

Xamarin Dev Days – Hands on Lab

Luis Ruvalcaba

Technical Evangelist

lruval@microsoft.com

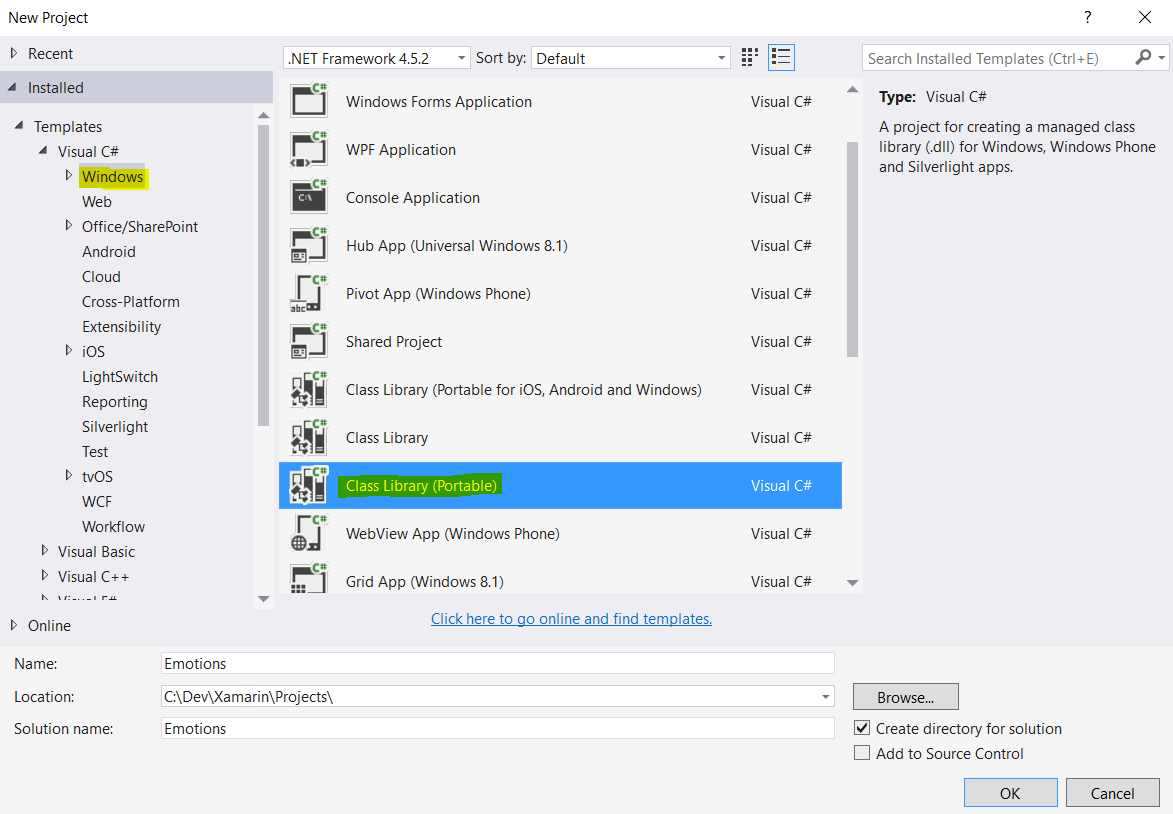
@ruvaluis

Documento actualizado el 3 de febrero de 2017

## Objetivo

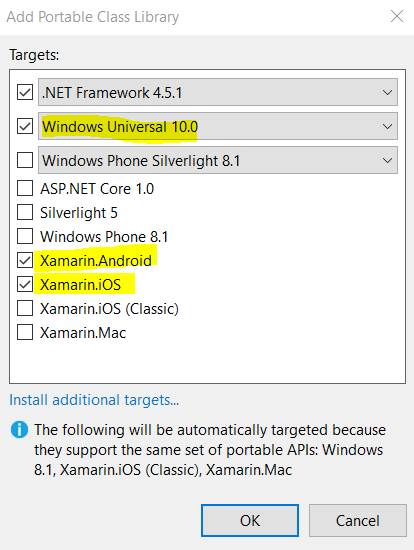
En este **Hands On Lab** aprenderás a desarrollar tu primera aplicación en Xamarin.Android y consumir servicios de Azure (Cognitive Services y Azure Mobile Apps) en donde podrás analizar una foto que tomes con tu celular y enviar el resultado de dicho análisis a una tabla en Mobile Apps de Azure.

