0.0 *****XAMARIN FORMS ******

martes, 23 de mayo de 2017 07:40 AM

James Montemagno, Evangelsita Xamarin

4 Pilares para una solución

- Satisfacer Experiencia de usuario.
  Visual studio 2017

  Opciones de desarrollo
  -Plataforma Especifica: Apple[Objective-c] Android[Java] WindowsPhone [c#]
  Tiene la mejor experiencia de usuario, pero se necesita un equipo por cada plataforma. Más costoso, más gente, más herramientas.

  -Aplicaciones Hibridas: Basadas en html5,css3. Utilizando el browser. Tiene la peor la experiencia de usuario. Fue popular por los costos económicos. Es muy limitante por que no permite acceder a las APIs y frameworks. Son lentas porque son interpretadas y no compiladas.

  -Cross Platform: [Interfaz de usuario nativas] [100% de APIs para c#] [Rendimiento Nativo]
  Al momento de compilar la aplicación, será necesario una mac y Windiwos para compilar.

  El desarrollo es basado en C#, pero se necesita utilizar las librerías específica de cada plataforma. Es necesario crear 2 interfaz para IOS y Android. Xamarin IOS y Xamarin Android.

- Escalabilidad[Cloud]:
  Azure Mobile Apps.
  Usar la aplicación cuando no tengo datos[Usar la aplicación, pero sincronizar cuando tenga datos]
  [Sync offlines] [Connect data] [Authenticate] [Push Notifications- poner notiifcations para acceder a la app, recordatorios sin necesidad]

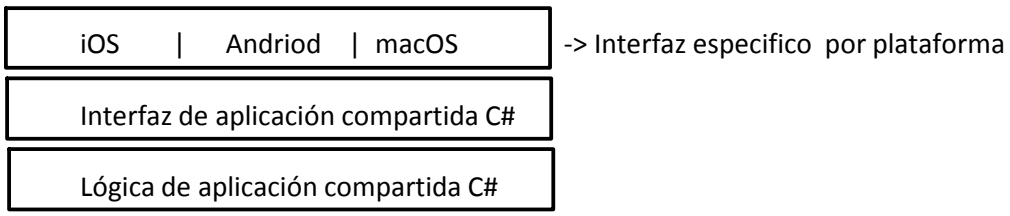
- Inteligencia: Machine learning, speech
  Cortana Intelligence Suite

1.0 Introducción

domingo, 20 de agosto de 2017 08:27 AM

¿Qué es Xamarin?
Plataforma para crear aplicaciones Andriod, IOS, macOS implmentando código C# que serán interfaces completamente nativas.

Se implementa de la siguiente manera:



¿Xamarin Forms vs Xamarin Nativo?

Nativo
- StoryBoards(iOS) y XML(Andriod)
- 75% de código compartido
- Funcionalidades Nativas de platforma

Forms
- XAML o C#
- 100% código compartido
- Sin funcionalidad nativa

2.0 Instalación

domingo, 20 de agosto de 2017 08:55 AM

1.0 Descargar Xamarin: https://www.xamarin.com/download

2.0 Si se tiene un ambiente windows es necesario, tener conectado una maquina mac concectada a la misma red para poder probaren un emulador las aplicaciones para iOS.

3.0 Xamarin Forms

martes, 22 de agosto de 2017 07:08 AM

1.0 Creamos un nuevo proyecto como Cross Platform

2.0 Seleccionamos Xamarin forms y Portable Class Library

3.0 Este tipo de proyecto con estas caracteristicas utiliza tecnología .PCL para compartir código entre las plataformas

4.0 XAML

martes, 22 de agosto de 2017 07:27 AM

Es un lenguaje de marco muy parecido a HTML o XML, que utilizaremos para deifnir la interfaz grafica que se mostrara en Andriod y iOS.

Dentro de cada página principal de interfaz se cuenta con una etiqueta llamada **<ContentPage>,** esta etiqueta solo permite un contenedor. Pero se puede seleccionar un contenedor que si permita mas de un contenedor en su interior. Es posible usar el **<StackLayout>,<Grid>**

Para ingresar algun compenente dentro de un contenedor podemos usar **<Entry>** y con el objeto **x** podemos llamar todos los atributos como (Name, placaHolder, etc..)

