PROGRAMACIÓN MULTIMEDIA

Y

DISPOSITIVOS MÓVILES

UT2. Interfaces de usuario de aplicaciones móviles multiplataforma

Departamento de Informática y Comunicaciones

CFGS Desarrollo de Aplicaciones Multiplataforma

Segundo Curso

IES Virrey Morcillo

Villarrobledo (Albacete)

Índice

1. Introducción

Una **interfaz de usuario (UI)**, hace referencia a la interfaz con la que las personas interaccionan con los dispositivos. Al mismo tiempo, una interfaz de usuario también permite que el dispositivo envíe una retroalimentación al usuario, de modo que éste pueda dar cuenta de si su acción se ha llevado a cabo con éxito.

Hace tiempo que el objetivo ya no sólo es conseguir una interfaz de usuario útil, sino que el aspecto estético también juega un papel muy importante. Por consiguiente, la interfaz de usuario es imprescindible para una buena **experiencia de usuario (UX)**, es decir, para la experiencia que un usuario tiene a grosso modo con una aplicación. A este respecto, el objetivo de los diseñadores es sentar las bases de una buena experiencia de usuario con una interfaz de usuario intuitiva, lo que funciona especialmente bien a través de una interfaz gráfica de usuario.

En definitiva, la interfaz de usuario de una aplicación es todo aquello que el usuario puede ver y todo aquello con lo que éste puede interactuar.

2. La interfaz de usuario de Xamarin.Forms

El conjunto de herramientas de **Xamarin.Forms** permite a los diseñadores crear interfaces de usuario de forma nativa para aplicaciones multiplataforma.

Proporciona su propia abstracción para la interfaz de usuario, que se representa mediante controles nativos de iOS, Android o Plataforma universal de Windows (UWP). Esto significa que las aplicaciones pueden compartir gran parte del código de su interfaz de usuario y conservar la apariencia nativa de la plataforma de destino.

Existen dos técnicas para crear interfaces de usuario en Xamarin.Forms:

1) La primera técnica consiste en crear las interfaces de usuario por completo con código fuente de C#.
2) La segunda técnica consiste en usar XAML o el Lenguaje de Marcado Extensible para Aplicaciones (eXtensible Application Markup Language).

En el siguiente ejemplo se muestra como instanciar e inicializar un objeto de tipo **Label** de Xamarin.Forms utilizando código:

```
public class App : Application {
	public App () {
		MainPage = new ContentPage {
			Content =  new StackLayout {
				VerticalOptions = LayoutOptions.CenterAndExpand,
				Children = {
					new Label {
						Text = "Hola desde el código!",
						HorizontalOptions = LayoutOptions.Center
					}
				}
			}
		};
	}
}
```

En este otro ejemplo se muestra como instanciar e inicializar un objeto de tipo **Label** en Xamarin.Forms utilizando XAML:

```
<?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:HolaXamarin"
             x:Class="HolaXamarin.MainPage">

    <StackLayout>
        <Label Text="Hola desde XAML!"
           HorizontalOptions="Center"
           VerticalOptions="CenterAndExpand" />
    </StackLayout>

</ContentPage>
```

XAML es un lenguaje basado en el estándar XML creado por Microsoft como una alternativa al código de programación en un lenguaje determinado, por ejemplo, en C# para instanciar e inicializar objetos y organizar estos objetos en jerarquías de tipo padre – hijo.

XAML también forma parte de Xamarin.Forms, de esta manera es posible diseñar la interfaz de aplicaciones multiplataforma para dispositivos móviles basadas en iOS, Android y UWP. En el archivo XAML, el desarrollador de Xamarin.Forms puede definir las interfaces de usuario utilizando los elementos propios Xamarin.Forms como las vistas, los diseños y las páginas, así como clases personalizadas.

El archivo XAML se puede compilar o incrustar en el archivo ejecutable. En cualquier caso, la información contenida en el archivo XAML se analiza en tiempo de compilación para localizar el nombre de los objetos y en tiempo de ejecución para instanciar e inicializar esos objetos y para establecer las relaciones entre esos objetos y el código de programación en C#.

XAML tiene varias ventajas con respecto al código equivalente en lenguaje C#:

- ✓ XAML suele ser más concisas y legibles que el código equivalente.
- ✓ La jerarquía de elementos padre – hijo inherente en XML permite imitar con mayor claridad visual los elementos de la jerarquía de objetos de la interfaz de usuario.
- ✓ XAML puede ser fácilmente escrito a mano por los programadores, pero también se presta para ser generado por las herramientas de diseño visual.