Comencemos por crear un nuevo proyecto tipo *Class Library (Portable)* en Visual Studio.



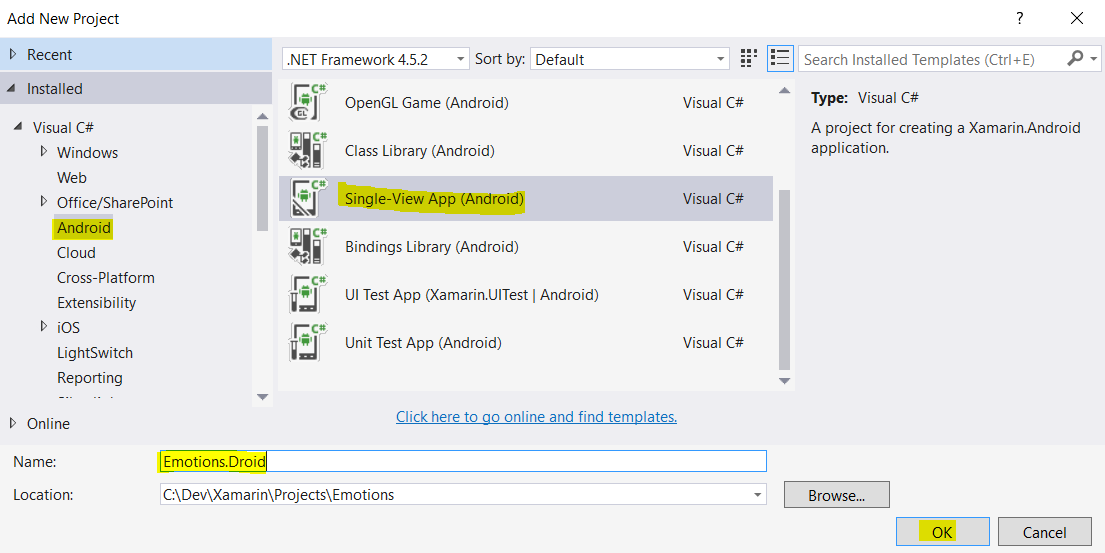
La idea de crear un proyecto de tipo *Class Library* es que podamos compartir el código entre diferentes plataformas (Android, iOS y Universal Windows Platform), debido a lo anterior es necesario seleccionar las siguientes opciones al momento de la creación del proyecto portable:

* .NET Framework 4.5.1
* Windows Universal 10.0 (Opcional, si tienes otra versión de Windows, puede dejar sin seleccionar esta casilla)
* Xamarin.Android
* Xamarin.iOS

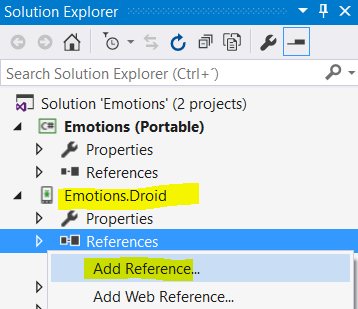


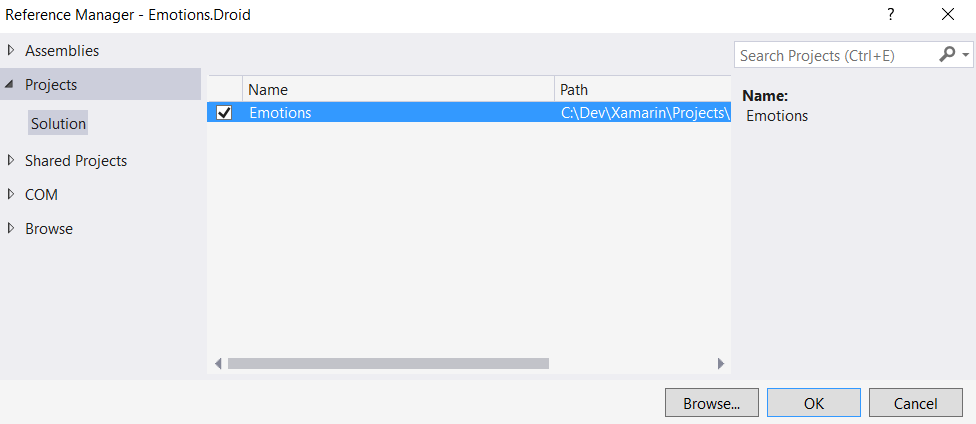
Borra el archivo Class1.cs que se generó en el proyecto portable, ya que agregaremos posteriormente las diferentes clases que utilizaremos en el proyecto portable.

