CENTRO PARA EL DESARROLLO DE LA INDUSTRIA DEL SOFTWARE | UANL

Xamarin

Desarrollo Móvil con Xamarin

1 - Introducción a Xamarin.Forms

Contenido

- ¿Qué es Xamarin?
- ¿Qué es Xamarin.Forms?
- Componentes de Xamarin.Forms
  - Pages
  - Views
  - Layouts
- Código específico a las plataformas (C#)
  - Xamarin.Essentials
- Definir la UI con XAML
  - Código específico a las plataformas (XAML)

¿Qué es Xamarin?

- Los sistemas operativos de los dispositivos móviles (Android y iOS) proveen abstracciones de alto nivel que contienen herramientas que la mayoría de los desarrolladores utilizan para construir apps.

- Estas abstracciones son los **SDKs** (Software Development Kits).

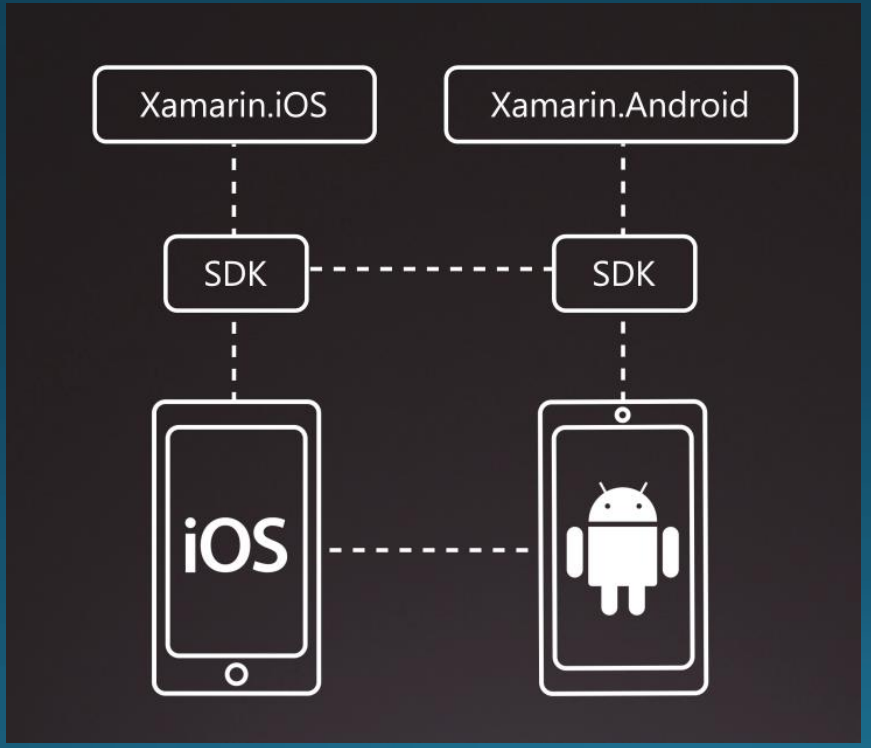
- Típicamente, para trabajar con el SDK de Android se utiliza Java o Kotlin, y para el SDK de iOS se utiliza Objective-C o Swift.

- **Xamarin** es una plataforma para el desarrollo móvil, que permite construir aplicaciones nativas para iOS y Android, compartiendo el código entre estas plataformas.

- Las aplicaciones se desarrollan utilizando C# y librerías de .NET, en Visual Studio.

