



TUTORIAL DE XAMARIN

Desarrollo de una aplicación móvil multiplataforma

Universidad de Costa Rica
Sede de Occidente
Informática Empresarial
Lenguajes para Aplicaciones Comerciales

José Andrés Hernández Vargas
B43324

Mauricio Mourraille Rojas
B34752

Yoswel Steven Ulate Camacho
B16633

Gerald Josué González Barrantes
B42939

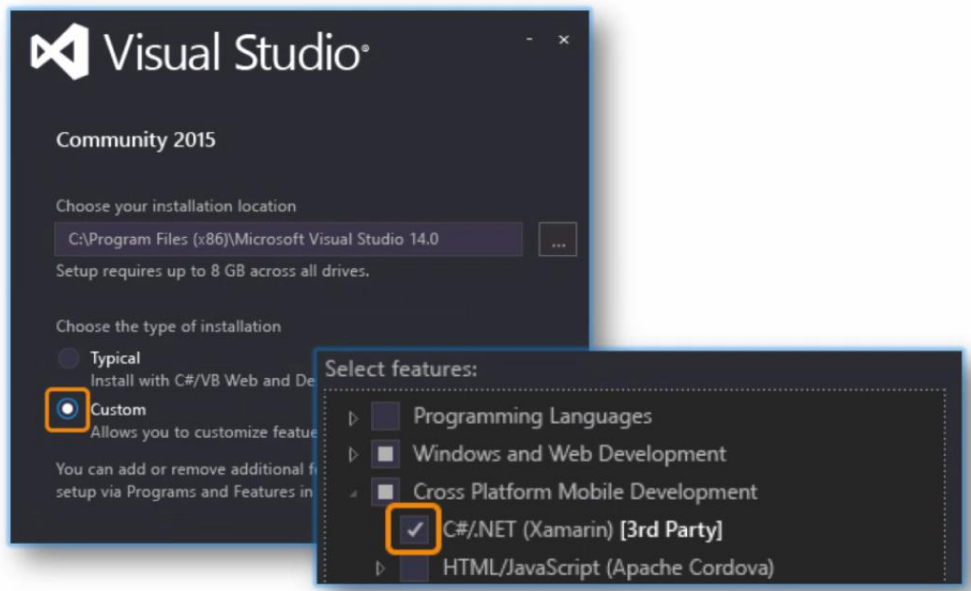
Objetivo:

El objetivo de este tutorial es enseñar una solución real a una aplicación usando distintos elementos de Xamarin, está dividida por partes para que el usuario pueda escoger cuales elementos le son útiles para su aprendizaje. El código completo de la aplicación lo pueden encontraren el repositorio de Github:

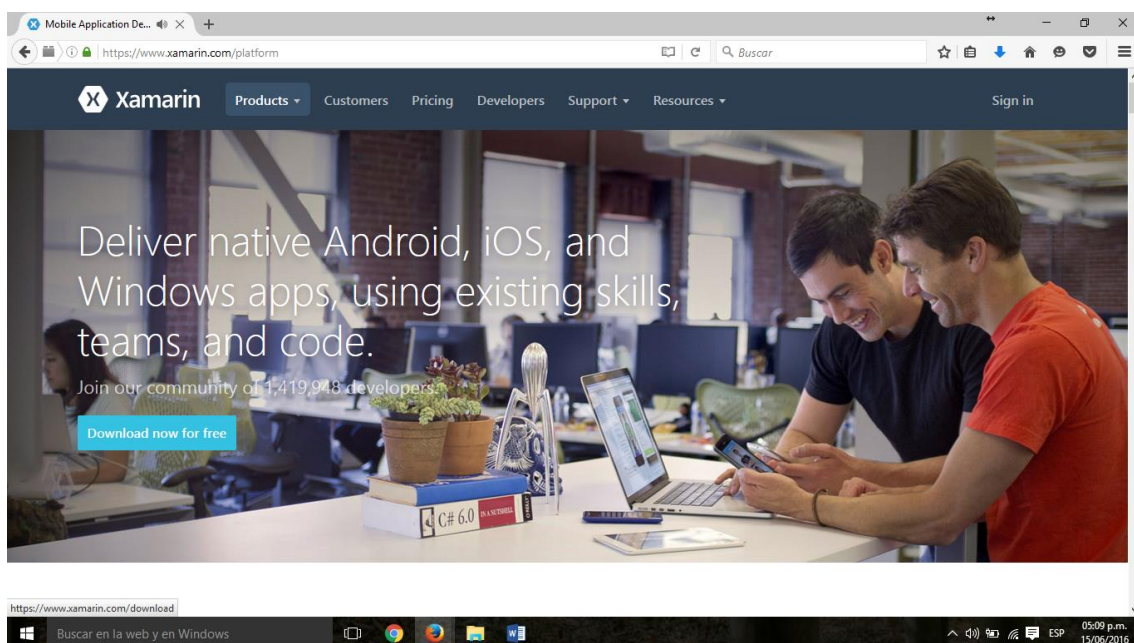
<https://github.com/machohv/XamarinTutorial.git>

Primera Parte: Instalación de Xamarin

1. Xamarin viene incluido en el paquete de instalación de Visual Studio 2015. Solamente se necesita dar click en Custom -> Cross Platform Mobile Development y seleccionar la opción de C#/.Net (Xamarin).



2. Si no se instaló Xamarin con Visual Studio no hay ningún problema, simplemente hay que dirigirse a <https://www.xamarin.com/platform> y dar click en descargar para Visual Studio si es con Windows, o Xamarin Studio en Mac, este último es independiente de Visual Studio. Deberán registrarse en el sitio de Xamarin para poder realizar la descarga.

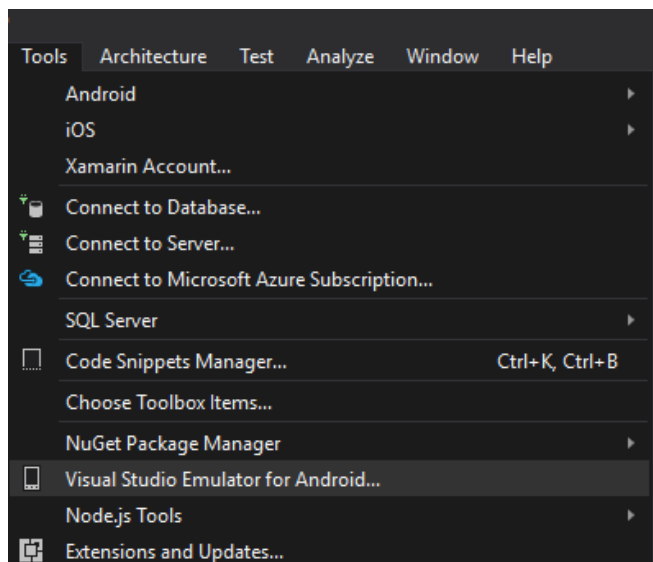


3. Simplemente se deben seguir los pasos para completar la instalación la cual es un poco lenta (un tiempo similar a la instalación de Visual Studio).

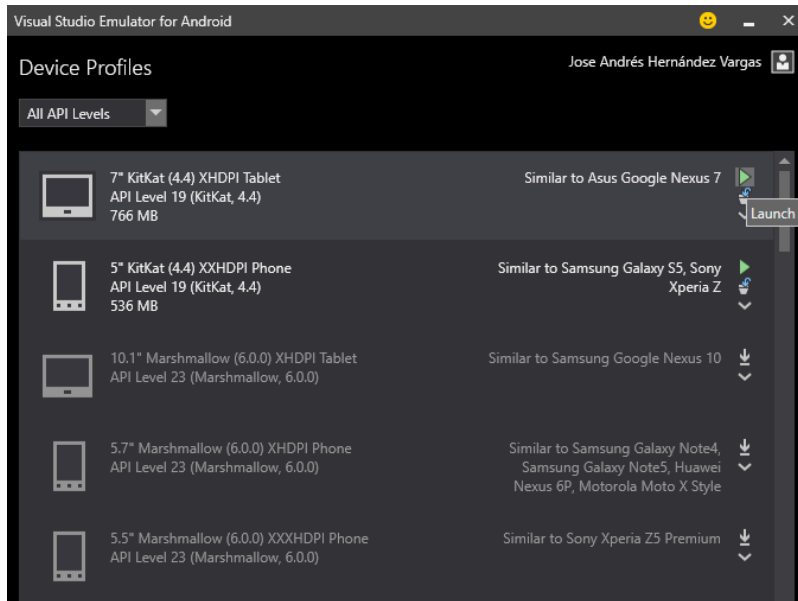


Segunda Parte: Preparando el ambiente de trabajo

4. Para el ejemplo se estarán realizando pruebas en Android. Se puede utilizar a través de las máquinas virtuales que incluye Android para Visual Studio, el problema es que éstas consumen mucha memoria RAM, otra opción es trabajar con un dispositivo conectado al ordenador. La guía para probar las aplicaciones en el dispositivo se encuentra en este link:
https://developer.xamarin.com/guides/android/getting_started/installation/set_up_device_for_development/
5. Para usar la máquina virtual selecciona el Visual Studio Emulator for Android en el menú Tools.



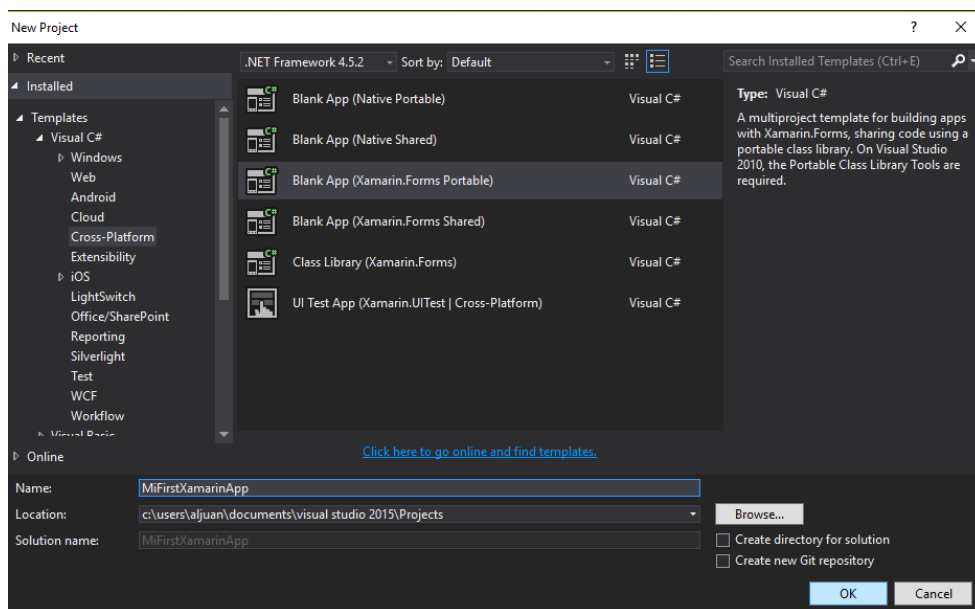
6. Selecciona la máquina virtual de tu preferencia y da click en Launch.



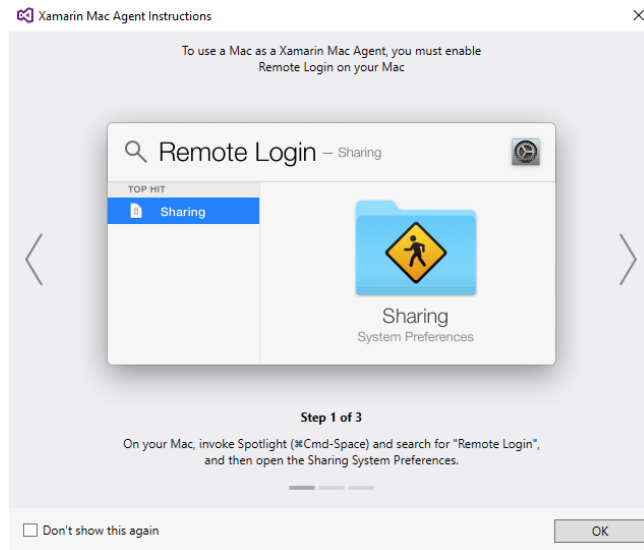
7. Una vez con la máquina virtual iniciada. Cuando se corra el programa empezará automáticamente en el dispositivo y será guardada como una aplicación.

Parte tres: Creación del proyecto y primeras pruebas.

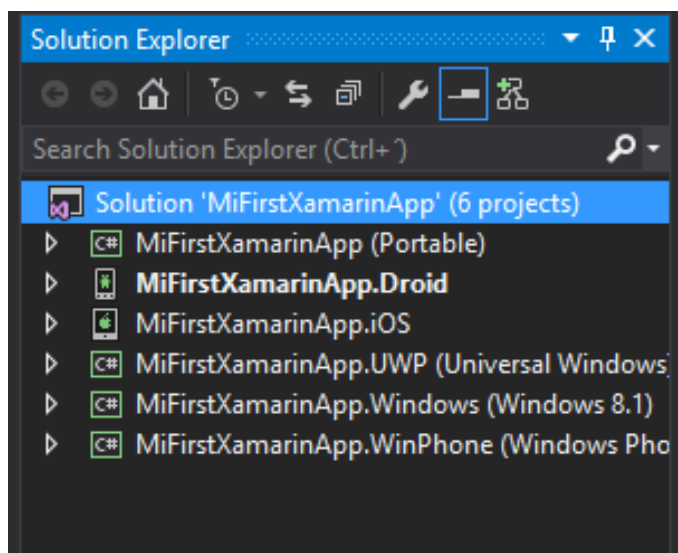
8. Dirijase a crear un nuevo proyecto y seleccione Visual C#, Cross-Platform, Blank App (Xamarin.Forms Portable). De un nombre al proyecto y de click en OK.