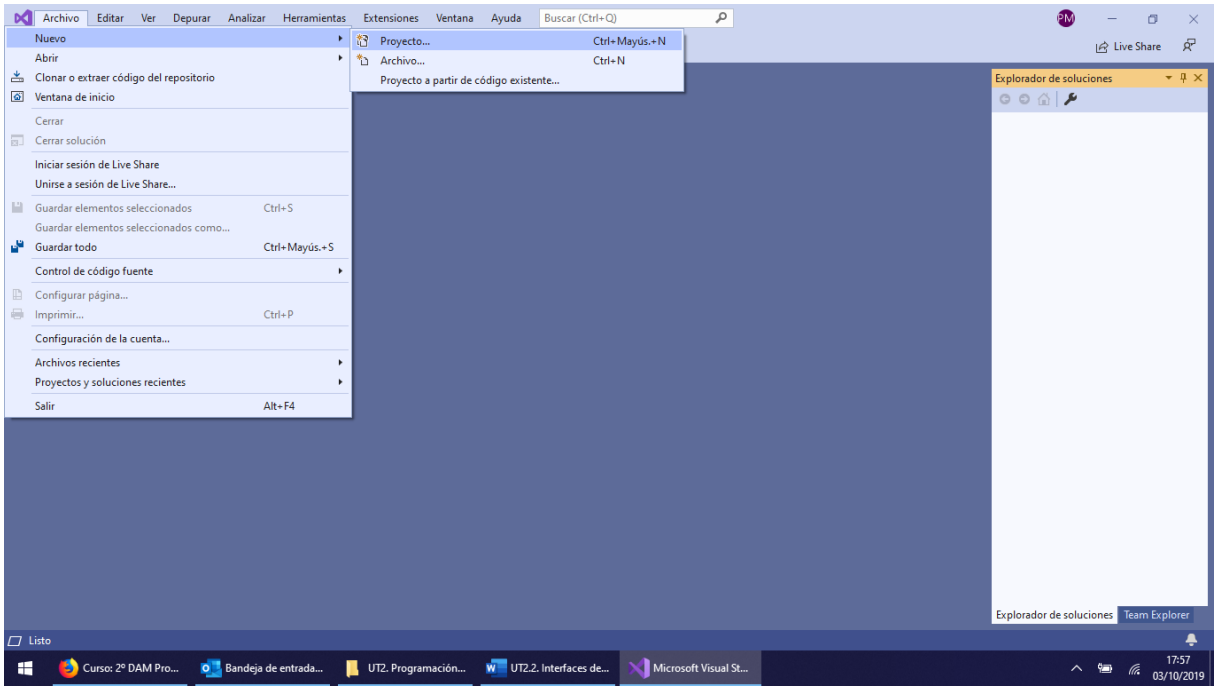
3. Estructura de una aplicación de Xamarin.Forms

En una aplicación de Xamarin.Forms, el lenguaje **XAML** se usa principalmente para definir el contenido visual de una página y funciona junto con un archivo de código subyacente (code-behind) en **C#** que se usa para proporcionarle la funcionalidad correspondiente.

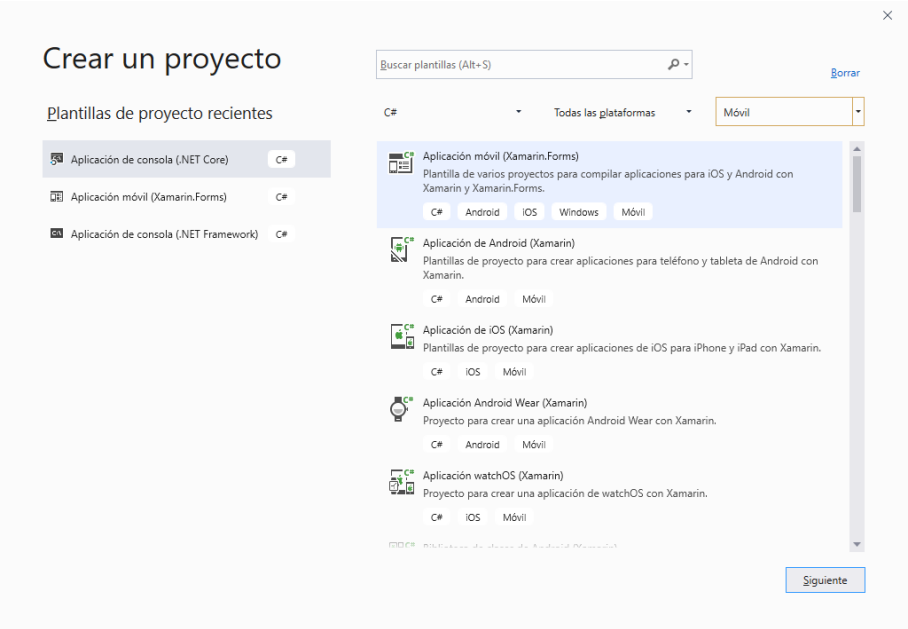
El archivo de código subyacente facilita la compatibilidad de código para el marcado. Juntos, estos dos archivos contribuyen a una nueva definición de clase que incluye vistas secundarias y la inicialización de las propiedades. En el archivo XAML, se hace referencia a las clases y propiedades con atributos y elementos XML y se establecen los vínculos entre el código y el marcado.

3.1. Crear una solución simple

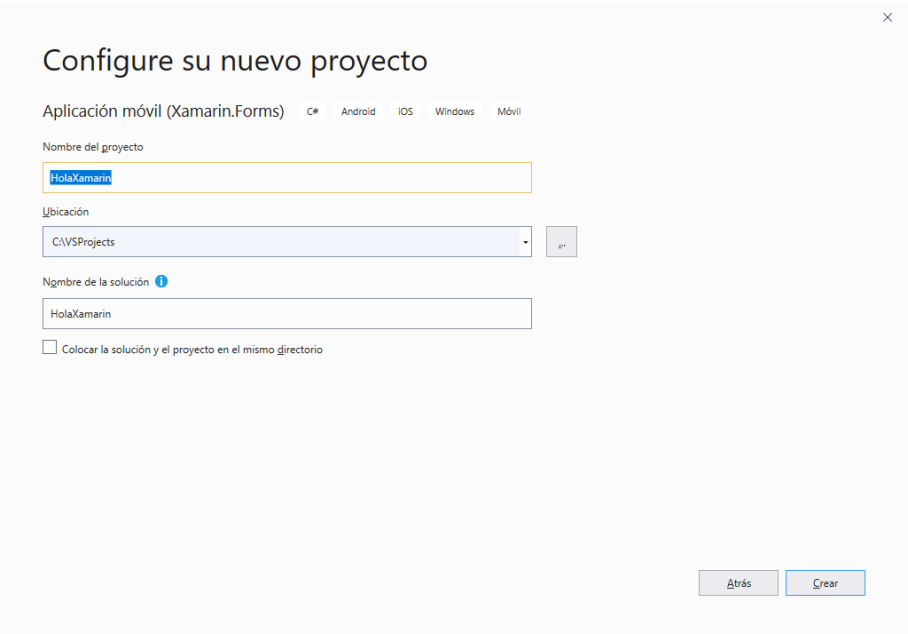
Para crear una nueva solución en **Visual Studio** debes seleccionar **Archivo > Nuevo > proyecto**.