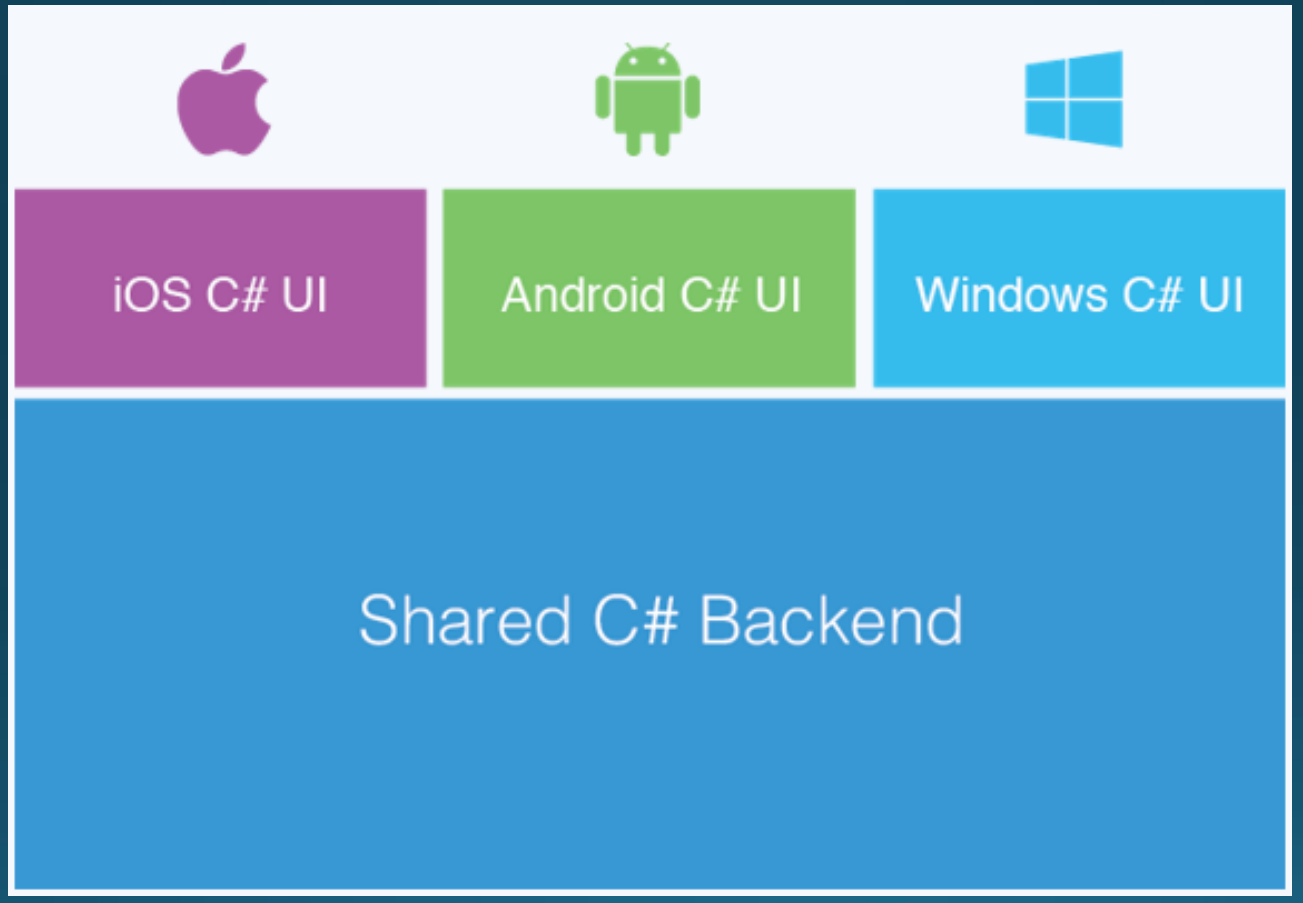
- En Xamarin, siempre habrá código específico a las plataformas (Android, iOS). En promedio, este código conformará el 15-30% de los proyectos.

- Xamarin.Android y Xamarin.iOS trabajan con los SDKs de cada plataforma utilizando C# y librerías de .NET.



