

# 0.0 \*\*\*\*\*XAMARIN FORMS \*\*\*\*\*

martes, 23 de mayo de 2017 07:40 AM

James Montemagno, Evangelista 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 Híbridas: 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 Windiws 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

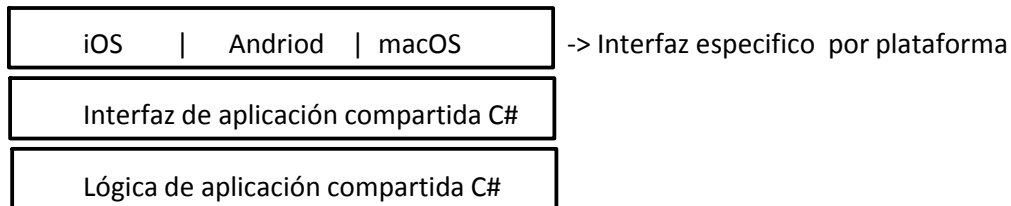
domingo, 20 de agosto de 2017

08:27 AM

## ¿Qué es Xamarin?

Plataforma para crear aplicaciones Android, iOS, macOS implementando código C# que serán interfaces completamente nativas.

Se implementa de la siguiente manera:



## ¿Xamarin Forms vs Xamarin Nativo?

### Nativo

- StoryBoards(iOS) y XML(Android)
- 75% de código compartido
- Funcionalidades Nativas de plataforma

### 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 conectada a la misma red para poder probar en 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 características 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 definir la interfaz grafica que se mostrara en Android 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 componente dentro de un contenedor podemos usar **<Entry>** y con el objeto **x** podemos llamar todos los atributos como (Name, placeholder, etc..)

Para cada componente se le debe asociar un nombre el cual sera el identificador para utilizarlo 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 utilizar 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 contextPage 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 adicionalmente estilos que se aplicaran a cada elemento especifico sin tener que crear la propiedad dentro de cada elemento, lo podemos realizar definiendo 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 etiqueta 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.xml en la pestaña de diseñador que se encuentra en la carpeta de Layout.

Podemos arrastrar 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 EditText.

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 archivo xml.

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

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

#### Main.xml

```
<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 .xml se debe guardar los cambios y recompilar la solución.

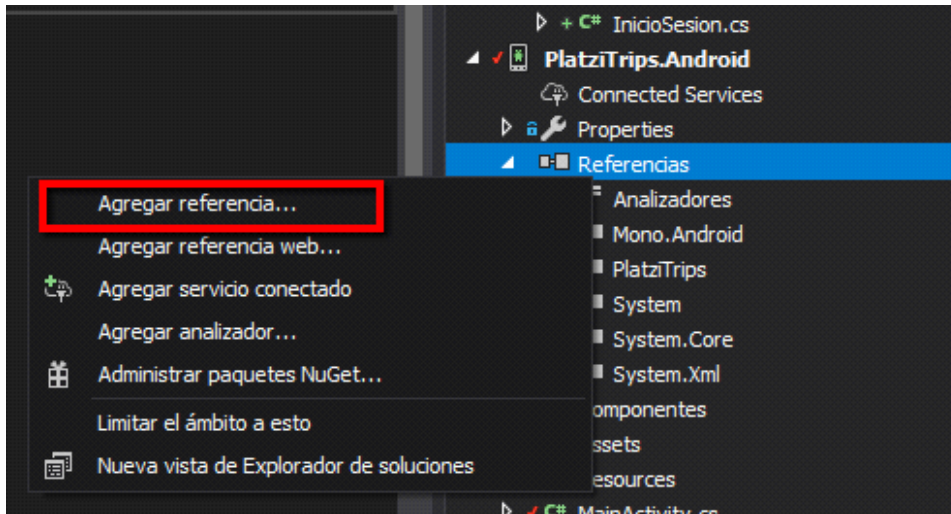
**SOLUCIÓN ERROR:** En caso de que en algún momento no este funcionando correctamente las funcionalidades de VisualStudio, no completa las palabras reservadas, No reconoce las variables, se debe ir a la carpeta del proyecto y borrar el archivo .vs y volver a Correr.

## 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.

## 10. Metodos Genericos

sábado, 16 de septiembre de 2017 12:33 PM

Podemos crear metodos genericos que reciban un una referencia al tipo de dato y el método se comporte dicamicamente según el parametro.

Por ejemplo podemos crear un metodo generico de inserción a la base de datos, según el tipo que llegue en el paramtro de la siguiente manera:

```
Public bool static insert<T>(ref T item, string ruta_db)
{
    using (SQLite.SQLiteConnection conn = new SQLite.SQLiteConnection(ruta_db))
    {
        conn.CreateTable<T>();

        if (conn.Insert(item) > 0)
            estado_insercion = true;
    }

    return estado_insercion;
}
```

# 11. Agregar nueva vista

sábado, 16 de septiembre de 2017

12:43 PM

En Adroid:

Para agregar una vista a la aplicación en la plataforma de android debemos ubicarnos en la carpeta **layout** dentro de Reources y agregar un nuevo elemento .xml. Por cada elemento de vista agregado, debemos crear el correspondiente archivo Activity, por lo tanto nos ubicamos en la raiz del proyecto Android y agregamos un nuevo elemento Activity.cs por convención la idea es que inicie con el mismo nombre del layout.

Si los elemenos que vamos agregar en una vista son muy grandes de tal manera que sobrepase el tamaño de la pantalla entonces cambiamos el contenedor por defecto a **ScrollView**

**<ScrollView>**

Android:layout\_width="match\_parent" // esta propiedad indica que el tamaño es igual al del padre en ancho

Android:layout\_height="match\_parent" // esta propiedad indica que el tamaño es igual al del padre en alto

**<LienarLayout>**

Android:orientation="vertical" // propiedad que indica que todos los elementos agregados seran apildos

<EditText android:id="@+id/lugarEditTex" // todas los id comienzan por @+id

Android:layout\_height="wrap\_content"/> //propiedad indica una altura definida

**</LienarLayout>**

**</ScrollView>**

## 12. Vincular la vista axml y la clase Activity

sábado, 16 de septiembre de 2017 01:44 PM

Para poder controlar los componentes de la vista desde la clase activity, debemos establecer la propiedad **SetContentView**:

```
SetContentView(Resource.Layout.nuevaClaseActivity);
```

Ahora debemos definir la variables que van a controlar los elemenetos de la vista:

```
EditText variableText;
```

```
variableText = findViewById<EditText>(Resources.Id.variableText);
```

Luego de tener los elementos mapeados, y dentro de esta vista debemos guardar en la base de datos, podemos definir el controlador para el boton y programar el boton.

