Manual de estudiante

JORNADA XAMARIN

Humberto Jaimes

humberto@humbertojaimes.net

CONTENIDO

1. ¿Qué es Xamarin.Forms?

Xamarin.Forms, es un enfoque de Xamarin en el que un Proyecto Xamarin tradicional tiene instalado el Framework llamado Xamarin.Forms.

Este Framework provee una serie de clases que permiten maximizar la cantidad de código compartido, la principal característica de Xamarin.Forms es que la interfaz de usuario se puede programar una sola vez y Xamarin se encarga de transformar ese código en una interfaz de usuario nativa.

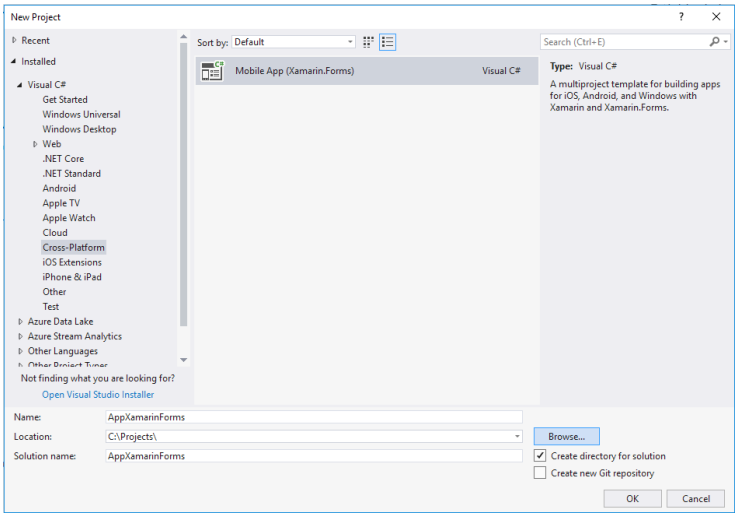
Utilizando este enfoque se puede compartir mas el 90% del código y como en el fondo es un proyecto de Xamarin tradicional seguimos teniendo acceso a todas las funcionalidades nativas.

1. Primera app con Xamarin.Forms

1.1 Creando el proyecto

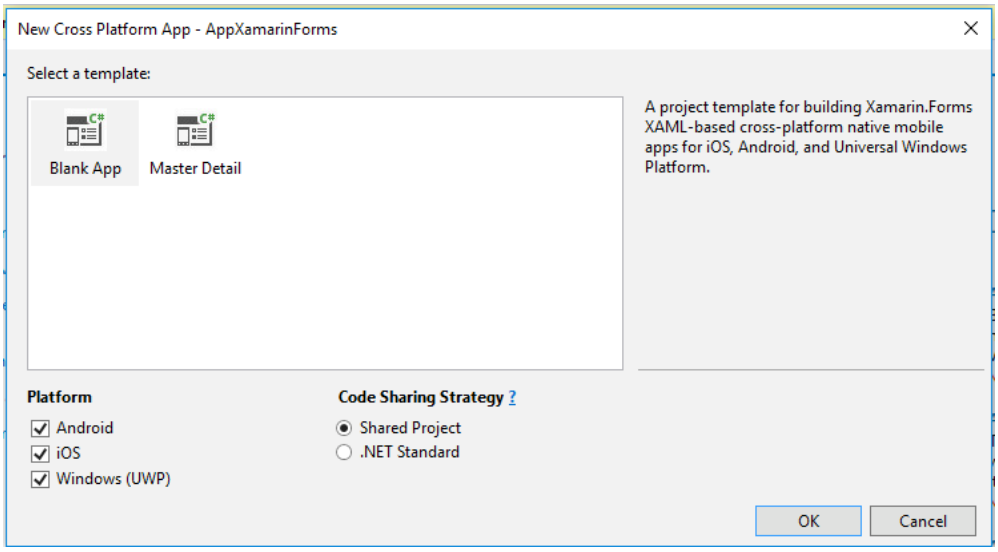
Para crear una app Xamarin.Forms desde Visual Studio entraremos a la opción nuevo proyecto, elegiremos la opción “Mobile App (Xamarin.Forms)” dentro de la sección “Cross-Platform”.

Como nombre la llamaremos “PrimeraAppForms”



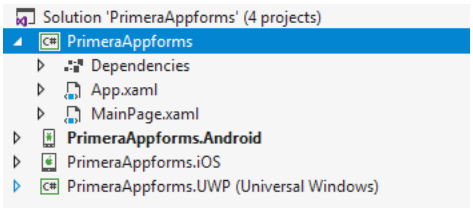
1.2 ELIGIENDO LA PLANTILLA

En la ventana que aparece debemos elegir “Blank App” en la parte superior y “.NET Standard” en la parte inferior



1.3 ESTRUCTURA DEL PROYECTO

La estructura de nuestro nuevo proyecto es la siguiente:



Un proyecto .NET Standard el cual contendrá toda la lógica compartida, en este caso la interfaz de usuario es código compartido a diferencia de cómo funciona en Xamarin.Tradicional.

En este proyecto se encuentra un archivo llamado App.xaml, el cual normalmente se usa para crear recursos que se usaran a lo largo de toda la app. Recursos como cadenas de texto comunes o estilos visuales para ciertos elementos como botones.

La otra parte de esta clase es el archivo "App.xaml.cs" el cual es el punto de entrada de la app donde definimos la página principal y en el futuro podemos inicializar algunos componentes.

Otro archivo dentro de la plantilla es el archivo MainPage.xaml, este archivo es la única página del proyecto que en este momento contiene los elementos de un "Hola Mundo".

Además de la biblioteca .NET Standard existe un proyecto por cada plataforma, y estos proyectos tienen una referencia al proyecto .NET Standard. Normalmente se hace uso de estos proyectos para inicializar componentes o crear controles o funcionalidades muy personalizadas.