Para cada componente se le debe asociar un nombre el cual sera el identificador para utilizaro desde su .cs.

```
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
         xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
         xmlns:local="clr-namespace:ToDoForms"
         x:Class="ToDoForms.MainPage">
  <StackLayout VerticalOptions="Center" Margin="20">
    <Entry x:Name="userEntry" Placeholder="User Name"/>
    <Entry x:Name="PassEntry" Placeholder="Password" IsPassword="True" />
    <Button Text="Sign In" Clicked="Button_Clicked" />
    <Label x:Name="resultLabel" HorizontalOptions="Center"/>
  </StackLayout>
</ContentPage>
```

```
public partial class MainPage : ContentPage
  {
    public MainPage()
    {
      InitializeComponent();
    }

    async private void Button_Clicked(object sender, EventArgs e)
    {
      if (string.IsNullOrEmpty(userEntry.Text) || string.IsNullOrEmpty(PassEntry.Text))
      {
        resultLabel.Text = "User Name and password is required";
      }
      else
      {
        resultLabel.Text = "Login Success";
        await Navigation.PushAsync(new NewItem());
      }
    }
  }
```

5.0 SQLite

sábado, 26 de agosto de 2017 02:39 PM

SQLite es un paquete que nos permite realizar la conexión a la base de desde nuestra aplicación.

Los paquetes de SQLite se deben agregar por cada plataforma (PCL, Android, iOS),

Se ubica sobre cada proyecto clic derecho-> Administrar paquetes NuGet,y se instala el paquete sqlite-net-pcl

Para asociar cada atributo de una clase a la tabla de base de datos y sus validaciones se deben utilizar las siguientes anotaciones, pero primero se debe importar el paquete necesario para que la clase pueda utilizar las funciones y anotaciones de SQLite:

using SQLite;

[PrimaryKey, AutoIncrement]
Public int Id {set; get;}

SQLite es un base de datos que se almacenara en una carpeta del celular, por cada paltaforma debemos especificar la ruta donde se guardaran los registros.

Para Andriod
Debemos definir en la clase MainActivity cual es la ruta del archivo donde estará alamacenada la bd.

```
string nombreArchivoBD = "baseDatos.sqlite";
string ruta = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);

 string ruta_db = Path.Combine(ruta, nombreArchivoBD);

 LoadApplication(new App(ruta_db));
```

Para crear la conexión de la bd, utilizamos un componente de los paquetes de SQLite llamada SQLiteConnection, solo se puede tener una conexión de SQLite Connection a la vez por eso es necesario utlizar el componente using dentro de esta declaración.

El componente Using significa que lo que este denfinido dentro de el solo tendra ese ambito de exitenciabilidad.

```
using (SQLite.SQLiteConnection conn = new SQLite.SQLiteConnection(App.RutaDB))
{

}
```

6.0 Estilos{Resources}

lunes, 04 de septiembre de 2017 07:24 AM

Se pueden definir estilos a nivel de cada pagina de la siguiente manera, ubicando los recursos dentro de cada contectPage de cada página:

```
<ContentPage.Resources>
    <ResourceDictionary>
      <Color x:key="accentColor">#7FC719</Color>
      <Color x:key="buttonTextColor">#FFFFFF</Color>
    </ResourceDictionary>
  </ContentPage.Resources>
  <StackLayout VerticalOptions="Center" Margin="20">
    <Entry x:Name="userEntry" Placeholder="User Name" TextColor="{StaticResource accentColor}"/>
    <Entry x:Name="PassEntry" Placeholder="Password" IsPassword="True" TextColor="{StaticResource accentColor}"/>
    <Button Text="Sign In" Clicked="Button_Clicked" BackgroundColor="{StaticResource accentColor}"
```

O Se pueden definir a nivel de aplicación en el archivo app.xaml:

```
<Application.Resources>
    <ResourceDictionary>
      <Color x:key="accentColor">#7FC719</Color>
      <Color x:key="buttonTextColor">#FFFFFF</Color>
    </ResourceDictionary>
</Application.Resources>
```

