Laboratorio

Examinando la anatomía de una aplicación Xamarin.Forms

Versión: 1.0.0
Octubre de 2016

Miguel Muñoz Serafín
@msmdotnet

Soporte: mmunoz@mail.ticapacitacion.com

CONTENIDO

Introducción

Una interfaz de usuario moderna está construida de varios tipos de objetos visuales. Dependiendo del sistema operativo, estos objetos visuales pueden tener diferentes nombres, tales como, Control, Element, View o Widget. Sin embargo, todos ellos tienen la responsabilidad de realizar el trabajo de presentación, interacción o ambos.

En Xamarin.Forms, los objetos que aparecen en la pantalla son llamados colectivamente ***Elementos Visuales***. Los elementos visuales se encuentran agrupados en 3 principales categorías:

- Page
- Layout
- View

La API Xamarin.Forms define clases llamadas ***VisualElement***, ***Page***, ***Layout*** y ***View***. Estas clases y sus descendientes forman la columna vertebral de la interfaz de usuario Xamarin.Forms. ***VisualElement*** es una clase excepcionalmente importante en Xamarin.Forms. Un objeto ***VisualElement*** es cualquier cosa que ocupa un espacio en la pantalla.

Una aplicación Xamarin.Forms consiste de una o más páginas. Una página usualmente ocupa toda (o al menos un área muy grande) de la pantalla. Algunas aplicaciones consisten únicamente de una sola página mientras que otras permiten navegar entre múltiples páginas. La mayoría de las veces, las aplicaciones trabajan simplemente con un tipo de página llamada ***ContentPage***.

En cada página, los elementos visuales se encuentran organizados en una jerarquía de padre e hijo. El hijo de una ***ContentPage*** es generalmente un elemento contenedor *(Layout)* que permite organizar los elementos visuales. Algunos contenedores tienen un solo hijo mientras que la mayoría de los contenedores tienen múltiples hijos que son organizados dentro de él. Estos hijos pueden ser otros contenedores o Vistas *(Views)*. Diferentes tipos de contenedores organizan a sus hijos en un Stack, en una cuadrícula de dos dimensiones o de una forma libre.

El término ***View*** en Xamarin.Forms denota tipos familiares de objetos de presentación e interacción: texto, mapas de bits, botones, campos para entrada de texto, deslizadores (sliders), conmutadores (switches), barras de progreso, selectores de fecha y hora, y otros objetos de creación propia. Estos objetos son llamados frecuentemente como Controles o Widgets en otros ambientes de programación.

En este laboratorio, exploraremos la estructura de una aplicación multiplataforma Xamarin.Forms. En un siguiente laboratorio, analizaremos las distintas opciones para compartir código en una aplicación multiplataforma y exploraremos el *View* ***Label*** para mostrar texto en una aplicación multiplataforma desarrollada con Xamarin.Forms.

Objetivos

Al finalizar este laboratorio, los participantes serán capaces de:

- Describir la estructura de una aplicación Xamarin.Forms.

Requisitos

Para la realización de este laboratorio es necesario contar con lo siguiente:

- Un equipo de desarrollo con sistema operativo Windows 10 y Visual Studio 2015 Community, Professional o Enterprise con la plataforma Xamarin.
- Un equipo Mac con la plataforma Xamarin.

Tiempo estimado para completar este laboratorio: **60 minutos**.

Ejercicio 1: Explorando la estructura de una aplicación Xamarin.Forms

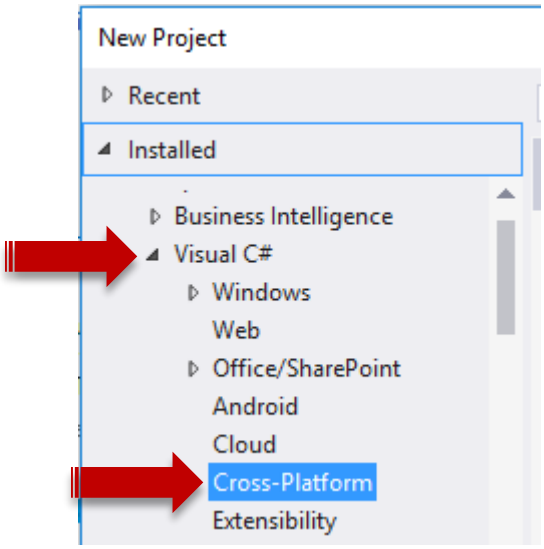
En este ejercicio crearás una aplicación Xamarin.Forms y examinarás la estructura de archivos de cada proyecto que forman parte de una solución multiplataforma Xamarin.Forms.

Tarea 1. Crear la aplicación Xamarin.Forms.

En esta tarea crearás una aplicación Xamarin.Forms utilizando Microsoft Visual Studio y una plantilla estándar. El proceso creará una solución que contendrá 6 proyectos:

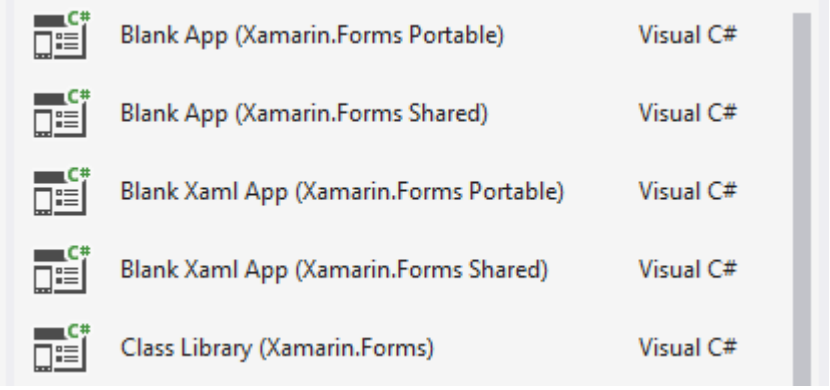
- 5 proyectos específicos de cada plataforma. iOS, Android, UWP, Windows 8.1 y Windows Phone.
- 1 proyecto común para la mayor parte del código compartido de la aplicación.