A continuación, procederemos a agregar un proyecto Xamarin.Android, para lo cual deberás seleccionar la siguiente plantilla:



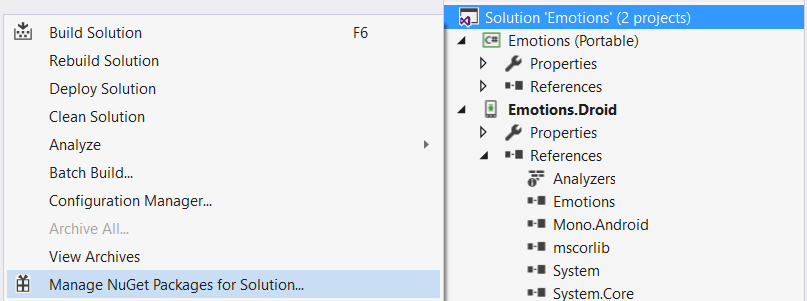
Selecciona el proyecto Emotions.Droid y agrega una referencia hacia el proyecto Emotions



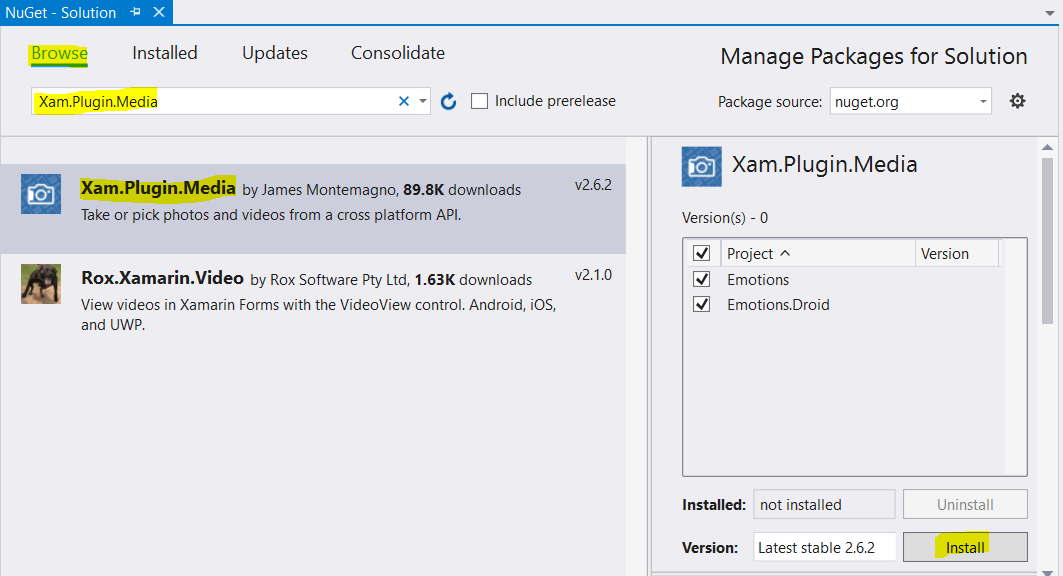


Selecciona el proyecto *Emotions.Droid*, da clic derecho y selecciona *Set as StartUp Project*

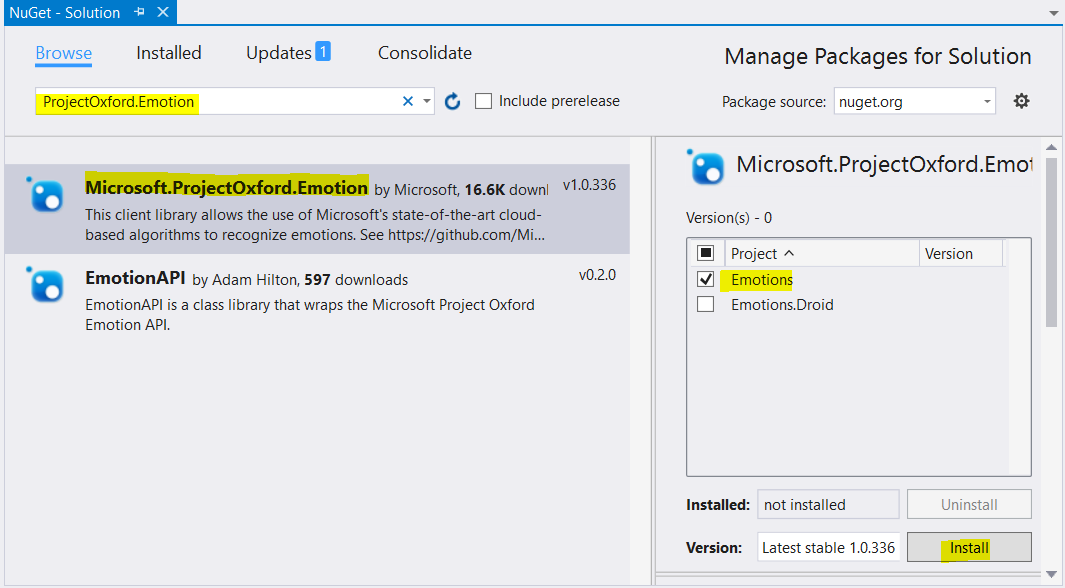
Ahora procederemos a agregar los diferentes paquetes de Nuget que utilizaremos en este proyecto para acceder a la cámara, detectar la conectividad, analizar una imagen utilizando Cognitive Services y registrar información en una tabla de Mobile Apps en Azure, para lo cual debes seleccionar la solución y dar clic derecho en *Manage NuGet Packages for Solution*