Podemos definir adicionalemnte estilos que se aplicaran a cada elemento especifico sin tener que crear la propidad dentro de cada elemento, lo podemos realizar deifiniendo estilos a nivel de aplicación:

```
<Application.Resources>
    <ResourceDictionary>
        <Color x:Key="accentColor">#293275Color>
        <Color x:Key="buttonTextColor">#FFFFFFColor>

        <Style TargetType="Entry"> <Setter Property="TextColor" Value="{StaticResource
        accentColor}"/> Style> ResourceDictionary> Application.Resources>
```

Otra forma de definir los estilos asociados a cada etiquete pero mas personalizados es de la siguiente manera:

```
<Style x:Key="titleLabel" TargetType="Label">
    <Setter Property="FontSize" Value="25"/>
    <Setter Property="FontAttributes" Value="Bold"/>
 Style>
 <Style x:Key="subtitleLabel" TargetType="Label">
    <Setter Property="TextColor" Value="{StaticResource accentColor}"/>
Style>

<Label Text="Nueva tarea" Style="{StaticResource titleLabel}"/>
<Label Text="Nueva tarea" Style="{StaticResource subtitleLabel}" Grid.Row="1"/>
```

7.0 **XAMARIN Classic Andriod

martes, 05 de septiembre de 2017 10:13 PM

ANDROID

Los desarrollos se estaran trabajando principalmente en el archivo Main.axml en la pestaña de diseñador que se encuentra en l a carpeta de Layout.

Podemos arratrar y pegar los componentes en la pestaña Designar, buscamos la vista de Cuadro de herramientas se encuentra en **ver->Cuadro de herramientas**

Para agregar un campo de texto utilizamos un componente llamada Text Plain que en el código se llama EdtiText.
Podemos agregar el placeholder(nombre que indica para que es el texto) equivale a **android:hint**

ACCEDER A LOS ELEMENTOS DEL .AXML DESDE CÓDIGO C#

Desde el archivo MainActivity.cs podemos controlar los elementos creados en el archvo axml.

Por cada componente creado en la página .axml debemos tener una propiedad en la clase MainActivity

Debemos obtener la referencia del objeto creado en el .axml para manipularlos, las propiedades en la clase MainActivity las c reamos haciendo referencia al mismo tipo de dato que se definio en el .axml con el mismo nombre:

Main.axml

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Contraseña"
    android:id="@+id/passwordEditText" />
```

MainActivity.cs

```
Public class MainActivity{
      EditText passwordEditText
}
```

Para obtener el objeto se utiliza la función **FindViewById** y se utiliza la propiedad Resource la cual almacena todos los ID de los objetos definidos.

```
Protected override void OnCreate(){
      passwordEditText = FindViewById<EditText>(Resource.id.passwordEditText);
}
```

Para el caso del button, o demas elemenos que desencanden un evento, debemos crear el manejador para los eventos del click.

Se debe crear un metodo nuevo con el operador de manejador (**+=**):

```
      Protected override void OnCreate(){
       loginButton.Click += LoginButton_Click;
      }