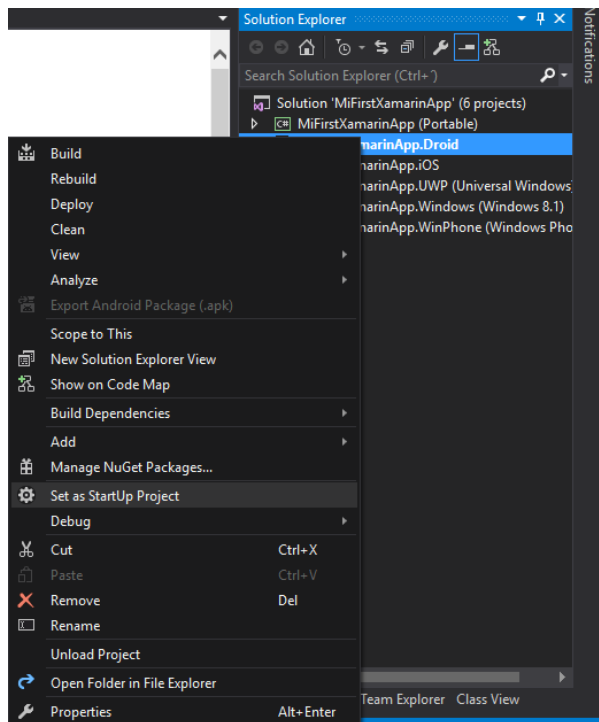
9. Aparecerán las opciones de Xamarin Mac Agent, esto es para probar las aplicaciones de IOS conectándose de forma remota vía SSH a una Mac. En el siguiente link puedes encontrar una guía detallada para conectarse a una Mac: https://developer.xamarin.com/guides/ios/getting_started/installation/windows/connecting-to-mac/



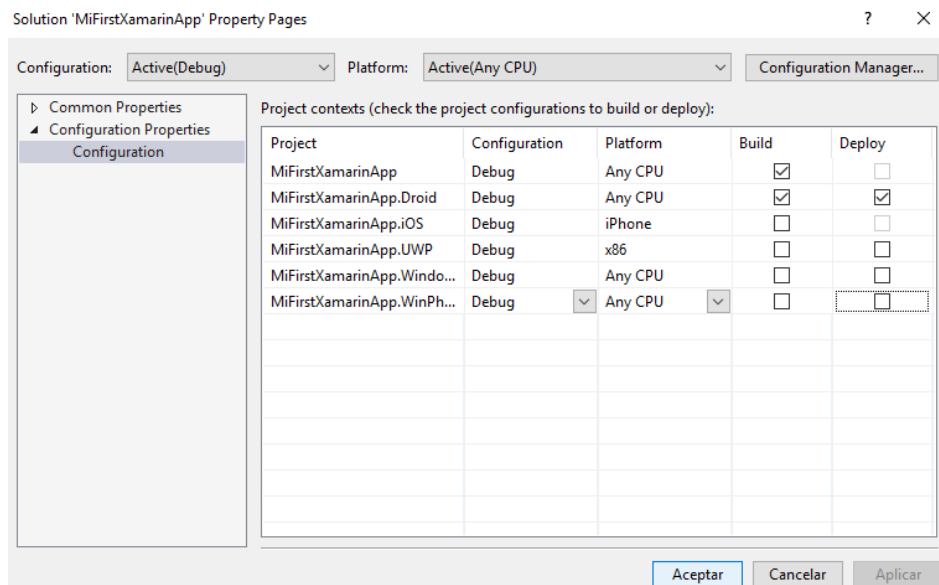
10. Note como se genera un proyecto por cada dispositivo y un proyecto Portable (En este se desarrolla la mayor parte de la lógica y diseño de la app, el resto de los proyectos referencian a este).



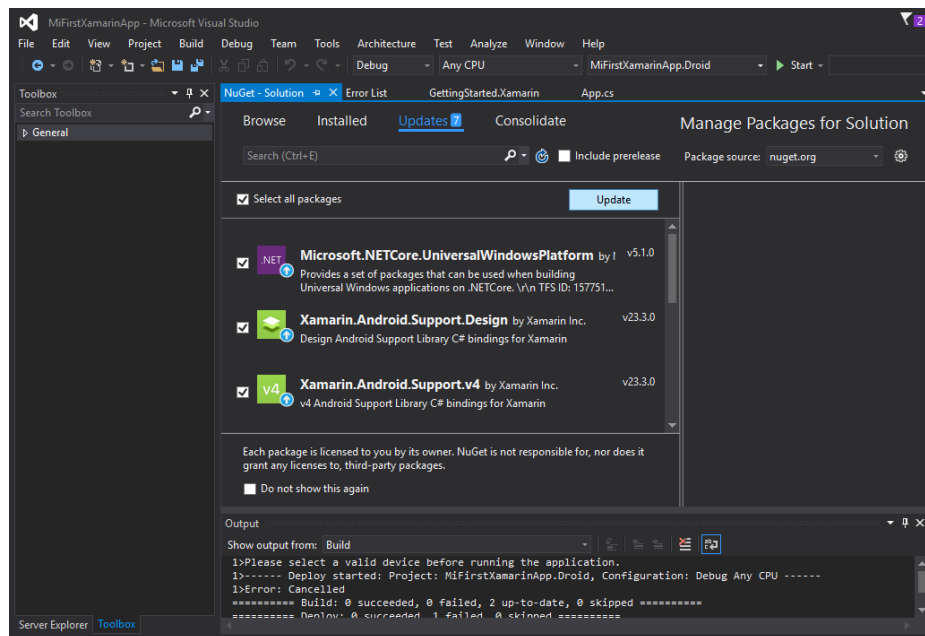
11. Selección el proyecto .Droid como proyecto de inicio para probar la aplicación en Android.



12. Para asegurar que el proyecto va a correr sobre Android. De click en el solution y vaya a Properties. Seleccione solo las opciones para Android y de click en Aplicar y finalmente de click en Aceptar



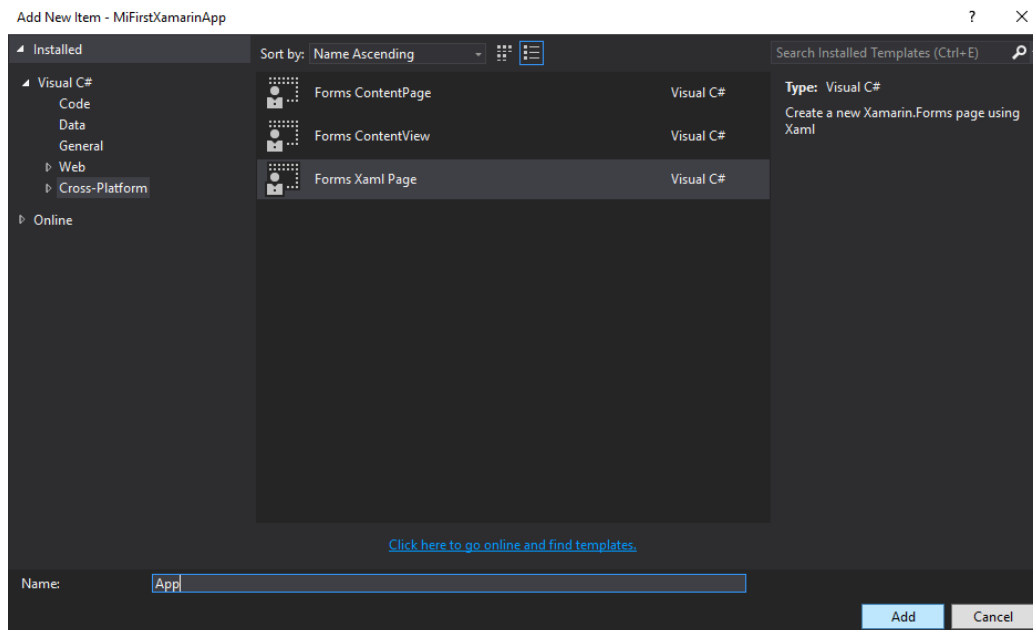
13. Corra la aplicación para verificar que todo funciona, es recomendable dar Build o Rebuild antes de Run, esto porque son tareas muy pesadas y es mejor separlas. Si hay algún problema verifique que tiene todo actualizado, tanto Xamarin con el SDK de Android. Los proyectos de Xamarin son muy propensos a fallar si no todo está actualizado.
14. Si el proyecto no inicia correctamente, intente actualizar todos los paquetes. Esto dándole click derecho en el solution y dando click en Manage Nuget Packages for Solution. O bien puede actualizar todo Xamarin.



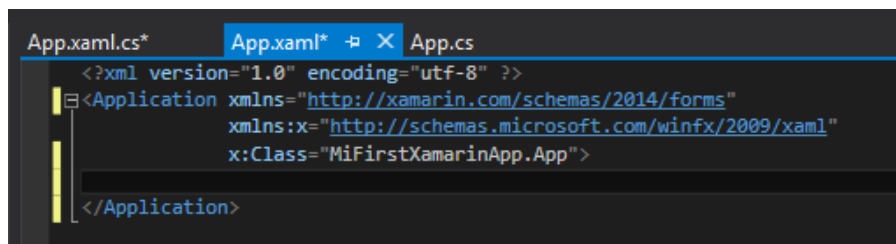
15. La aplicación debería verse de esta manera.



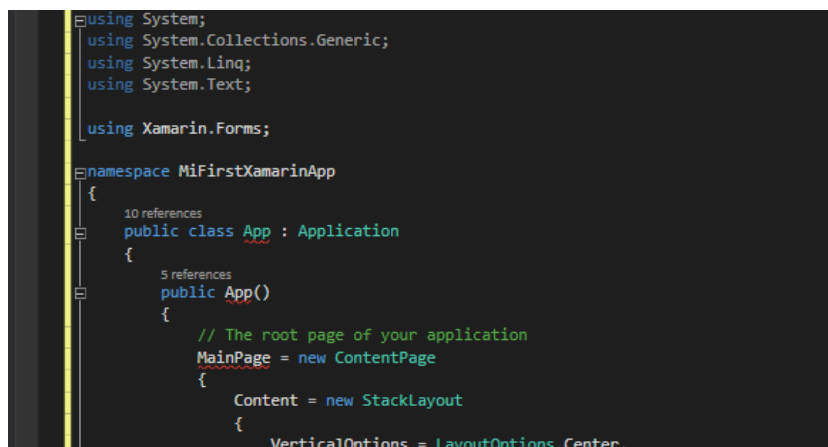
16. Para este ejemplo se trabajará con XAML, por lo que mudaremos la aplicación de la siguiente manera. En el proyecto Portable da click derecho, Add, Add New Item. Selecciona Forms XAML page y nombralo “App”. La aplicación App es el proyecto de inicio en Xamarin.



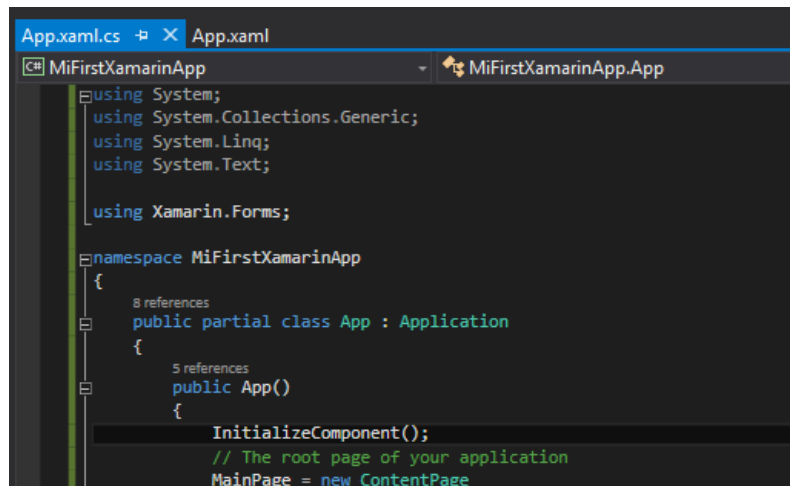
17. Cambie las etiquetas de ContentPage para que sean de tipo Application y borre la etiqueta del Label.



18. Copie el contenido del App.cs y péguelo en el App.xaml.cs

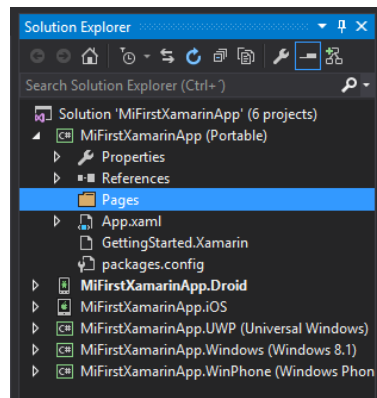


19. Defina la clase como parcial para que funcione correctamente y agregue la línea `InitializeComponent()` al principio del método constructor `App()`. Luego proceda a borrar la clase original `App.cs`

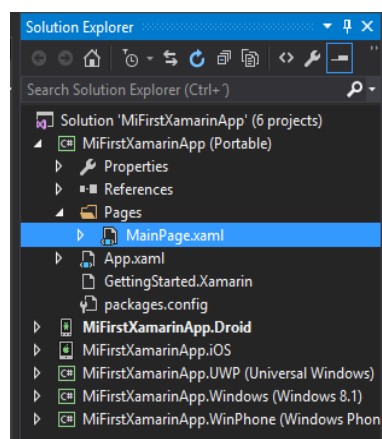


```
App.xaml.cs App.xaml
MiFirstXamarinApp MiFirstXamarinApp.App
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Xamarin.Forms;
namespace MiFirstXamarinApp
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();
            // The root page of your application
            MainPage = new ContentPage
```

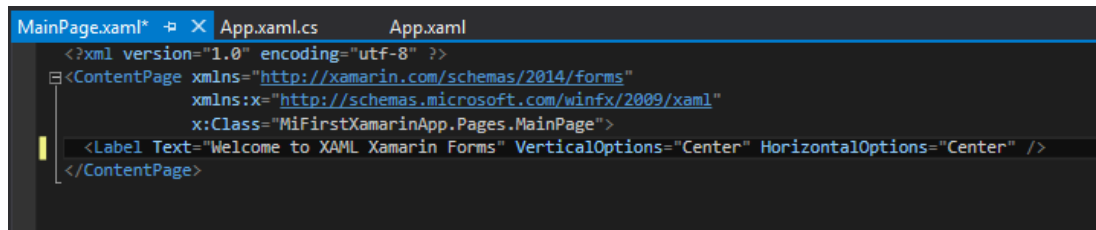
20. Agregamos un nuevo folder al proyecto llamado Pages. Aquí guardaremos todas las páginas de la aplicación.