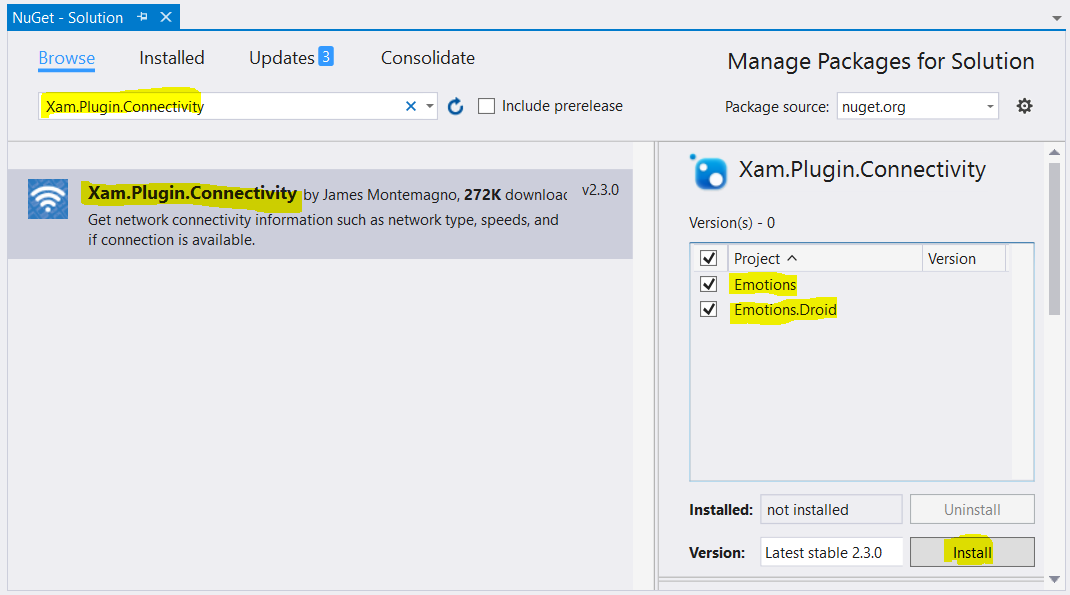
Selecciona el tab de **Browse** y busca el paquete **Xam.Plugin.Media** *,* el cual deberás instalar en el proyecto *Emotions* y *Emotions.Droid*