1. Selecciona la opción **File > New > Project** desde Visual Studio.
2. En el panel izquierdo de la ventana **New Project** selecciona **Visual C# > Cross-Platform**.

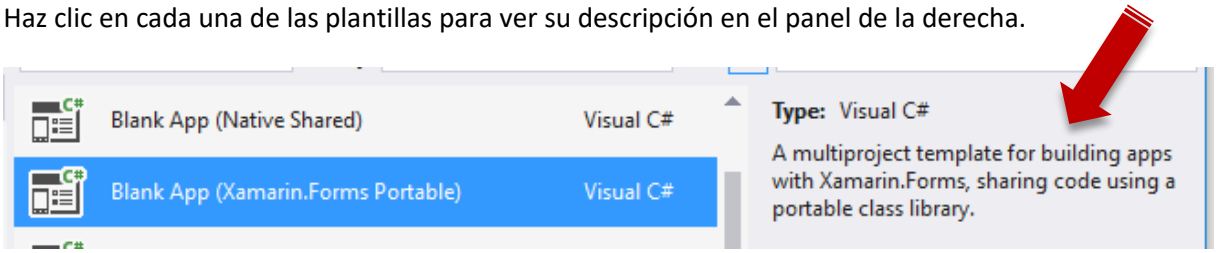


3. En el panel central de la ventana **New Project** puedes ver distintas plantillas de solución disponibles incluyendo 5 de Xamarin.Forms:

    a. Blank App (Xamarin.Forms Portable)
    b. Blank App (Xamarin.Forms Shared)
    c. Blank Xaml App (Xamarin.Forms Portable)
    d. Blank Xaml App (Xamarin.Forms Shared)
    e. Class Library (Xamarin.Forms)



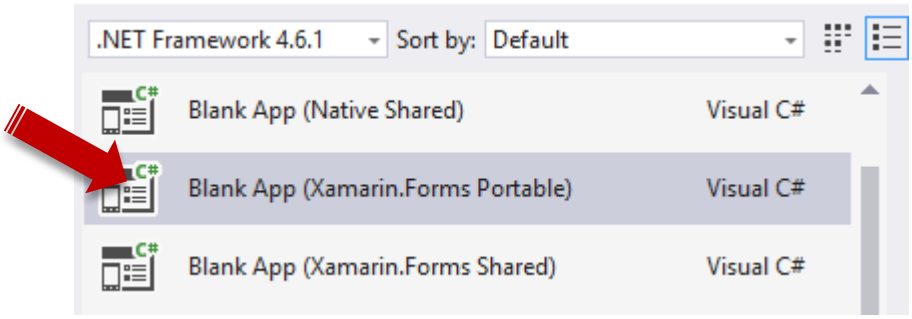
Haz clic en cada una de las plantillas para ver su descripción en el panel de la derecha.



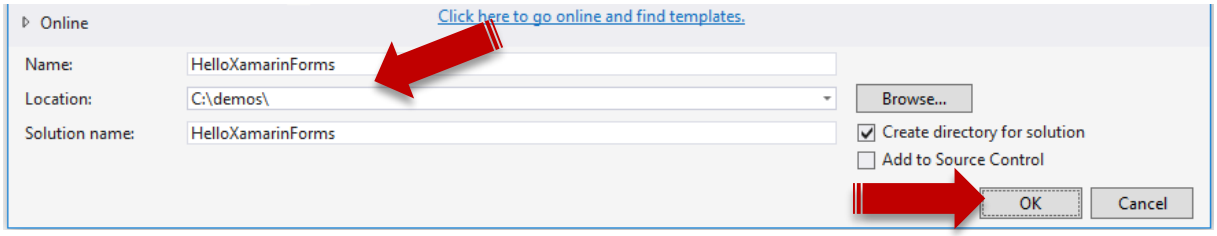
El término "**Portable**" en este contexto se refiere a una Biblioteca de Clases Portable (Portable Class Library – PCL). Todo el código común de la aplicación se convierte en una biblioteca DLL que es referenciada por todos los proyectos de plataforma individuales.

El término "**Shared**" en este contexto se refiere a un Proyecto de Recursos Compartidos (Shared Asset Project – SAP). Este proyecto contiene archivos de código y otro tipo de archivos que son compartidos con los proyectos de cada plataforma, volviéndose esencialmente parte de cada proyecto de cada plataforma.