21. Seguidamente crea una página llamada `MainPage` dentro del folder del tipo Forms Xaml Page.

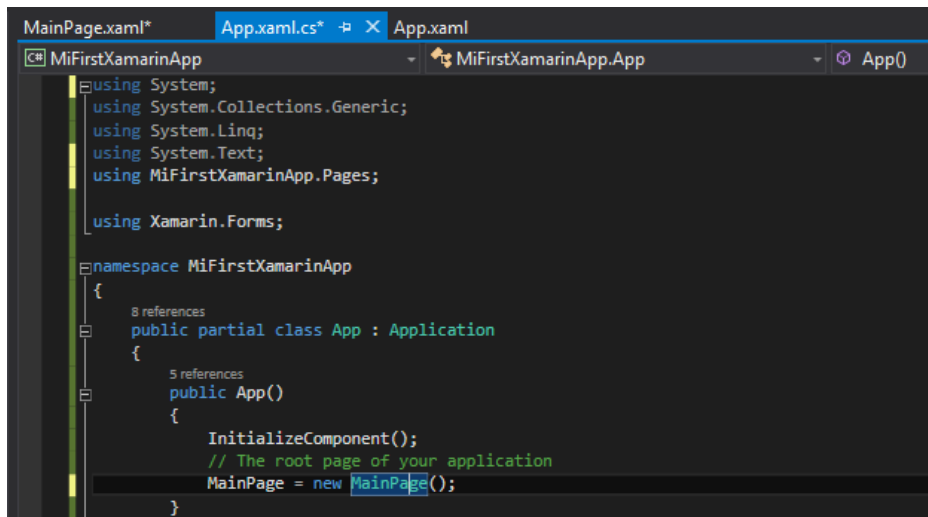


22. Cambie el texto del Label por “Welcome to XAML Xamarin Forms”.



```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MiFirstXamarinApp.Pages.MainPage">
    <Label Text="Welcome to XAML Xamarin Forms" VerticalOptions="Center" HorizontalOptions="Center" />
</ContentPage>
```

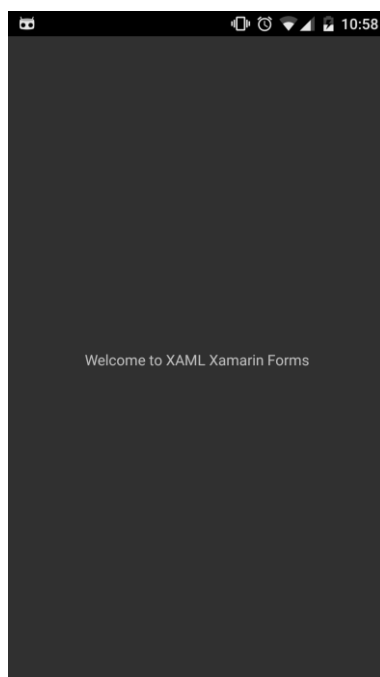
23. En el App.xaml.cs cambie el valor de la variable MainPage y cree un nuevo MainPage(). Recuerde importar la carpeta Pages.



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MiFirstXamarinApp.Pages;
using Xamarin.Forms;

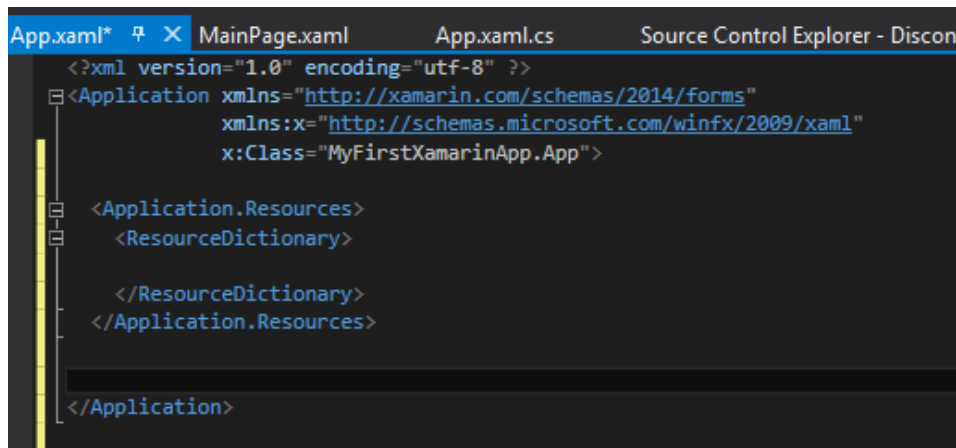
namespace MiFirstXamarinApp
{
    [8 references]
    public partial class App : Application
    {
        [5 references]
        public App()
        {
            InitializeComponent();
            // The root page of your application
            MainPage = new MainPage();
        }
    }
}
```

24. Al correr la aplicación la página deberá de verse de la siguiente manera.



Parte cuatro: Creación de la interfaz gráfica.

25. Lo primero será definir un diccionario de recursos. Esto para no repetir estilos ni configuraciones y solamente referenciar a elementos del diccionario. En el App.xaml definimos las etiquetas Application.Resources y Resource Dictionary.



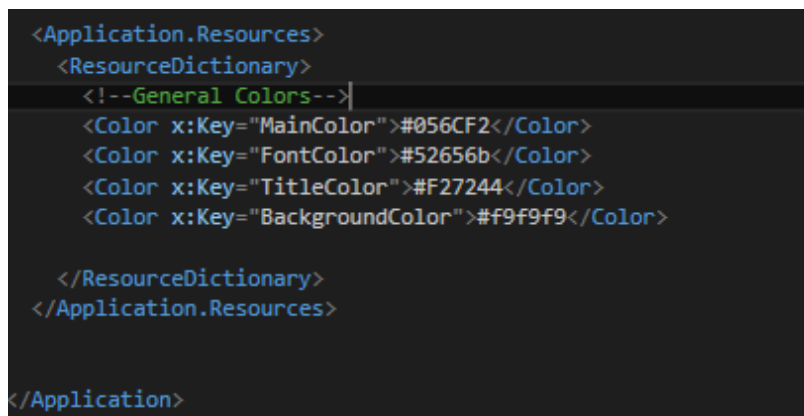
```
<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MyFirstXamarinApp.App">

    <Application.Resources>
        <ResourceDictionary>

        </ResourceDictionary>
    </Application.Resources>

</Application>
```

26. Agregamos algunas definiciones básicas de colores.

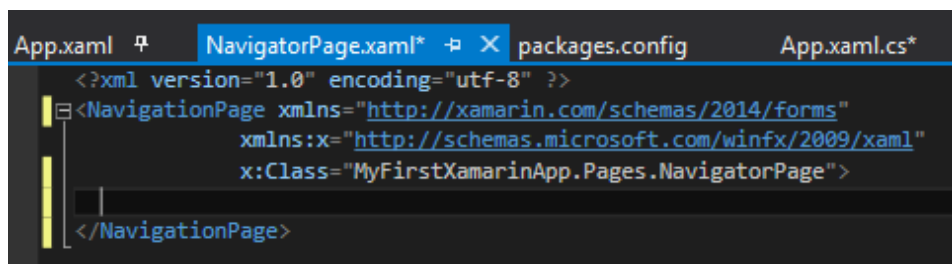


```
<Application.Resources>
    <ResourceDictionary>
        <!--General Colors-->
        <Color x:Key="MainColor" #056CF2</Color>
        <Color x:Key="FontColor" #52656b</Color>
        <Color x:Key="TitleColor" #F27244</Color>
        <Color x:Key="BackgroundColor" #f9f9f9</Color>

    </ResourceDictionary>
</Application.Resources>

</Application>
```

27. En la carpeta pages agregue un nuevo page llamado NavigatorPage.xaml, cambie las etiquetas de ContentPage por NavigationPage, además borre el Label que trae por defecto. Los navigation page son los encargados de manejar el cambio de páginas.



```
<?xml version="1.0" encoding="utf-8" ?>
<NavigationPage xmlns="http://xamarin.com/schemas/2014/forms"
               xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
               x:Class="MyFirstXamarinApp.Pages.NavigationPage">

    </NavigationPage>
```

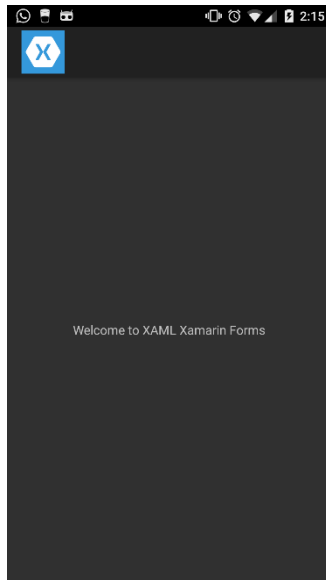
28. En el NavigatorPage.xaml.cs cambié la herencia para que sea de tipo `NavigationPage`. Haga que el constructor reciba una variable de tipo `ContentPage` y use el método `PushAsync(page)` para agregar la página al `NavigationPage`.

```
namespace MyFirstXamarinApp.Pages
{
    3 references | 0 changes | 0 authors, 0 changes
    public partial class NavigatorPage : NavigationPage
    {
        0 references | 0 changes | 0 authors, 0 changes
        public NavigatorPage(ContentPage page)
        {
            this.PushAsync(page);
            InitializeComponent();
        }
    }
}
```

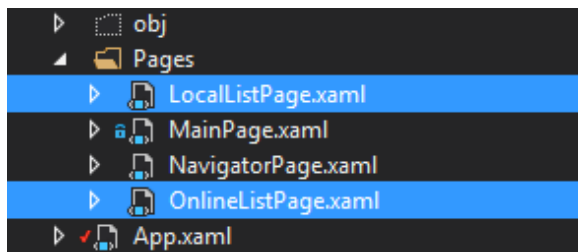
29. En el App.xaml.cs agregue un atributo estático de tipo `NavigationPage`, y en el constructor de la clase inicialice este atributo como un `NavigatorPage` enviando a la `MainPage` como parámetro, luego dele a la variable `MainPage` el valor del atributo.

```
namespace MyFirstXamarinApp
{
    8 references | Aljuan, 11 hours ago | 1 author, 1 change
    public partial class App : Application
    {
        2 references | 0 changes | 0 authors, 0 changes
        public static NavigationPage Navigator { get; internal set; }
        5 references | Aljuan, 11 hours ago | 1 author, 1 change
        public App()
        {
            InitializeComponent();
            Navigator = new NavigatorPage(new MainPage());
            MainPage = Navigator;
        }
    }
}
```

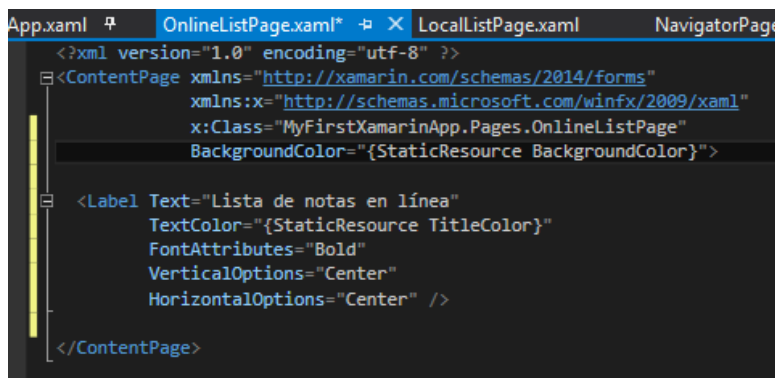
30. Corra la aplicación y verifique que tiene una barra superior con el logo de Xamarin.



31. Lo siguiente es cambiar la navegación de la aplicación para que sea de tipo Carrusel, estas aplicaciones son las que para cambiar de páginas solamente se debe deslizar el dedo hacia la derecha o la izquierda, un ejemplo de estas aplicaciones es Snapchat. Agregue dos páginas en el folder pages. En una se verán las notas que se guardan localmente, en la otra se verán las notas que se guardan en la base de datos de Azure.



32. En cada página cambie el texto del label por defecto por un mensaje. En estas páginas utilizaremos recursos del diccionario. En la etiqueta que abre el ContentPage defina el BackgroundColor como "{StaticResource BackgroundColor}". En los label defina el TextColor como "{StaticResource TitleColor}" y el FontAttributes como "Bold".



33. Una vez realizado el paso anterior en las dos páginas, modifique el MainPage para que sea de tipo CarouselPage en las etiquetas y en la herencia.

```
namespace MyFirstXamarinApp.Pages
{
    4 references | Aljuan, 12 hours ago | 1 author, 1 change
    public partial class MainPage : CarouselPage
    {
        1 reference | Aljuan, 12 hours ago | 1 author, 1 change
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

34. Agregue la siguiente propiedad donde abre la etiqueta CarouselPage xmlns:pages="clr-namespace:ProjectName.Pages;assembly= [SolutionName](#) ", Esta propiedad permite que se agreguen contentPages externos para mantener el código más ordenado. Agregue las referencias a la página de la siguiente manera: <pages:PageName></pages: PageName >.

```
OnlineListPage.xaml  LocalListPage.xaml  MainPage.xaml*  MainPage.xaml.cs
<?xml version="1.0" encoding="utf-8" ?>
<CarouselPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  xmlns:pages="clr-namespace:MyFirstXamarinApp.Pages;assembly=MyFirstXamarinApp"
  x:Class="MyFirstXamarinApp.Pages.MainPage">

  <pages:LocalListPage></pages:LocalListPage>

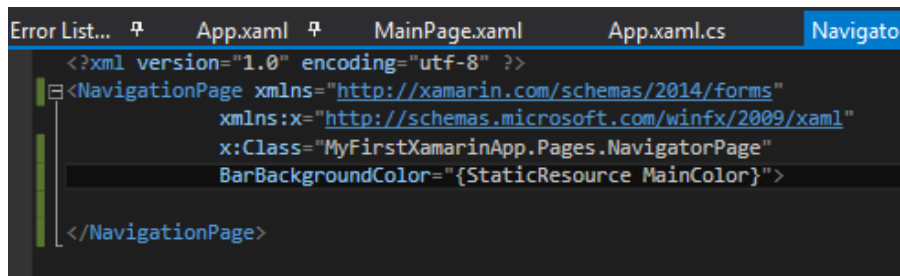
  <pages:OnlineListPage></pages:OnlineListPage>

</CarouselPage>
```