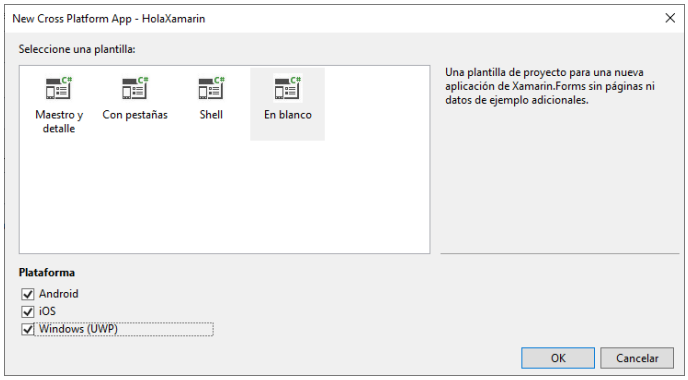
En el nuevo cuadro de diálogo, selecciona como tipo de proyecto **Móvil** situado a la derecha y, a continuación, **Aplicación Móvil (Xamarin.Forms)** de la lista central y haz clic en **Siguiente**.



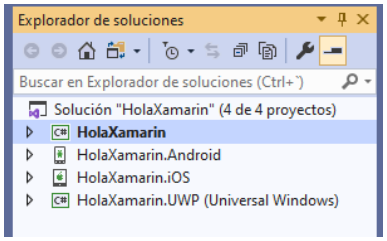
En la siguiente pantalla, asígnale al proyecto el nombre de **HolaXamarin**, opcionalmente puedes seleccionar una ubicación distinta de la propuesta para la solución y observa el nombre de la misma. Finalmente, haz clic en **Crear**.



A continuación, selecciona alguna de las plantillas propuestas, en nuestro caso **En blanco**. Selecciona las plataformas de destino, en nuestro caso **Android, iOS y Windows (UWP)**. Finalmente, haz clic en **OK**.

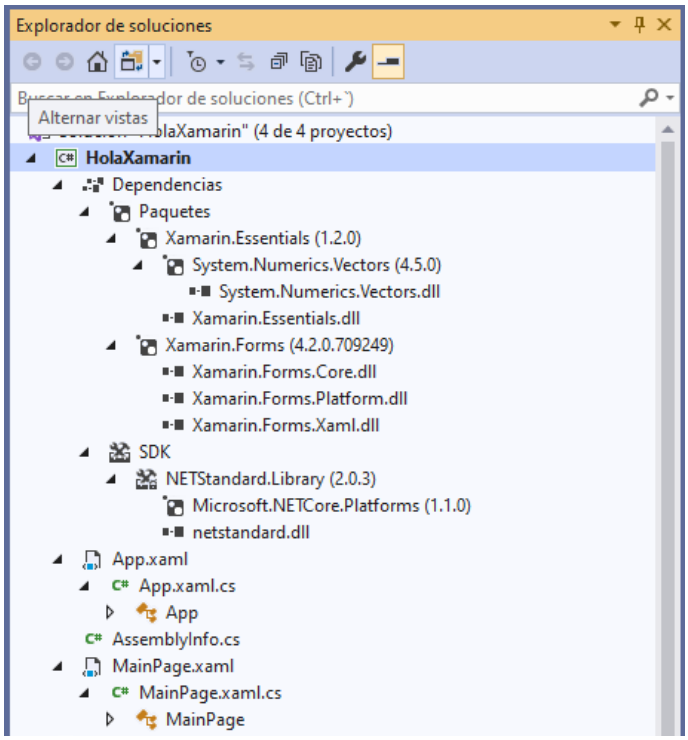


Se crean cuatro proyectos en la solución: el proyecto común en **HolaXamarin**, el proyecto Android en **HolaXamarin.Android**, el proyecto iOS en **HolaXamarin.iOS** y el proyecto UWP en **HolaXamarin.UWP**.



3.2. Estructura del proyecto común

Para maximizar la reutilización del código de inicio, las aplicaciones de Xamarin.Forms tienen una clase única denominada **App** que es responsable de crear instancias de la primera página o **Page** que se mostrará en pantalla.



3.2.1. Descripción de la aplicación

En el proyecto común, llamado **HolaXamarin**, existen dos archivos que implementan la aplicación móvil propiamente dicha, mediante una clase denominada **App** derivada de la clase **Application**. Estos archivos son: **App.xaml**, el archivo de código XAML y **App.xaml.cs**, el archivo de código subyacente C# asociado con el archivo XAML.