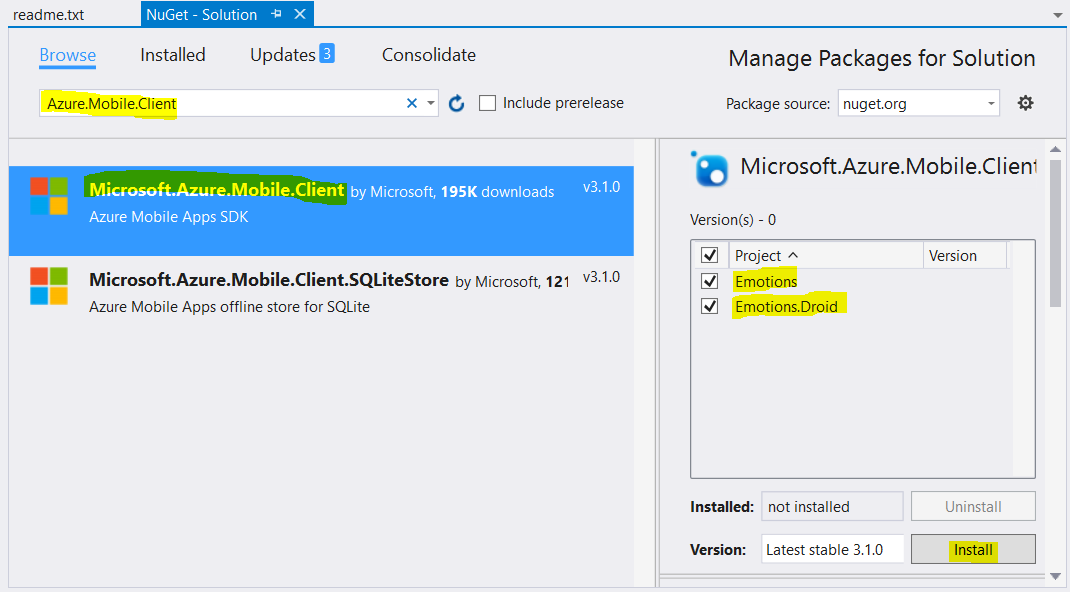
Busca el paquete **ProjectOxford.Emotion** *e* instálalo en el proyecto **Emotions**



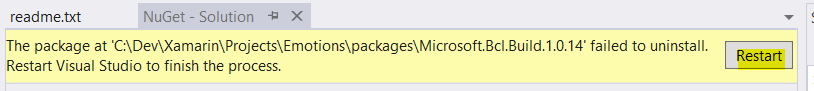
Busca el paquete **Xam.Plugin.Connectivity** *e* instálalo en el proyecto **Emotions** y **Emotions.Droid**



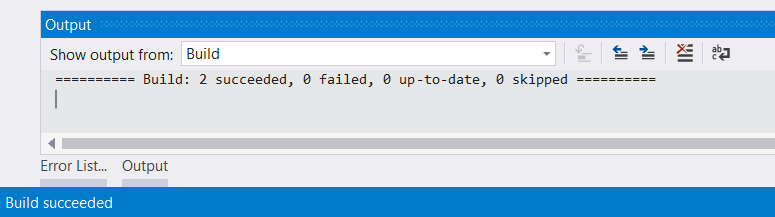
Busca el paquete **Azure.Mobile.Client** *e* instálalo en el proyecto **Emotions** y **Emotions.Droid**



Es probable que Visual Studio te pida reiniciar para que la instalación de los paquetes sea exitosa, confirma el reinicio presionando el siguiente botón:



Ahora que tenemos instalados todos los paquetes de NuGet necesarios para el proyecto, procede a compilar la solución, deberías de obtener el siguiente resultado:

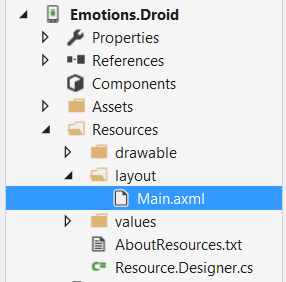


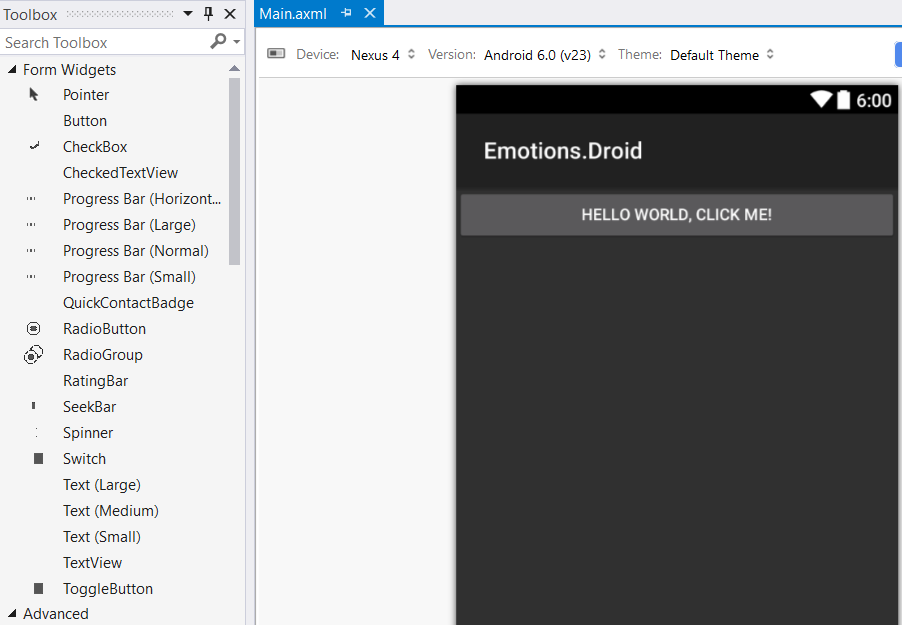
Conectar tu dispositivo Android que esté desbloqueado en modo desarrollador y habilitado para debugear por USB y procede a hacer el deployment de la aplicación, opcionalmente puedes utilizar el emulador de Android





