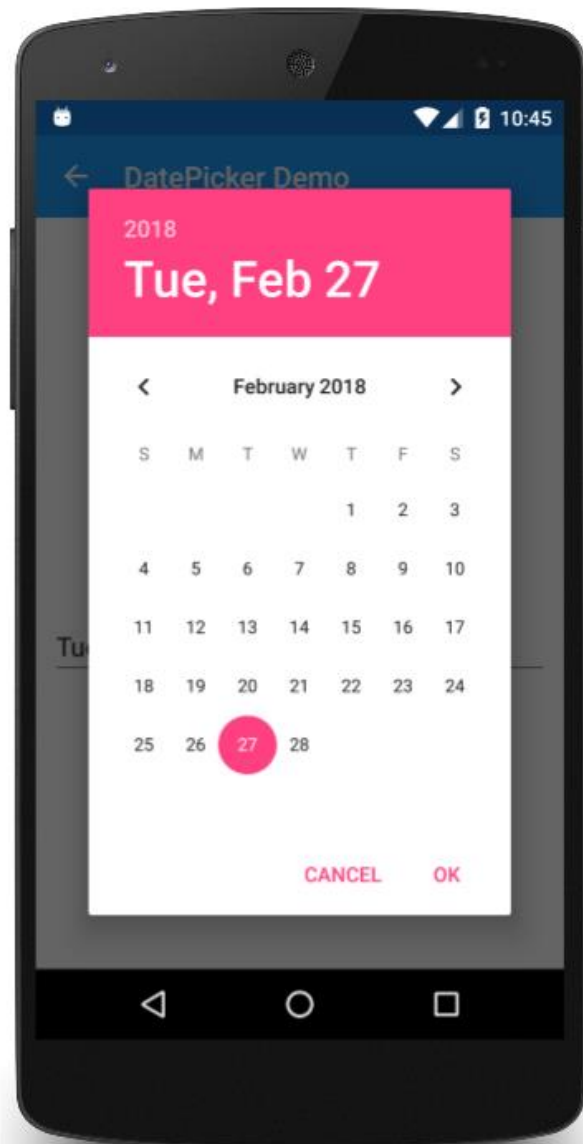


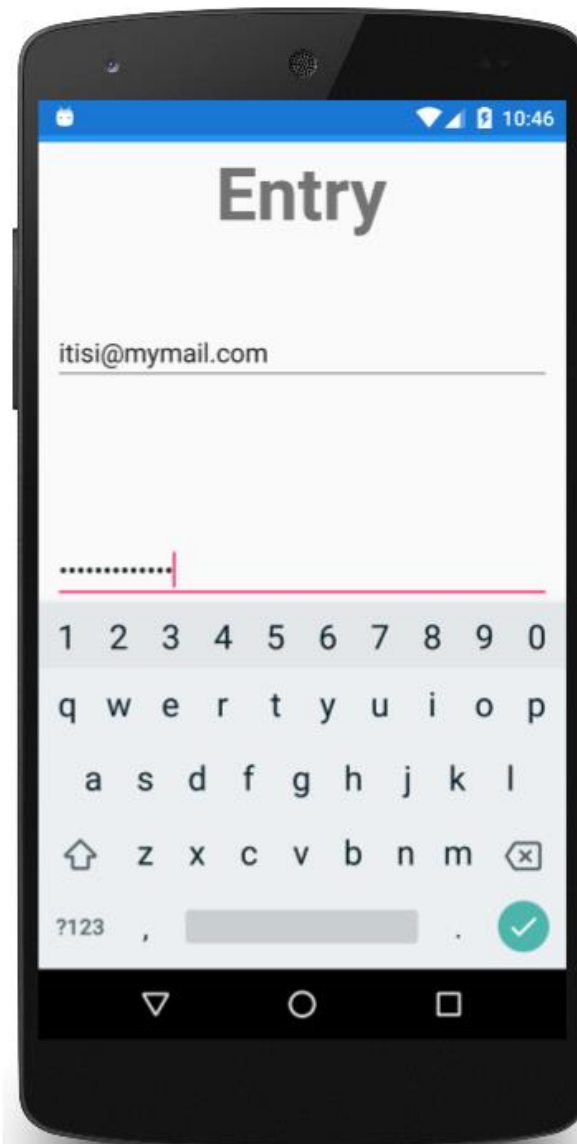
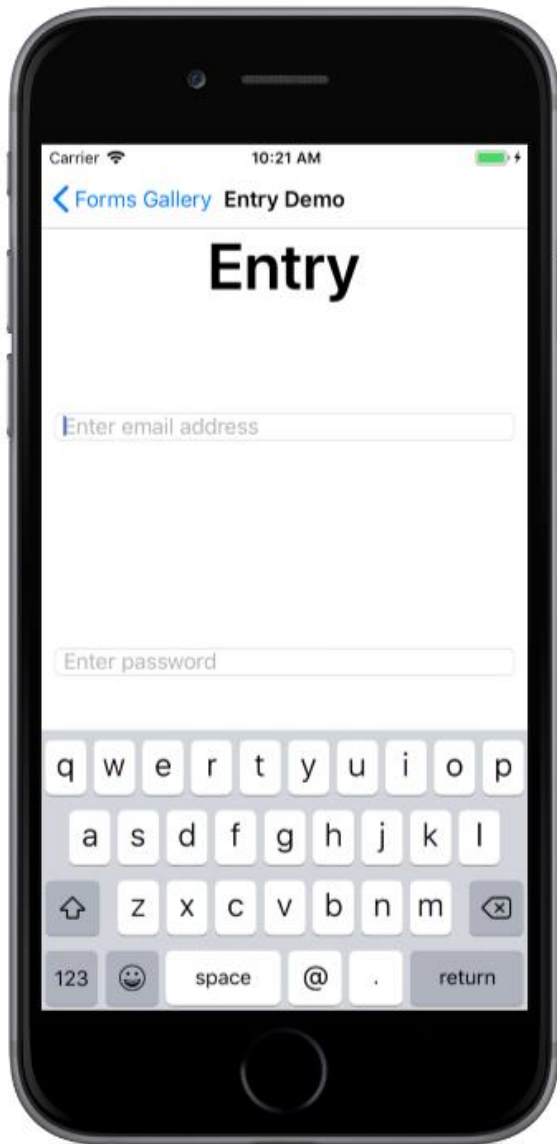