```
guardarViajeButton = findViewById<Button>(Resource.Id.guardarViajeButton);  
guardarViajeButton.Click += GuardarViajeButton_Click;
```

```
private void GuardarViajeButton_Click(object sender, EventArgs e)  
{  
    var nuevoViaje = new Viaje() //Manera de inicializar un objeto  
    {  
        Nombre = lugarViajeEdtiText.Text,  
        FechaInicio = fechaIdaDatePicker.DateTime,  
        FechaRegreso = fechaRegresoDatePicker.DateTime  
    };  
  
    if (DataBaseHelper.Insertar(ref nuevoViaje, obtenerRutaBaseDatos()))  
        Toast.MakeText(this, "Insertado Correctamente", ToastLength.Short).Show();  
    else  
        Toast.MakeText(this, "Error al guardar", ToastLength.Short).Show();  
}
```

**Nota:** El elemento Toast se utiliza para mostrar un mensaje en la parte inferior de la pantalla, tiene como parametro el contexto que es la clase en la que estamos, el mensaje y tiempo en pantalla.

## 13.0 Crear Navegacion

sábado, 16 de septiembre de 2017 04:20 PM

Para crear la navegacion en Android, debemos utilizar el componente Intent el cual nos permite ir a otra vista o incluso dirigirnos a otra aplicación como por ejemplo Maps.

Debemos inicializar el componente Intent con el contexto y el tipo al que vamos a dirigirnos:

```
Intent intent = new Intent(this, typeof(claseActivity));  
StartActivity(intent);
```

## 14. Leer Listas y Agregar ToolBar

martes, 19 de septiembre de 2017 09:40 PM

Para leer la información de la base de datos y presentarla en forma de lista en una vista, creamos un nuevo activity en el proyecto Android, modificamos la herencia por ListActivity

```
Public class ListaViajesActivity: ListActivity
{
```

```
}
```

Ya por defecto ListActivity tiene su propia vista para listar los elementos en celdas, no es necesario crear un axml, lo que debemos hacer es obtener los elementos de la base de datos.

### Metodo para leer los datos de la base de datos:

En el código del proyecto pcl, donde tenemos la clase DataBaseHelper, debemos crear el metodo para consultar los datos:

```
public static List<Viaje> LeerViajes(string ruta_db)
{
    List<Viaje> viajes = new List<Viaje>();

    using (var conexion = new SQLite.SQLiteConnection(ruta_db))
    {
        viajes = conexion.Table<Viaje>().ToList();
    }

    return viajes;
}
```

Luego de tener el método para la consultas de datos, en la clase ListaViajesActivity debemos utilizar este método para traer la info y mostrarla en la vista, para esto necesitamos un adaptador que me convierta los datos en una lista para mostrar en la vista:

```
List<Viaje> viajes;
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);

    // Create your application here
    string nombreArchivo = "viajes_db.sqlite";
    string rutaCarpeta = System.Environment.GetFolderPath(System.Environment.SpecialFolder.Personal);
    string ruta = Path.Combine(rutaCarpeta, nombreArchivo);

    viajes = new List<Viaje>();
    viajes = DataBaseHelper.LeerViajes(ruta);

    var arrayAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleListItem1, viajes>();
    ListAdapter = arrayAdapter;
}
```

Esta implementación esta en el onCreate de la página la cual se ejecuta cuando se crea por primera vez la página de ListaViajesActivity, si se realiza alguna actualización nuevos datos, el método onCreate no se ejecuta nuevamente, por lo tanto debemos sobre escribir le método **onRestart**

```
protected override void OnRestart()
{
    base.OnRestart();

    viajes = new List<Viaje>();
    viajes = DataBaseHelper.LeerViajes(MainActivity.ObtenerRutaBaseDatos());
    var arrayAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleListItem1, viajes>();
    ListAdapter = arrayAdapter;
}
```

La propiedad ListAdapter que tiene los registros de viajes, para mostrar la información por defecto utiliza el metodo toString que tienen todos los objetos, por lo tanto para mostrar la información como deseamos, debemos sobre escribir el método toString() de la clase **Viaje**

**NOTA:** El método anterior es una forma de mostrar una lista en una pantalla con una plantilla de listarListas.

A continuación vamos realizar la misma ventana implementando un interfaz normal heredando de la clase Activity

#### \*\*\*\*\* INTERFAZ PARA LISTAR ELEMENTOS DE UNA LISTA \*\*\*\*\*

En este momento tenemos la clase **ListaViajeActivity** heredando de **ListActivity**, lo que vamos hacer es cambiar dicha herencia por **Activity** y crear el respectivo **.axml** en la carpeta **Resources->Layout**

- 1- Crear archivo .axml en Resources->Layout-> Diseño.
- 2- Agregar por medio del cuadro de herramientas un ToolBar y un ListView.
- 3- Cambiar la herencia de ListaViajeActivity por Activity.
- 4- Asociar la vista del axml con el .c por medio del setContentView(Resource.Layout.id)
- 5- Crear como atributos de la clase las propiedades que deseamos controlar del axml (toolbar y listview)
- 6- Por medio del FindById Asociar los objetos de la vista.
- 7- Asignar a la propiedad adapter del ListVlew la lista de objetos de la BD para ser listada.
- 8- Agregamos el ToolBar creado al metodo SetActionBar, y le definimos el titulo que mostrara en la app.

#### AGREGAR FUNCIONALIDAD(BOTON) AL ACTIONBAR

Vamos agregar un icono al ToolBar para que cumple la función de regresarnos a la vista NuevoViaje.axml.

- |  |  |
|--|--|
| <p>1- Debemos elegir el Icono que utilizaremos, para esto debemos tener el mismo icono en diferentes resoluciones para que este varie dependiendo de la resolución de la pantalla, utilizamos la siguiente página la opción <b>Generic Icon Generator</b> <a href="https://romannurik.github.io/AndroidAssetStudio/index.html">h</a></p> | <p><a href="https://romannurik.github.io/AndroidAssetStudio/index.html">https://romannurik.github.io/AndroidAssetStudio/index.html</a></p> |
|--|--|
- 2- Luego de descargar el icono en las resoluciones, se debe agregar a cada carpeta.
  - 3- Debemos crear una carpeta menu dentro de Resource para definir un archivo con las características del boton. **Resources->menu**
  - 4- Dentro de menu creamos el archivo agregar.xml, el cual es un archivo .xml unico por cada elemento.
  - 5- En este archivo vamos a definir la estructura de menu con sus propiedades:



<!--ShowAsAction Esta propiedad es para indicar que si hay espacio en el toolbar se muestre el boton, sino se puede agregar a los tres puntos de menu de abreviación -->