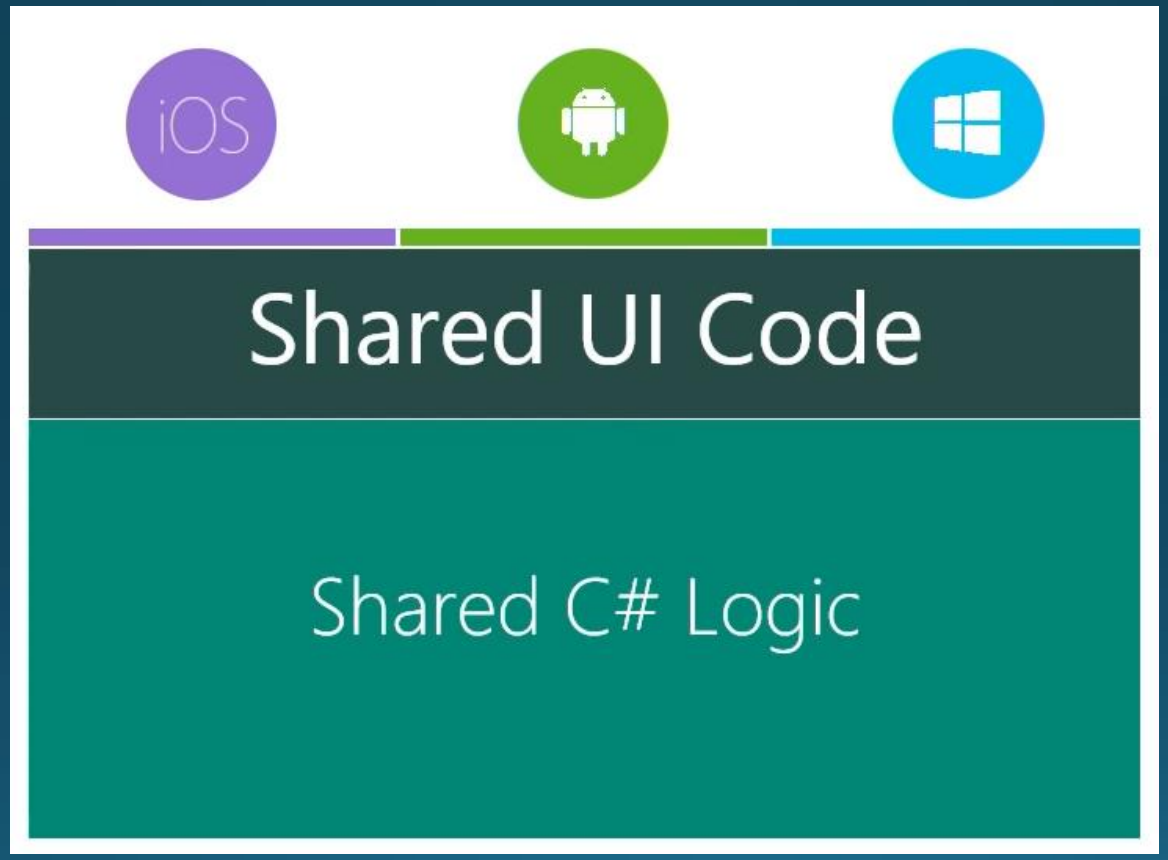
- Xamarin.Android y Xamarin.iOS son una opción muy buena cuando el objetivo es compartir código de backend (lógica de negocios, acceso a servicios web, operaciones a bases de datos, etc).

- Pero con este enfoque, se debe crear la UI (interfaz del usuario) para cada plataforma de forma separada.