El archivo **App.xaml** contiene el siguiente código:

```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="HolaXamarin.App">
    <Application.Resources>

    </Application.Resources>
</Application>
```

La primera línea del archivo forma la declaración XML del documento. Consiste en la definición del número de versión (**version="1.0"**) y la declaración de la codificación del archivo (**encoding="utf-8"**).

A continuación, la etiqueta **Application** establece la página raíz de la aplicación móvil multiplataforma, es decir, el núcleo de una aplicación de Xamarin.Forms.

Dentro de la etiqueta **Application**, las declaraciones de los dos espacios de nombres **xmlns** hacen referencia a identificadores URI, la primera al sitio de web de Xamarin y el segundo de Microsoft. Son simplemente URIs que pertenecen a Microsoft y Xamarin y, básicamente funcionan como identificadores de versión.

La declaración del primer espacio de nombres **xmlns** indica que las etiquetas definidas en el archivo XAML que no tienen ningún prefijo hacen referencia a las clases de Xamarin.Forms.

La declaración del segundo espacio de nombres **xmlns:x**, define un prefijo **x**, que indica que algunos elementos y atributos que son básicos de XAML son compatibles con otras implementaciones de XAML.

La declaración del segundo y tercer espacio de nombres **xmlns:d** y **xmlns:mc**, se utilizan para que los controles sean más fáciles de visualizar en el visor de vista previa de XAML, dado que los datos en tiempo de diseño son datos falsos.

El atributo **mc:Ignorable="d"** especifica que prefijos XML del espacio de nombres que se encuentran en un archivo XAML de marcado pueden ser omitidos por un procesador.

El atributo **x:Class** especifica el nombre de una clase y el espacio de nombres donde está definida en XAML. El nombre de la clase debe coincidir con el nombre de clase del archivo de código subyacente C#. Hay que tener en cuenta que esta construcción sólo puede aparecer en el elemento raíz de un archivo XAML. En este caso la clase **App** perteneciente al espacio de nombres **HolaXamarin** que se deriva de la clase **Application**.

La propiedad **Resources** define el diccionario de recursos para el objeto de tipo **Application**. Los recursos XAML son definiciones de objetos que se pueden compartir y volver a utilizarse en una aplicación de Xamarin.Forms. Estos objetos de recursos se almacenan en un diccionario de recursos. Recursos típicos que se almacenan en un diccionario de recursos incluyen: estilos, plantillas de control, plantillas de datos, colores y convertidores de tipos.

El archivo **App.xaml.cs** contiene el siguiente código:

```
using System;
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace HolaXamarin {
    public partial class App : Application {
        public App() {
            InitializeComponent();
            MainPage = new MainPage();
        }

        protected override void OnStart() {
            // Handle when your app starts
        }

        protected override void OnSleep() {
            // Handle when your app sleeps
        }

        protected override void OnResume() {
            // Handle when your app resumes
        }
    }
}
```

La directiva **using** permite el uso de tipos en un espacio de nombres de dominio de manera que no se tenga que cualificar por completo el uso de dicho tipo. Por ejemplo, mediante **using Xamarin.Forms** se puede utilizar la clase **Application** sin tener que escribir **Xamarin.Forms. Application** cada vez que se use dentro del código.

XAML se puede compilar opcionalmente directamente en lenguaje intermedio (IL) con el compilador XAML (XAMLC). XAMLC está deshabilitado de forma predeterminada para garantizar la compatibilidad con versiones anteriores.

La sentencia **namespace HolaXamarin { }** se usa para declarar un espacio de nombres o ámbito que contiene un conjunto de objetos relacionados. Se suele usar para organizar los elementos de código y crear tipos únicos globales. Los espacios de nombres tienen implícitamente acceso público y esto no se puede modificar.

A continuación, se define la clase **App** como pública y parcial que hereda de la clase **Application**.

La palabra clave **public** es el modificador de acceso de la clase. El acceso público es el nivel de acceso más permisivo, es decir, no hay ninguna restricción para el acceso a la clase desde cualquier punto de la aplicación.

Las definiciones de tipo **partial** permiten dividir la definición de la clase en varios archivos. Esto sugiere que podría existir otra definición de clase parcial para **App**.

La clase **App** contiene un constructor que se ocupa de llamar al método **InitializeComponent** que se utiliza para cargar y analizar el archivo **App.xaml** asociado e instancia un objeto que pertenece a la clase **MainPage**.

A continuación, la propiedad **MainPage** de la clase **Application** establece la página raíz de la aplicación.

Finalmente, la clase **Application** contiene tres métodos virtuales que se pueden reescribir para controlar el ciclo de vida de la aplicación:

- **OnStart**, se llama cuando se inicia la aplicación.
- **OnSleep**, se llama cada vez que la aplicación entra en segundo plano.
- **OnResume**, se llama cuando se reanuda la aplicación, después de que se va a enviar al fondo.

Hay que tener en cuenta que no hay un método de finalización de la aplicación. En circunstancias normales se realizará la finalización de la aplicación desde el OnSleep, sin ninguna notificación adicional en el código.

Todos estos métodos tienen un nivel de accesibilidad **protected** (protegido), es decir, son accesibles dentro de la clase **App** y por parte de instancias de clase derivadas de ésta.

El modificador **override** es necesario para ampliar o modificar la implementación abstracta o virtual de los métodos heredados.