```
<!--For all properties see: http://developer.android.com/reference/android/support/design/appbar/AppBar.html-->
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      android:layout="@layout/menubar"
      app:menu="menu_agregar">
    <item
        android:id="@+id/menu_agregar"
        android:icon="@drawable/ic_add"
        android:showAsAction="ifRoom"
        android:title="Agregar"
    />
</menu>
```

- 6- Para dar funcionalidad al icono creado en el toolbar, se deben sobrescribir dos metodos en la clase **ListaViajesActivity.cs** uno para crear la opciones de menu y otro para responder al evento

```
public override bool OnCreateOptionsMenu(IMenu menu)
{
    MenuInflater.Inflate(Resource.Menu.agregar, menu);

    return base.OnCreateOptionsMenu(menu);
}
```

```
public override bool OnOptionsItemSelected(IMenuItem item)
{
    if ("Agregar".Equals(item.TitleFormatted.ToString()))
    {
        Intent intent = new Intent(this, typeof(NuevoViajeActivity));
        StartActivity(intent);
    }

    return base.OnOptionsItemSelected(item);
}
```

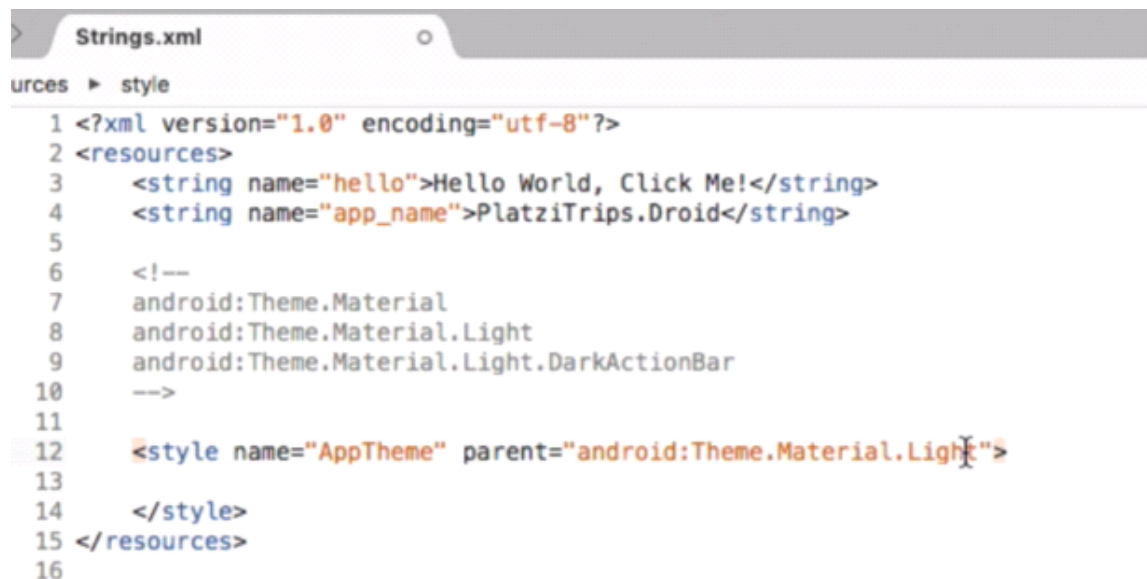
# 15. Material Design

domingo, 24 de septiembre de 2017 03:06 PM

El diseño proporcionado por google para para crear interfaces con mejores graficas y colores.

En el archivo **Strings.xml** ubicado en el proyecto Andriod en la carpeta **Resources->values->String.xml** podemos definir las propiedades, estilos que nos permitan establecer los colores a otras propiedades que modifican la forma en que se ve la app.

Para crear estilos en andriod, debemos crear una etiqueta **<style>** asociarle un nombre **name="AppTheme"** y un estilo padre que será un estilo de referencia ya creado por google que podemos utilizar y modificar que estará especificado en la propiedad **parent**.