2. XAML

2.1 ¿QUÉ ES XAML?

XAML (E**X**tensible **A**pplication **M**arkup **L**anguage) es una de las formas de crear interfaces de usuario en Xamarin.Forms (la otra es con C#). XAML es muy parecido a XML con algunas ligeras diferencias, por ejemplo que es sensible a mayúsculas y minúsculas.

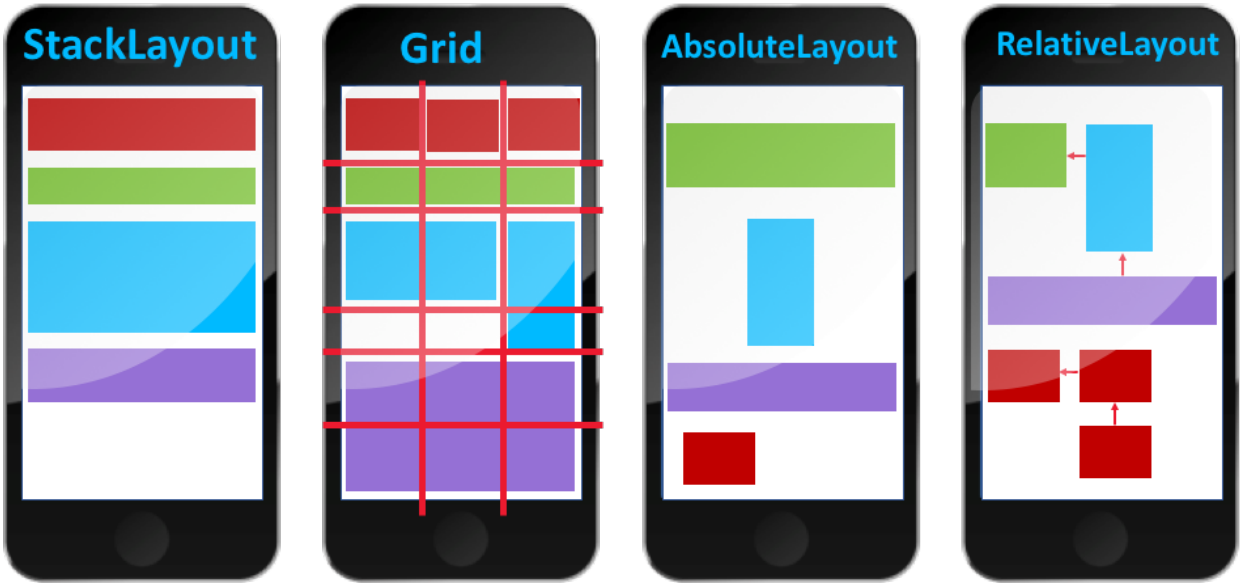
Actualmente no existe un diseñador Drag and Drop para crear las interfaces de usuario por lo cual es necesario crear nuestras interfaces escribiendo todo el código XAML sin ayuda de herramientas.

Así es como se define una pantalla con XAML

Cada elemento creado en XAML tiene como respaldo una clase que también puede ser creada con C#. El nombre de la etiqueta en XAML es el nombre de la clase y cada uno de sus atributos son las diferentes propiedades que se pueden asignar a un objeto de esa clase.

2.2 Contenedores en Xamarin.Forms

Dentro de Xamarin.Forms existen los siguientes contenedores



StackLayout: Nos permite apilar elementos de izquierda a derecha o de arriba hacia abajo.

Grid: Nos permite definir filas y columnas para acomodar nuestros elementos

AbsoluteLayout: Sirve para acomodar elementos utilizando coordenadas X y Y

RelativeLayout: Se usa para acomodar elementos tomando en cuenta la posición de otros elementos.

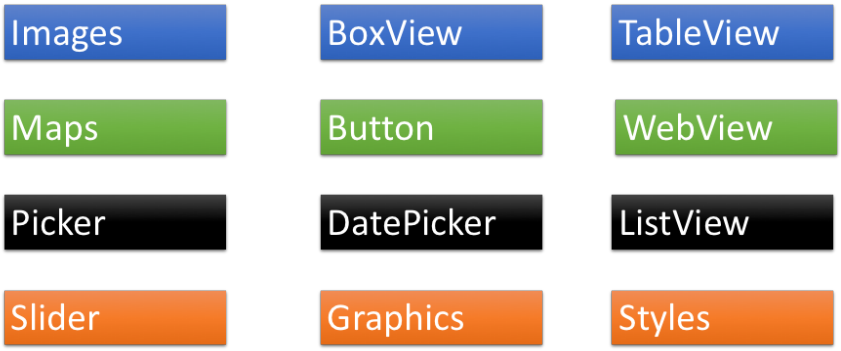
En este curso nos enfocaremos en StakLayout y Grid al ser los mas comunes.

2.3 Controles

Xamarin.Forms contiene los controles mas utilizados para crear apps. Dentro de los contenedores vamos a ir agregando las vistas que conforman la pantalla de nuestras apps.

Cuando ejecutemos el proyecto Xamarin de forma interna convierte estas vistas en los controles nativos, de forma que la app mantenga su esencia de haber sido creada para la plataforma en que esta ejecutándose.