35. Cambie el constructor del NavigatorPage para que reciba un CarouselPage

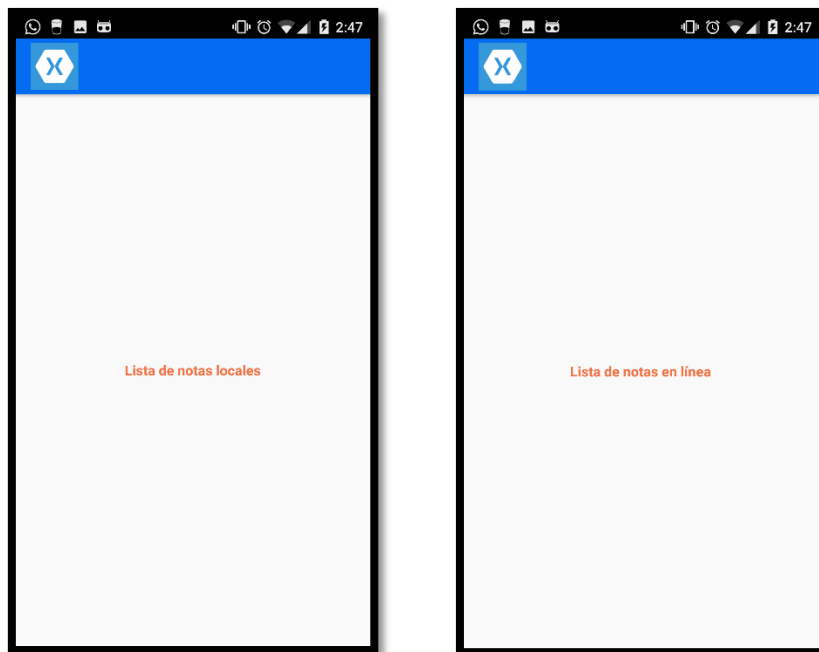
```
1 reference | 0 changes | 0 authors, 0 changes
public NavigatorPage(CarouselPage page)
{
    this.PushAsync(page);
    InitializeComponent();
}
```

36. Defina el BarBackgroundColor como el recurso estático MainColor.

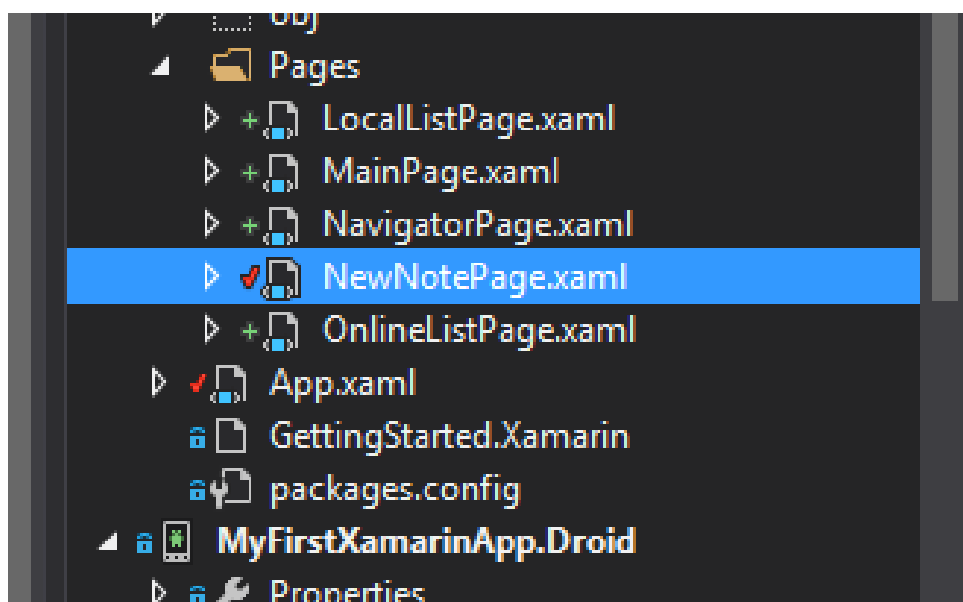


```
<?xml version="1.0" encoding="utf-8" ?>
<NavigationPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="MyFirstXamarinApp.Pages.NavigatorPage"
  BarBackgroundColor="{StaticResource MainColor}">
  </NavigationPage>
```

37. Las vistas de la aplicación deberán verse de la siguiente manera.



38. Agregue una nueva página y llámela “NewNotePage”, en esta página estará el formulario para agregar notas. Cambie el label para que diga un mensaje simple.



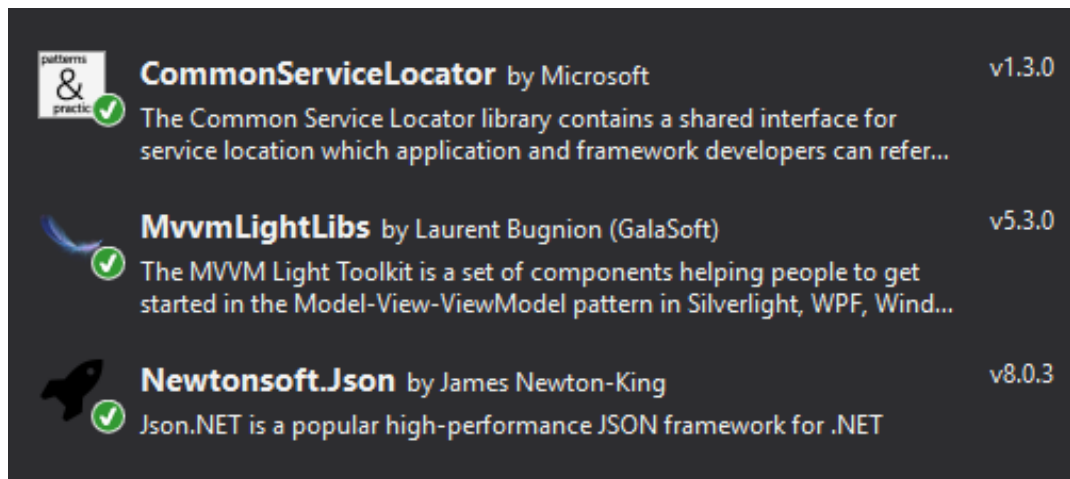

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="MyFirstXamarinApp.Pages.NewNotePage">
    <Label Text="Crear nota" VerticalOptions="Center" HorizontalOptions="Center" />
</ContentPage>

```

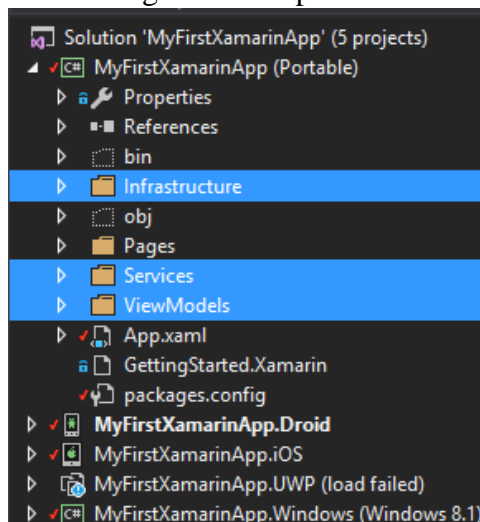
Parte cinco: Agregando la lógica de navegación.

39. Lo primero es instalar algunos paquetes, dale click derecho al solution y ve “Manage NuGet Packages for Solution...” y busque por los siguientes tres paquetes:



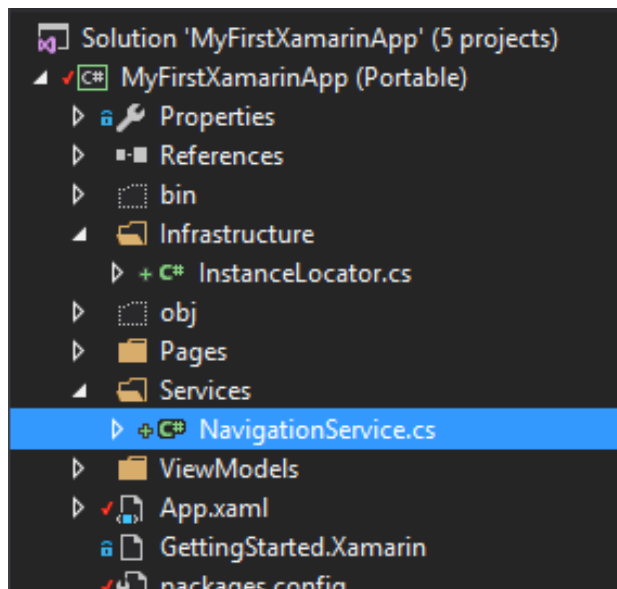
Asegurese de que se instalen en todo el proyecto, esto debido a que a pesar de que solo trabajamos en el Portable, al momento de compilación todos los proyectos referencian a este porque trabajan de forma independiente y si no encuentran los mismos paquetes dará error.

40. Cree las siguientes carpetas: Infrastructure, Services y ViewModels.



Estas carpetas se usarán para separar el código por funcionalidades y no mezclar todo en una sola clase. La carpeta Services contendrá el código específico de algunos elementos como el de la Navegación. La carpeta ViewModels contendrá los códigos de backend de las páginas, el Infrastructure es usado para comunicar los ViewModels con sus páginas.

41. En la carpeta Services cree una clase llamada NavigationService



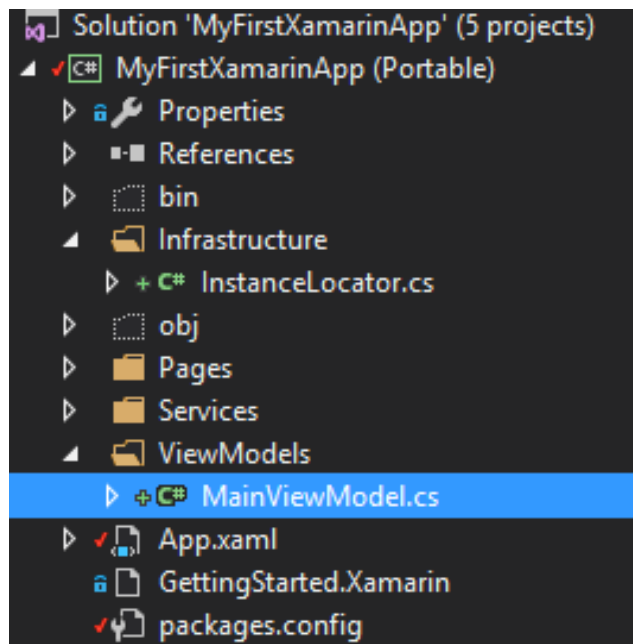
42. Recuerde definir la clase como pública y copie el siguiente contenido.

```
public class NavigationService
{
    1 reference | 0 changes | 0 authors, 0 changes
    public async void Navigate(string PageName)
    {
        switch (PageName)
        {
            case "NewNotePage":
                await Navigate(new NewNotePage());
                break;
            default:
                break;
        }
    }
}

1 reference | 0 changes | 0 authors, 0 changes
private static async Task Navigate<T>(T page) where T : Page
{
    await App.Navigators.PushAsync(page);
}
}
```

Recuerde importar las librerías necesarias para que el código funcione, solamente posicione el cursor sobre un error y Visual Studio le propondrá cuales librerías agregar.

43. En la carpeta ViewModels, agregue una clase llamada MainViewModel



44. Copie el siguiente código dentro de la clase.

```
public class MainViewModel
{
    NavigationService navigationService;

    1 reference | 0 changes | 0 authors, 0 changes
    public MainViewModel()
    {
        navigationService = new NavigationService();
    }

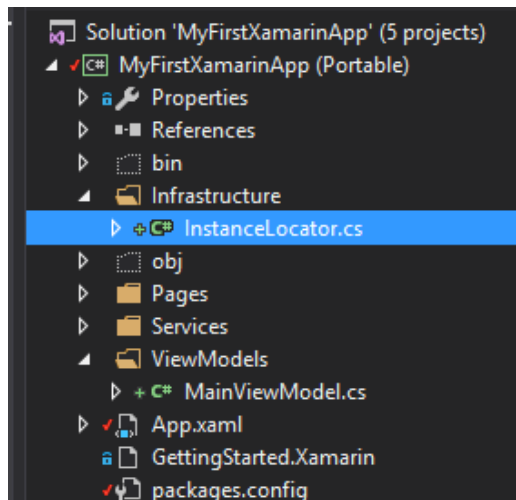
    0 references | 0 changes | 0 authors, 0 changes
    public ICommand GoToCommand { get { return new RelayCommand<string>(GoTo); } }

    1 reference | 0 changes | 0 authors, 0 changes
    private void GoTo(string pageName)
    {
        navigationService.Navigate(pageName);
    }
}
```

Como notará el RelayCommand necesita de la librería de MvvmLight para funcionar. Hasta este momento lo que se hizo fue crear un comando de navegación. Este comando llama a un método que trae un parámetro. En el método se hace llamado al método Navigate de NavigationService en el cual se

crea una nueva página dependiendo de su nombre y en el Task Navigate es puesta en el Navigator con su método PushAsync.

45. En la carpeta Infrastructure cree una nueva clase denominada InstanceLocator.



46. Copie el siguiente código dentro de la clase

47.

```
1 reference | 0 changes | 0 authors, 0 changes
public class InstanceLocator
{
    0 references | 0 changes | 0 authors, 0 changes
    public InstanceLocator()
    {
        Main = new MainViewModel();
    }
    1 reference | 0 changes | 0 authors, 0 changes
    public MainViewModel Main { get; set; }
}
```

48. Lo siguiente es definir esta infraestructura como un recurso estático. Para esto diríjase al diccionario de recursos en el App.xaml. Para que se puedan agregar infraestructuras copie la siguiente línea en la etiqueta de Application.

xmlns:infra="clr-

namespace:ProjectName.InfrastructureFolderName;assembly=SolutionName ".

Luego agregue la etiqueta de Infrastructure

<infra:InstanceLocator x:Key="Locator"></infra:InstanceLocator>

```

<?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:infra="clr-namespace:MyFirstXamarinApp.Infrastructure;assembly=MyFirstXamarinApp"
  x:Class="MyFirstXamarinApp.App">

  <Application.Resources>
    <ResourceDictionary>
      <!--General Colors-->
      <Color x:Key="MainColor">#056CF2</Color>
      <Color x:Key="FontColor">#52656b</Color>
      <Color x:Key="TitleColor">#F27244</Color>
      <Color x:Key="BackgroundColor">#f9f9f9</Color>

      <!--Locator-->
      <infra:InstanceLocator x:Key="Locator"></infra:InstanceLocator>
    </ResourceDictionary>
  </Application.Resources>

</Application>

```

49. Ahora en las páginas de LocalListPage y OnlineListPage defina el BindinContext en la etiqueta de ContentPage. Este BindingContext definirá al MainViewModel a través del Locator que creamos anteriormente. Además agregue un ToolbarItem, este elemento llamará al comando del MainViewModel y pasará de parámetro del nombre de la nueva página. Los ToolbarItems se crean dentro de las etiquetas de ContentPage.ToolbarItems.

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="MyFirstXamarinApp.Pages.LocalListPage"
  BackgroundColor="{StaticResource BackgroundColor}"
  BindingContext="{Binding Main, Source={StaticResource Locator}}">

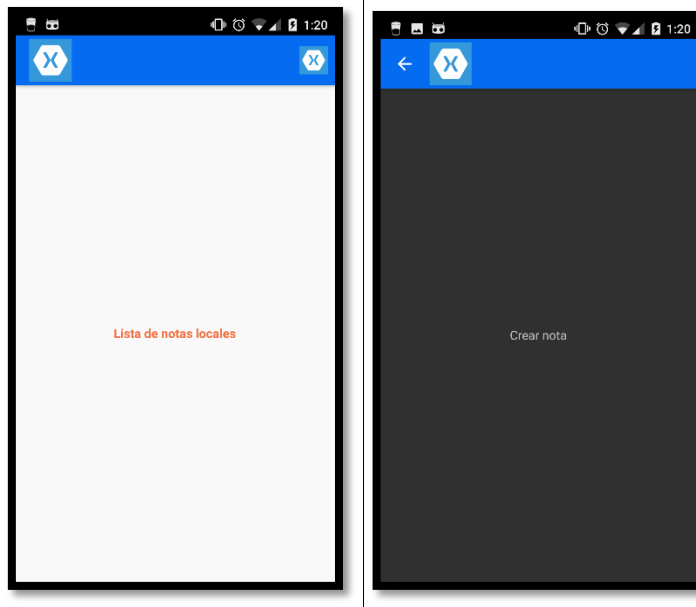
  <Label Text="Lista de notas locales"
    TextColor="{StaticResource TitleColor}"
    FontAttributes="Bold"
    VerticalOptions="Center"
    HorizontalOptions="Center" />

  <ContentPage.ToolbarItems>
    <ToolbarItem Icon="icon.png" Command="{Binding GoToCommand}" CommandParameter="NewNotePage">
    </ToolbarItem>
  </ContentPage.ToolbarItems>

</ContentPage>

```

50. Al correr la aplicación, notará como aparece un nuevo ícono de Xamarin en la esquina superior derecha, al hacer click en este ícono se abre la nueva página y automáticamente el Navigator pone el ícono de volver a la izquierda, al darle click se redirige a la página anterior.