```
> Strings.xml
urces ▶ style
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="hello">Hello World, Click Me!</string>
4   <string name="app_name">PlatziTrips.Droid</string>
5
6   <!--
7   android:Theme.Material
8   android:Theme.Material.Light
9   android:Theme.Material.Light.DarkActionBar
10  -->
11
12   <style name="AppTheme" parent="android:Theme.Material.Light">
13
14   </style>
15 </resources>
16
```

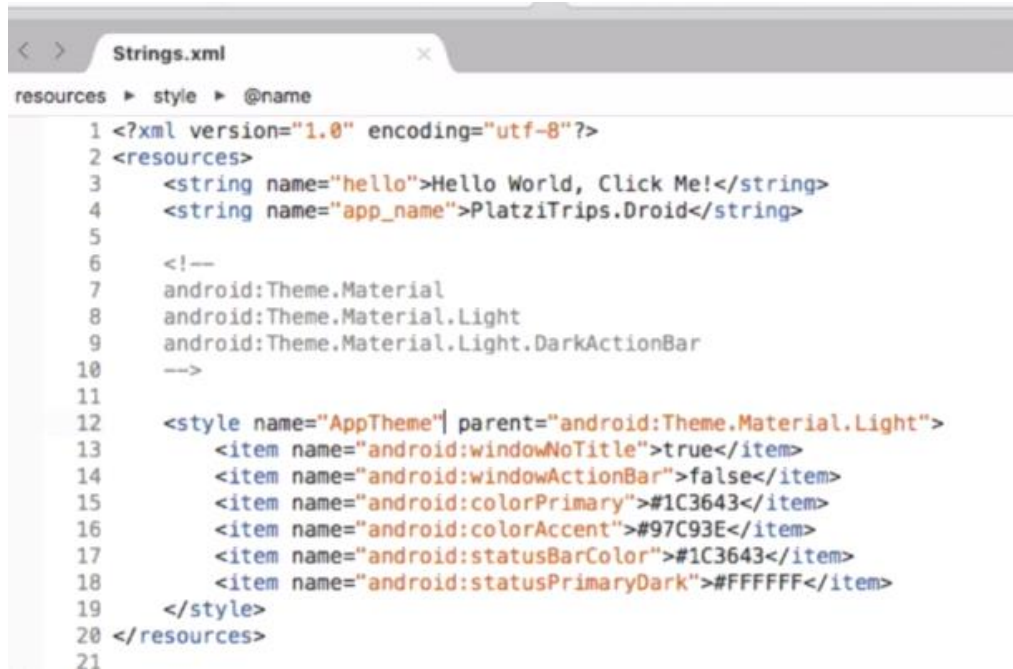
Las propiedades las podemos definir de la siguiente manera:

Colorprimary: color principal

colorAccent: color de acento o secundario

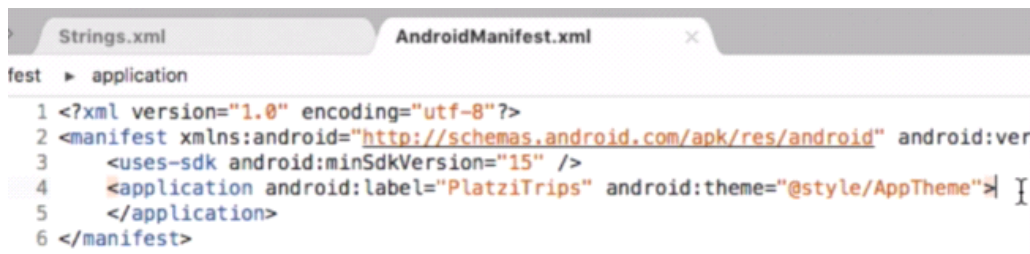
statusBarColor: color de barra de estado, (donde esta la imagen de la señal del wifi)

statusPrimaryDark: color cuando el tema es completamente oscuro.



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <string name="hello">Hello World, Click Me!</string>
4   <string name="app_name">PlatziTrips.Droid</string>
5
6   <!--
7     android:Theme.Material
8     android:Theme.Material.Light
9     android:Theme.Material.Light.DarkActionBar
10  -->
11
12   <style name="AppTheme" parent="android:Theme.Material.Light">
13     <item name="android:windowNoTitle">true</item>
14     <item name="android:windowActionBar">false</item>
15     <item name="android:colorPrimary">#1C3643</item>
16     <item name="android:colorAccent">#97C93E</item>
17     <item name="android:statusBarColor">#1C3643</item>
18     <item name="android:statusPrimaryDark">#FFFFFF</item>
19   </style>
20 </resources>
21
```

Luego de definir el estilo debemos establecer que vamos a utilizar este tema durante toda la aplicación para esto debemos modificar el Manifest de la aplicación de Android ir al archivo **properties->AndroidManifest.xml**, y definir sobre la propiedad application la definición del tema:



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" android:versionName="1.0">
3   <uses-sdk android:minSdkVersion="15" />
4   <application android:label="PlatziTrips" android:theme="@style/AppTheme">
5   </application>
6 </manifest>
```

Ahor para poder utilizar Material Design debemos realizar algunas configuraciones a nivel del proyecto de Android:

- 1- Clic derecho propiedades->Manifiesto de Android-> Version de Android mínima: level 21
- 2- Android Application->Versión de Android destino: API 23
- 3- General-> Plataforma Destino->Usar la última versión

## 16. Servicios Rest

martes, 17 de octubre de 2017 10:11 PM

Vamos a utilizar una API publica expuesta en la siguiente página:

<https://developer.foursquare.com>

Estando en la página nos ubicamos en los endpoints para mirar los servicios expuestos y los ejemplos:

<https://developer.foursquare.com/docs/api/venues/search>

Para iniciar el consumo de los servicios debemos adecuar el ambiente de Visual Studio para la aplicación pueda acceder a Internet, para esto debemos descargar los siguientes paquetes por el administrador de Nuget y asociarlos a los 3 proyectos:

-Microsoft.Net.Http [Conexión Internet]

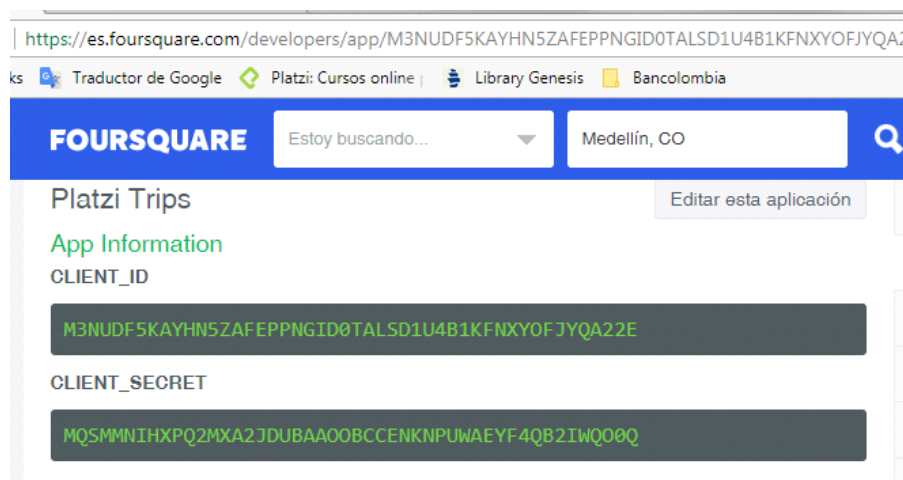
-NewtonSoft.Json [Paquete para deserializar las estructuras json - pasar json a c#]

**NOTA:**En caso de que se presente algún problema descargando el Microsoft.Net.Http, se recomienda instalar primero el **Micrisoft.Bcl.Build**, luego intentar nuevamente el **Microsoft.Net.Http**.

### USO de Funciones de Foursquare

Para usar las funciones expuestas debemos validar que condiciones tiene y que forma de autenticación:

Para las app que no requieren autenticación, se debe crear una app nuevo dentro de foursquare y generar el **Client\_ID** y **Client\_Secret**:



Vamos luego a crear la clase Constantes dentro de una carpeta llamada Helper que crearemos en la carpeta classes del proyecto PCL. En esta clase vamos a guardar como constantes estos dos datos y la URL completa para la invocación del servicio:

```
public class Constantes
{
    public const string CLIENT_ID = "M3NUDF5KAYHN5ZAFEPNGID0TALSD1U4B1KFNXYOFJYQA22E";
    public const string CLIENT_SECRET = "MQSMNNIHX PQ2MXA2JDUBAAO0BCCENKNPUWAEYF4QB2IWQ00Q";

    public static string ObtenerUrlCategorias()
    {
        return $"https://api.foursquare.com/v2/venues/categories?client_id={CLIENT_ID}&client_secret={CLIENT_SECRET}&v={DateTime.Now.ToString("yyyyMMdd")}";
    }
}
```

Ahora vamos a consumir el metodo y a guardarlo en un objeto C#, para realizar esto debemos crear un Deserializador de Json:

- La clase la creamos con la ayuda de la página <https://jsonutils.com/> y copiando una URL tal como la invocariamos, esta página nos proporciona las clases que debemos utilizar:

```
namespace PlatziRips.Classes
{
    public class Categorias
    {
        public Response response { get; set; }
    }

    public class Response
    {
        public List<Category> categories { get; set; }
    }

    public class Category
    {
        public string id { get; set; }
        public string name { get; set; }
        public string shortName { get; set; }
    }
}
```

- Luego de tener la clase con la estructura del json que queremos deserializar, debemos crear una clase para realizar las consultas a la API:

```
public async Task<List<Category>> ObtenerCategorias()
{
    List<Category> listCategorias = new List<Category>();
    string url = Constantes.ObtenerUrlCategorias();

    using (HttpClient client = new HttpClient() )
    {
        var foursquareCategory = await client.GetStringAsync(url);
        Categorias categorias = JsonConvert.DeserializeObject<Categorias>(foursquareCategory);

        listCategorias = categorias.response.categories;
    }

    return listCategorias;
}
```

- Ahora vamos a crear una vista donde expondremos la información consultada de los servicios: Para esto agregamos un nuevo axml en **Layout del proyecto Android**. Dentro de los componente del nuevo Layout debemos tener un ListView para listar la información del servicio.
- Ahora debemos crear la clase Activity que contendra el funcionamiento del axml creado anteriormente.

## 17. Pasar Parametros entre activities

miércoles, 25 de octubre de 2017 10:14 PM

Primero, el evento que desencadenará la acción de llamar otra vista y pasar los parametros será la selección de un objeto de la lista de un List View:

-Para esto debemos crear un manejador del evento de click, el cual sera:

**ListView.ItemClick += ListaViajesListView\_ItemClick;**

```
List<Viaje> viajes;
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);

    SetContentView(Resource.Layout.ListaViajes);

    viajesToolbar = FindViewById<Toolbar>(Resource.Id.ViajesToolbar);
    listaViajesListView = FindViewById<ListView>(Resource.Id.ListaViajesView);
    listaViajesListView.ItemClick += ListaViajesListView_ItemClick;
```

Dentro del metodo **ListaViajesListView\_ItemClick** (sender , args e) podemos detectar la posición seleccionada con el parametro **e**.

Dentro de este metodo debemos crear un objeto Bundle el cual sera el que pasaremos al otro activity por medio de un bundle el cual funciona como un map.

```
private void ListaViajesListView_ItemClick(object sender, AdapterView.ItemClickEventArgs e)
{
    var ciudad_seleccionada = viajes[e.Position];

    Intent intent = new Intent(this, typeof(DetallesViajeActivity));

    var bundle = new Bundle();
    bundle.PutString("nombre_ciudadSeleccionada", ciudad_seleccionada.Nombre);
    bundle.PutString("fechaIda_ciudadSeleccionada", ciudad_seleccionada.FechaInicio.ToString("MMM dd"));
    bundle.PutString("fechaRegreso_ciudadSeleccionada", ciudad_seleccionada.FechaRegreso.ToString("MMM dd"));
    bundle.PutInt("id_ciudadSeleccionada", ciudad_seleccionada.Id);

    intent.PutExtras(bundle);
    StartActivity(intent);
}
```

Ahora desde el activity que se invoco podemos capturar la información haciendo uso del bundle enviado como parametro.

En el metodo OnCreate capturamos el bundle y recorremos todos sus parametros con el objeto Intent.Extras

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

    SetContentView(Resource.Layout.DetallesViaje);

    detallesToolbar = FindViewById<Toolbar>(Resource.Id.detallesToolbar);
    fechaTextView = FindViewById<TextView>(Resource.Id.fechaTextView);
    ciudadTextView = FindViewById<TextView>(Resource.Id.ciudadTextView);
    detalleListView = FindViewById<ListView>(Resource.Id.detalleListView);

    SetSupportActionBar(detallesToolbar);

    nombreCiudadSeleccionada = Intent.Extras.GetString("nombre_ciudadSeleccionada");
    fechaIda = Intent.Extras.GetString("fechaIda_ciudadSeleccionada");
    fechaRegreso = Intent.Extras.GetString("fechaRegreso_ciudadSeleccionada");
    idCiudadSeleccionada = Intent.Extras.GetInt("id_ciudadSeleccionada");
}
```

Ya con los valores de la pagina anterior podemos asignarlos a la vista.