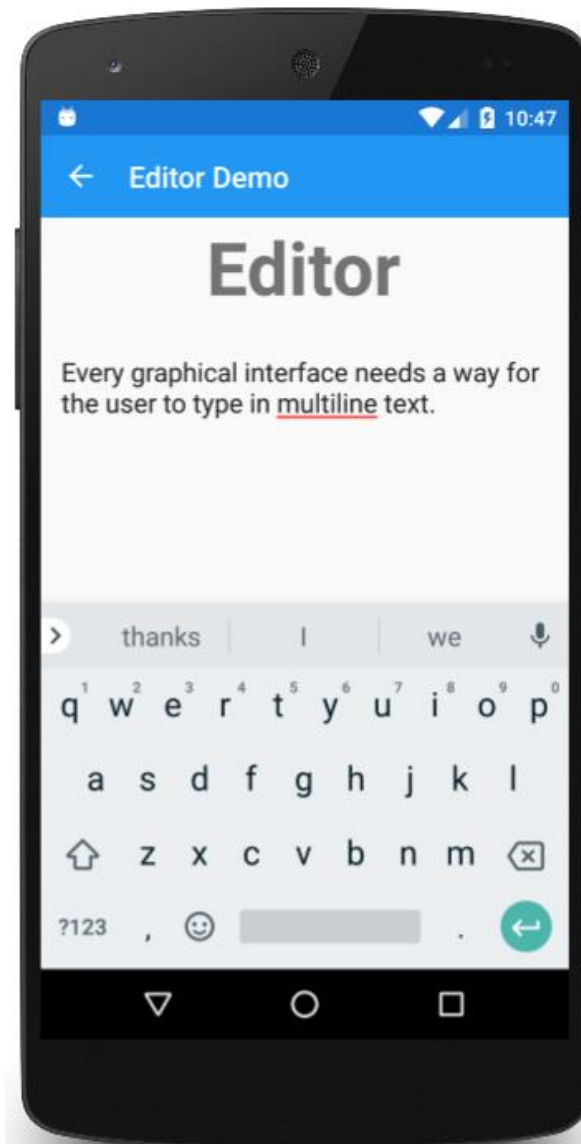


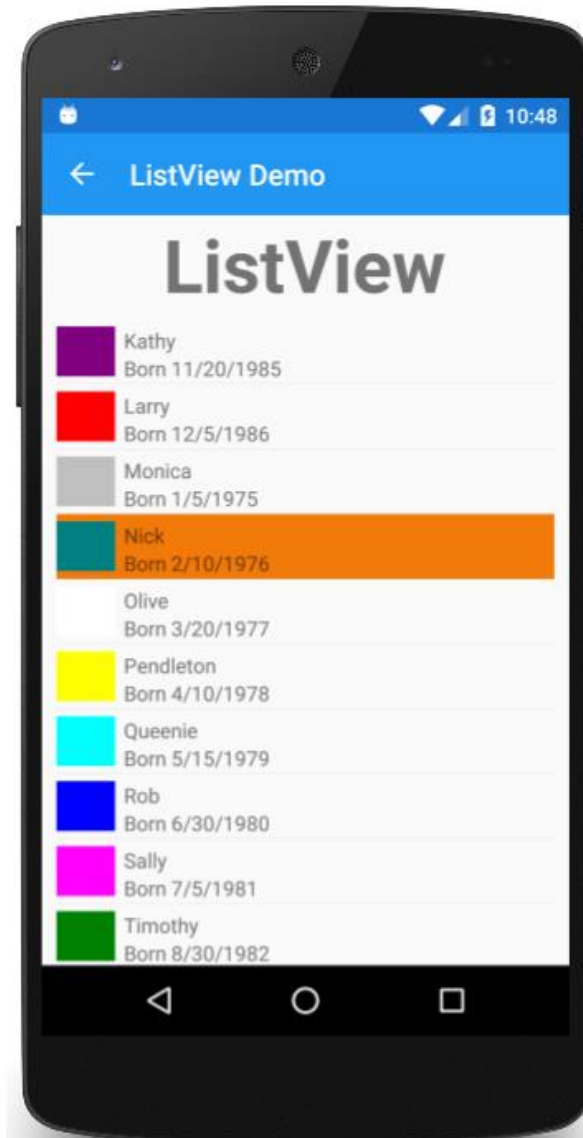














DÉMO

— live



Ressources

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

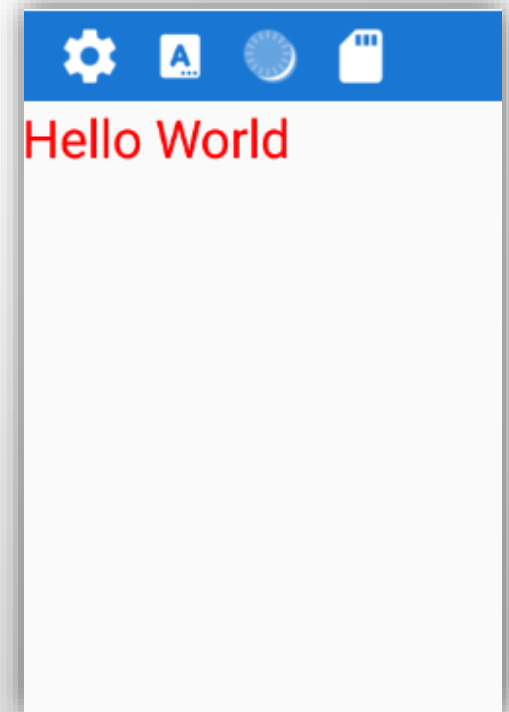
```
<Application.Resources>  
  <x:Double x:Key="LabelSize">14</x:Double>  
</Application.Resources>
```

```
<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

Bindings

```
public class HomeViewModel : ViewModelBase
{
    private string _title;
    private ObservableCollection<Todo> _todos;

    public string Title
    {
        get => _title;
        set => SetProperty(ref _title, value);
    }

    public ObservableCollection<Todo> Todos
    {
        get => _todos;
        set => SetProperty(ref _todos, value);
    }
}
```

Bindings

- La vue définit un contexte pour les bindings dans son constructeur
- Ici HomeViewModel

```
public HomePage()  
{  
    InitializeComponent();  
    BindingContext = new HomeViewModel();  
}
```

Bindings

- Une fois le contexte définit, toutes les **PROPRIÉTÉS PUBLIC** du contexte deviennent disponibles pour le binding

```
<Label Text="{Binding Title}" />
```



```
<Label Text="{Binding _title}" />
```



Bindable Properties

- Bindable Property = Propriété supportant le binding
- Dans une vue/control custom, c'est le seul moyen d'avoir du binding pour vos nouvelles propriétés
- Exemple: je crée un control qui permet de gérer un champ texte et un label au-dessus. J'y ajoute deux BP
 - Title: le texte qui ira dans le label
 - Text: le texte saisi dans l'input

Bindable Properties