3.2.2. Descripción del contenido visual

Por otro lado, también en el proyecto común, tenemos otros dos archivos utilizados para definir el contenido visual de la página completa. Estos archivos son: **MainPage.xaml,** el archivo con código XAML y **MainPage.xaml.cs,** el archivo de código subyacente C#.

El archivo **MainPage.xaml** contiene el siguiente código:

```
<?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:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
            mc:Ignorable="d"
            x:Class="HolaXamarin.MainPage">

    <StackLayout>
        <!-- Place new controls here -->
        <Label Text="Welcome to Xamarin.Forms!"
           HorizontalOptions="Center"
           VerticalOptions="CenterAndExpand" />
    </StackLayout>

</ContentPage>
```

La primera línea del archivo forma la declaración XML del documento. Consiste en la definición del número de versión (**version="1.0"**) y la declaración de la codificación del archivo (**encoding="utf-8"**).

A continuación, la etiqueta **Application** establece la página raíz de la aplicación móvil multiplataforma, es decir, el núcleo de una aplicación de Xamarin.Forms.

Dentro de la etiqueta **Application**, las declaraciones de los dos espacios de nombres **xmlns** hacen referencia a identificadores URI, la primera al sitio de web de Xamarin y el segundo de Microsoft. Son simplemente URIs que pertenecen a Microsoft y Xamarin y, básicamente funcionan como identificadores de versión.

La declaración del primer espacio de nombres **xmlns** indica que las etiquetas definidas en el archivo XAML que no tienen ningún prefijo hacen referencia a las clases de Xamarin.Forms, por ejemplo ContentPage.

La declaración del segundo espacio de nombres **xmlns:x**, define un prefijo **x**, que indica que algunos elementos y atributos que son básicos de XAML son compatibles con otras implementaciones de XAML.

La declaración del segundo y tercer espacio de nombres **xmlns:d** y **xmlns:mc**, se utilizan para que los controles sean más fáciles de visualizar en el visor de vista previa de XAML, dado que los datos en tiempo de diseño son datos falsos.

El atributo **mc:Ignorable="d"** especifica que prefijos XML del espacio de nombres que se encuentran en un archivo XAML de marcado pueden ser omitidos por un procesador.

El atributo **x:Class** indica un nombre de clase totalmente cualificado, en este caso la clase **MainPage** perteneciente al espacio de nombres **HolaXamarin**. Esto significa en .NET. En este caso concreto se define una nueva clase denominada **MainPage** en el espacio de nombres **HolaXamarin** que se deriva de **ContentPage**, es decir, la etiqueta que aparece en el atributo **x:Class**. El atributo **x:Class** solo puede aparecer en el elemento raíz de un archivo XAML para definir una clase derivada de C#. Ésta es la única nueva clase definida en el archivo XAML. Todo lo demás que aparece en el archivo XAML son simples instanciaciones e inicializaciones de clases existentes.

Tras la definición de la estructura de la página, se define un tipo de diseño o layout llamado **StackLayout**. Este tipo de diseño organiza los elementos correspondientes a las vistas en una línea unidimensional o pila, ya sea horizontal o verticalmente.

Finalmente, dentro del diseño utilizado se define un elemento de tipo vista o view llamado **Label**, utilizado para mostrar el texto asignado a la propiedad **Text**, centrado horizontalmente mediante la propiedad **HorizontalOptions** y centrado expandido verticalmente mediante la propiedad **VerticalOptions**.