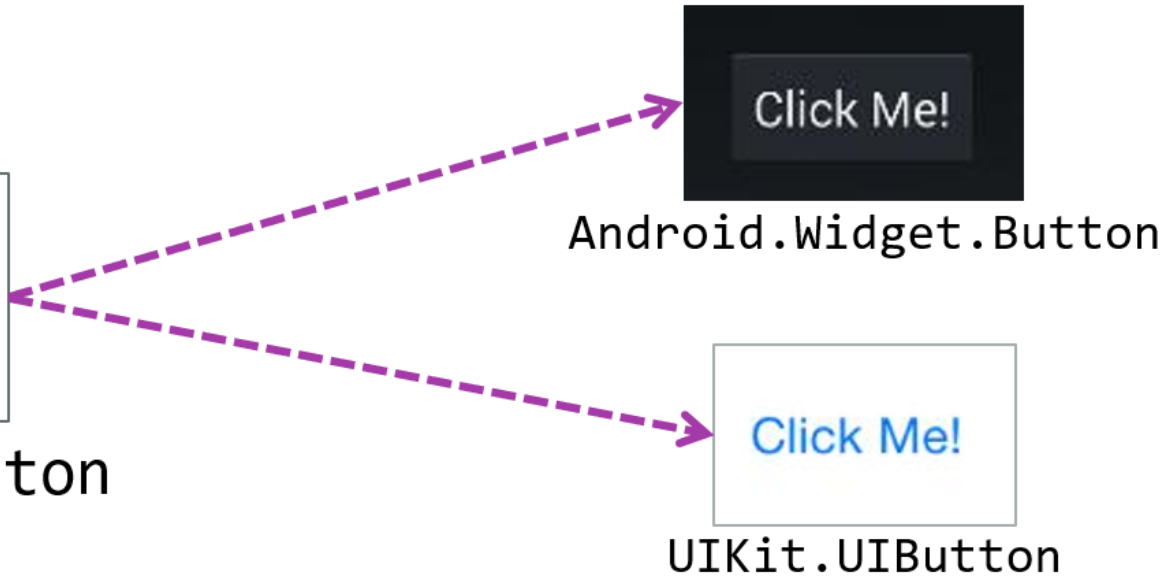

iOS C# UI
Android C# UI
Windows C# UI
Shared C# Backend

¿Qué es Xamarin.Forms?

- En el enfoque de Xamarin.Forms, además de compartir el código del backend, también se comparte el código de la UI.
- Esto se logra al definir la UI (pantallas, layouts, controles) una sola vez, en código C# o XAML.
- En tiempo de ejecución, cada plataforma transforma la UI definida en Xamarin.Forms a sus respectivos controles nativos.


iOS
Shared UI Code
Shared C# Logic

```
var button = new Button {
    Text = "Click Me!"
};
```