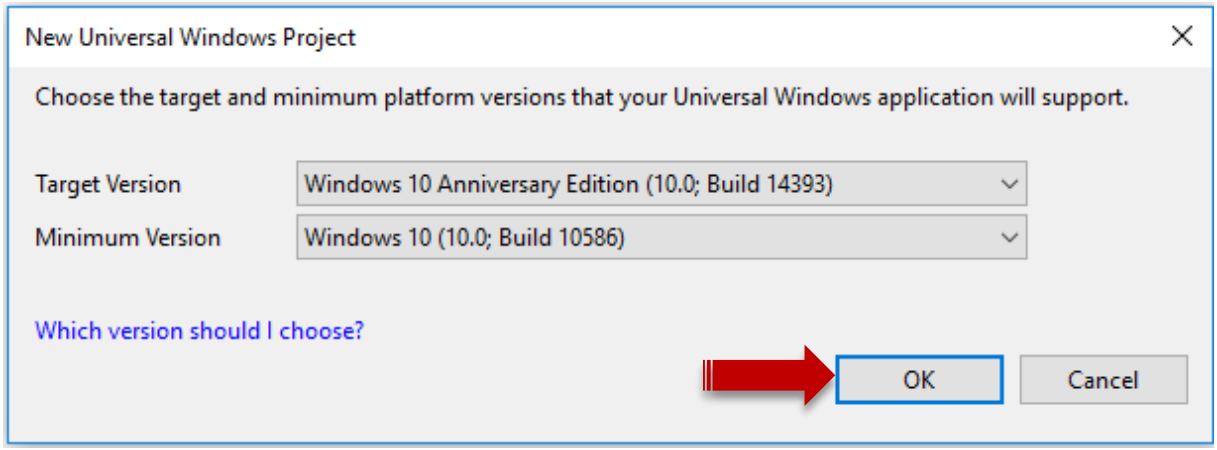
4. Selecciona la plantilla **Blank App (Xamarin.Forms Portable)**.



5. Proporciona el nombre, ubicación y haz clic en **OK** para crear el proyecto.

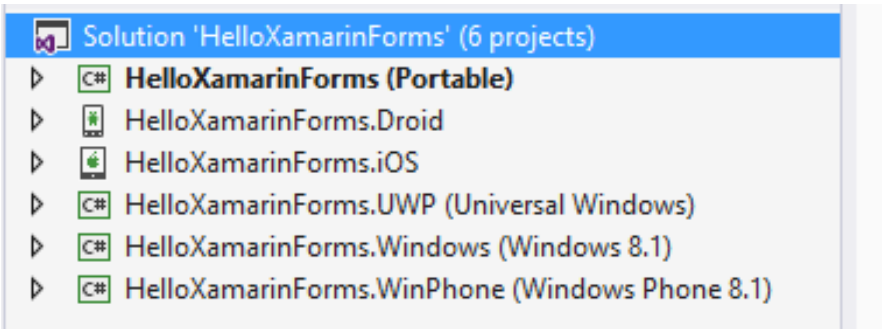


6. Haz clic en el botón **OK** del cuadro de diálogo **New Universal Windows Project** para aceptar las versiones sugeridas para la aplicación UWP que será creada.



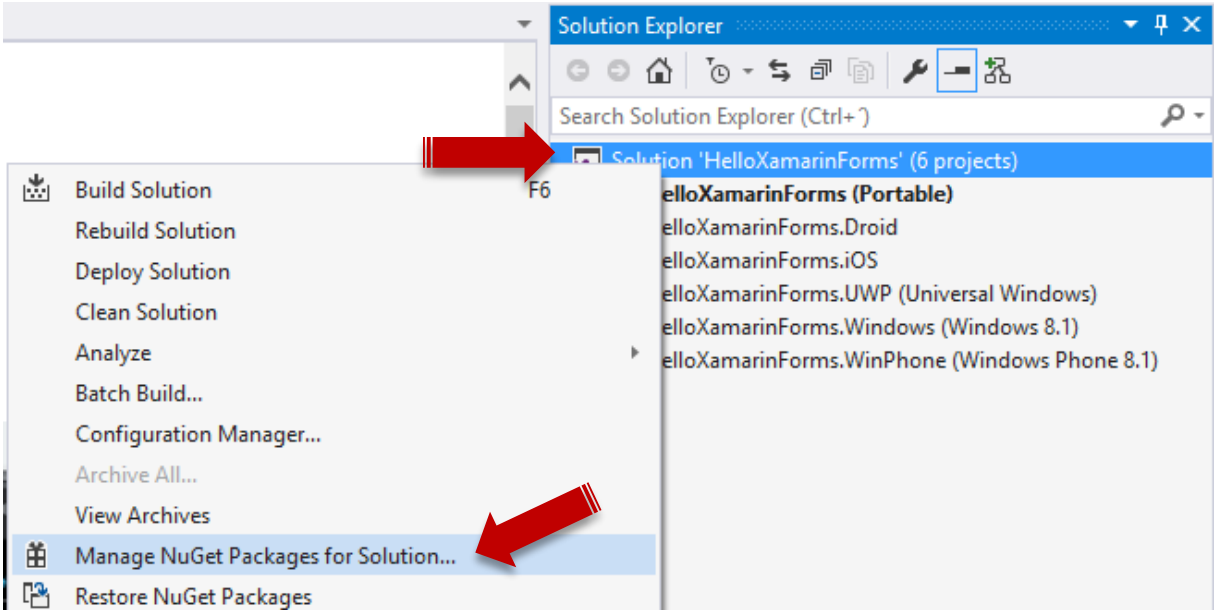
Seis proyectos son creados. Para una solución llamada **HelloXamarinForms**, estos proyectos son:

a. Un proyecto PCL llamado **HelloXamarinForms** que es referenciado por los otros 5 proyectos.
b. Un proyecto de aplicación para Android llamado **HelloXamarin.Droid**.
c. Un proyecto de aplicación para iOS llamado **HelloXamarin.iOS**.
d. Un proyecto de aplicación para la UWP de Windows 10 y Windows Mobile 10 llamado **HelloXamarinForms.UWP**.
e. Un proyecto de aplicación para Windows 8.1 llamado **HelloXamarinForms.Windows**.
f. Un proyecto de aplicación para Windows Phone 8.1 llamado **HelloXamarinForms.WinPhone**.