La aplicación que se despliega es muy sencilla, el siguiente paso es crear la interface de usuario utilizando controles nativos de Android, para lo cual da doble clic sobre el archivo Emotions.Droid\Resources\Layout\**Main.axml**

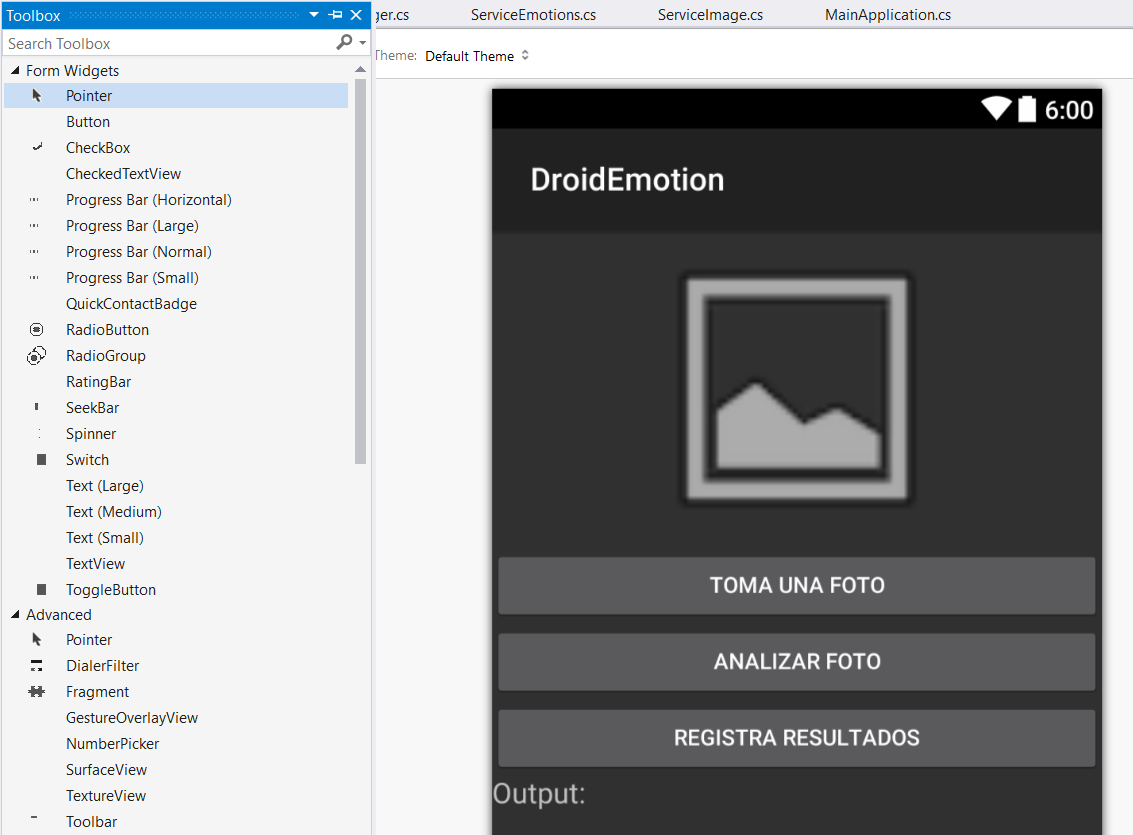




A continuación, procederemos a agregar los siguientes controles

* 1 ImageView, nombra este control como imageViewFoto
* 3 botones, cuyos nombres serán: btnCamara, btnAnalizaFoto y btnRegistraResultados
* 1 TextView, nombre este control como txtOutput

El resultado final deberá de verse de la siguiente manera:



Cambia a la vista de *Source* y valida que se muestre el siguiente código XML

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:minWidth="25px"

android:minHeight="25px">

<ImageView

android:src="@android:drawable/ic\_menu\_gallery"

android:layout\_width="match\_parent"

android:layout\_height="200dp"

android:id="@+id/imageViewFoto" />

<Button

android:text="Toma una foto"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/btnCamara" />

<Button

android:text="Analizar foto"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/btnAnalizaFoto" />

<Button

android:text="Registra resultados"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:id="@+id/btnRegistraResultados" />

<TextView

android:text="Output: "

android:textAppearance="?android:attr/textAppearanceMedium"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

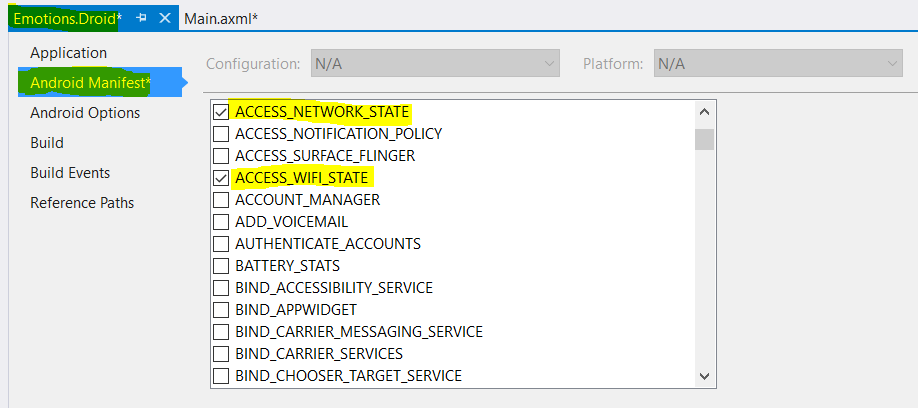
android:id="@+id/txtOutput" />

</LinearLayout>

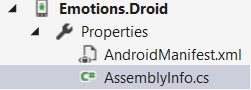
Posterior a esto, en el proyecto **Emotions.Droid**, demos doble clic sobre **Properties**, en la pestaña de **Android** **Manifest** deberemos de marcar los permisos:

Estos permisos nos permitirán interactuar con la cámara del dispositivo, así como detectar si el teléfono cuenta con una conexión a internet para poder consumir los webservices de Cognitive y Mobile Apps de Azure.

* ACCESS\_NETWORK\_STATE
* ACCESS\_WIFI\_STATE
* READ\_EXTERNAL\_STORAGE
* WRITE\_EXTERNAL\_STORAGE



Guardemos y cerremos el tab de propiedades del proyecto de Android. Ahora abramos el archivo *AssemblyInfo.cs* (expande *Properties* en el Explorador de Soluciones del proyecto Android)



Inserta las siguientes líneas de código al final del archivo:

[assembly: UsesFeature("android.hardware.camera", Required = false)]

[assembly: UsesFeature("android.hardware.camera.autofocus", Required = false)]

Selecciona el proyecto portable (Emotions) y agrega una nueva clase llamada *ServiceImage.cs*, el código de dicha clase es el siguiente:

using Plugin.Media;

using System.Threading.Tasks;

using Plugin.Media.Abstractions;

namespace Emotions

{

public class ServiceImage

{

public static async Task<MediaFile> TakePicture(bool useCam = true)

{

await CrossMedia.Current.Initialize();

if (useCam)

{

if (!CrossMedia.Current.IsCameraAvailable || !CrossMedia.Current.IsTakePhotoSupported)

{

return null;

}

}

var file = useCam

? await CrossMedia.Current.TakePhotoAsync(new StoreCameraMediaOptions

{

Directory = "DevDayMX",

Name = "DevDay\_Test.jpg"

})

: await CrossMedia.Current.PickPhotoAsync();

return file;

}

}

}

¿Para qué sirve esta clase? Se expone un método que decide si la imagen se tomará de la cámara del dispositivo o de la galería de imágenes existentes. Una vez obtenida la imagen, se almacena de manera temporal para ser utilizado dentro de la aplicación.

# Cognitive Services

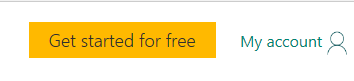
Procederemos a utilizar Cognitive Services para analizar imágenes que tomemos del celular o seleccionemos de la galería de imágenes del dispositivo.