UI defined using a Xamarin.Forms **Button**

- Ejercicio #1 – Crear app de Xamarin.Forms

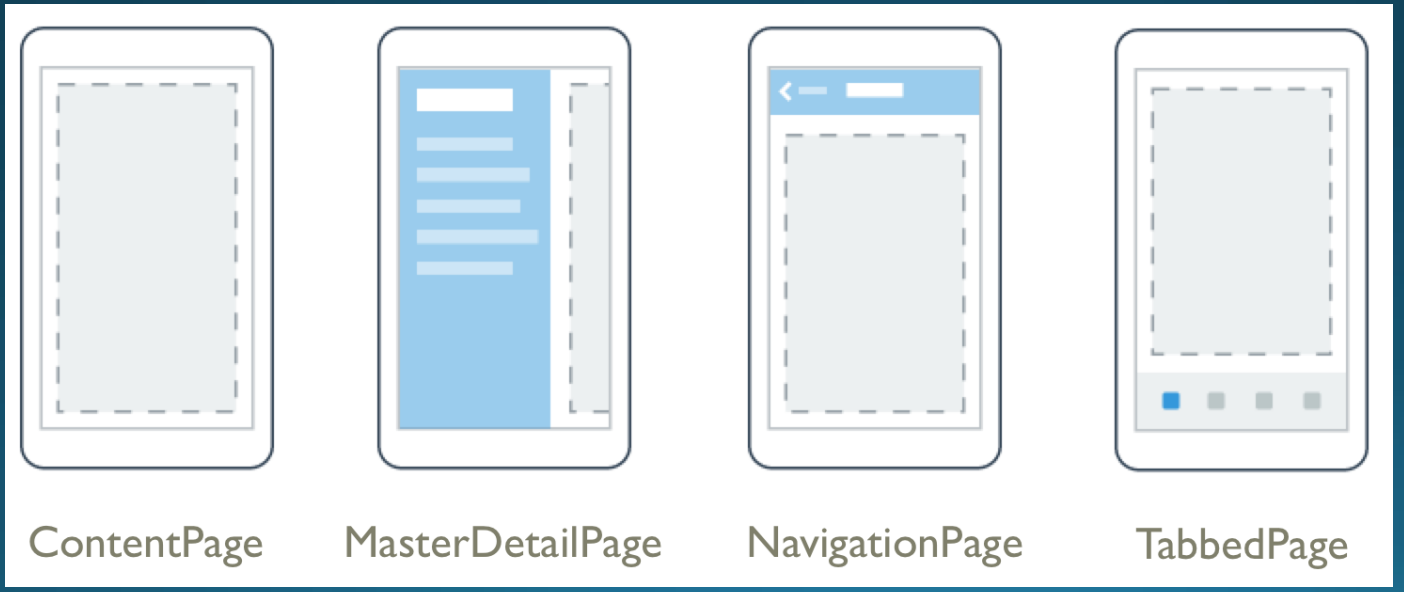
Componentes de Xamarin.Forms

- La clase **Application** es el punto de entrada de la aplicación:
  - Es utilizada por el código de las plataformas específicas para inicializar la aplicación.
  - Siempre va "apuntar" a la primera pantalla de la aplicación.
  - Define métodos para tratar con los eventos del ciclo de vida de la aplicación.

Pages

- La clase **Page** define una pantalla en la aplicación.

- Varias clases derivan de **Page**, y representan diferentes tipos de pantallas:



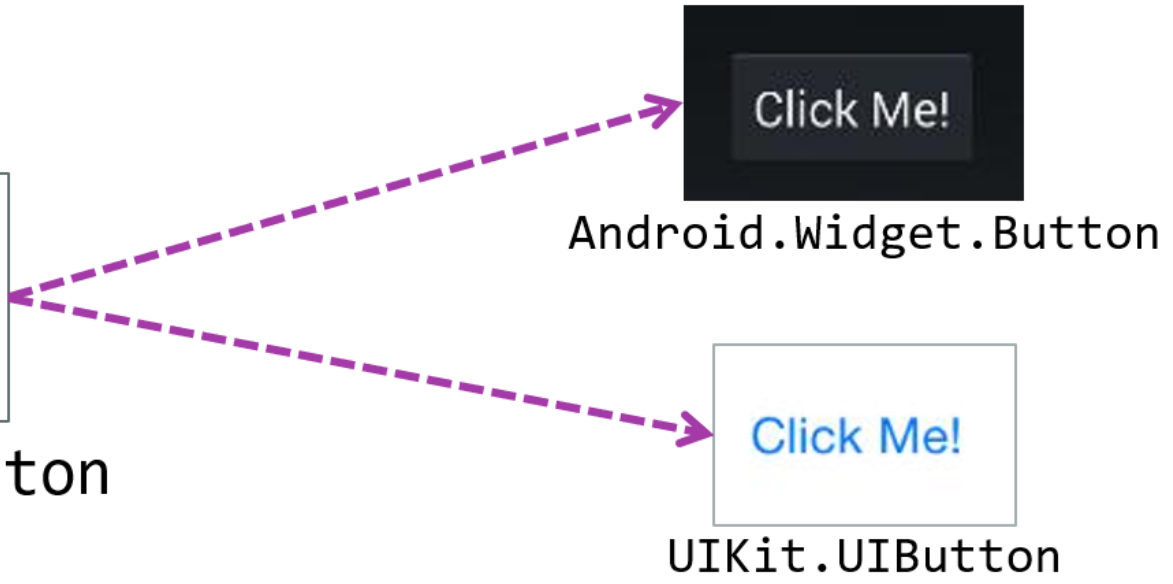
Views

- La clase **View** define los controles típicos que se tendrán en la aplicación.

- Varias clases derivan de **View**, por ejemplo:
  - Label, Image, Map -> Controles de presentación.
  - Button, SearchBar -> Controles para iniciar comandos (eventos).
  - Slider, DatePicker, TimePicker -> Controles para asignar valores.

- En tiempo de ejecución, cada plataforma convierte los controles definidos con la clase **View** a controles nativos.

```
var button = new Button {
    Text = "Click Me!"
};
```