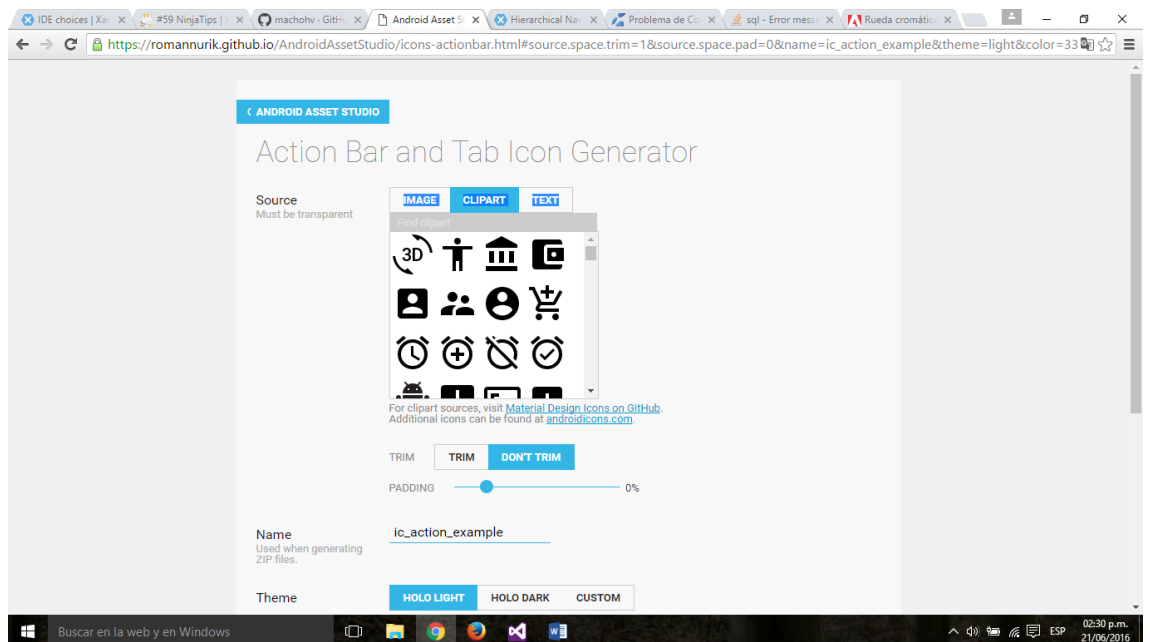


Parte seis: Agregando Íconos.

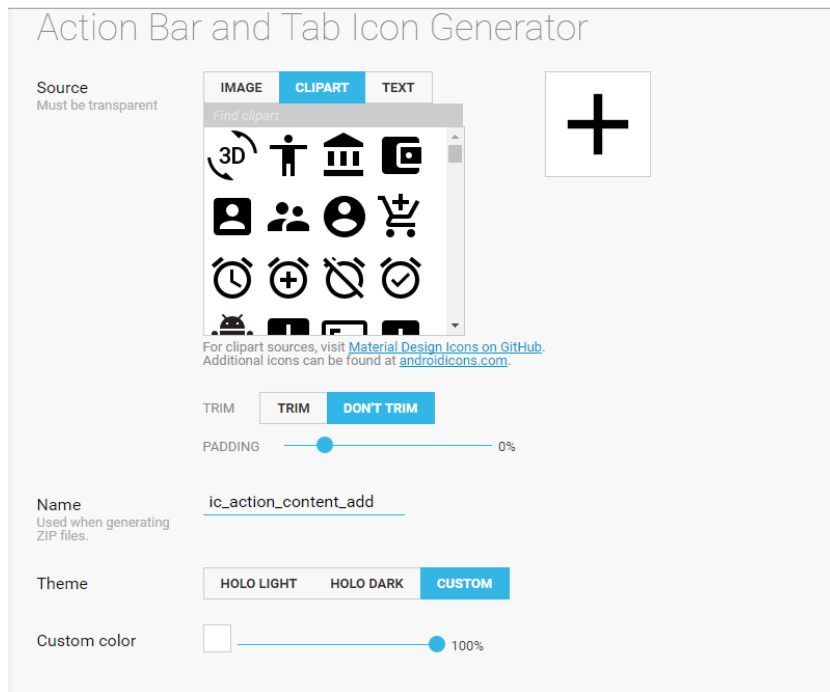
51. Lo primero en esta parte es buscar íconos, en el siguiente link se pueden crear los íconos necesarios para nuestra app:

<https://romannurik.github.io/AndroidAssetStudio/icons-actionbar.html>

Seleccione CLIPART para usar los predeterminados.



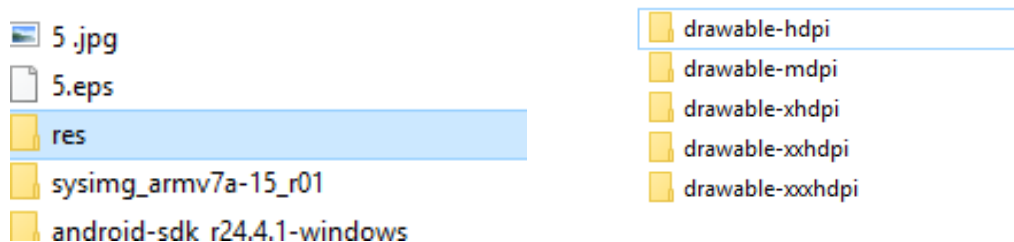
52. Agregue un ícono de +. Puede poner el nombre que quiera, para este ejemplo dejaremos el nombre predeterminado. Cambie el color a blanco y ponga la barra al 100% (Esta es la opacidad). Descargue el zip.



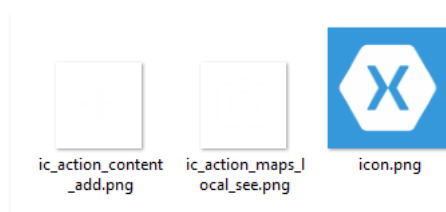
53. Descargue el ícono de una cámara.



54. Extraiga todos los zips, notará como todas las imágenes se guardan en un folder llamado res.



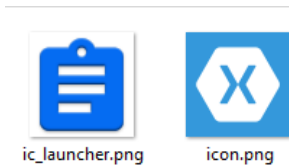
55. Copie las carpetas dentro de rest y péguelas en la siguiente dirección:
SolutionFolder/SolutionName/SolutionName.Droid/Resources/
Notará como ya existían unos folder llamados de la misma forma.



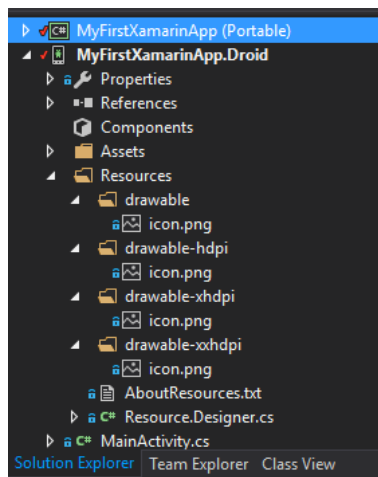
56. Para el ícono del app diríjase a esta dirección:
<https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html> .Escoja el ícono que desee para el app. Puede subir una imagen en formato de vectores. Para este ejemplo se usará el siguiente ícono con el mismo color que el MainColor con el Shape Bevel



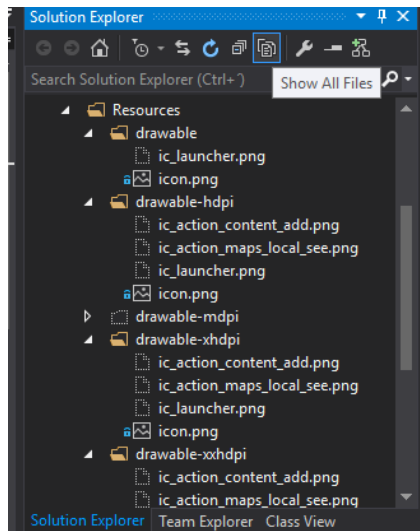
57. Descomprima el archivo que descargó. Notará como estas carpetas se llaman diferente, pero tienen el mismo código de tamaño. Copie y pegue cada archivo en su respectiva carpeta. En la carpeta drawable pegue el archivo de hdpi



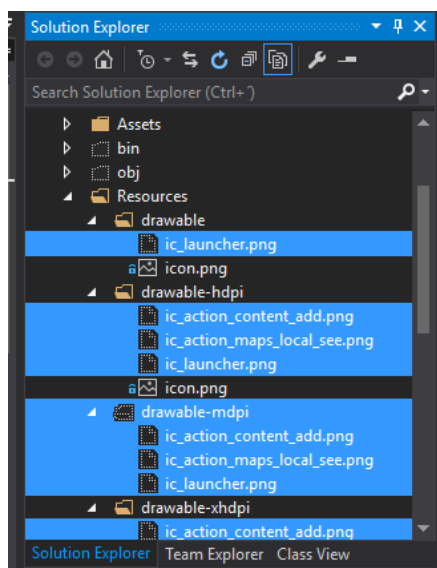
58. Dentro de Visual Studio en el Solution Explorer diríjase a las carpetas que deberían contener las imágenes.



59. Dele click en Show All Files para que salgan las imágenes.



60. Seleccione todas las imágenes, de click derecho y seleccione Include in Project.



61. En la clase MainActivity dentro del proyecto de Android cambie el icono en el Label de Activity por “ic_launcher” o el nombre que definieron para su icono. Aprovechando que estamos en esta clase haremos dos ajustes más, cambiaremos el tema por defecto para que sea fondo blanco, esto porque los campos de texto tendrían letras blancas si el fondo fuera oscuro y con nuestro tema de fondo, no se vería lo que se escribe.

```
namespace MyFirstXamarinApp.Droid
{
    [Activity(Label = "MyFirstXamarinApp", Icon = "@drawable/ic_launcher",
        Theme = "@android:style/Theme.Material.Light.DarkActionBar",
        MainLauncher = true, ConfigurationChanges = ConfigChanges.ScreenSize
        0 references | Aljuan, 5 days ago | 1 author, 1 change
    public class MainActivity : global::Xamarin.Forms.Platform.Android.Forms
    {
        1 reference | Aljuan, 5 days ago | 1 author, 1 change
        protected override void OnCreate(Bundle bundle)
    }
}
```

62. El otro ajuste que se hará es quitar el ícono de la barra de navegación. Esto con la siguiente línea de `ActionBar.SetIcon` dentro del método `OnCreate`.

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

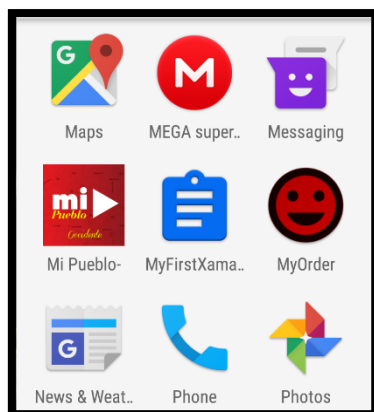
    global::Xamarin.Forms.Forms.Init(this, bundle);
    LoadApplication(new App());

    ActionBar.SetIcon(
        new ColorDrawable(Resources.GetColor(Android.Resource.Color.Transparent)));
}
```

63. Ahora en el proyecto portable en las páginas `LocalListPage` y `OnlineListPage` cambie los iconos del `ToolBarItem` por los del +.

