Xamarin Desarrollo rápido de aplicaciones

[**Introducción** 2](#_Toc521531849)

[**1.1** **Que es Xamarin** 2](#_Toc521531850)

[**1.2** **Xamarin Classic Vs Xamarin Forms** 2](#_Toc521531851)

[**1.3** **XAML vs C#** 3](#_Toc521531852)

# **Introducción**

* 1. **Que es Xamarin**

Xamarin es la forma fácil de crear aplicaciones móviles multiplataforma teniendo como base el lenguaje C#. Ya sea para la creación de la UI o para plasmar la lógica de negocios. Con Xamarin un programador sólo se preocupará por factorizar la funcionalidad común entre las diferentes plataformas más usadas hoy día: Android, iOS como principales. Las aplicaciones se crean utilizando el lenguaje de programación C# del Framework .NET, y para hacer más ágil el desarrollo, podemos disponer del ambiente integrado de desarrollo Visual Studio.

La manera en que trabaja Xamarin es que accede directamente a las APIs nativas para renderizar los componentes, es por eso que la salida es básicamente un componente nativo, a diferencia de las apps hibridas, su core seria el webkit del sistema, renderizan un control.

* 1. **Xamarin Classic Vs Xamarin Forms**

*UI*

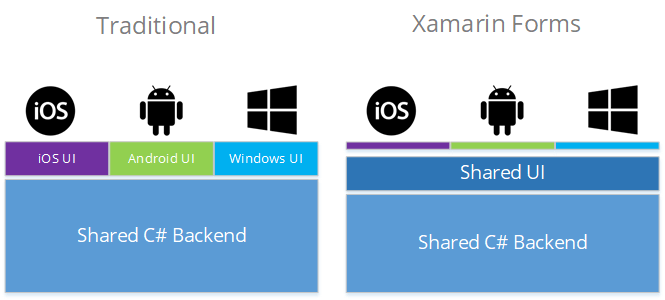
Con Xamarin Classic, tendrás que irte al respectivo archivo de interfaz gráfica de la plataforma que estés codificando, por ejemplo, para Android tendrás que abrir el ó los archivos (.AXML), mientras que en iOS los archivos (.Storyboard) Dentro de estos archivos, tendrás que crear los elementos visuales por plataforma, sin poder compartir ningún elemento gráfico entre los tipos de proyectos que te encuentres trabajando.

Por otra parte, en Xamarin.Forms tienes los archivos (.XAML) en el proyecto común, dentro de los cuales, tendrás que colocar las etiquetas para armar tu interfaz gráfica.

*Lógica de negocios*

En una solución tipo Xamarin Classic, contaremos con un proyecto tipo PCL o Shared, en el cual podremos colocar toda la lógica de negocios requerida por nuestros proyectos, por ejemplo, lo que tiene que ver con servicios web, bases de datos, operaciones, etc.

En un proyecto tipo Xamarin Forms, por otra parte, podremos compartir toda la lógica de negocios requerida por nuestros proyectos, pero adicionalmente podremos compartir nuestros elementos que tienen que ver con la interfaz gráfica. Hay que notar que los elementos se renderizarán de forma nativa de acuerdo a la plataforma en el que lo despleguemos.



* 1. **XAML vs C#**

XAML es un lenguaje de marcado declarativo que puede utilizarse para definir las interfaces de usuario. La interfaz de usuario se define en un archivo XML utilizando la sintaxis XAML, mientras que el comportamiento de tiempo de ejecución se define en un archivo de código subyacente independiente.

Con C# podremos conseguir el mismo resultado, solo que con un lenguaje mas “común”.

Ventaja y desventaja no tienen ninguna, cada desarrollador se adapta al que mejor le parezca, o facilite, y sobre todo en cualquiera de los 2 casos podemos combinar entre XAML y C#.

*Con XAML*

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<ContentPage** xmlns="http://xamarin.com/schemas/2014/forms"
3. xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
4. x:Class="LayoutSamples.StackLayoutDemo"
5. Title="StackLayout Demo"**>**
6. **<ContentPage.Content>**
7. **<StackLayout** Spacing="10" x:Name="layout"**>**
8. **<Button** Text="StackLayout" VerticalOptions="Start"
9. HorizontalOptions="FillAndExpand" **/>**
10. **<BoxView** Color="Yellow" VerticalOptions="FillAndExpand"
11. HorizontalOptions="FillAndExpand" **/>**
12. **<BoxView** Color="Green" VerticalOptions="FillAndExpand"
13. HorizontalOptions="FillAndExpand" **/>**
14. **<BoxView** HeightRequest="75" Color="Blue" VerticalOptions="End"
15. HorizontalOptions="FillAndExpand" **/>**
16. **</StackLayout>**
17. **</ContentPage.Content>**
18. **</ContentPage>**

Con C#

1. **public** **class** StackLayoutCode : ContentPage
2. {
3. **public** StackLayoutCode ()
4. {
5. var layout = **new** StackLayout ();
6. var button = **new** Button { Text = "StackLayout", VerticalOptions = LayoutOptions.Start,
7. HorizontalOptions = LayoutOptions.FillAndExpand };
8. var yellowBox = **new** BoxView { Color = Color.Yellow, VerticalOptions = LayoutOptions.FillAndExpand, HorizontalOptions = LayoutOptions.FillAndExpand };
9. var greenBox = **new** BoxView { Color = Color.Green, VerticalOptions = LayoutOptions.FillAndExpand, HorizontalOptions = LayoutOptions.FillAndExpand };
10. var blueBox = **new** BoxView { Color = Color.Blue, VerticalOptions = LayoutOptions.FillAndExpand,
11. HorizontalOptions = LayoutOptions.FillAndExpand, HeightRequest = 75 };
13. layout.Children.Add(button);
14. layout.Children.Add(yellowBox);
15. layout.Children.Add(greenBox);
16. layout.Children.Add(blueBox);
17. layout.Spacing = 10;
18. Content = layout;
19. }
20. }