El archivo **MainPage.xaml.cs** contiene el siguiente código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace HolaXamarin {
    // Learn more about making custom code visible in the Xamarin.Forms previewer
    // by visiting https://aka.ms/xamarinforms-previewer
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage {
        public MainPage() {
            InitializeComponent();
        }
    }
}
```

La directiva **using** permite el uso de tipos en un espacio de nombres de dominio de manera que no se tenga que cualificar por completo el uso de dicho tipo. Por ejemplo, mediante **using Xamarin.Forms** se puede utilizar la clase **Application** sin tener que escribir **Xamarin.Forms. Application** cada vez que se use dentro del código.

La sentencia **namespace HolaXamarin { }** se usa para declarar un espacio de nombres o ámbito que contiene un conjunto de objetos relacionados. Se suele usar para organizar los elementos de código y crear tipos únicos globales. Los espacios de nombres tienen implícitamente acceso público y esto no se puede modificar.

El atributo **[DesignTimeVisible(false)]** Habilita la representación en tiempo de diseño para controles personalizados.

A continuación, se define la clase **MainPage** como pública y parcial que hereda de la clase **ContentPage**.

La palabra clave **public** es el modificador de acceso de la clase. El acceso público es el nivel de acceso más permisivo, es decir, no hay ninguna restricción para el acceso a la clase desde cualquier punto de la aplicación.

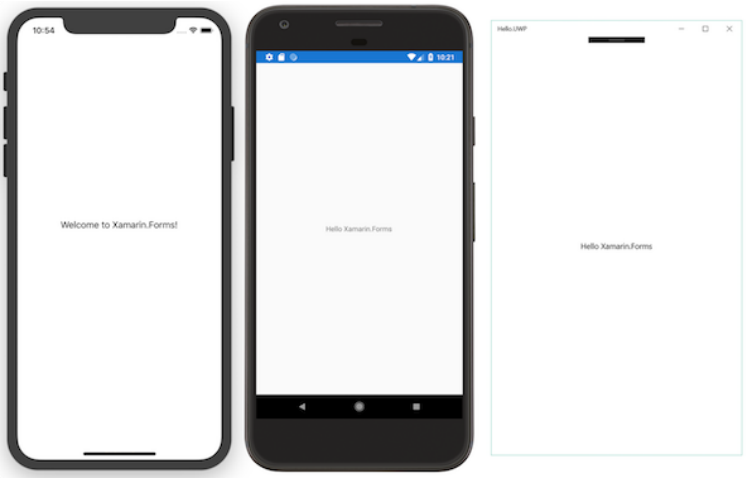
Las definiciones de tipo **partial** permiten dividir la definición de la clase, estructura o interfaz en varios archivos. Esto sugiere que podría existir otra definición de clase parcial para **MainPage**.

La clase **MainPage** contiene un constructor que se ocupa de llamar al método **InitializeComponent** que se utiliza para cargar y analizar el archivo **MainPage.xaml** asociado.

3.3. Compilar y ejecutar la solución

Si se ejecuta la aplicación, debería tener un aspecto similar al que se muestra en las capturas de pantallas mostradas.

Cada una de las pantallas de estas capturas corresponde a una página o **Page** de Xamarin.Forms. Un elemento Page representa una **Actividad** en Android, un **Controlador de vista** en iOS o una **Página** en la Plataforma universal de Windows (UWP). En el ejemplo de las capturas de pantalla anteriores se crea una instancia de un objeto **ContentPage** y se usa para mostrar un elemento **Label**.



Cuando Visual Studio compila la solución, analiza el archivo XAML para generar un archivo de código C#. Si observas el directorio **HolaXamarin\HolaXamarin\obj\Debug**, encontrarás un archivo llamado **XamlSamples.MainPage.xaml.g.cs**, la "g" son las siglas de generadas. Se trata de la otra definición de clase parcial de **MainPage** que contiene la definición del método **InitializeComponent** invocado desde el constructor de **MainPage**. Estas dos definiciones de clase parciales de **MainPage**, a continuación, se compilan juntas. Dependiendo de si se compila el código XAML o no, el archivo XAML o un formato binario del archivo XAML se incrusta en el ejecutable.

En tiempo de ejecución, el código del proyecto de una plataforma particular llama al método **LoadApplication**, pasándole una nueva instancia de la clase **App** perteneciente a la biblioteca .NET Standar. El constructor de la clase **App** instancia un objeto perteneciente a la clase **MainPage**.

El constructor de esa clase llama al método **InitializeComponent**, que llama al método **LoadFromXaml** que extrae el archivo XAML (o su binario compilado) de la biblioteca .NET Standard. El método **LoadFromXaml** inicializa todos los objetos definidos en el archivo XAML, los conecta a través de una relación padre – hijo, adjunta los controladores de eventos definidos en el código a los eventos que se definen en el archivo XAML y establece el árbol de objetos resultante como el contenido de la página.

Cuando se compila y ejecuta este programa, el elemento de tipo diseño **StackLayout** organiza las vistas linealmente en la pantalla del dispositivo, ya sea horizontal o verticalmente y, finalmente, el elemento de tipo vista **Label** aparece en el centro de la página como sugiere el código XAML.

4. Elementos visuales de Xamarin.Forms

En una aplicación de Xamarin.Forms, los objetos que ocupan espacio en la pantalla se conocen como elementos visuales, encapsulados por la clase **VisualElement**. Los elementos visuales pueden dividirse en tres categorías correspondientes a estas clases:

- **Pages (Páginas)**: las páginas representan pantallas de aplicaciones móviles multiplataforma.
- **Layouts (Diseños)**: los diseños son contenedores que se usan para crear vistas en estructuras lógicas.
- **Views (Vistas)**: las vistas son los controles que se muestran en la interfaz de usuario, como etiquetas, botones y cuadros de entrada de texto.
- **Cells (Celdas)**: las celdas son componentes especializados que se usan en los elementos de una lista y describen cómo debe dibujarse cada uno de ellos.

En tiempo de ejecución, cada control se asignará a su equivalente nativo, que es lo que se representará.

Una **Page** ocupa toda la pantalla, o casi toda la pantalla. A menudo, el elemento secundario de una página es un **Layout** que se utiliza para organizar los elementos visuales hijos. Los elementos visuales hijos de un Layout puede ser otro Layout o una **View**, que son objetos familiares como texto, los mapas de bits, los controles deslizantes, botones, cuadros de lista y así sucesivamente.

4.1. Páginas o Pages

Las páginas de Xamarin.Forms representan pantallas de aplicaciones móviles multiplataforma. Todos los tipos de página que se describen a continuación se derivan de la clase **Page**. Estos elementos visuales ocupan toda o la mayoría de la pantalla. Un objeto **Page** representa un **ViewController** en iOS y un **Page** en la plataforma Universal de Windows. En Android, cada página ocupa la pantalla como una **Activity**, pero las páginas de Xamarin.Forms son no objetos Activity.

Existen los siguientes tipos de páginas Xamarin.Forms:

- **ContentPage.** Es el tipo más sencillo y común de página. Se trata de una página que muestra una vista única, a menudo como un contenedor de un StackLayout o ScrollView.
- **MasterDetailPage.** Administra dos paneles de información: una página maestra que presenta los datos en un nivel alto y una página de detalles que muestra los detalles de bajo nivel sobre la información de la página maestra.