```
<ContentPage.ToolbarItems>
  <ToolBarItem Icon="ic_action_content_add.png" Command="{Binding GoToCommand}" CommandParameter="NewNote" />
</ToolBarItem>
</ContentPage.ToolbarItems>
</ContentPage>
```

64. Notará como ahora el logo de la aplicación ha cambiado y en el `Navigation Bar` aparece el símbolo de + y no estará el logo de Xamarin.



Parte siete: Creación del formulario de notas.

65. En la clase página NewNotePage edite la etiqueta de ContentPage, defina el BackgroundColor, agregue un título y defina el Binding Context.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
  xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
  x:Class="MyFirstXamarinApp.Pages.NewNotePage"
  BackgroundColor="{StaticResource BackgroundColor}"
  Title="Nueva Nota"
  BindingContext="{Binding Main, Source={StaticResource Locator}}">

  <Label Text="Crear nota" VerticalOptions="Center" HorizontalOptions="Center" />
</ContentPage>
```

66. Borre el Label y copie el siguiente formulario dentro de NewNotePage. Este formulario contendrá un espacio para el título, otro para la descripción, otro para la fecha y por último un botón que abrirá la cámara del dispositivo para agregar fotos.

```
<ScrollView>
  <StackLayout Padding="8">
    <Label TextColor="{StaticResource TitleColor}" Text="Título"></Label>
    <Entry TextColor="{StaticResource FontColor}" BackgroundColor="White" ></Entry>

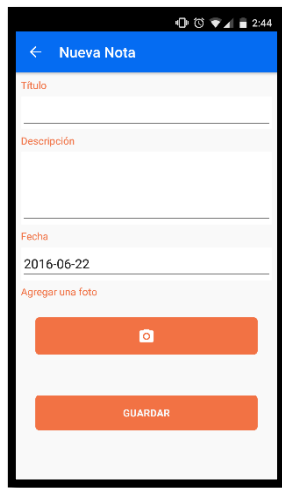
    <Label TextColor="{StaticResource TitleColor}" Text="Descripción"></Label>
    <Editor TextColor="{StaticResource FontColor}" BackgroundColor="White"
      HeightRequest="100" ></Editor>

    <Label TextColor="{StaticResource TitleColor}" Text="Fecha"></Label>
    <DatePicker BackgroundColor="White">
      <DatePicker.Format>yyyy-MM-dd</DatePicker.Format>
    </DatePicker>

    <Label TextColor="{StaticResource TitleColor}" Text="Agregar una foto"></Label>
    <Button BackgroundColor="{StaticResource TitleColor}"
      Image="ic_action_maps_local_see.png"
      Margin="20"></Button>

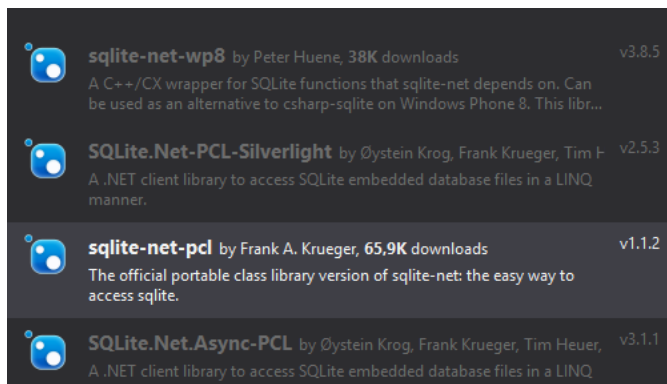
    <Button BackgroundColor="{StaticResource TitleColor}"
      Text="Guardar"
      TextColor="{StaticResource BackgroundColor}"
      FontAttributes="Bold"
      Margin="20, 30"></Button>
  </StackLayout>
</ScrollView>
```

67. Compruebe que el formulario se creó de manera correcta:

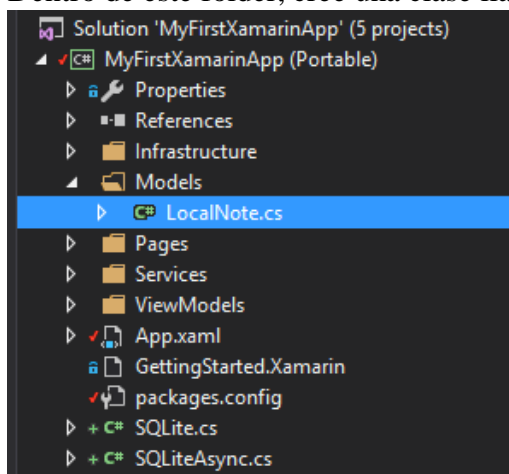


Parte ocho: Guardando las notas en SQLite.

68. Instale el Nuget Package de SQLite-net-pcl



69. Cree un nuevo folder llamado Models, este contendrá las clases de objetos. Dentro de este folder, cree una clase llamada LocalNote.



70. Agregue los siguientes properties a la clase LocalNote.

```

public class LocalNote
{
    0 references | 0 changes | 0 authors, 0 changes
    public int Id { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string Title { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string Description { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public DateTime Date { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public string ImagePath { get; set; }
}

```

71. Importe SQLite en la clase y agregue las etiquetas de Table, PrimaryKey Autoincrement para el Id y MaxLength para el Title.

```

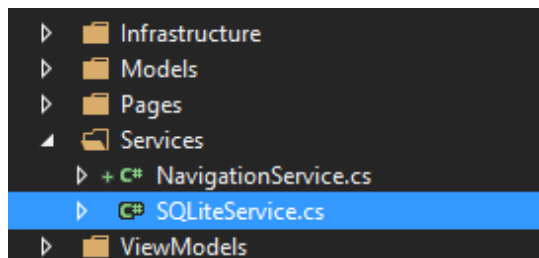
using SQLite;

namespace MyFirstXamarinApp.Models
{
    [Table("Note")]
    0 references | 0 changes | 0 authors, 0 changes
    public class LocalNote
    {
        [PrimaryKey, AutoIncrement]
        0 references | 0 changes | 0 authors, 0 changes
        public int Id { get; set; }

        [MaxLength(250)]
        0 references | 0 changes | 0 authors, 0 changes
        public string Title { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public string Description { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public DateTime Date { get; set; }
        0 references | 0 changes | 0 authors, 0 changes
        public string ImagePath { get; set; }
    }
}

```

72. Cree una clase en Services llamada SQLiteService, esta clase contendrá toda la lógica de acceso a datos.



73. Importe SQLite en la clase y el Folder Models, cree una conexión de atributo y un mensaje de estatus. Inicialice la conexión en el constructor y cree la tabla para NoteTable.

```
using SQLite;
using MyFirstXamarinApp.Models;

namespace MyFirstXamarinApp.Services
{
    1 reference | 0 changes | 0 authors, 0 changes
    public class SQLiteService
    {
        private readonly SQLiteConnection conn;

        2 references | 0 changes | 0 authors, 0 changes
        public string StatusMessage { get; set; }

        0 references | 0 changes | 0 authors, 0 changes
        public SQLiteService(string dbPath)
        {
            conn = new SQLiteConnection(dbPath);
            conn.CreateTable<LocalNote>();
        }
    }
}
```

74. Cree un método para agregar notas. Este método recibirá un objeto de tipo LocalNote como parámetro, validará si el título es nulo e insertará el objeto en la base de datos.

```
1 reference | 0 changes | 0 authors, 0 changes
public void AddNewNote(LocalNote note)
{
    try
    {
        if (string.IsNullOrEmpty(note.Title))
            throw new Exception("Valid title required");

        //insert a new note into the Note table
        var result = conn.Insert(new LocalNote {
            Title = note.Title,
            Description = note.Description,
            Date = note.Date,
            ImagePath = note.ImagePath
        });

        StatusMessage = "Se agregó la nota correctamente";
    }
    catch (Exception ex)
    {
        StatusMessage = "Error agregando la nota";
    }
}
}
```

75. Finalmente, agregue un método para ver las notas guardadas en la base de datos. Este método devolverá una lista de objetos de tipo LocalNote.

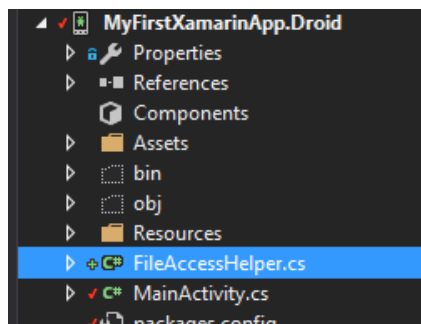
```
0 references | 0 changes | 0 authors, 0 changes
public List<LocalNote> GetAllNotes()
{
    //return a list of notes saved to the Note table in the database
    return conn.Table<LocalNote>().ToList();
}
```

76. En el App.xaml.cs agregue una propiedad estática de tipo SQLiteService, haga que el constructor reciba un parámetro de tipo string, que será la ruta de la base de datos e inicialice la variable en el constructor.

```
3 references | 0 changes | 0 authors, 0 changes
public static SQLiteService sqliteService { get; private set; }

4 references | 0 changes | 0 authors, 0 changes
public App(string dbPath)
{
    InitializeComponent();
    Navigator = new NavigatorPage(new MainPage());
    sqliteService = new SQLiteService(dbPath);
    MainPage = Navigator;
}
```

77. Diríjase al proyecto de Android y cree una clase llamada FileAccessHelper, esta clase contendrá el método que buscará la ruta de las bases de datos del dispositivo.



78. En la clase cree el siguiente método estático.

```
namespace MyFirstXamarinApp.Droid
{
    1 reference | 0 changes | 0 authors, 0 changes
    public class FileAccessHelper
    {
        1 reference | 0 changes | 0 authors, 0 changes
        public static string GetLocalFilePath(string filename)
        {
            string path = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
            return System.IO.Path.Combine(path, filename);
        }
    }
}
```

79. Ahora en el MainActivity haga un llamado a este método, mande de filename el nombre que desee para su base de datos, si no existe, se crea automáticamente, luego envíe el path como parámetro cuando crea el objeto App.

```

public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsApplicationActivity
{
    1 reference | Aljuan, 6 days ago | 1 author, 1 change
    protected override void OnCreate(Bundle bundle)
    {
        base.OnCreate(bundle);

        global::Xamarin.Forms.Forms.Init(this, bundle);

        string dbPath = FileAccessHelper.GetLocalFilePath("Note.db3");

        LoadApplication(new App(dbPath));

        ActionBar.SetIcon(
            new ColorDrawable(Resources.GetColor(Android.Resource.Color.Transparent)));
    }
}

```

80. Ya está toda la lógica para agregar y ver notas, el siguiente paso es hacer que el formulario se comunice con el método de agregar notas. Diríjase a la página NewNotePage y agregue una acción al botón para llamar a un nuevo método denominado SaveNote.

```

<Button BackgroundColor="{StaticResource TitleColor}"
    Text="Guardar"
    TextColor="{StaticResource BackgroundColor}"
    FontAttributes="Bold"
    Margin="20, 30"
    Clicked="SaveNote"></Button>

```

81. Dele nombre a cada campo de texto con la propiedad de Xaml x:Name

```

<ScrollView>
    <StackLayout Padding="8">
        <Label TextColor="{StaticResource TitleColor}" Text="Título"></Label>
        <Entry x:Name="Title" TextColor="{StaticResource FontColor}" BackgroundColor="White" ></Entry>

        <Label TextColor="{StaticResource TitleColor}" Text="Descripción"></Label>
        <Editor x:Name="Description" TextColor="{StaticResource FontColor}" BackgroundColor="White"
            HeightRequest="100" ></Editor>

        <Label TextColor="{StaticResource TitleColor}" Text="Fecha"></Label>
        <DatePicker BackgroundColor="White" x:Name="Date">
            <DatePicker.Format>yyyy-MM-dd</DatePicker.Format>
        </DatePicker>

        <Label TextColor="{StaticResource TitleColor}" Text="Agregar una foto"></Label>
        <Button BackgroundColor="{StaticResource TitleColor}"
            Image="ic_action_maps_local_see.png"
            Margin="20"></Button>
    </StackLayout>
</ScrollView>

```

82. Ahora hay que crear este método. En la clase NewNotePage.xaml.cs cree un nuevo método llamado SaveNote y agregue la lógica para el llamado al método que guarda en la base de datos.


```

public void SaveNote(object sender, EventArgs args)
{
    LocalNote note = new LocalNote
    {
        Title = Title.Text,
        Description = Description.Text,
        Date = Date.Date,
        ImagePath = "icon.png"
    };

    App.sqliteService.AddNewNote(note);
    string message = App.sqliteService.StatusMessage;

    App.Navigator.DisplayAlert("Información", message, "Ok");
}

```

Parte nueve: Accediendo a las notas de SQLite.

83. En el diccionario de recursos en el App.xaml cree un DataTemplate para mostrar una lista de elementos, en este caso las notas.

```

<!--Data template-->
<DataTemplate x:Key="NoteItemTemplate">
    <ViewCell>
        <Grid Padding="8">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto"></ColumnDefinition>
                <ColumnDefinition Width="*"></ColumnDefinition>
            </Grid.ColumnDefinitions>
            <Image WidthRequest="50" HeightRequest="50" VerticalOptions="Start"
                Source="{Binding ImagePath}"></Image>
            <StackLayout Grid.Column="1">
                <Label FontAttributes="Bold" VerticalOptions="Center"
                    TextColor="{StaticResource FontColor}" Text="{Binding Title}"></Label>
                <Label VerticalOptions="Center" TextColor="{StaticResource FontColor}"
                    Text="{Binding Description}"></Label>
                <Label VerticalOptions="Center" TextColor="{StaticResource MainColor}"
                    Text="{Binding Date, StringFormat='{0:yyyy/MM/dd}'}"></Label>
            </StackLayout>
        </Grid>
    </ViewCell>
</DataTemplate>

```

84. Ahora diríjase a la página LocalListPage, cree un ListView, defina un x:Name y defina como ItemTemplate el recurso estático NoteItemTemplate que creó anteriormente. Además modifique el label del título para que tenga un margin de 8 y el tamaño de letra de 20. Encierre todos los elementos en un StackLayout

```

<StackLayout>
  <Label Text="Lista de notas locales"
        TextColor="{StaticResource TitleColor}"
        FontAttributes="Bold"
        FontSize="20"
        VerticalOptions="Center"
        Margin="8"
        HorizontalOptions="Center" />

  <ListView x:Name="listaNotas"
            ItemTemplate="{StaticResource NoteItemTemplate}"
            HasUnevenRows="True">
  </ListView>
</StackLayout>

```

85. En el LocalListPage.xaml.cs agregue un property de tipo ObservableCollection<LocalNote>, inicialícela en el constructor. Agregue un método override OnAppearing(), este método se activa cada vez que la página aparece, dentro de este método llame al GetAllNotes del SqliteService, y guarde sus valores en Notes, finalmente llame a la variable del ListView y defina la variable Notes como el ItemsSource