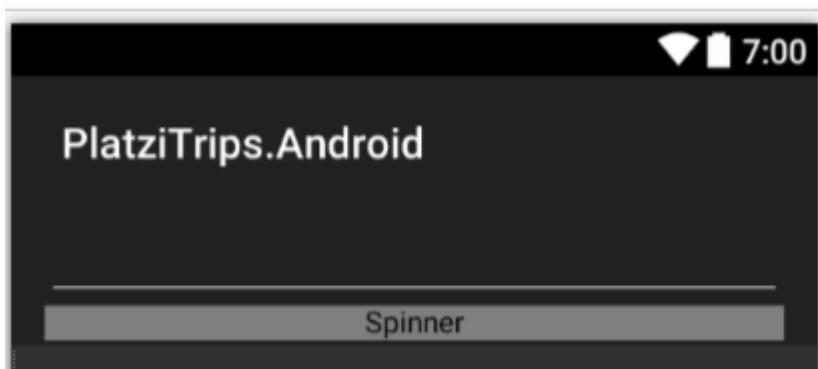
## 18. Llenar Spinner[ComboBox]

sábado, 28 de octubre de 2017 08:10 AM

Vamos a crear una página para consumir un servicio y filtrar de los resultados el lugar deseado por medio de un Spinner:

Creemos un archivo axml en Layout y le asignamos un toolbar, un PlainText, Spinner.

Deben estar ubicados verticalmente[Usar para esto el LinearLayout - android.orientation="vertical"]



```
<Toolbar
    android:background="?android:attr/colorPrimary"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/nuevoLugarToolbar">
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/filtroEditText" />
        <Spinner
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/CategoriaSpinner" />
    </LinearLayout>
</Toolbar>
```

Luego creamos un nuevo activity para controlar la vista de anterior.

Dentro del nuevo activity el cual será el controlador de la vista que contiene el spinner, vamos a realizar la consulta al metodo de foursquare para cargar el spinner con las categorias que devuelva el servicio.

```

EditText filtroEditText;
Spinner categoriaSpinner;
ListView venuesListView;
Toolbar nuevoLugarToolbar;

List<Category> categorias;

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

    SetContentView(Resource.Layout.NuevoLugar);

    filtroEditText = FindViewById<EditText>(Resource.Id.filtroEditText);
    categoriaSpinner = FindViewById<Spinner>(Resource.Id.CategoriaSpinner);
    venuesListView = FindViewById<ListView>(Resource.Id.venuesListView);
    nuevoLugarToolbar = FindViewById<Toolbar>(Resource.Id.nuevoLugarToolbar);

    FoursquareHelper helper = new FoursquareHelper();
    categorias = await helper.ObtenerCategorias();

    var spinnerAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleSpinnerItem, categorias>;
    categoriaSpinner.Adapter = spinnerAdapter;
}

```

Debemos usar la clase FoursquareHelper para traer las categorías, como el método ObtenerCategorias es un método Async y retorna inicialmente un Task, debemos poner la palabra await para indicar que debe esperar a que termine y en ese caso si retornar un tipo de dato List<Category>.

Se define el Adapter y se asigna el resultado del método.

Finalmente para que se muestre correctamente la información en el spinner, debemos sobrescribir el método ToString() en la clase Category retornando solamente el name.

```

public class Category
{
    public string id { get; set; }
    public string name { get; set; }
    public string shortName { get; set; }

    public override string ToString()
    {
        return name;
    }
}

```

## CONSULTAR LUGARES DE ACUERDO A LA CATEGORIA SELECCIONADA EN EL SPINNER

Para retornar los lugares de una categoría seleccionada, debemos identificar cuando se selecciona un elemento dentro del spinner y luego invocar al método de Foursquare que nos retorna los lugares.

- 1- Crear el controlador del evento que identifica cuando se selecciona una nueva categoría, para esto vamos a crear en el método onCreate la siguiente línea:

```

a. categoriaSpinner.ItemSelected += CategoriaSpinner_ItemSelected;

private void CategoriaSpinner_ItemSelected(object sender, AdapterView.ItemSelectedEventArgs e)
{
    throw new NotImplementedException();
}

```



- 2- Ahora debemos almacenar la categoria seleccionada usando la lista que tiene todos los datos y el parametro e del evento el cual cuenta con la propiedad posicion.
- 3- Luego pasamos el parametro de la ciudad seleccionada por medio del bundle para poder invocar el metodo Foursquare que trae los lugares.
- 4- Luego creamos un adapter para asignar un arrayAdapter con algun tipo de vista al componente ListView que me mostrara los lugares:

a.

```
private async void CategoriaSpinner_ItemSelected(object sender, AdapterView.ItemSelectedEventArgs e)
{
    var categoriaSeleccionada = categorias[e.Position];

    FoursquareHelper helper = new FoursquareHelper();
    listVenues = await helper.ObtenerLugares(nombreCiudadSelecccionad, categoriaSeleccionada.id);

    var venuesAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleListItem1, listVenues>;
    venuesListView.Adapter = venuesAdapter;
}
```

## 19. Crear Filtro para ListView

miércoles, 01 de noviembre de 2017 09:48 PM

Vamos a crear un filtro a partir del uso del componente Plain Text, a medida que el usuario escriba palabras se va realizando la búsqueda de acuerdo a las palabras registradas.