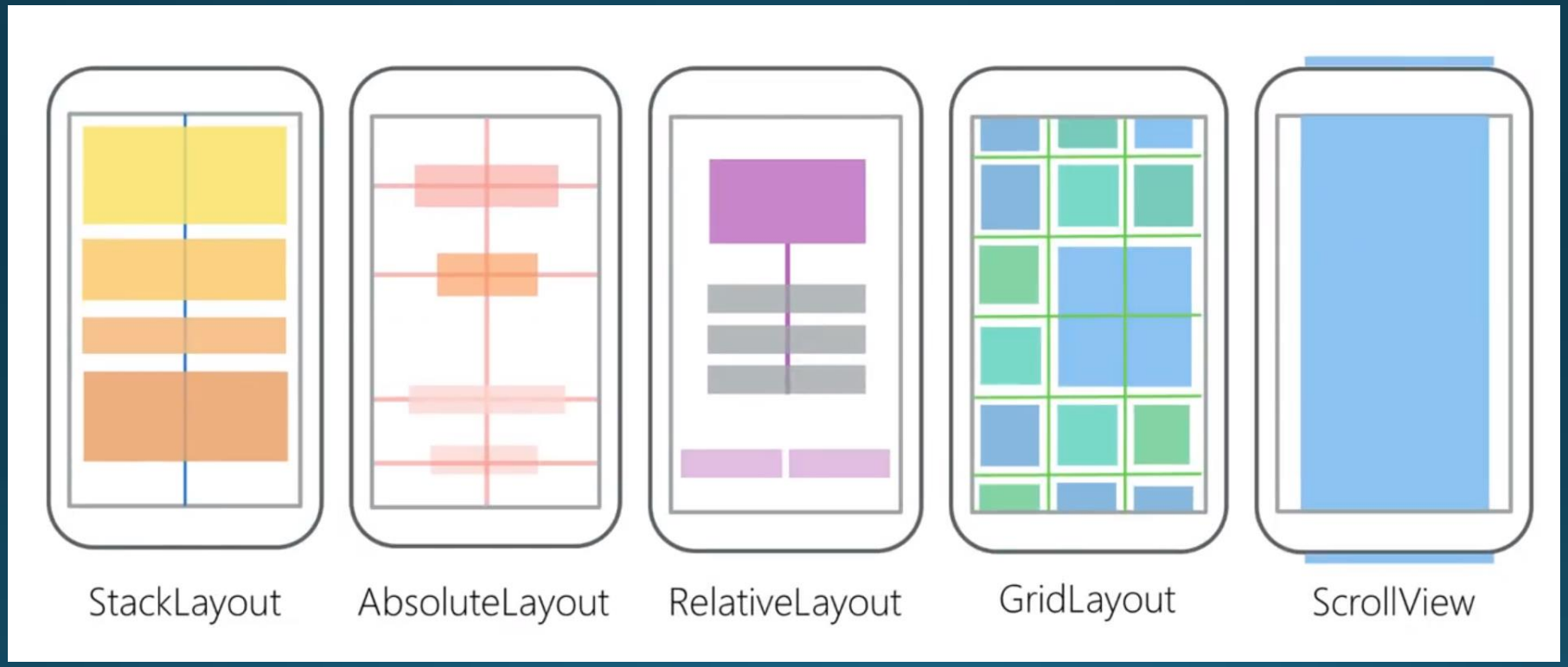
UI defined using a Xamarin.Forms **Button**

Layouts

- Para crear múltiples controles (**Views**) en una pantalla, éstos deben estar definidos dentro de un contenedor.

- La clase **Layout** actúa como un contenedor tanto de controles (**Views**), como de otros contenedores (**Layouts**).

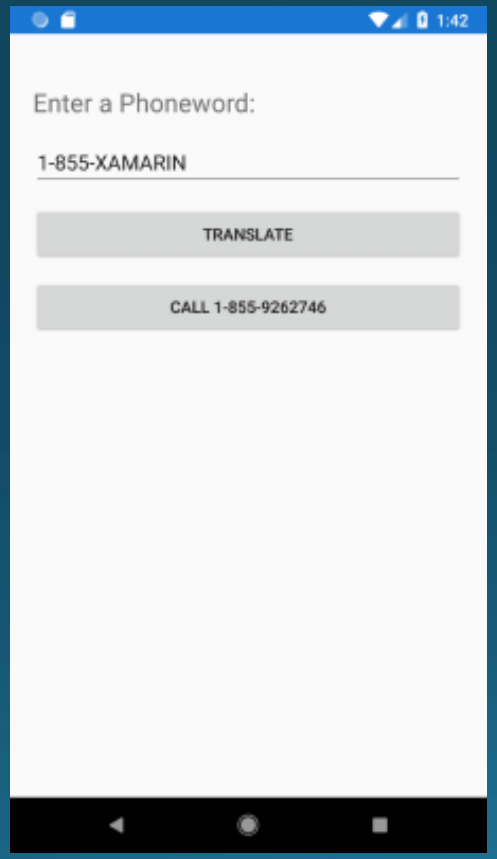
- Define el tamaño y la posición de sus elementos hijos (los elementos que contiene).