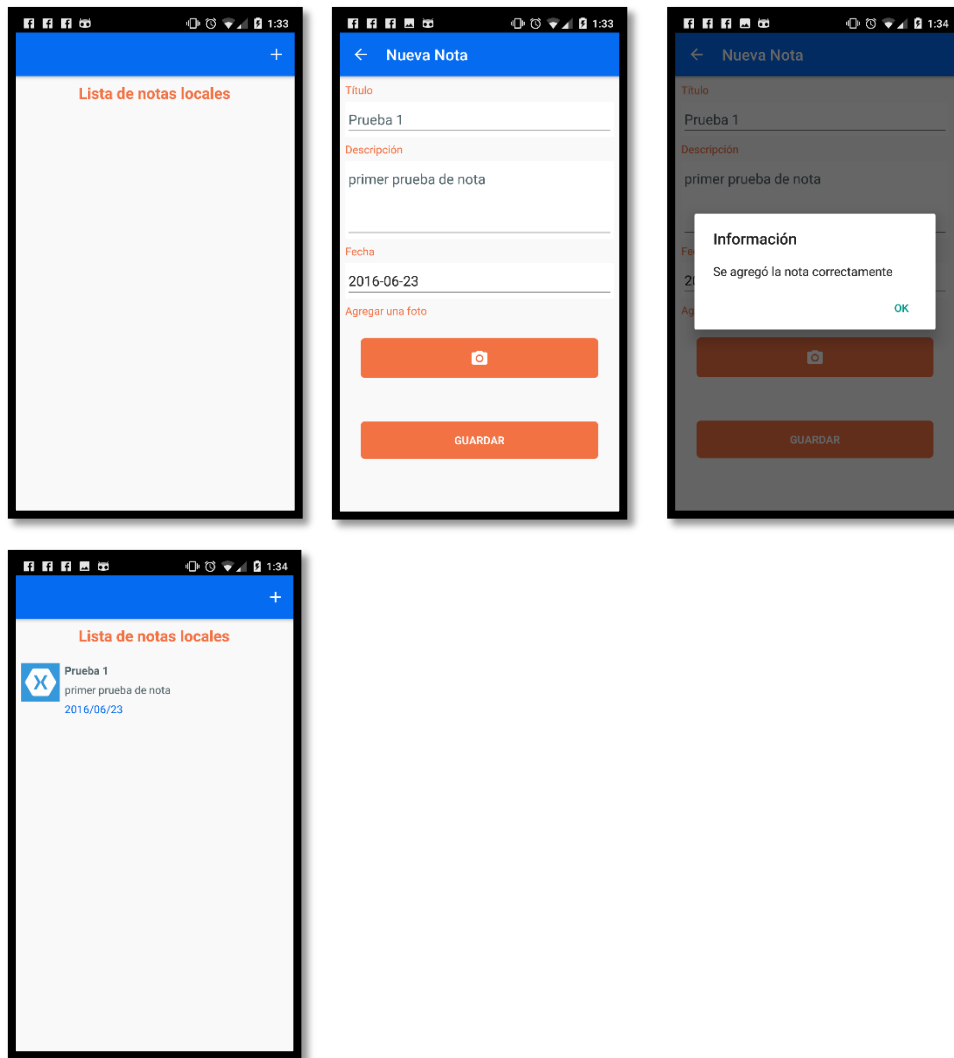
```

public partial class LocalListPage : ContentPage
{
    4 references | 0 changes | 0 authors, 0 changes
    public ObservableCollection<LocalNote> Notes { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public LocalListPage()
    {
        InitializeComponent();
        Notes = new ObservableCollection<LocalNote>();
    }

    0 references | 0 changes | 0 authors, 0 changes
    protected override async void OnAppearing()
    {
        var list = App.sqliteService.GetAllNotes();
        Notes.Clear();
        foreach (var item in list)
        {
            Notes.Add(item);
        }
        listaNotas.ItemsSource = Notes;
    }
}

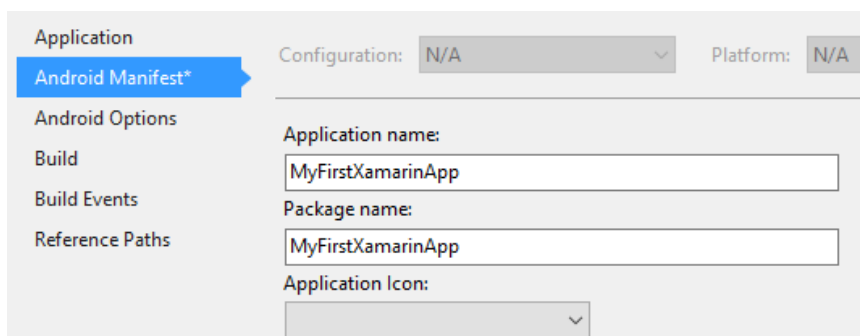
```

86. Al correr la aplicación se debe ser capaz de agregar notas y que se muestren automáticamente.

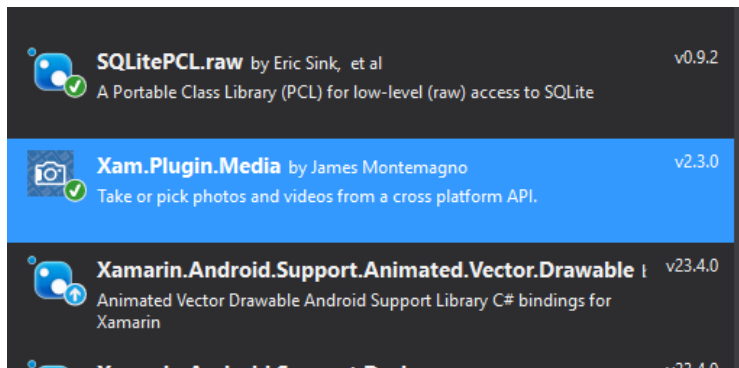


Parte diez: Accediendo a la cámara.

87. Llegó la hora de agregarle funcionalidad al botón de la cámara. Para poder usar la cámara en los dispositivos Android, diríjase a las propiedades del proyecto .Droid y en Android Manifest active las opciones de CAMERA y WRITE_EXTERNAL_STORAGE



88. Instale el Nuget Package Xam.Plugin.Media. Este paquete nos ayudará a acceder a la cámara desde el proyecto Portable de manera muy simple



89. Diríjase al proyecto de Android, en el MainActivity.cs importe el Plugin.Media. Inicialice el objeto CrossMedia antes de la llamada al App.

```
protected override async void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    global::Xamarin.Forms.Forms.Init(this, bundle);

    string dbPath = FileAccessHelper.GetLocalFilePath("Note.db3");

    await CrossMedia.Current.Initialize();

    LoadApplication(new App(dbPath));

    ActionBar.SetIcon(
        new ColorDrawable(Resources.GetColor(Android.Resource.Color.Transparent)));
}
```

90. En la página NewNotePage.xaml agregue una etiqueta Image para ver el preview de la foto, defina un x:Name para llamarla desde el Code Behind. Añádale al botón para tomar la foto la acción de Clicked.

```
<Button BackgroundColor="{StaticResource TitleColor}"
        Image="ic_action_maps_local_see.png"
        Margin="20"
        Clicked="TakePhoto"></Button>

<Image x:Name="ImagePreview"></Image>

<Button BackgroundColor="{StaticResource TitleColor}"
        Text="Guardar"
        TextColor="{StaticResource BackgroundColor}"
        FontAttributes="Bold"
        Margin="20, 30"
        Clicked="SaveNote"></Button>
```

91. En el NewNotePage.xaml.cs cree un atributo de tipo string para guardar el path de la foto.

```
public partial class NewNotePage : ContentPage
{
    private string imagePath = "";
```

92. Cree el método que activará el evento Clicked del Botón para agregar la foto. En la primera línea inicialice la variable CrossMedia, valide si la cámara está disponible. Cree un file para que guarde la foto tomada. El método CrossMedia.Current.TakePhotoAsync abre la cámara y ejecuta una acción si se toma una foto. Valide si el file es nulo (será nulo si no se toma ninguna foto). Guarde el path de la imagen en el atributo que creó anteriormente y defínale el Source al Image que creó para mostrar el preview de la imagen.

```
public async void TakePhoto(object sender, EventArgs args)
{
    await CrossMedia.Current.Initialize();

    if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported)
    {
        await DisplayAlert("No Camera", "No camera available.", "Ok");
        return;
    }

    var file = await CrossMedia.Current.
        TakePhotoAsync(new Plugin.Media.Abstractions.StoreCameraMediaOptions {
            SaveToAlbum = true
        });

    if (file == null)
    {
        return;
    }

    imagePath = file.AlbumPath;

    ImagePreview.Source = ImageSource.FromStream(() =>
    {
        var stream = file.GetStream();
        file.Dispose();
        return stream;
    });
}
```

93. En el método para guardar la nota dele al imagePath de la nueva nota el valor de del atributo que creó anteriormente. Opcionalmente puede darle el valor al atributo de "icon.png" o un ícono descargado que especifique que no tiene imagen.

```

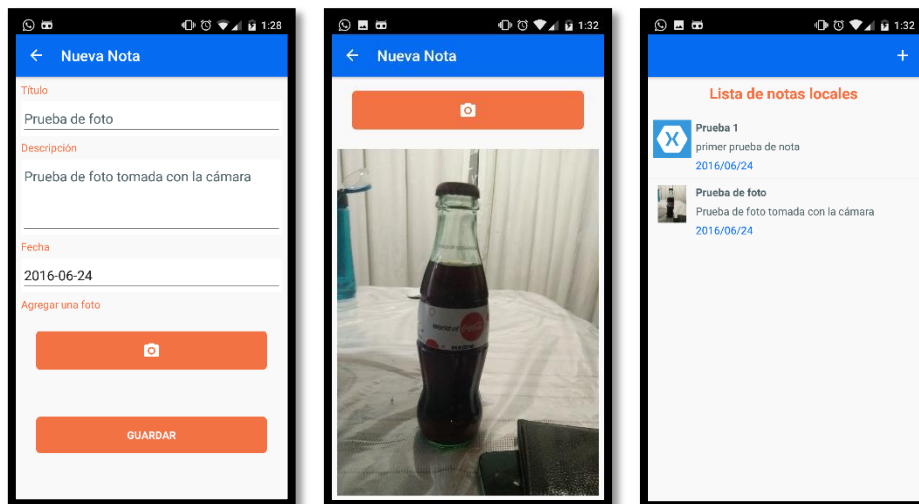
public void SaveNote(object sender, EventArgs args)
{
    if (imagePath == "")
    {
        imagePath = "icon.png";
    }
    LocalNote note = new LocalNote
    {
        Title = Title.Text,
        Description = Description.Text,
        Date = Date.Date,
        ImagePath = imagePath
    };

    App.sqliteService.AddNewNote(note);
    string message = App.sqliteService.StatusMessage;

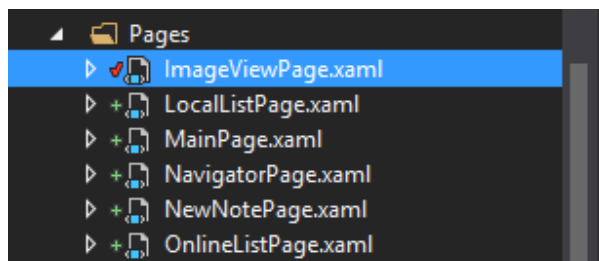
    App.Navigator.DisplayAlert("Información", message, "Ok");
}

```

94. Al correr la aplicación se podrán tomar fotos y al guardar las notas con fotos se podrá ver el thumbnail de la foto al inicio de la nota.



95. Para mostrar las imágenes en grande agregue una nueva página que se llame ImageViewPage.



96. Quite el label por defecto y agregue un Image con un x:Name

```

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
              xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
              x:Class="MyFirstXamarinApp.Pages.ImageViewPage"
              BindingContext="{Binding Main, Source={StaticResource Locator}}">

    <Image x:Name="BigImage"></Image>

</ContentPage>

```

97. Cambie el constructo de la página para que reciba el path de la imagen y asígnele el Source al Image.

```

public partial class ImageViewPage : ContentPage
{
    1 reference | 0 changes | 0 authors, 0 changes
    public ImageViewPage(string imagePath)
    {
        InitializeComponent();
        BigImage.Source = imagePath;
    }
}

```

98. En el LocalListPage agregue la acción al ListView de ItemTapped.

```

<ListView x:Name="listaNotas" ItemTapped="OpenImage"|
          ItemTemplate="{StaticResource NoteItemTemplate}"
          HasUnevenRows="True">
</ListView>

```

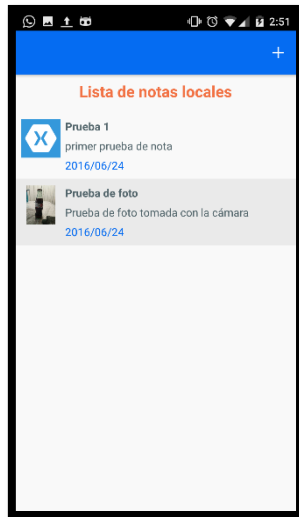
99. Agregue el método de la acción del botón. Llame al App.Navigator y agregue una nueva página de ImageViewPage y mande por parámetro el ImagePath del Item seleccionado.

```

public void OpenImage(object sender, ItemTappedEventArgs e)
{
    if (e.Item != null )
    {
        App.Navigator.PushAsync(
            new ImageViewPage(((LocalNote)listaNotas.SelectedItem).ImagePath));
    }
}

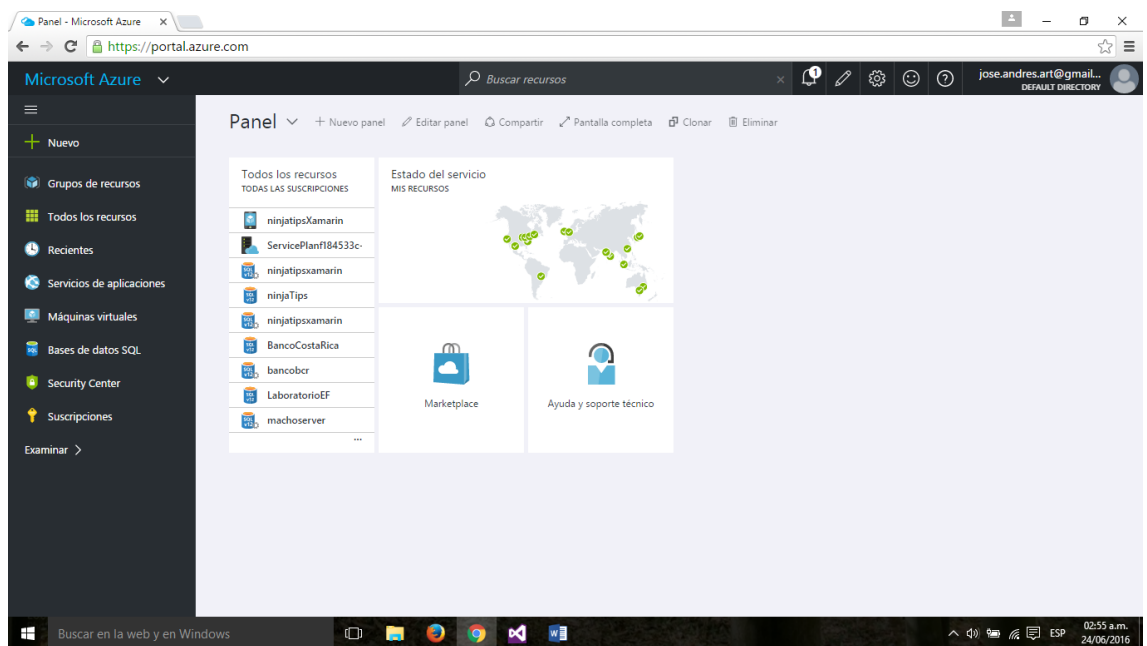
```