- 1- Primero debemos crear el controlador del evento que realizara la búsqueda cuando cambie el texto en el componente PlainText, para realizar esto vamos a la clase VenuesActivity al metodo onCreate y creamos el siguiente controlador:

```
filtroEditText.TextChanged += FiltroEditText_TextChanged;
```

```
private void FiltroEditText_TextChanged(object sender, Android.Text.TextChangedEventArgs e)
{
    var listaFiltrada = listVenues.Where(v => v.name.ToLower().Contains(filtroEditText.Text.ToLower())).ToList();

    var venusAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleListItem1, listaFiltrada>;
    venuesListView.Adapter = venusAdapter;
}
```

\*\*\*\*\* AGREGAR MENU DE GUARDAR \*\*\*\*\*

Ahora vamos a agregar un boton al toolbar que nos permita guardar los elementos seleccionados de la lista (Como lo vimos en la sección 14):

- 1- Descargamos los iconos y los agregamos las carpetas correspondientes.
- 2- Creamos guardar.xml en la carpeta menu.
- 3- Luego debemos ir a la clase VenuActivity en el metodo onCreate establecer(set) el toolbar al actionBar y poner un titulo.
- 4- Luego debemos sobrescribir los metodos onCreateOptionsMenu [para agregar la nueva opción de menu] y onOptionsItemSelected.

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

    SetContentView(Resource.Layout.NuevoLugar);

    filtroEditText = FindViewById<EditText>(Resource.Id.filtroEditText);
    categoriaSpinner = FindViewById<Spinner>(Resource.Id.CategoriaSpinner);
    venuesListView = FindViewById<ListView>(Resource.Id.venuesListView);
    nuevoLugarToolbar = FindViewById<Toolbar>(Resource.Id.nuevoLugarToolbar);

    FoursquareHelper helper = new FoursquareHelper();
    categorias = await helper.ObtenerCategorias();

    var spinnerAdapter = new ArrayAdapter<this, Android.Resource.Layout.SimpleSpinnerItem, categorias>;
    categoriaSpinner.Adapter = spinnerAdapter;

    categoriaSpinner.ItemSelected += CategoriaSpinner_ItemSelected;
    nombreCiudadSeleccionada = Intent.Extras.GetString("nombre_ciudadSeleccionada");

    filtroEditText.TextChanged += FiltroEditText_TextChanged;

    SetActionBar(nuevoLugarToolbar);
    ActionBar.Title = "Elegir Destinos";
}
```

```

public override bool OnCreateOptionsMenu(IMenu menu)
{
    MenuInflater.Inflate(Resource.Menu.guardar, menu);

    return base.OnCreateOptionsMenu(menu);
}

b. public override bool OnOptionsItemSelected(IMenuItem item)
{
    if ("Guardar".Equals(item.TitleFormatted.ToString()))
    {
        // ...
    }

    return base.OnOptionsItemSelected(item);
}

```

\*\*\*\*\* CREAR NUEVA TABLA EN BD PARA AGREGAR SITIOS SELECCIONADOS \*\*\*

Ahora dentro del metodo que identifica que seleccionamos la opción guardar, debemos crear una tabla para almacenar los datos seleccionados:

- 1- Antes de crear la tabla debemos crear la clase que maneja los datos de dicha tabla, esta clase debe contener las propiedades para reconocer un lugar de Interes:

```

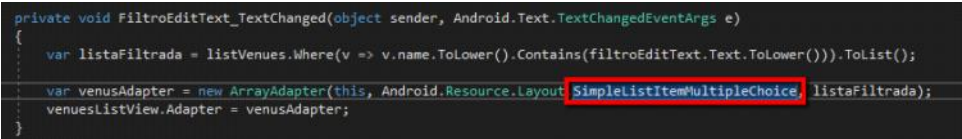
public class LugarDeInteres
{
    public LugarDeInteres(Venue venue, int idViaje)
    {
        IdViaje = idViaje;
        Nombre = venue.name;
        Latitud = (float) venue.location.lat;
        Longitud = (float) venue.location.lng;
        Categoria = venue.categories.First().name;
    }

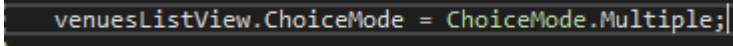
    a. [PrimaryKey, AutoIncrement]
    public int Id { get; set; }

    public string Nombre { get; set; }
    public float Latitud { get; set; }
    public float Longitud { get; set; }
    public int IdViaje { get; set; }
    public string Categoria { get; set; }
}

```

- 2- Vamos a especificar la manera para que un ListView permita la selección de varios elementos, para esto debemos modificar el argumento del ArrayAdapter por SimpleListItemMultipleChoice:

- a. 
- b. Adicional se debe especificar en el OnCreate que el ListView permite Choice Multiple estableciendo la siguiente propiedad:

- i. 

## 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 desees utilizar.

Agregar caja de texto es un TextField en el cuadro de herramientas.

Se debe poner cuidado donde se ubica, ya que en IOS debemos especificar exactamente donde lo ponemos (No es tan automatico como en android).

**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 designer.

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 Designer por que cuando se crea el elemento en la interfaz este automaticamete se crea el correspondiente objeto en el archivo **ViewController.designer.cs**.

Por lo tanto definimos directamente los eventos de los botones:

#### **ViewController.cs**

```
Public override void ViewDidLoad(){  
    Button.TouchUpInside += button_TouchUpInside;  
}  
  