Estos son algunos de los controles disponibles en Xamarin.Forms

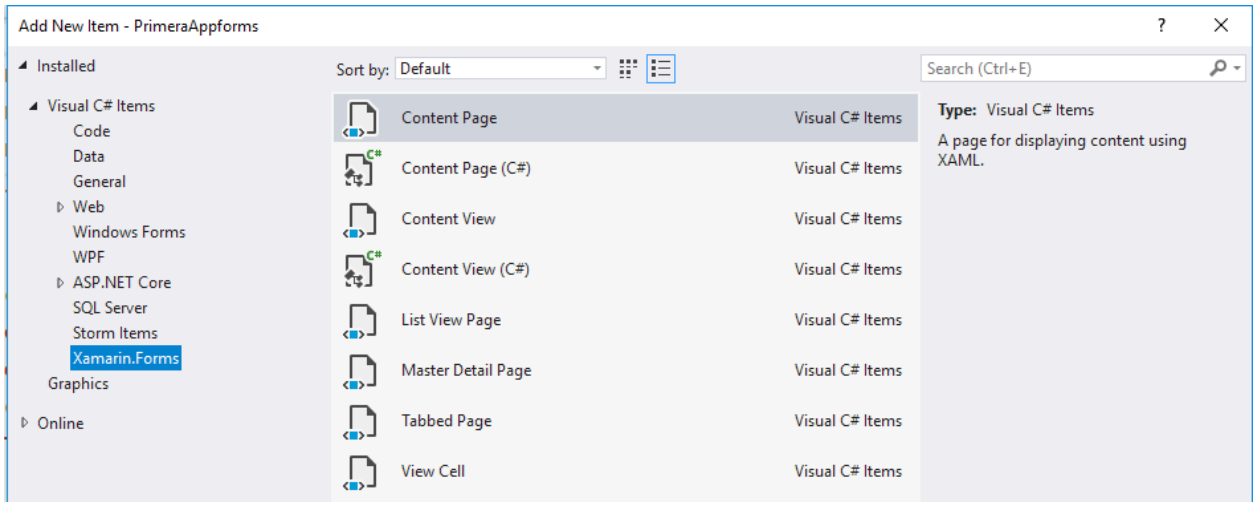


3. AGREGANDO CONTENIDO VISUAL A NUESTRA APP

5.1 CREANDO NUESTRAS PRIMERAS PÁGINAS

Ahora que ya conocemos la teoría comencemos a aplicarla en nuestro proyecto creado anteriormente.

Dentro del proyecto “PrimeraAppForms” crea una carpeta llamada “Paginas” y dentro de esta genera archivos de tipo “Content Page”, esta opción se encuentra dentro de la sección Xamarin.Forms



Los archivos a generar son:

- InicioDeSesionPagina
- RegistroPagina
- CamaraPagina
- UbicacionPagina
- MenuPrincipalPagina
- AcercaDePagina
- DirectorioPagina
- AltaDirectorioPagina

A cada página hay que asignarle un título, este se asignar en la propiedad “Title” de la ContentPage, este es un ejemplo de la página de Inicio:

```
<ContentPage Title="Inicio" xmlns="http://xamarin.com/schemas/2014/forms" xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="PrimeraAppForms.Paginas.InicioDeSesionPagina" />
```

De forma adicional dentro de la misma carpeta genera un archivo de tipo “Content Page (C#)” llamado “MaestroDetallePagina”

3.2 TRABAJANDO CON CONTENEDORES Y CONTROLES VISUALES

*Antes de comenzar una buena ayuda para diseñar pantallas con Xamarin.Forms es instalar el “Gorilla Player”. Esta es una herramienta que permite visualizar nuestro diseño XAML en el emulador o teléfono en tiempo real.

Para instalar la herramienta accede a https://grialkit.com/gorilla-player/ y sigue las instrucciones. (La herramienta es gratuita)

1. Vamos a comenzar con la pantalla de inicio de sesión, entre las etiquetas `<ContentPage.Content> </ContentPage.Content>` escribe el siguiente contenido.

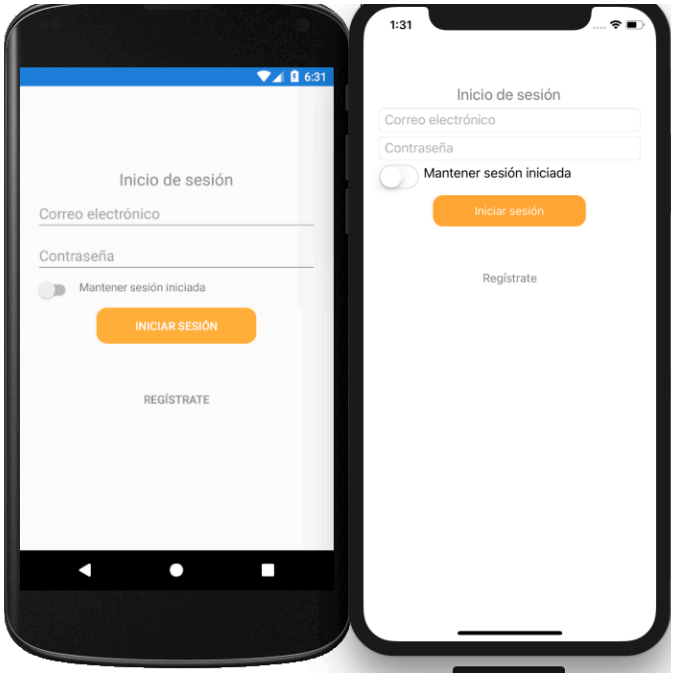
```
<StackLayout Padding="20,100,20,0">
    <Label Text="Inicio de sesión" TextColor="Gray" FontSize="20" HorizontalOptions="Center" />
    <Entry Placeholder="Correo electrónico" Keyboard="Email" />
    <Entry Placeholder="Contraseña" IsPassword="True" />
```

```
    <StackLayout Orientation="Horizontal">
      <Switch />
      <Label Text="Mantener sesión iniciada" />
    </StackLayout>
    <Button HorizontalOptions="Center" Text="Iniciar sesión" TextColor="White"

      BackgroundColor="#FFA733" BorderRadius="15" BorderColor="White" BorderWidth="2"

      WidthRequest="200" />
    <ActivityIndicator HeightRequest="30" HorizontalOptions="CenterAndExpand" />
    <Button Text="Regístrate" TextColor="Gray" BackgroundColor="Transparent" />
  </StackLayout>
```

Para probar el resultado modifica la siguiente linea en el archive App.xaml.cs asignando la página principal de la siguiente manera.

```
MainPage = new Paginas.InicioDeSesionPagina();
```

El resultado debe ser el siguiente en Android y iOS:



2. La siguiente página con la que trabajaremos es la de registro de usuarios (RegistroPagina.xaml), en la página anterior usamos un StackLayout para acomodar los elementos. Para la de registro usaremos un control llamada “TableView” el cual permite acomodar elementos por secciones y dar la apariencia de un formulario.