```
public static readonly BindableProperty TitleProperty =  
    BindableProperty.Create(  
        propertyName: nameof(Title),  
        returnType: typeof(string),  
        declaringType: typeof(FormControl),  
        defaultValue: null,  
        defaultBindingMode: BindingMode.OneWay,  
        propertyChanged: OnTitlePropertyChanged  
    );  
  
public string Title  
{  
    get => (string)GetValue(TitleProperty);  
    set => SetValue(TitleProperty, value);  
}
```



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 ?



DevOps

DevOps

- La théorie : unifier le développement et l'exploitation
 - Le développeur développe la fonctionnalité, la teste, la pousse en prod et en assure le suivi
 - Tester au plus tôt et de manière automatisé
 - Tester dans un environnement proche de la production
 - Release régulière
 - Boucle de feedback rapide

DevOps

- Le point le plus problématique : release régulière
- Mise en place de :
 - CI = Continuous Integration
 - CT = Continuous Testing
 - CD = Continuous Deployment

DevOps

- Plusieurs solutions existent :
 - Bitrise (Build + Test)
 - Azure DevOps (Build + Test)
 - **AppCenter** (Build + Test + Déploiement)
 - ...

Add new app



App name:

LiveCoding

Icon:



Description:

Enter a brief description (optional)

Owner:



OS:

- ☐ iOS
- ☒ Android
- ☐ Windows
- ☐ macOS Preview

Platform:

- ☐ Java / Kotlin
- ☐ React Native
- ☐ Cordova Preview
- ☒ Xamarin
- ☐ Unity

Using a different platform? [Let us know.](#)

Add new app



LiveCoding

Android



Overview



Build



Test



Distribute



Diagnostics



Analytics



Push



Settings

Select a service



Azure DevOps



GitHub



Bitbucket



● master



This branch has not been configured to build yet.

It sure looks intriguing, though!

LAST COMMIT



Add postman collection + update cours/td su...
Julien Mialon

2 weeks ago

[Configure build](#)

Build app

Project:

LiveCoding2.Android.csproj



Configuration:

Debug



SDK version:

Xamarin.Android 9.0



Build scripts:

None

Learn more about [custom build scripts](#)

Build frequency:



Build this branch on every push



Manually choose when to run builds

Automatically increment version code

Choose a format to increment your builds.



Off

Sign builds

☒ On

Builds must be signed to run on devices.

Keystore



Keystore file:
livecoding.keystore



Environment variables

Keystore password:

.....

Key alias:

.....

Key password:

.....

Test on a real device Free

☒ On

Verify that your build works on a real device by running a launch test.

[!\[\]\(4729e517bc6a7cd81c8025b9646574fb_img.jpg\) Do not use personal data in your tests](#)

Distribute builds

☒ On

Release your app to testers or to a store.

Collaborators



1 tester



Running...

00:37

STARTED
just now

LAUNCH TESTED
N/A

SIGNED
N/A



Build 1

Manual build

DURATION
5 min 59 sec

LAUNCH TESTED
[Results](#)

SIGNED
Yes

Test runs

Date	Version	Duration	Status	Results
Feb 23, 2019, 6:07:39 PM	1.0 (1)	1 min	 PASSED	1 test passed <div></div>

Releases

Release history

Number ↓

Version

Destinations

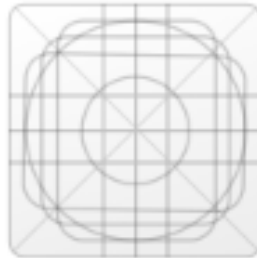


1

1.0 (1)

Collaborators

A new version of **LiveCoding for Android** is
available.



LiveCoding

1.0 (1)

for Android

Install

What's new

Add postman collection + update cours/td subjects

DevOps

- Au final, une fois le build configuré, il ne vous reste plus qu'à coder. À chaque push, votre code est :
 - **Compilé**
 - **Testé**
 - **Déployé**



Aa Bb Cc Dd

$2 + 2 = \dots$

Questions ?

TD

- <https://github.com/Julien-Mialon/Cours-Xamarin>
- Connectez vous au serveur discord
<http://discord.julienmialon.com> et changer votre nom
- Faites un groupe de 3 ou 4 et tagguer les membres de votre groupes dans #groupes