StackLayout
AbsoluteLayout
RelativeLayout
GridLayout
ScrollView

- Definir espacio (whitespace) entre los controles que forman parte de un layout, mejorará notablemente el aspecto de las pantallas. Los siguientes atributos definen este espacio:

- **Margin:** define el espacio de un control con respecto a controles adyacentes.
- **Padding:** define el espacio del layout con respecto al borde de la pantalla.
- **Spacing:** define el espacio entre todos los controles dentro de un layout.

- Ejercicio #2 – Crear la pantalla Phoneword (API nativa)



Código específico a las plataformas (C#)

- En Xamarin.Forms, **siempre** habrá código específico para cada plataforma.
- Dependiendo del tamaño y los requerimientos de la aplicación, el porcentaje de este código específico puede variar (lo normal es que esté en un rango del 15% al 30%).
- Una parte importante del código específico tiene que ver con la utilización de APIs nativas de cada plataforma.

- **DependencyService** es una clase que permite la utilización de código específico a las plataformas, desde una llamada en el código compartido.

- Una interfaz se define en el código compartido, y **DependencyService** encuentra la implementación de esa interfaz en los proyectos de las plataformas.

- El código específico que se define en la implementación de la interfaz típicamente tiene que ver con la utilización de APIs nativas.

- Continuar con el Ejercicio #2

Xamarin.Essentials

- **Xamarin.Essentials** provee el acceso a múltiples APIs nativas de las plataformas (Android, iOS) desde una única API en el código compartido.

- Entre las APIs nativas se incluyen a las que proporcionan los siguientes servicios/componentes:

  - Llamadas
  - Batería
  - Acelerómetro
  - Mapas
  - Sistema de archivos
  - Conectividad
  - Correo
  - SMS

- Ejercicio #3 – Modificar la pantalla Phoneword para utilizar Xamarin.Essentials

Definir la UI con XAML

- **XAML** (e**X**tensible **A**pplication **M**arkup **L**anguage) es un lenguaje de marcado para crear interfaces de usuario (UI).
- En nuestro ejercicio, tanto la UI como su comportamiento están definidos en C#.
- **XAML** permite que la UI esté definida en su propio archivo, de manera que nuestro código C# solo defina comportamiento.

```
public MainPage()
{
  prompt          = new Label ();
  phoneNumberText = new Entry ();
  translateButton = new Button();
  callButton      = new Button();

  var panel = new StackLayout();
  panel.Children.Add(prompt);
  panel.Children.Add(phoneNumberText);
  panel.Children.Add(translateButton);
  panel.Children.Add(callButton);

  this.Content = panel;
}
```

```
<ContentPage>

    <ContentPage.Content>

        <StackLayout >
            <Label  />
            <Entry  />
            <Button />
            <Button />
        </StackLayout>

    </ContentPage.Content>

</ContentPage>
```

- Para acceder a los elementos definidos en XAML a través del codebehind (C#), hay que asignarles un nombre.

- Esto se logra con la propiedad **x:Name** de XAML.

- Ejercicio #4 – Definir la UI de Phoneword con XAML

Código específico a las plataformas (XAML)

- La experiencia visual de la aplicación será diferente en cada plataforma.
- Para lograr una experiencia similar en ambas plataformas, se deberán especificar ciertos valores de los elementos de la UI de manera distinta para cada plataforma.
- Esto se puede lograr en C# y en XAML, pero como estamos hablando de UI, es más apropiado trabajar en XAML.

- Ejercicio #5 – Agregar código específico a las plataformas (XAML)