Otra diferencia es que no usaremos un botón, ahora usaremos un ToolbarItem el cual se posiciona en la barra de navegación la cual agregaremos mas adelante.

Para agregar este ToolbarItem, agrega las siguientes líneas después de la línea donde esta la etiqueta ContentPage

```
<ContentPage.ToolbarItems>
    <ToolbarItem Text="Crear" Priority="1" Order="Primary"  />
  </ContentPage.ToolbarItems>
```

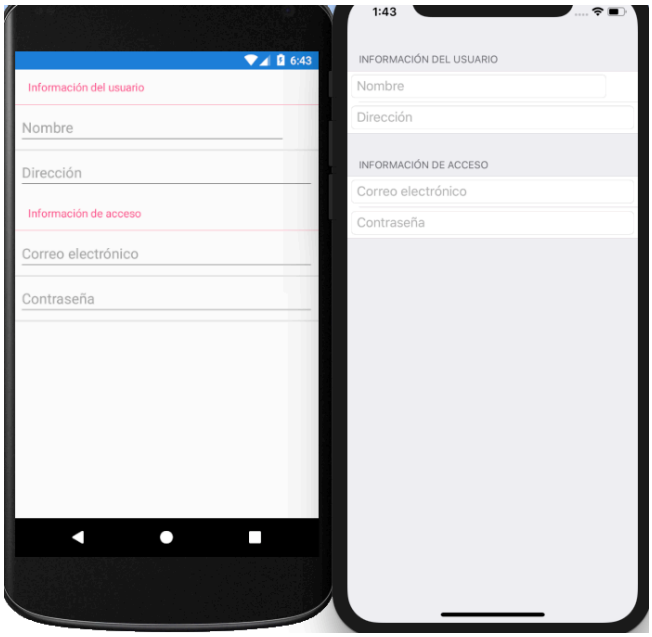
Para agregar el contenido dentro de la página agrega el siguiente código entre las etiquetas `<ContentPage.Content>` `</ContentPage.Content>` como hiciste en la página de inicio

```
<TableView Intent="Form" HasUnevenRows="true">
        <TableSection Title="Información del usuario">
          <ViewCell>
            <StackLayout Padding="5,5,5,5" Orientation="Horizontal">
              <Entry Placeholder="Nombre" HorizontalOptions="FillAndExpand" />
              <Image HeightRequest="30" WidthRequest="30" />
            </StackLayout>
          </ViewCell>
          <ViewCell>
            <StackLayout Padding="5,5,5,5">
              <Entry Placeholder="Dirección" HorizontalOptions="FillAndExpand" />
            </StackLayout>
          </ViewCell>
        </TableSection>
        <TableSection Title="Información de acceso">
          <ViewCell>
            <StackLayout Padding="5,5,5,5">
              <Entry Placeholder="Correo electrónico" HorizontalOptions="FillAndExpand" />
            </StackLayout>
          </ViewCell>
          <ViewCell>
            <StackLayout Padding="5,5,5,5">
              <Entry Placeholder="Contraseña" IsPassword="true" HorizontalOptions="FillAndExpand" />
            </StackLayout>
          </ViewCell>
```

```
    </TableSection>
  </TableView>
```

En el caso del TableView es importante que cada elemento o contenedor que usemos se encuentre de una etiqueta de tipo <ViewCell> de lo contrario obtendremos un error al ejecutar nuestra app.

Al igual que en el caso anterior podemos modificar el archivo “App.xaml.cs” para ver el resultado, el cual debe ser el siguiente.



Este es el contenido de las siguientes páginas que usaremos a lo largo del curso:

3. UbicacionPagina.xaml

```
<StackLayout Margin="20">
    <Label Text="¿Donde estoy?"                                                    />
    <BoxView HeightRequest=".5" Margin="0,0,0,5" />
    <Grid>
      <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
```

```
    </Grid.ColumnDefinitions>
    <Label Text="Latitud" Grid.Column="0" Grid.Row="0"                                    />
    <Label                                                  Grid.Column="1" Grid.Row="0" />
    <Label Text="Longitud" Grid.Column="0" Grid.Row="1"                                   />
    <Label                                                  Grid.Column="1" Grid.Row="1" />
    <Label Text="Dirección" Grid.Column="0" Grid.Row="2"                                  />
    <Label                                                  Grid.Column="1" Grid.Row="2" />
  </Grid>
</StackLayout>
```

4. AcercaDePagina.xaml

```
<ScrollView>
  <StackLayout Margin="20">
    <Label Text="Mi Primera app"                                                          />
    <BoxView HeightRequest=".5" Margin="0,0,0,5" />
    <Grid Margin="0,0,0,20">
      <Grid.RowDefinitions>
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
        <RowDefinition Height="Auto" />
      </Grid.RowDefinitions>
      <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
      </Grid.ColumnDefinitions>
      <Label Text="Versión App" Grid.Row="0" Grid.Column="0"                               />
      <Label                                  Grid.Row="0" Grid.Column="1"                />
      <Label Text="Nombre del Dispositivos" Grid.Row="1" Grid.Column="0"                   />
      <Label                                  Grid.Row="1" Grid.Column="1"                />
      <Label Text="Fabricante de dispositivo" Grid.Row="2" Grid.Column="0"                 />
      <Label                                  Grid.Row="2" Grid.Column="1"                />
      <Label Text="Modelo de dispositivo" Grid.Row="3" Grid.Column="0"  />
      <Label                                  Grid.Row="3" Grid.Column="1"                />
      <Label Text="Verion S.O." Grid.Row="4" Grid.Column="0"                               />
      <Label                                                Grid.Row="4" Grid.Column="1" />
```

```
        </Grid>
      </StackLayout>
    </ScrollView>
```