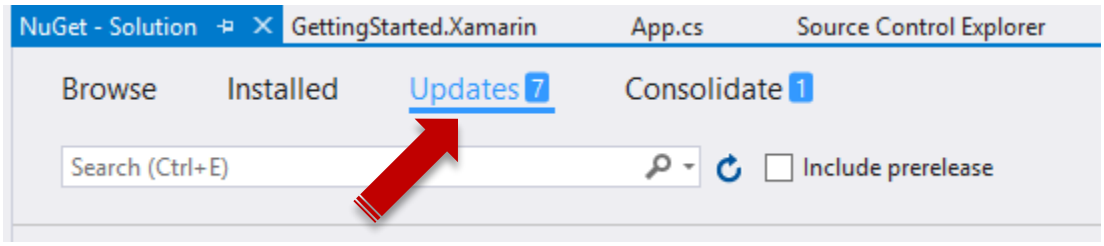


Cuando creas una nueva solución Xamarin.Forms, las bibliotecas Xamarin.Forms y otras bibliotecas auxiliares son descargadas automáticamente desde el administrador de paquetes NuGet. Visual Studio almacena esas bibliotecas en un directorio llamado **packages** dentro del directorio de la solución. Sin embargo, la versión particular de la biblioteca Xamarin.Forms que es descargada es especificada dentro de la plantilla de la solución y, por lo tanto, una nueva versión podría estar disponible.

7. Selecciona la opción **Manage NuGet Packages for Solution** desde el menú contextual del nombre de la solución.

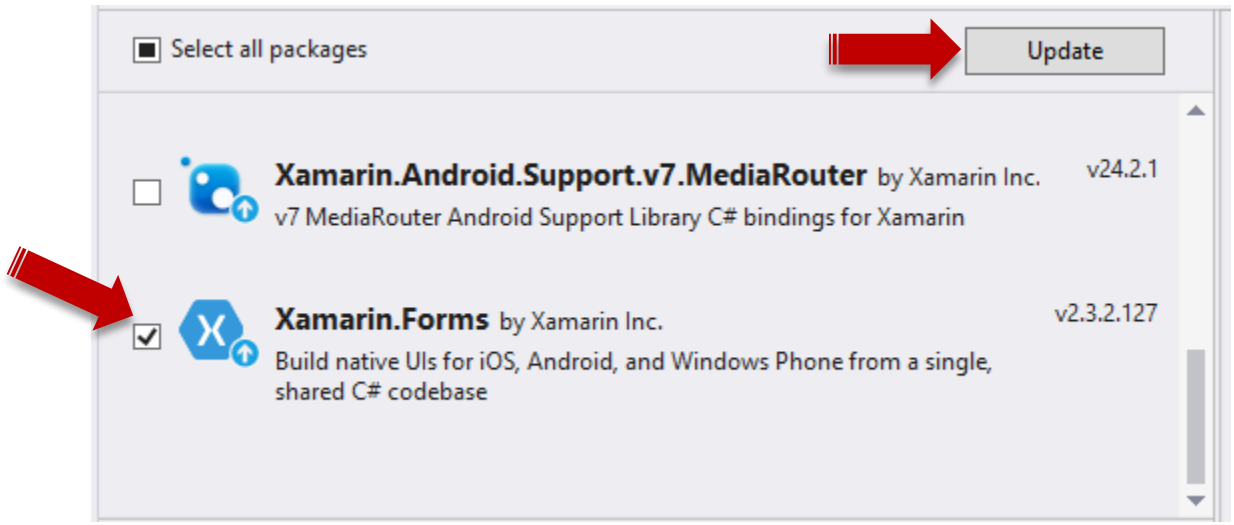


8. Haz clic en la opción **Updates** para ver las actualizaciones disponibles.



Visual Studio puede indicarte que existen actualizaciones para el paquete NuGet Xamarin.Forms y todas sus dependencias, sin embargo, Xamarin.Forms está configurado para dependencias de versiones específicas. Por lo tanto, aunque Visual Studio te indique que hay nuevas versiones disponibles de paquetes Xamarin.Android.Support, Xamarin.Forms no es necesariamente compatible con esas nuevas versiones.

9. Selecciona el paquete **Xamarin.Forms** y haz clic en **Update** para iniciar la actualización.



Es probable que te sea solicitado aceptar el acuerdo de licencias y reiniciar Visual Studio para concluir la instalación.

En caso de que selecciones todos los paquetes, podrías obtener un mensaje de error similar al siguiente debido al problema de compatibilidad de versiones mencionado anteriormente:

Unable to resolve dependencies. 'Xamarin.Android.Support.Design 24.2.1' is not compatible with 'Xamarin.Forms 2.3.2.127 constraint: Xamarin.Android.Support.Design (= 23.3.0)'