100. Al seleccionar elementos de la lista se deberán abrir nuevas páginas con su imagen en grande.

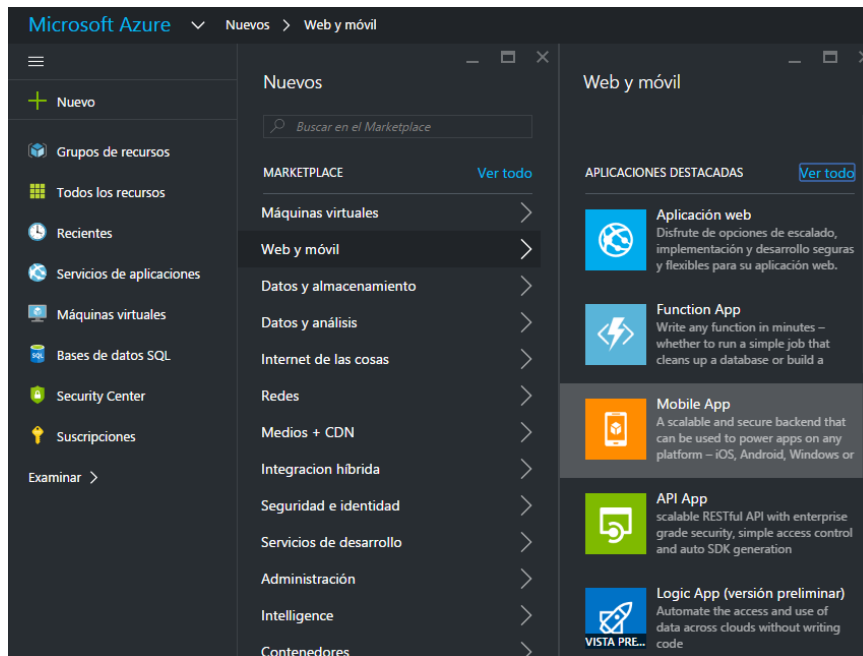


Parte once: Guardando datos en la nube.

101. Ingrese a portal.azure.com e inicie sesión con su cuenta.



102. Seleccione Nuevo, Web y Móvil, Mobile App.



103. Cree una aplicación móvil.

* Nombre de aplicación

XamarinNotesApp ✓

.azurewebsites.net

* Suscripción

DreamSpark ▼

* Grupo de recursos ⓘ

☒ Crear nuevo ☐ Usar existente

Apps ✓

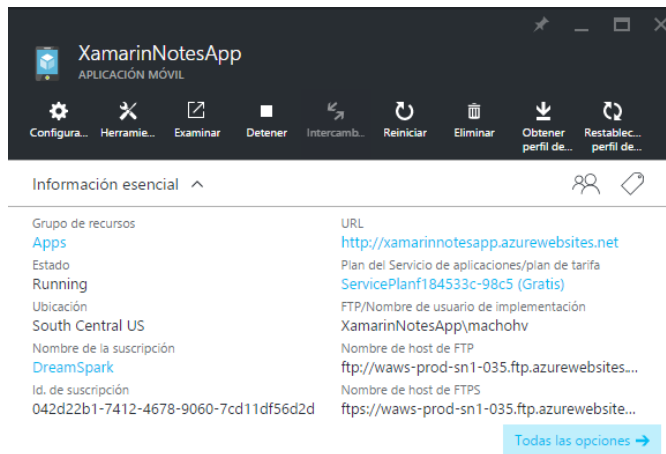
* Plan del Servicio de aplicaciones/Ubica...

ServicePlanf184533c-98c5(South ... >

☐ Anclar al panel

Crear

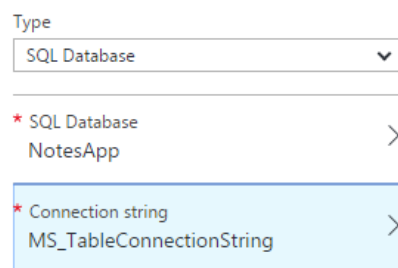
104. Abra la ventana de la aplicación



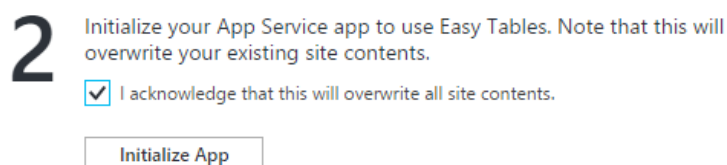
105. En Configuración, Móvil, seleccione Tablas Fáciles o Easy Tables.



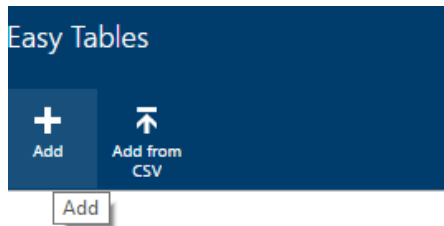
106. Configure el Easy Tables, cree una base de datos siguiendo todos los pasos.



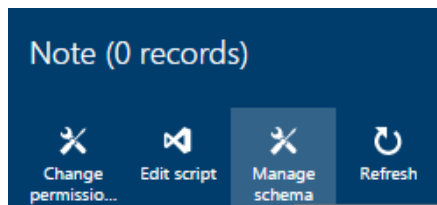
107. Una vez que haya llenado todos los campos cree la conexión. Cuando la conexión esté creado inicialice el servidor de la app.



108. De click en el + para agregar una base.



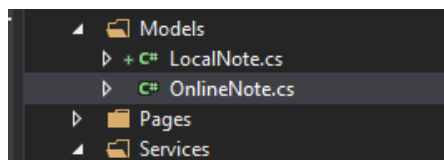
109. Llame a la tabla Note, y deje todos los permisos para acceso de anónimos. Seleccione Manage Schema para agregar y eliminar columnas.



110. Edite las columnas para que queden como se necesitan para las notas.

NAME	TYPE	IS INDEX	
id	String	true	...
Title	String	false	...
Description	String	false	...
Date	Date	false	...
ImagePath	String	false	...

111. En VisualStudio, cree una clase llamada OnlineNote en models.



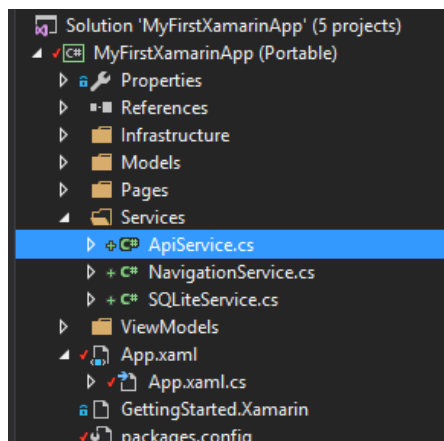
112. Agregue las misma propiedades que LocalNote pero sin las etiquetas de SQLite. Use las etiquetas JsonProperty para darle a los properties el valor de columnas con diferente nombre.

```

public class OnlineNote
{
    [JsonProperty("id")]
    public string Id { get; set; }
    public string Title { get; set; }
    public string Description { get; set; }
    public DateTime Date { get; set; }
    public string ImagePath { get; set; }
}

```

113. Cree una nueva clase en Services, llámela ApiService, esta contendrá los métodos para conectarse a la base de datos que acaba de crear.



114. Cree una tarea asíncrona para traer todas las notas. Defina de url la dirección suministrada en la aplicación móvil creada en Android, seguido de / tables / el nombre de la tabla.

```

public class ApiService
{
    public async Task<List<OnlineNote>> GetAllNotes()
    {
        using (HttpClient client = new HttpClient())
        {
            string url = "http://xamarinnotesapp.azurewebsites.net/tables/Note";
            client.DefaultRequestHeaders.Add("ZUMO-API-VERSION", "2.0.0");
            var result = await client.GetAsync(url);

            string data = await result.Content.ReadAsStringAsync();

            if (result.IsSuccessStatusCode)
            {
                return JsonConvert.DeserializeObject<List<OnlineNote>>(data);
            }
            else
            {
                return new List<OnlineNote>();
            }
        }
    }
}

```

115. Cree otra tarea para agregar una nueva nota.

```
public async Task<OnlineNote> AddNote(OnlineNote note)
{
    using (HttpClient client = new HttpClient())
    {
        string url = "http://xamarinnotesapp.azurewebsites.net/tables/Note";
        client.DefaultRequestHeaders.Add("ZUMO-API-VERSION", "2.0.0");

        string content = JsonConvert.SerializeObject(note);
        StringContent body = new StringContent(content, Encoding.UTF8, "application/json");
        var result = await client.PostAsync(url, body);

        string data = await result.Content.ReadAsStringAsync();

        if (result.IsSuccessStatusCode)
        {
            return JsonConvert.DeserializeObject<OnlineNote>(data);
        }
        else
        {
            return null;
        }
    }
}
```

116. Ya está toda la lógica para agregar una nueva nota y ver notas. Lo siguiente es unirlos con la interfaz. Se usará la misma página para agregar notas. En el App.xaml.cs agregue una propiedad estática para el ApiService y una propiedad de tipo string para guardar el tipo de nota. Esto lo utilizaremos para diferenciar si se va a agregar una nota local o una nota en línea.

```
public static ApiService apiService { get; internal set; }

3 references | 0 changes | 0 authors, 0 changes
public static string noteType { get; internal set; }

4 references | 0 changes | 0 authors, 0 changes
public App(string dbPath)
{
    InitializeComponent();

    Navigator = new NavigatorPage(new MainPage());
    sqliteService = new SQLiteService(dbPath);
    apiService = new ApiService();
    MainPage = Navigator;
}
```

117. Vaya al OnlineListPage.xaml. En el ToolbarItem envíe de CommandParameter "NewOnlineNotePage".

```
<ContentPage.ToolbarItems>
    <ToolbarItem Icon="ic_action_content_add.png"
        Command="{Binding GoToCommand}"
        CommandParameter="NewOnlineNotePage">
    </ToolbarItem>
</ContentPage.ToolbarItems>
```

118. En el método Navigate del NavigationService agregue el caso para “NewOnlineNotePage” y defínalo igual al caso anterior con la diferencia de setear en cada uno el valor al noteType del App.

```
public async void Navigate(string PageName)
{
    switch (PageName)
    {
        case "NewNotePage":
            App.noteType = "local";
            await Navigate(new NewNotePage());
            break;
        case "NewOnlineNotePage":
            App.noteType = "online";
            await Navigate(new NewNotePage());
            break;
        default:
            break;
    }
}
```

119. En el método SaveNote de NewNotePage.xaml.cs valide el tipo de nota que se va a agregar para agregar una LocalNote o una OnlineNote y guardarlas en sus bases de datos respectivas.

```
if (App.noteType == "local")
{
    LocalNote note = new LocalNote
    {
        Title = Title.Text,
        Description = Description.Text,
        Date = Date.Date,
        ImagePath = imagePath
    };

    App.sqliteService.AddNewNote(note);
    message = App.sqliteService.StatusMessage;
}
else
{
    OnlineNote note = new OnlineNote
    {
        Title = Title.Text,
        Description = Description.Text,
        Date = Date.Date,
        ImagePath = imagePath
    };

    await App.apiService.AddNote(note);
    message = "Nota agregada correctamente a la nube";
}
await App.Navigator.DisplayAlert("Información", message, "Ok");
}
```

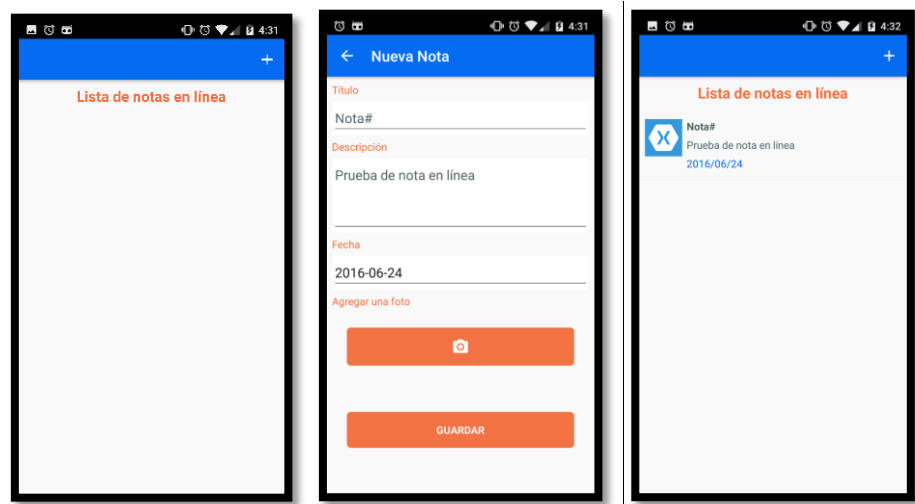
120. Edite el OnlineListPage para que sea igual al LocalListPage, con excepción al título. En el Code Behind haga solamente los cambios de LocalNote por OnlineNote y SQLiteService por ApiService

```
public partial class OnlineListPage : ContentPage
{
    4 references | 0 changes | 0 authors, 0 changes
    public ObservableCollection<OnlineNote> Notes { get; set; }
    0 references | 0 changes | 0 authors, 0 changes
    public OnlineListPage()
    {
        InitializeComponent();
        Notes = new ObservableCollection<OnlineNote>();
    }

    1 reference | 0 changes | 0 authors, 0 changes
    protected override async void OnAppearing()
    {
        var list = await App.apiService.GetAllNotes();
        Notes.Clear();
        foreach (var item in list)
        {
            Notes.Add(item);
        }

        listaNotas.ItemsSource = Notes;
    }
}
```

121. Para concluir, note como ahora se agregan notas a la base de datos en Azure y son recuperadas en la aplicación.



Note (1 record)

ID	TITLE	DESCRIPTION	DATE	IMAGEPATH	VERSION	CREATEDAT	UPDATEDAT	DELETED
c67a7e1b-8745-4c...	Nota#	Prueba de nota en ...	2016-06-24T00:00:...	icon.png	AAAAAAAAAB9E=	2016-06-24T10:31:...	2016-06-24T10:31:...	false