5. CamaraPagina.xaml

```
<StackLayout Margin="20">
        <Image                                                    HeightRequest="300" WidthRequest="300" />
        <Button Text="Tomar Foto" />
</StackLayout>
```

6. DirectorioPagina.xaml

```
 <StackLayout>
        <ListView                                                                                    >
          <ListView.ItemTemplate>
            <DataTemplate>
              <TextCell                                                                              />
            </DataTemplate>
          </ListView.ItemTemplate>
        </ListView>
    </StackLayout>
```

7. MenuPrincipalPagina.xaml

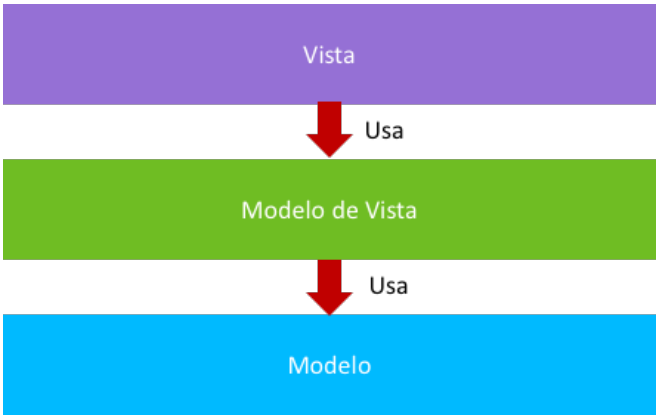
```
<StackLayout>
        <Button Text="Ubicación"
        <Button Text="Cámara"
        <Button Text="Directorio"                                                                    />
        <Button Text="Acerca de" />
        <Button Text="Salir"                                                                         />
 </StackLayout>
```

8. El archivo “MaestroDetallePagina” lo ocuparemos después.

4. MVVM y DataBinding

La forma mas común de organizar el código de una app Xamarin.Forms es usando el patrón MVVM, el cual permite desacoplar totalmente la vista de la funcionalidad.

El patrón esta conformado por las siguientes capas



Cada una de las capas tiene un objetivo.

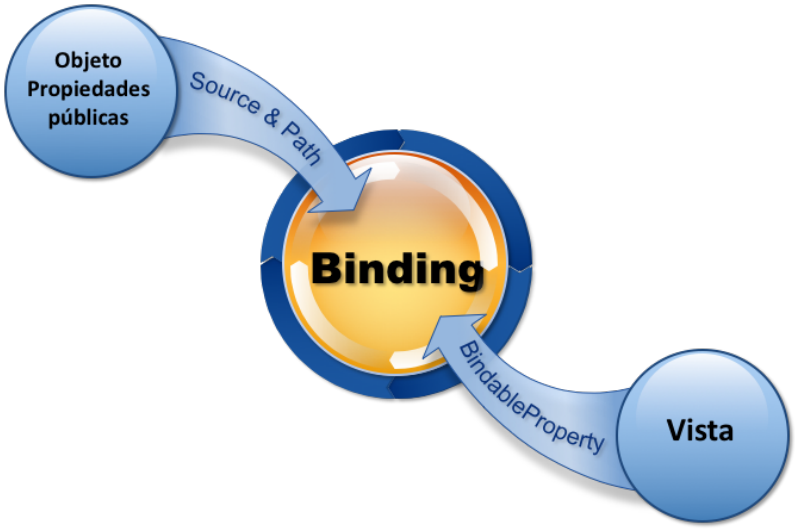
- El modelo contiene las clases que conforman el núcleo de nuestra app, normalmente aquí se encuentran clases POCO (Plain Old C# Object).
- La vista es la parte visual de nuestra App, esta capa esta conformada por nuestras páginas XAML.
- El modelo de vista es la capa que une el modelo y la vista, generalmente aquí se encuentran métodos para ejecutar acciones disparadas por los elementos gráficos y se procesan datos para hacerlos aptos para ser mostrados en la parte visual, por ejemplo, aquí se define el formato en que se mostrara una fecha.

Parte fundamental de MVVM es el concepto de DataBinding, el cual es el mecanismo que permite enlazar las propiedades de los elementos visuales con las propiedades de nuestros modelos de vista.

Un ejemplo simple sería usando nuestra página de inicio de sesión, dentro de la página tenemos dos campos de entrada de texto uno para el usuario y otro para la contraseña, por otro lado tendremos un Modelo de Vista con una propiedad Usuario y otra Contraseña. El DataBinding permite que se conecte la propiedad Text de las cajas con las propiedades del Modelo de Vista, todo esto sin asignarle un nombre a la caja de texto para acceder a sus valores.

La ventaja de usar un Binding es que si en algún momento decidimos cambiar el campo de texto por otro control, como un listado es mas fácil el cambio al no haber dependencia a un control en especifico.

De forma gráfica así podemos entender un DataBinding

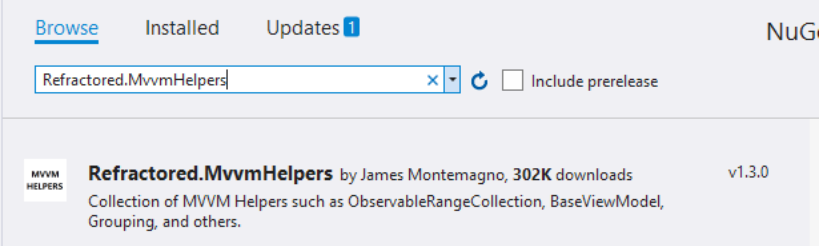


Ya revisados los conceptos comencemos a implementar MVVM siguiendo los pasos que se enumeran a continuación.