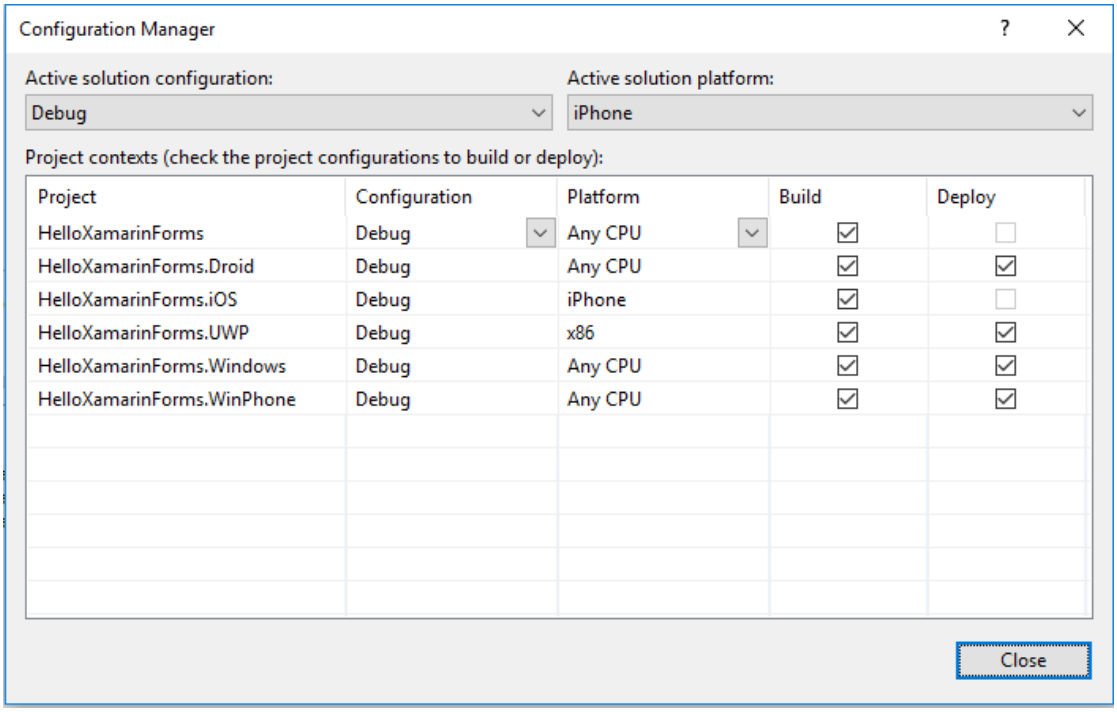
Puedes consultar el siguiente enlace para mayor información acerca del mensaje de error.

https://developer.xamarin.com/guides/xamarin-forms/troubleshooting/

10. Una vez que haya concluido la actualización selecciona la opción **Build > Configuration Manager**.

11. En la ventana de diálogo **Configuration Manager** podrás ver el proyecto PCL y los otros 5 proyectos de aplicaciones.

    Asegúrate que la casilla de verificación **Build** este seleccionada para todos los proyectos y que la casilla **Deploy** este seleccionada para todos los proyectos de aplicación (a menos que la casilla esté gris).



Observa que en la columna **Platform**:

- El proyecto PCL tiene establecido el único valor disponible **Any CPU**.
- El proyecto Android tiene establecido el único valor disponible **Any CPU**.
- El proyecto iOS puede tomar los valores **iPhone** o **iPhoneSimulator** dependiendo de cómo estaremos probando la aplicación.
- El proyecto UWP puede tomar el valor **x86** para desplegar la aplicación hacia el escritorio de Windows o hacia un emulador. También puede tomar el valor **ARM** para desplegar la aplicación hacia un teléfono.
- El proyecto Windows puede tomar el valor **x86** para desplegar la aplicación hacia el escritorio de Windows o hacia un emulador. También puede tomar el valor **ARM** para desplegar la aplicación hacia una tableta. Incluso puede tomar el valor **x64** para desplegar la aplicación hacia plataformas de 64 bits o el valor **Any CPU** para desplegar la aplicación en el escritorio de Windows, emulador, tableta o plataformas de 64 bits.
- El proyecto WinPhone puede tomar el valor **x86** para desplegar la aplicación hacia un emulador, el valor **ARM** para desplegar la aplicación hacia un teléfono real o el valor **Any CPU** para desplegar la aplicación en ambos, emulador y teléfono real.

12. Haz clic en **Close** para cerrar la ventana de dialogo **Configuration Manager**.

13. Opcionalmente, en caso de que desees mostrar la barra de herramientas de iOS y Android para acceder a los dispositivos y emuladores correspondientes, puedes habilitar las opciones **View > Toolbars > iOS** y **View > Toolbars > Android**.

14. Selecciona la opción **Set As StartUp Project** del menú contextual del proyecto Android para establecer la aplicación Android como proyecto de inicio.

15. Despliega la aplicación en el emulador de tu preferencia.

    Podrás ver la aplicación desplegada en el emulador Android que seleccionaste.

16. Regresa a Visual Studio y detén la ejecución.

17. Selecciona la opción **Set As StartUp Project** del menú contextual del proyecto iOS para establecer la aplicación iOS como proyecto de inicio.

18. Despliega la aplicación en el emulador de tu preferencia.

En la Mac, podrás ver la aplicación desplegada en el emulador iOS que seleccionaste.

19. Regresa a Visual Studio y detén la ejecución.
20. Selecciona la opción **Set As StartUp Project** del menú contextual del proyecto UWP para establecer la aplicación UWP como proyecto de inicio.
21. Despliega la aplicación en el emulador Windows Mobile de tu preferencia.

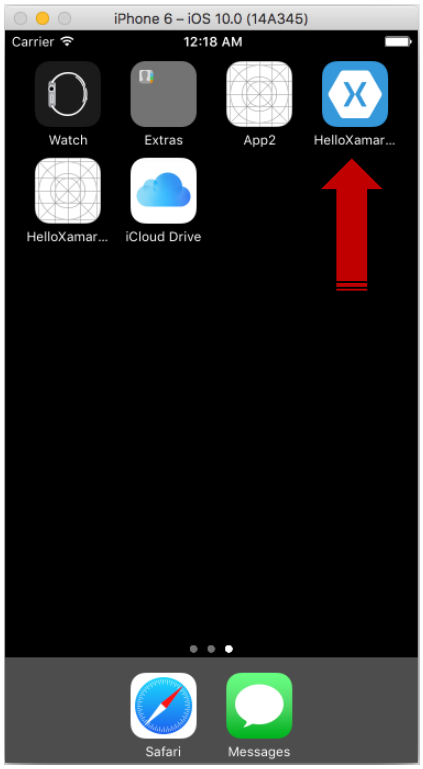
    Podrás ver la aplicación desplegada en el emulador Windows Mobile que seleccionaste.