- **NavigationPage.** Administra la navegación entre otras páginas con una arquitectura basada en la pila.
- **TabbedPage.** Muestra un con conjunto de pestañas, cada una de las cuales carga contenido en la pantalla. En iOS, aparece la lista de pestañas en la parte inferior de la pantalla y el área de detalles está por encima. En los teléfonos Android y Windows, las pestañas aparecen en la parte superior de la pantalla.
- **TemplatedPage.** Muestra el contenido a pantalla completa con una plantilla de control y la clase base de ContentPage.
- **CarouselPage.** Los usuarios pueden deslizar de un lado a otro para mostrar las páginas de contenido, como una galería. Proporciona una experiencia de navegación sencilla y muy natural.

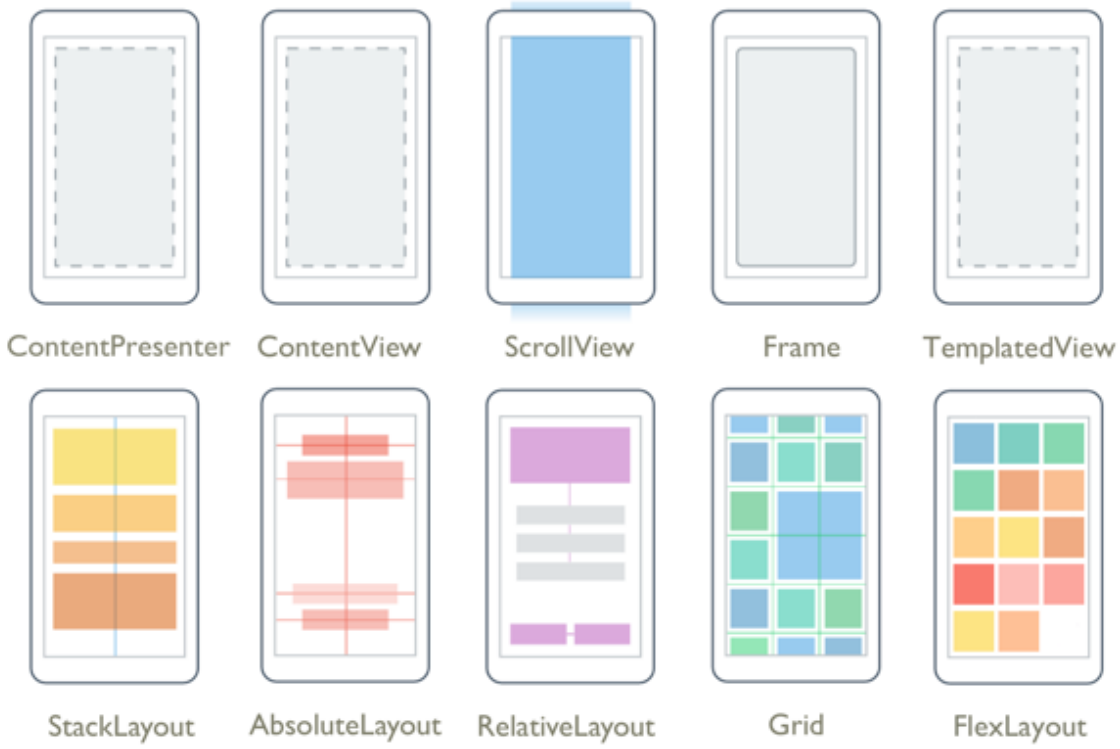
4.2. Diseños o Layouts

Los diseños de Xamarin.Forms se utilizan para crear controles de interfaz de usuario en estructuras visuales.

Las clases **Layout** y **Layout<T>** de Xamarin.Forms son subtipos especializadas de las vistas que actúan como contenedores para otros diseños y vistas. La clase propia **Layout** se deriva de **View**. Un Layout derivado normalmente contiene lógica para establecer la posición y tamaño de los elementos hijos en las aplicaciones de Xamarin.Forms.

Los diseños Xamarin.Forms pueden dividirse en varias categorías:

- Diseños con contenido único.
  - **ContentView**. Elemento que contiene un único elemento secundario.
  - **Frame**. Elemento que contiene un único elemento secundario, con algunas opciones de tramas.
  - **ScrollView**. Elemento capaz de desplazarse si el contenido lo requiere.
  - **TemplatedView**. Vista que muestra el contenido con una plantilla de control y la clase base de ContentView.
  - **ContentPresenter**. Administrador de diseño de las vistas con plantilla.
- Diseños con varios elementos secundarios.
  - **StackLayout**. Coloca los elementos secundarios en una sola línea que se puede orientar de forma vertical u horizontal.
  - **Grid**. Dispone las vistas en filas y columnas.
  - **AbsoluteLayout**. Coloca los elementos secundarios en posiciones absolutas.
  - **RelativeLayout**. coloca los elementos secundarios respecto a la RelativeLayout propia o a sus elementos relacionados.
  - **FlexLayout**. Diseño similar al de una caja flexible que dispone los elementos secundarios en filas o columnas de elementos secundarios.



4.3. Vistas o Views

Las vistas de Xamarin.Forms son los bloques de creación de interfaces de usuario en aplicaciones móviles multiplataforma.

Las vistas son objetos de interfaz de usuario, como etiquetas, botones y los controles deslizantes que se conocen normalmente como controles o widgets en otros entornos de programación de gráficos. Las vistas compatibles con Xamarin.Forms se derivan la clase **View**.

Las vistas Xamarin.Forms pueden dividirse en varias categorías:

- Vistas de presentación.
  - **Label**. Muestra las cadenas de texto de una línea o bloques de varias líneas de texto.
  - **Image**. Muestra un mapa de bits.
  - **BoxView**. Muestra un rectángulo relleno con color.
  - **WebView**. Muestra las páginas Web o contenido HTML.
  - **OpenGLView**. Muestra gráficos OpenGL en proyectos de iOS y Android.
  - **Map**. Muestra un mapa.
- Vistas que inician comandos.
  - **Button**. Muestra un objeto rectangular con un texto y que se desencadena un evento Clicked cuando se ha presionado.
  - **SearchBar**. Muestra un área para el usuario escriba una cadena de texto y un botón que le indica a la aplicación que realice una búsqueda.
- Vistas para establecer valores.
  - **Slider**. Permite al usuario seleccionar un valor desde un intervalo continuo.
  - **Stepper**. Permite al usuario seleccionar un valor de un intervalo incremental.
  - **Switch**. Adopta la forma de un modificador on/off para permitir al usuario seleccionar un valor booleano.
  - **DatePicker**. Permite al usuario seleccionar una fecha con el selector de fecha de la plataforma.
  - **TimePicker**. Permite al usuario seleccionar una hora con el selector de hora de la plataforma.
- Vistas para editar texto.
  - **Entry**. Permite al usuario escribir y editar una sola línea de texto.
  - **Editor**. Permite al usuario escribir y editar varias líneas de texto.
- Vistas para indicar actividad.
  - **ActivityIndicator**. Usa una animación para mostrar que la aplicación esté implicada en una actividad larga sin dar ninguna indicación del progreso.
  - **ProgressBar**. Usa una animación para mostrar que la aplicación está progresando a través de una actividad larga.

- Vistas que muestran colecciones.
    - **Picker**. Muestra el elemento seleccionado de una lista de cadenas de texto.
    - **ListView**. Muestra una lista desplazable de elementos de datos.
    - **TableView**. Muestra una lista de filas de tipo celda con encabezados opcionales y subencabezados.

4.4. Celdas o Cells

Las celdas de Xamarin.Forms pueden agregarse a ListView y TableViews. Una celda es un elemento especializado utilizado por los elementos en una tabla y describe cómo se debe representar cada elemento en una lista.

Una celda no es un elemento visual, en realidad es una plantilla para crear un elemento visual. Las celdas Xamarin.Forms se derivan la clase **Cell**.

Existen los siguientes tipos de celdas Xamarin.Forms:

- **TextCell**. Muestra una o dos cadenas de texto.
- **ImageCell**. Muestra la misma información que TextCell pero incluye un mapa de bits.
- **SwitchCell**. Muestra una etiqueta junto con un conmutador de encendido y apagado.
- **EntryCell**. Muestra una etiqueta y un campo de entrada de texto de línea única.

5. Colores

Xamarin.Forms proporciona una clase llamada **Color** flexible y multiplataforma. La clase proporciona una serie de métodos para crear una instancia de color:

- **Colores con nombre** - una colección de comunes colores con nombre, incluidos Red, Green, y Blue.
- **FromHex** - valor similar a la sintaxis utilizada en HTML, por ejemplo, "00FF00". El canal alfa o grado de opacidad se puede especificar opcionalmente como el primer par de caracteres ("CC00FF00").
- **FromHsla** - valores double que determinan el matiz, la saturación y la luminosidad, con el valor canal alfa opcional (0.0-1.0).
- **FromRgb** - valores int que determinan la cantidad de rojo, verde y azul (0-255).
- **FromRgba** - valores int que determinan la cantidad de rojo, verde, azul y canal alfa (0-255).
- **FromUint** - establecer un único valor double que representa el rojo, verde, azul y canal alfa.

Presentamos algunos colores de ejemplo, asignados a la BackgroundColor de algunas etiquetas utilizando diferentes variaciones de la sintaxis permitida:

```
var red = new Label { Text = "Red", BackgroundColor = Color.Red };
var orange = new Label { Text = "Orange", BackgroundColor = Color.FromHex("FF6A00") };
var yellow = new Label { Text = "Yellow", BackgroundColor = Color.FromHsla(0.167, 1.0, 0.5,
1.0) };
var green = new Label { Text = "Green", BackgroundColor = Color.FromRgb(38, 127, 0) };
```

```
var blue = new Label { Text = "Blue", BackgroundColor = Color.FromRgba(0, 38, 255, 255) };
var indigo = new Label { Text = "Indigo", BackgroundColor = Color.FromRgb(0, 72, 255) };
var violet = new Label { Text = "Violet", BackgroundColor = Color.FromHsla(0.82, 1, 0.25, 1)
};

var transparent = new Label { Text = "Transparent", BackgroundColor = Color.Transparent };
var @default = new Label { Text = "Default", BackgroundColor = Color.Default };
var accent = new Label { Text = "Accent", BackgroundColor = Color.Accent };
```

Estos colores se muestran en cada plataforma de la siguiente manera:

También se puedan encontrar fácilmente los colores en XAML mediante los nombres de colores definidos o las representaciones hexadecimales que se muestran aquí:

```
<Label Text = "Sea color" BackgroundColor="Aqua" />
<Label Text = "RGB" BackgroundColor="#00FF00" />
<Label Text = "Alpha plus RGB" BackgroundColor="#CC00FF00" />
<Label Text = "Tiny RGB" BackgroundColor="#0F0" />
<Label Text = "Tiny Alpha plus RGB" BackgroundColor="#C0F0" />
```