1. Crea una carpeta llamada “ModelosDeVista” y para cada una de las pantallas creadas anteriormente crea una clase equivalente modificando la palabra “Pagina” por “ModeloDeVista”. Las clases requeridas serían las siguientes:

- AcercaDeModeloDeVista.cs
- DirectorioModeloDeVista.cs
- InicioDeSesionModeloDeVista.cs
- MenuPrincipalModeloDeVista.cs
- UbicacionModeloDeVista.cs
-

2. Ahora hay que instalar el paquete NuGet llamado “Refractored.MvvmHelpers”



Este componente nos permite implementar MVVM de forma mas rápida al contener las clases mínimas requeridas para trabajar con el patrón.

3. Después de instalar el paquete cada una de nuestras clases “ModeloDeVista” debemos modificarlas para que hereden de la clase “BaseViewModel”, por ejemplo, la clase “AcercaDeModeloDeVista” quedaría así.

```
public class AcercaDeModeloDeVista: BaseViewModel
```

4. Ahora es momento de enlazar los Modelo de Vista con sus respectivas páginas, este proceso requiere de mucho cuidado para evitar errores.

   Lo primero es agregar la siguiente definición en la misma línea de la etiqueta <ContentPage>

```
xmlns:mv="clr-namespace:PrimeraAppForms.ModelosDeVista"
```

   Por ejemplo, al modificar la página “AcercaDePagina.xaml”, el resultado es el siguiente

```
<ContentPage Title="Acerca De" xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml" x:Class="PrimeraAppForms.Paginas.AcercaDePagina"
        xmlns:mv="clr-namespace:PrimeraAppForms.ModelosDeVista">
```

Seguido a esto hay que agregar las siguientes líneas en la línea posterior a la que modificamos anteriormente.

```
<ContentPage.BindingContext>
    <mv:AcercaDeModeloDeVista />
  </ContentPage.BindingContext>
```

La línea resaltada en negritas debe contener el nombre del Modelo de Vista correspondiente a la vista con la que trabajaremos.

En las siguientes secciones comenzaremos a utilizar estos Modelos de Vista para agregar funcionalidad a nuestra app.

5. NAVEGACIÓN

En este momento tenemos ya todas las páginas que usaremos junto con su estructura visual mas básica. Ahora trabajaremos en estructurar la navegación de nuestra app.

Existen varios tipos de navegación en las apps, las mas comunes son:

- Menú lateral: Usando esta navegación podemos sacar un menú del lado izquierdo de la app, este menú contiene las diferentes páginas a las que queremos acceder y se irá reemplazando la página visible conforma elijamos una opción del menú.
- Navegación lineal: Es una forma de navegar donde vamos apilando pantallas para poder avanzar de atrás hacia adelante y de adelante hacia atrás.
- Pestañas: Se agregan algunos botones en la parte superior o inferior de la pantalla que permiten acceder de formas rápida a las pantallas de la app.
- Reemplazar la página inicial, esta opción es la ideal cuando queremos que se elimine una página dentro de la navegación de forma que el usuario no pueda volver acceder a ella.

6.1 AGREGANDO PESTAÑAS A LA APP

Para nuestro ejemplo vamos a modificar la pantalla principal de modo que la página de inicio y de registro sean de fácil acceso a través de pestañas, para lograr esto realizaremos los siguientes cambios.

1. Hay que modificar el archivo “App.xaml.cs” para cambiar la página principal, cambiemos esta línea

```
MainPage = new Paginas.InicioDeSesion();
```

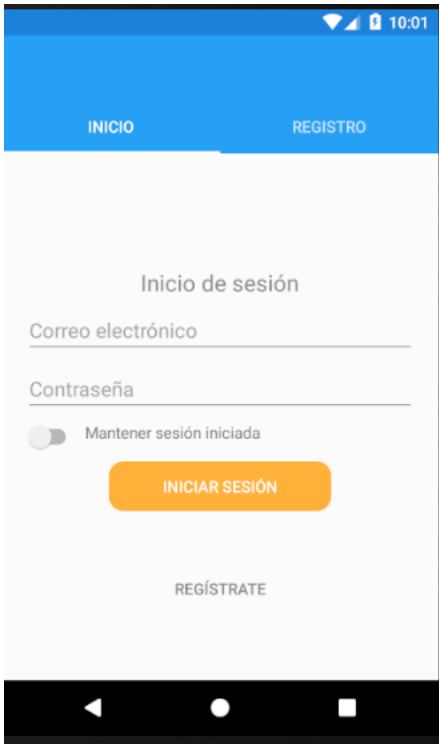
Por estas otras:

```
    TabbedPage pestañas = new TabbedPage();
pestañas.Children.Add(new Paginas.InicioDeSesionPagina());
pestañas.Children.Add(new Paginas.RegistroPagina());

MainPage = new NavigationPage(pestañas);
```

Estás líneas permiten crear la navegación mediante pestañas donde podemos ir agregando las páginas que se van a mostrar, las páginas se muestran en pestañas agregadas de izquierda a derecha en el mismo orden en que se agregan a la propiedad “Children”.

*Es importante que las páginas tengan asignada la propiedad “Title” de lo contrario las pestañas aparecerán sin ningún contenido. Adicionalmente si asignamos la propiedad “Icon” las pestañas contendrán un icono.

Al ejecutar en Android el resultado es el siguiente y al presionar las pestañas se debe mostrar la página adecuada.



6.2 Navegación con menú lateral

1. Ahora vamos a agregar navegación cuando presionemos el botón de iniciar sesión, para no ocupar tiempo innecesario no agregaremos validaciones a los campos y permitiremos navegar solo con presionar el botón.