22. Regresa a Visual Studio y detén la ejecución.

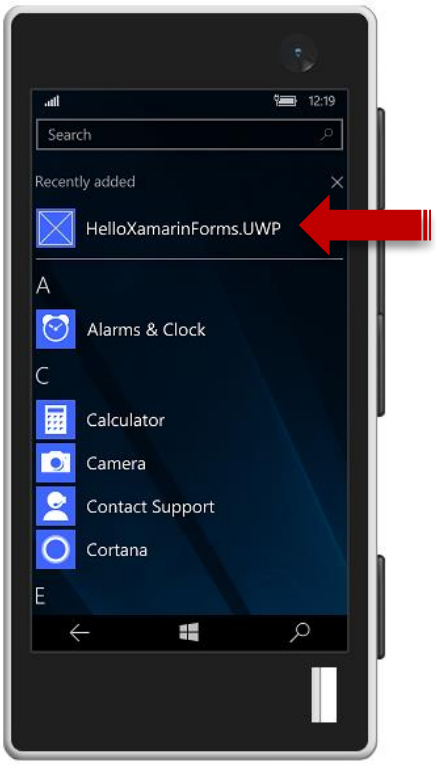
De manera predeterminada, todas las plataformas están habilitadas para cambios en la orientación. Puedes rotar cualquiera de los emuladores y podrás darte cuenta de que el texto se ajusta al nuevo centro de la pantalla.

La aplicación no solo se ejecuta en el dispositivo o emulador, sino que también es desplegada. La aplicación aparece con las otras aplicaciones del teléfono o emulador y puede ser ejecutada desde ahí.

Android



iOS



Windows 10 Mobile

Si no te gusta el icono de la aplicación o la forma en que se muestra el nombre de la aplicación, tu puedes cambiar eso en los proyectos individuales de cada plataforma.

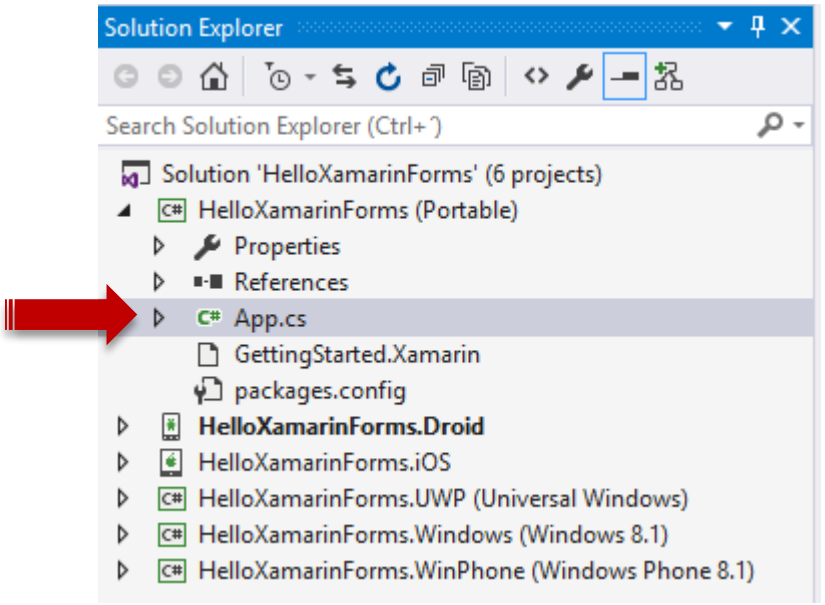
Tarea 2. Examinar los archivos de código generados.

Claramente, el programa creado por la plantilla Xamarin.Forms es muy simple, por lo mismo, representa una excelente oportunidad para examinar los archivos de código generados, descubrir la forma en que se interrelacionan y la forma en que trabajan.

El proyecto PCL es el proyecto que recibirá la mayor parte de nuestra atención cuando estemos escribiendo una aplicación Xamarin.Forms. En algunas circunstancias el código en este proyecto podría requerir algo de código particular de ciertas plataformas.

1. Empecemos con el código que es responsable de presentar el texto que ves en la pantalla. Esta es la clase **App** en el proyecto común PCL.

   Abre el archivo **App.cs** ubicado dentro del proyecto PCL.



2. Examina la definición de la clase App.

   Puedes notar que su espacio de nombres es el mismo que el nombre del proyecto.

```
namespace HelloXamarinForms
{
    public class App : Application
    {
```

   También puedes notar que la clase está definida como pública y que deriva de la clase **Xamarin.Forms.Application**.

3. Examina el código del constructor.

La única responsabilidad del constructor es la de establecer el valor de la propiedad **MainPage** de la clase **Application** a un objeto de tipo **Page**.

```
public App()
{
    // The root page of your application
    var content = new ContentPage
    {
        Title = "HelloXamarinForms",
        Content = new StackLayout
        {
            VerticalOptions = LayoutOptions.Center,
            Children = {
                new Label {
                    HorizontalTextAlignment = TextAlignment.Center,
                    Text = "Welcome to Xamarin Forms!"
                }
            }
        }
    };

    MainPage = new NavigationPage(content);
}
```

La clase **ContentPage** deriva de la clase **Page** y es muy común utilizarla en aplicaciones Xamarin.Forms de una sola página. **ContentPage** ocupa gran parte de la pantalla ya que, dependiendo de la plataforma, podría compartir espacio con barras de estado o herramientas.