Viod button_TouchUpInside(object sender, EventArgs e){  
    //Código  
}
```

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

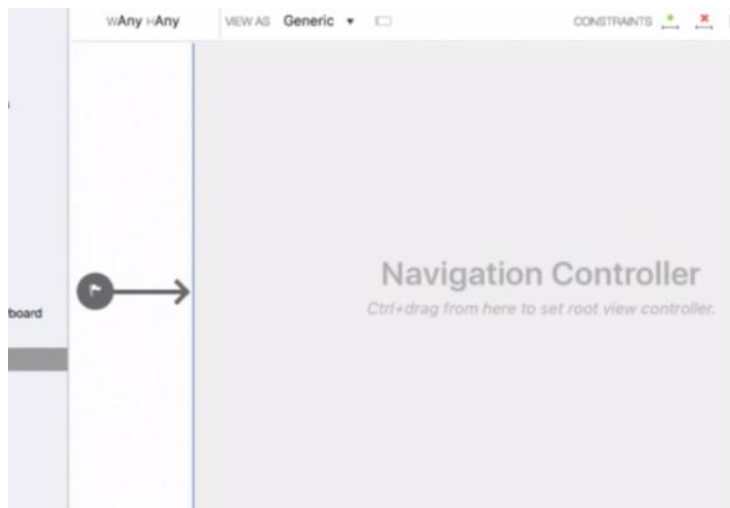
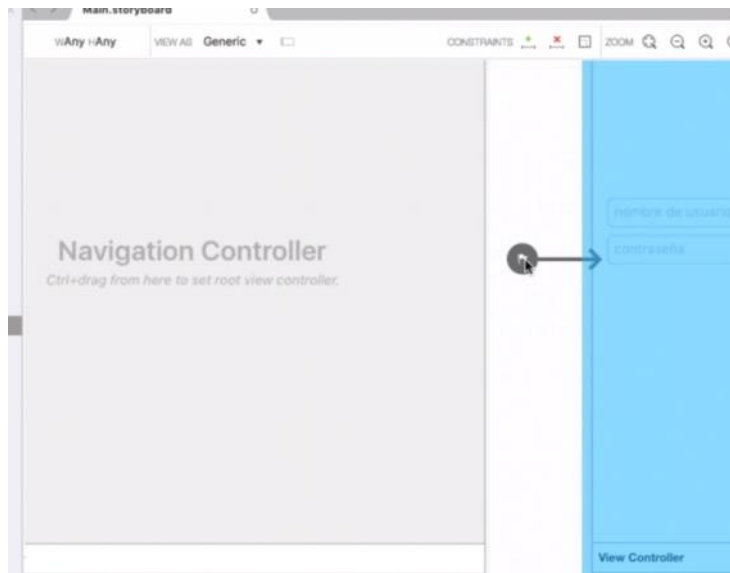
## 14. Agregar Nueva Vista

sábado, 16 de septiembre de 2017 04:30 PM

Para agregar una vista en IOS, como no tenemos acceso al código, debemos realizarlo por el cuadro de herramientas, Ubicamos el componente "**view Controller**" y lo arrastramos a la zona de diseño.

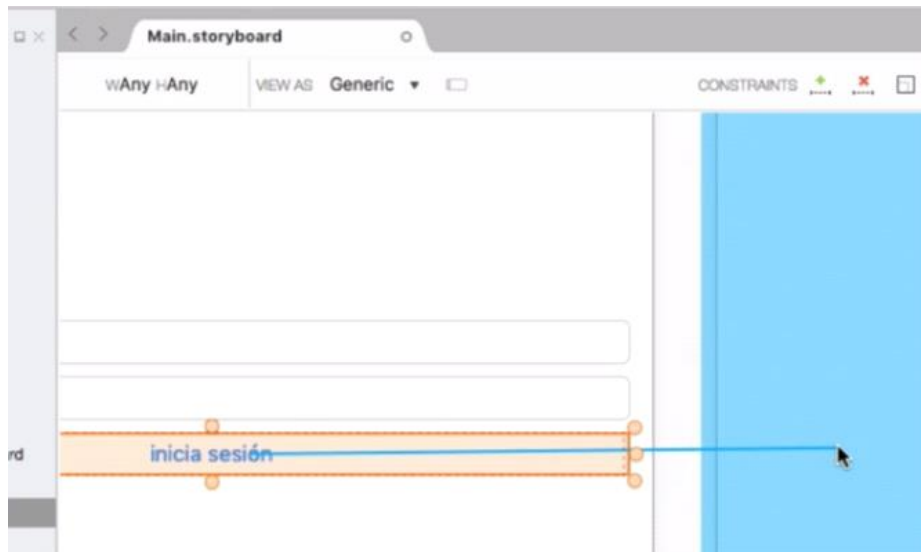
Ahora debemos definir la navegación entre las páginas, la navegación se realiza de la siguiente manera:

- En la barra de herramientas se busca el elemento **Navigation Controller**, este componente trae dos elementos debemos seleccionar el Table view y eliminarlo.
- Nos paramos en el Navigation Controller y ubicamos en el storyboard una flecha gris que indica la primera página que se ejecuta cuando inicia la app. Debemos seleccionar esta flecha y asignarla al nuevo **Navigation Controller**



Luego debemos asociar el navegador controller a la pagina de inicio de sesión, no ubicamos en **Navigation Controller** y con Ctrl + clic sostenido arrastramos hacia la otra página y se despliega la opción de Root la cual debemos seleccionar.

La funcionalidad anterior aplica tambien para indicar la navegabilidad para la acción de un boton.



Se debe seleccionar la opción Show

Para crear la página C# que controlara la vista, se debe ubicar en la vista y en las propiedades ubicar "**Class**" basta con copiar el nombre de la clase **NuevaVistaViewController** y dar enter, y automáticamente se crea el archivo.

Podemos ubicar botones en la parte superior del celular, para esto buscamos en el cuadro de herramientas y **Navigation Item** y luego un **Bar button item**, cambiar el identificador por una propiedad existente como (save).



## 15. Controlar elemento Bar button item

martes, 19 de septiembre de 2017 09:25 PM

Para controlar el elemento Bar button item, el cual es el boton creado en la parte superior del celular debemos realizarlo sobre la clase **viewController**:

Debemos sobre escribir el metodo **ViewDidLoad()** y posteriormente asignar el metodo al boton:

```
Public override void ViewDidLoad()
```

```
{
```

```
    Base.ViewDidLoad();
```

```
    guardarBarButtonItem.Clicked += GuardarBarButtonItem_Clicked;
```

```
}
```

```
Void GuardarBarButtonItem_Clicked(object sender, EventArgs e)
```

```
{
```

```
}
```

```
void GuardarBarButtonItem_Clicked(object sender, EventArgs e)
{
    string nombreArchivo = "viajes_db.sqlite";
    string rutaCarpeta = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Personal), "..", "Library");
    string rutaCompleta = Path.Combine(rutaCarpeta, nombreArchivo);

    var nuevoViaje = new Viaje()
    {
        Nombre = ciudadTextField.Text,
        FechaIda = (DateTime)idaDatePicker.Date,
        FechaRegreso = (DateTime)regresoDatePicker.Date
    };

    if (DatabaseHelper.Insertar(ref nuevoViaje, rutaCarpeta))
        Console.WriteLine("Inserción exitosa");
    else
        Console.WriteLine("Hubo un error");
}
```