   En el archivo “InicioDeSesionModeloDeVista.cs” debemos agregar las siguientes líneas

```
public ICommand IniciarComando
    {
      get
      {
        return new Command( () =>
        {
          Application.Current.MainPage = new MaestroDetallePagina();
        });
```

```
        }
    }
```

Un “Command” es un tipo de objeto que permite ejecutar métodos, generalmente es utilizado por botones para ejecutar una acción al ser presionados, aunque pueden ser usados por otros elementos como ListViews para ejecutar acciones al ser seleccionado un elemento de la lista.

Para poder hacer uso de este comando debemos hacer el DataBinding al botón del archivo “InicioDeSesionPagina”. Al botón debemos agregar la siguiente propiedad.

```
<Button HorizontalOptions="Center" Text="Iniciar sesión" TextColor="White" BackgroundColor="#FFA733" CornerRadius="15" Bord
erColor="White" BorderWidth="2" WidthRequest="200"

Command="{Binding IniciarComando}"/>
```

2. Antes de probar la app debemos agregar el contenido del archivo “MaestroDetallePagina.cs”, el cual es este:

```
public class MaestroDetalle : MasterDetailPage
  {
    public MaestroDetalle()
    {
      Master = new NavigationPage(new MenuPrincipalPagina()) { Title = "Menú"};
      Detail = new NavigationPage(new UbicacionPagina()) { Title = "Ubicación"};
    }
  }
```

El objetivo de estas líneas es asignar la página con el menú de opciones que se mostrará dentro del menú lateral y definir una página por defecto para la página de contenido que se mostrará en primer plano.

Es fundamental definir la propiedad “Title” para no obtener un error al ejecutar.

El uso de un “NavigationPage” permite que se muestre el titulo sobre cada página que se muestre en primer plano.

El resultado al presionar el botón en la página de inicio de sesión debe ser el siguiente

3. Es tiempo de darle funcionalidad al menú, el proceso es parecido al anterior. Hay que generar comandos para los botones y en cada comando agregar la navegación.

   El archivo “MenuPrincipalModeloDeVista” debe quedar de la siguiente manera:

```
public class MenuPrincipalModeloDeVista : BaseViewModel
  {

    public ICommand UbicacionComando
    {
      get
      {
        return new Command( () =>
        {
          (Application.Current.MainPage as MasterDetailPage).Detail = new NavigationPage(new
UbicacionPagina());
          (Application.Current.MainPage as MasterDetailPage).IsPresented = false;
        });
      }
    }




    public ICommand AcercaDeComando
    {
      get
      {
        return new Command( () =>
        {

          (Application.Current.MainPage as MasterDetailPage).Detail = new NavigationPage(new
AcercaDePagina());
          (Application.Current.MainPage as MasterDetailPage).IsPresented = false;
        });
      }
    }

    public ICommand SalirComando
    {
      get
      {
        return new Command(() =>
        {
          TabbedPage pestañas = new TabbedPage();
          pestañas.Children.Add(new Paginas.InicioDeSesionPagina());
          pestañas.Children.Add(new Paginas.RegistroPagina());
```

```
            Application.Current.MainPage = new NavigationPage(pestañas);
        });
    }
  }
}
```

En este caso estamos agregando navegación a través de reemplazar la página de detalle del “MasterDetailPage” y luego ocultamos el menú lateral.

Para ejecutar estos comando hay que hacer el Binding con los botones de la página “MenuPrincipalPagina”

```
                                                                    <StackLayout>
    <Button Text="Ubicación" Command="{Binding UbicacionComando}"/>
    <Button Text="Cámara" Command="{Binding CamaraComando}"/>
    <Button Text="Directorio" Command="{Binding DirectorioComando}"/>
    <Button Text="Acerca de" Command="{Binding AcercaDeComando}"/>
    <Button Text="Salir" Command="{Binding SalirComando}"/>
  </StackLayout>
```

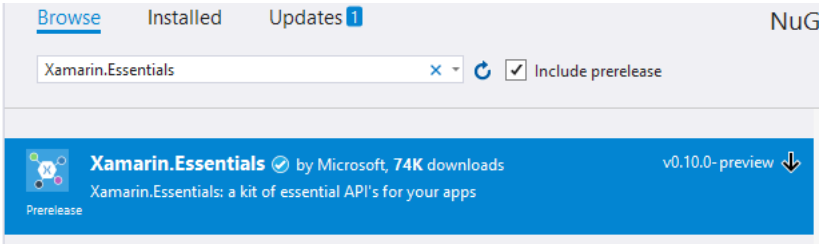
Al volver a ejecutar la app ya debemos poder navegar a las páginas que aún no tienen funcionalidad.

7 UBICACIÓN E INFORMACIÓN DE LA APP

Existe un componente llamado Xamarin.Essentials, el cual contiene las funcionalidades mas usadas al crear apps móviles. Algunas de las funcionalidades son: obtener la ubicación del dispositivo, obtener datos como el estado de la batería, la versión del sistema, abrir la aplicación de llamadas, abrir la app de mapas pasándole una ubicación especifica, etc.

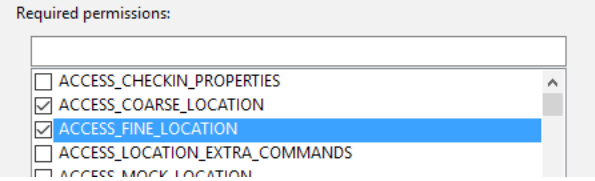
En este curso haremos uso de la funcionalidad de ubicación y la de obtener datos de la app y dispositivo.

1. El primer paso es instalar el paquete NuGet “Xamarin.Essentials”, para lograr hay que activar la opción de “Pre Releases” para poder ver este paquete que aun se encuentra en una versión de pruebas.



7.1 UBICACIÓN

1. Antes de crear la funcionalidad debemos agregar los permisos necesarios al manifiesto de Android, los permisos necesarios para este ejemplo son los siguientes.