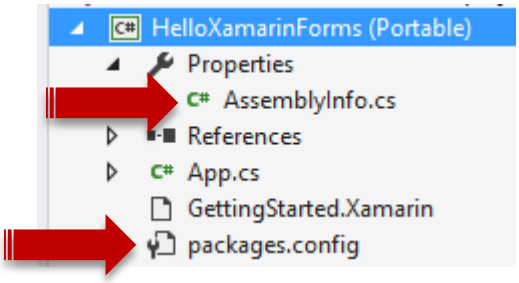
La clase **ContentPage** define una propiedad llamada **Content** a la cual le establecemos el contenido de la página. Generalmente el contenido es un contenedor que aloja diversos **Views**. En el caso del código generado, el contenedor es un **StackLayout** que acomoda a sus hijos en una pila.

La propiedad **VerticalOptions** del **StackLayout** permite especificar la alineación vertical de su contenido.

En el código generado, **StackLayout** contiene un único hijo que es un View **Label**. La clase **Label** deriva de la clase **View** y es utilizada en aplicaciones Xamarin.Forms para mostrar texto.
La propiedad **HorizontalTextAlignment** del **Label** permite especificar el alineamiento horizontal de su contenido almacenado en la propiedad **Text**.

En la vida real, en nuestras aplicaciones Xamarin.Forms de una sola página, generalmente definimos nuestra propia clase que deriva de **ContentPage**. El constructor de la clase **App** establece en su propiedad **MainPage** una instancia de la clase que definimos.

4. Observa que el proyecto PCL contiene el archivo **AssemblyInfo.cs** con información del Assembly. Contiene también el archivo **packages.config** que lista los paquetes NuGet requeridos por el programa.



5. Abre la sección **References** del proyecto PCL y nota que existen al menos 4 bibliotecas que el proyecto PCL requiere.

    - .NET
    - Xamarin.Forms.Core
    - Xamarin.Forms.Platform
    - Xamarin.Forms.Xaml

Los 5 proyectos de aplicación tienen sus propios recursos (assets) en forma de iconos y metadatos. Es importante poner atención en esos recursos al momento de publicar la aplicación hacia la tienda de la plataforma, pero mientras estés aprendiendo a desarrollar aplicaciones con Xamarin.Forms puedes ignorar por el momento esos recursos. Incluso, puedes mantener contraídos los proyectos de esas aplicaciones debido a que no necesitas preocuparte mucho de su contenido.

Veamos que hay dentro de esos proyectos de aplicación.

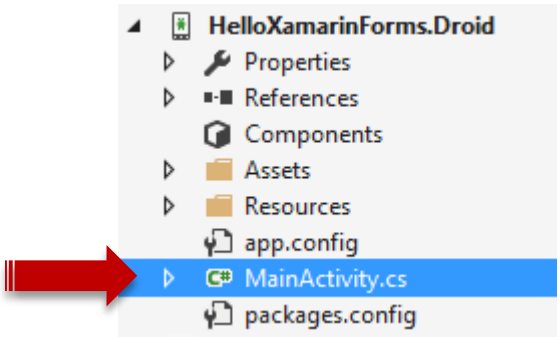
6. Examina la sección **References** de cada proyecto de aplicación. Ahí podrás ver la referencia al proyecto común PCL, así como varios Assemblies .NET, los Assemblies Xamarin.Forms listados anteriormente y los Assemblies adicionales Xamarin.Forms aplicables a cada plataforma:

    - Xamarin.Forms.Platform.Android
    - Xamarin.Forms.Platform.iOS
    - Xamarin.Forms.Platform.UAP (no mostrado explícitamente en el proyecto UWP)
    - Xamarin.Forms.Platform.WinRT
    - Xamarin.Forms.Platform.WinRT.Tablet
    - Xamarin.Forms.Platform.WinRT.Phone

    Cada una de estas bibliotecas define un método estático **Forms.Init** en el espacio de nombres **Xamarin.Forms** que inicializa el sistema Xamarin.Forms para esa plataforma particular. El código de arranque en cada plataforma debe realizar una llamada a ese

método. El código de arranque de cada plataforma también debe instanciar la clase **App** del proyecto PCL.
Veamos como realiza ese trabajo el código de arranque de cada plataforma.

7. Abre el archivo **MainActivity.cs** del proyecto Android.



En una aplicación Android, la clase típica **MainActivity** debe ser derivada de una clase Xamarin.Forms llamada **FormsAppCompatActivity** definida en el Assembly **Xamarin.Forms.Platform.Android**.

```
[Activity(Label = "HelloXamarinForms", Icon = "@drawable/icon",
    Theme = "@style/MainTheme", MainLauncher = true,
    ConfigurationChanges =
    ConfigChanges.ScreenSize | ConfigChanges.Orientation)]

public class MainActivity :
    global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
```

El atributo establecido en la clase **MainActivity** indica que la actividad no será recreada cuando el teléfono cambie su orientación (de horizontal a vertical y viceversa) o cuando cambie el tamaño de pantalla.

8. Examina el código del método **OnCreate**. En este método se realiza la llamada al método **Forms.Init**.

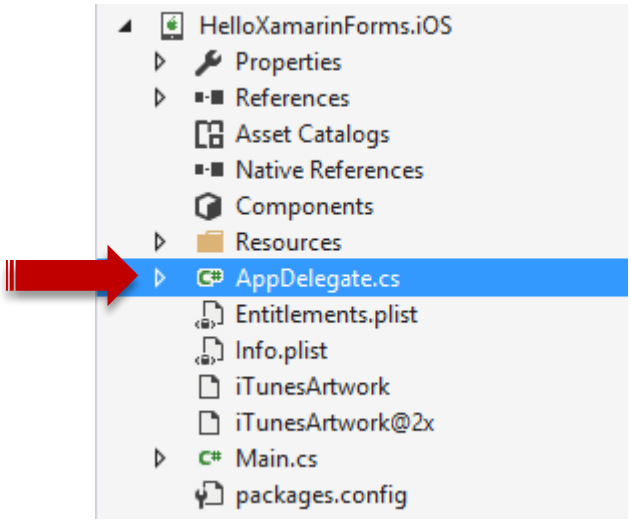
```
protected override void OnCreate(Bundle bundle)
{
    TabLayoutResource = Resource.Layout.Tabbar;
    ToolbarResource = Resource.Layout.Toolbar;

    base.OnCreate(bundle);

    global::Xamarin.Forms.Forms.Init(this, bundle);
    LoadApplication(new App());
}
```

Observa que la nueva instancia de la clase **App** definida en el proyecto PCL es pasada al método **LoadApplication** definido por **FormsAppCompatActivity**.