    private void LoginButton_Click(object sender, EventArgs e)
    {
       throw new NotImplementedException();
    }
```

NOTA: Para que nuestra clase MainActivity reconozca los nuevos elementos definidos en el .axml se debe guardar los cambios y recomp ilar la solución.

SOLUCIÓN ERROR: En caso de que en algún momento no este funcionando correctamente las funcionalidades de VisualStudio, no completa las palabr as reservadas,
No reconoce las variables, se debe ir a la carpeta del proyecto y borrar el archivo .vs y volver a Correr.

8.0 **XAMARIN Classic IOS

viernes, 08 de septiembre de 2017 06:03 PM

IOS

Para IOS contamos con el archivo **LaunchScreen.storyboard** que se encuentra en la carpeta Resources. En este archivo nos da la opción de Designer para arrastrar los elementos que deseas utilizar.

Agregar caja de texto es un TextField en el cuadro de herramientos.
Se debe poner cuidado donde se ubica, ya que en IOS debemos especificar exactamente donde lo ponemos (No es tan automatico como en andriod).

En IOS no tenemos acceso al código fuente de la interfaz, solo podemos utilizar la vista de propiedades presente en el storyboard para definir las propiedades.

Para el TextField, usamos la propiedad nombre para identificar el campo y la propiedad placeholder para el nombre el marca de agua.

Para el campo password, seleccionamos la opción Secure text entry para volverlo oculto.

La pestaña de Designer tiene la opción de diseñar en varias pantallas según el dispositivo iPhone 5, 6, 7 o en modo Generic. La idea es diseñar en modo Generic y posteriormente **[definir los constraint]** para que funcione en las demas versiones.

Constraint: Se debe crear un constraint por cada elemento, se realiza, haciendo dos clic izquiedos sobre el elemento y asociar las marcas extremas a cada borde.

Lo mismo se debe realizar para cada extremo del componente.

CONTROLAR LOS COMPONENTE DEL STORYBOARD DESDE C#

Existe un archivo llamado **VIewController.cs** para controlar los componentes creados en el designar.

En este archivo no debemos crear todos los componentes como en Android, el archivo **VIewController.cs** ya conoce la existencia de todos los elementos creados en el Designar por que cuando se crea el elemento en la interfaz este automaticamete se crea el correspondiente objeto en el archico **ViewController.designar.cs.**

Por lo tanto definimos directamente los eventos de los botones:

VIewController.cs

```
Public override void ViewDidLoad(){
        Button.touchUpInside += button_TouchUpInside;
}

Viod buttin_TouchUpIndise(object sender, EventArgs e){
        //Código
}
```

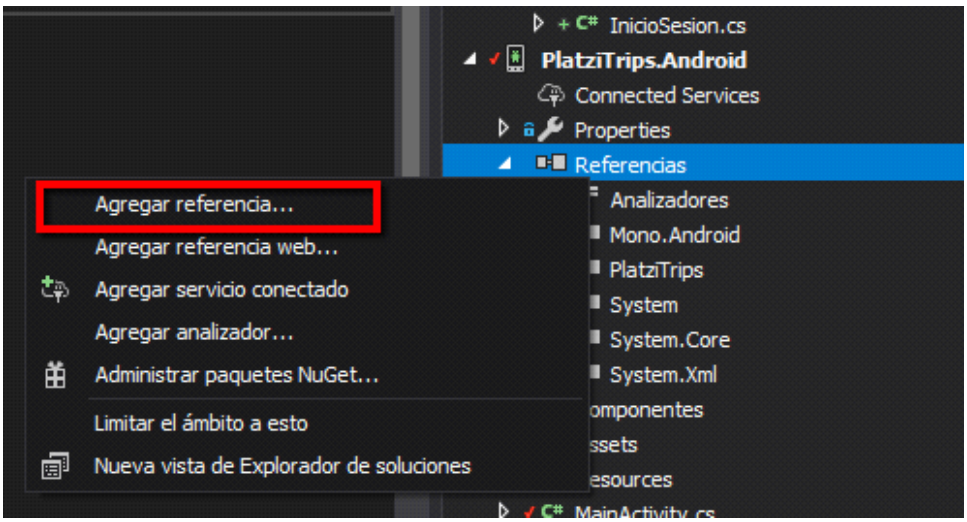
El método **TouchUpInside** es el metodo que responde al evente de clic.

9.0 Compartir Código PCL

lunes, 11 de septiembre de 2017 07:21 AM

Como tenemos dos plataformas con dos interfaces similares pero en distinto SO, la idea es crear una funcionalidad que dicho código se pueda compartir para las dos plataformas.

Luego de tener creado el código en el proyecto PCL, se debe agregar como una referencia a los proyectos IOS y Andriod.



Luego de agregada la referencia debemos usar la palabra **USING** en la parte superior de la clase donde vamos a utilizar el código del proyecto PCL. (**Using PlatziTrips.Classes**)

Nota: Tener en cuenta que la clase que deseamos compartir debe tener el modificar de acceso Public.