2. Ahora hay que modificar el Modelo de Vista “UbicacionModeloDeVista” para que sea capaz de obtener la ubicación, el contenido de la clase será este:

```
public class UbicacionModeloDeVista : BaseViewModel
  {
    public UbicacionModeloDeVista()
    {
      Ubicar();
    }

    private double latitud;

    public double Latitud
    {
      get => latitud;
      set => SetProperty(ref latitud, value);
    }

    private double longitud;
```

```
    public double Longitud
    {
      get => longitud;
      set => SetProperty(ref longitud, value);
    }

    private string direccion;

    public string Direccion
    {
      get => direccion;
      set => SetProperty(ref direccion, value);
    }

    private async Task Ubicar()
    {
      try
      {
        GeolocationRequest peticionUbicacion = new GeolocationRequest(GeolocationAccuracy.Best,
TimeSpan.FromSeconds(5));
        var ubicacion = await Geolocation.GetLocationAsync(peticionUbicacion);
        Latitud = ubicacion.Latitude;
        Longitud = ubicacion.Longitude;
        var direccionActual = (await Geocoding.GetPlacemarksAsync(ubicacion)).First();


        Direccion = $" Calle {direccionActual.Thoroughfare} No. {direccionActual.SubThoroughfare},      Colonia {direccionActual.Lo
cality}, C.P. {direccionActual.PostalCode}. {direccionActual.CountryName}";

      }
      catch (FeatureNotSupportedException fnsEx)
      {
        // Handle not supported on device exception
      }
      catch (PermissionException pEx)
      {
        // Handle permission exception
      }
      catch (Exception ex)
      {
        // Unable to get location
      }
```

```
    }
}
```

3. Para poder mostrar los resultados debemos modificar la vista “UbicacionPagina.xaml”, en esta página agregaremos los binding necesarios para mostrar el resultado de la lógica que obtiene la ubicación.

```
<StackLayout Margin="20">
        <Label Text="¿Donde estoy?" />
        <BoxView HeightRequest=".5" Margin="0,0,0,5" />
        <Grid>
          <Grid.RowDefinitions>
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="Auto" />
          </Grid.RowDefinitions>
          <Grid.ColumnDefinitions>
            <ColumnDefinition />
            <ColumnDefinition />
          </Grid.ColumnDefinitions>
          <Label Text="Latitud" Grid.Column="0" Grid.Row="0" />
          <Label Text="{Binding Latitud}" Grid.Column="1" Grid.Row="0" />
          <Label Text="Longitud" Grid.Column="0" Grid.Row="1" />
          <Label Text="{Binding Longitud}" Grid.Column="1" Grid.Row="1" />
          <Label Text="Dirección" Grid.Column="0" Grid.Row="2" />
          <Label Text="{Binding Direccion}" Grid.Column="1" Grid.Row="2" />
        </Grid>
    </StackLayout>
```

Al ejecutar la app, cuando se muestra la pantalla de ubicación después de unos segundos deben mostrarse los datos de nuestra ubicación actual.

7.2 DATOS DEL APP Y DISPOSITIVO

1. Para mostrar los datos en el acerca de el proceso es muy similar, primero modificar el Modelo de Vista “AcercaDeModeloDeVista”

```
public class AcercaDeModeloDeVista: BaseViewModel
  {
    public AcercaDeModeloDeVista()
    {
      Inicializar();
    }


    void Inicializar()
    {
      try
      {
        Version = AppInfo.VersionString;
        NombreDispositivo = DeviceInfo.Name;
        VersionSO = DeviceInfo.Version.ToString();
        FabricanteDispositivo = DeviceInfo.Manufacturer;
        ModeloDispositivo = DeviceInfo.Model;
      }
      catch(Exception ex)
      {

      }
    }

    private string nombreDispositivo;

    public string NombreDispositivo
    {
      get => nombreDispositivo;
      set => SetProperty(ref nombreDispositivo, value);
    }

    private string version;

    public string Version
    {
      get => version;
      set => SetProperty(ref version, value);
    }

    private string fabricanteDispositivo;

    public string FabricanteDispositivo
```

```
    {
        get => fabricanteDispositivo;
        set => SetProperty(ref fabricanteDispositivo, value);
    }


    private string modeloDispositivo;

    public string ModeloDispositivo
    {
        get => modeloDispositivo;
        set => SetProperty(ref modeloDispositivo, value);
    }

    private string versionSO;

    public string VersionSO
    {
        get => versionSO;
        set => SetProperty(ref versionSO, value);
    }

}
```

8 Posteriormente se van a crear los binding dentro de la vista “AcercaDePagina”

```
<ScrollView>
        <StackLayout Margin="20">
            <Label Text="{Binding Version, StringFormat='Mi Primera app {0}'}" />
            <BoxView HeightRequest=".5" Margin="0,0,0,5" />
            <Grid Margin="0,0,0,20">
                <Grid.RowDefinitions>
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                    <RowDefinition Height="Auto" />
                </Grid.RowDefinitions>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition />
```

```
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <Label Text="Versión App" Grid.Row="0" Grid.Column="0"                                    />
        <Label Text="{Binding Version}" Grid.Row="0" Grid.Column="1"                              />
        <Label Text="Nombre del Dispositivos" Grid.Row="1" Grid.Column="0"  />
        <Label Text="{Binding NombreDispositivo}" Grid.Row="1" Grid.Column="1"                    />
        <Label Text="Fabricante de dispositivo" Grid.Row="2" Grid.Column="0"  />
        <Label Text="{Binding FabricanteDispositivo}" Grid.Row="2" Grid.Column="1"                />
        <Label Text="Modelo de dispositivo" Grid.Row="3" Grid.Column="0"  />
        <Label Text="{Binding ModeloDispositivo}" Grid.Row="3" Grid.Column="1"                    />
        <Label Text="Verion S.O." Grid.Row="4" Grid.Column="0"                                    />
        <Label Text="{Binding VersionSO}" Grid.Row="4" Grid.Column="1/>
      </Grid>
    </StackLayout>
  </ScrollView>
```