9. Abre el archivo **AppDelegate.cs** dentro del proyecto iOS.



Un proyecto iOS contiene típicamente una clase que deriva de **UIApplicationDelegate**. Sin embargo, la biblioteca **Xamarin.Forms.Platform.iOS** define una clase base alternativa llamada **FormsApplicationDelegate**.

El siguiente es el código que define la clase **AppDelegate** que deriva de **FormsApplicationDelegate**.

```
[Register("AppDelegate")]
public partial class AppDelegate :
    global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
```

10. Examina el código del método **FinishedLaunching**.

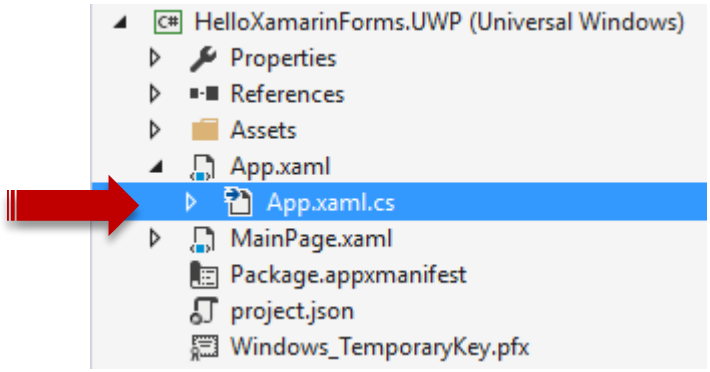
```
    public override bool FinishedLaunching(UIApplication app,
        NSDictionary options)
    {
        global::Xamarin.Forms.Forms.Init();
        LoadApplication(new App());

        return base.FinishedLaunching(app, options);
    }
```

Este método inicia con una llamada al método **Forms.Init** definido en el Assembly **Xamarin.Forms.Platform.iOS**. Posteriormente invoca al método **LoadApplication** definido en la clase **FormsApplicationDelegate** pasándole una nueva instancia de la clase **App** definida en el proyecto PCL.

El objeto Page asignado a la propiedad **MainPage** de ese objeto **App** puede ser utilizado para crear un objeto de tipo **UIViewController** que es responsable de generar el contenido de la página.

11. En el proyecto UWP o cualquiera de los proyectos Windows, abre el archivo **App.xaml.cs**.



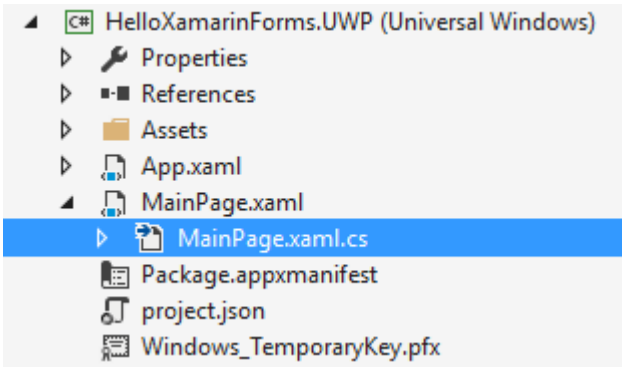
En el archivo **App.xaml.cs** se encuentra definida una clase llamada **App** que deriva de la clase **Application** definida en la biblioteca **Windows.UI.Xaml**.

```
sealed partial class App : Application
```

12. Examina el código del método **OnLaunched**. El código de este método invoca al método **Forms.Init** de la siguiente manera:

```
Xamarin.Forms.Forms.Init(e);
```

13. Abre ahora el archivo **MainPage.xaml.cs**.



Este archivo define una clase llamada **MainPage** que deriva de una clase Xamarin.Forms especificada en el elemento raíz del archivo **MainPage.xaml**.

14. Examina el código del constructor de la clase **MainPage**.

```
public MainPage()
{
    this.InitializeComponent();

    LoadApplication(new HelloXamarinForms.App());
}
```

En este método, una nueva instancia de la clase **App** es pasada al método **LoadApplication** definido en la clase base **Xamarin.Forms.Platform.UWP.WindowsBasePage**.

Resumen

En este laboratorio exploraste la anatomía de una aplicación Xamarin.Forms.

Si has creado una solución Xamarin.Forms y no quieres desarrollar para una o más plataformas, simplemente elimina esos proyectos.

Si posteriormente cambias de idea acerca de esos proyectos, puedes agregar nuevos proyectos de la plataforma deseada a la solución Xamarin.Forms.

En conclusión, no hay nada especial entre aplicaciones Xamarin.Forms comparadas con aplicaciones normales Xamarin o proyectos Windows, excepto las bibliotecas Xamarin.Forms.

Es tiempo de explorar ahora las diferencias entre las opciones disponibles para compartir código en una aplicación multiplataforma con Xamarin.Forms.

Cuando hayas finalizado este laboratorio publica el siguiente mensaje en Twitter y Facebook:

¡He finalizado el #Lab02 del #XamarinDiplomado y conozco la estructura de una aplicación Xamarin.Forms!