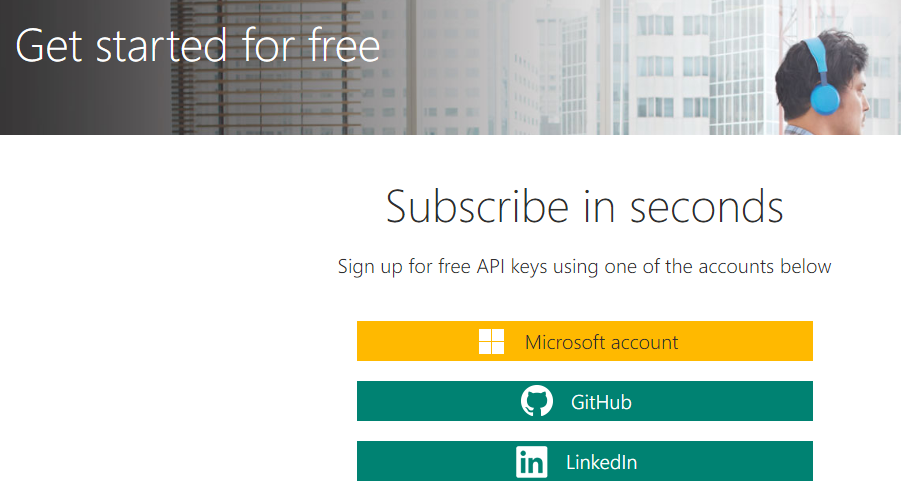
## ¿Cómo funciona Emotion API?

El Emotion API toma como entrada uno o varios rostros para analizar la expresión facial y regresa el grado de confianza (score) de un conjunto de emociones para cada rostro.

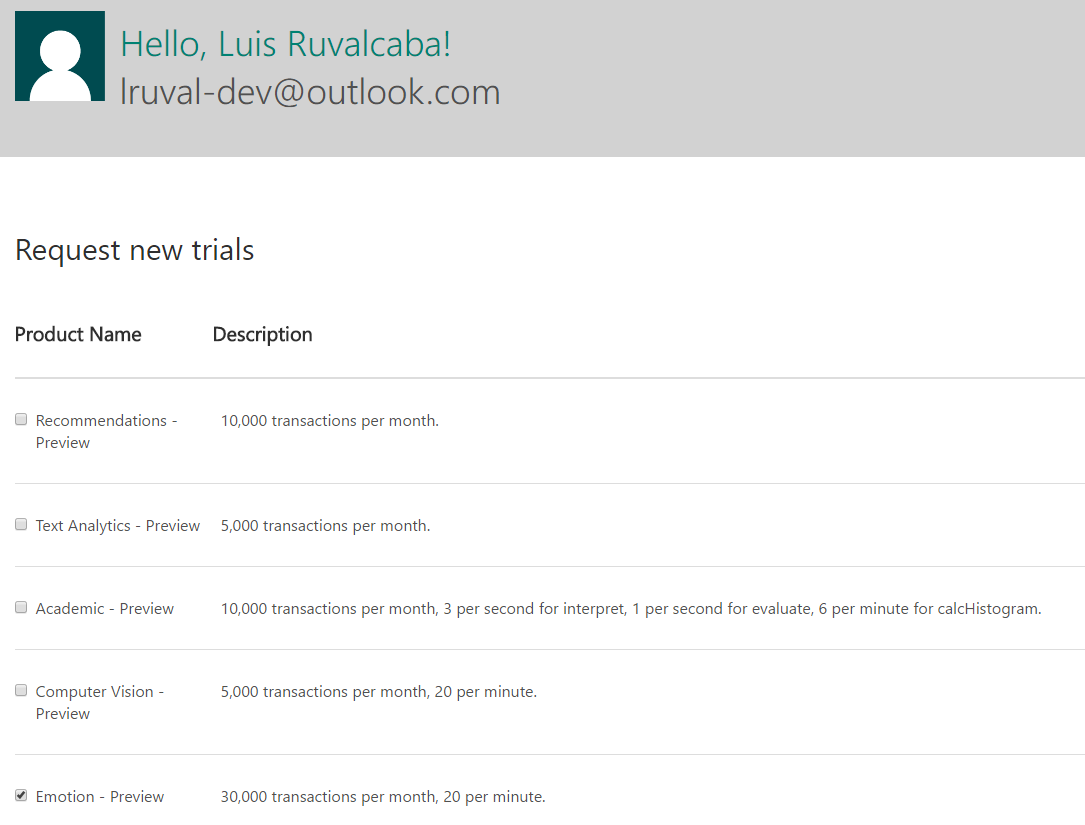
## 

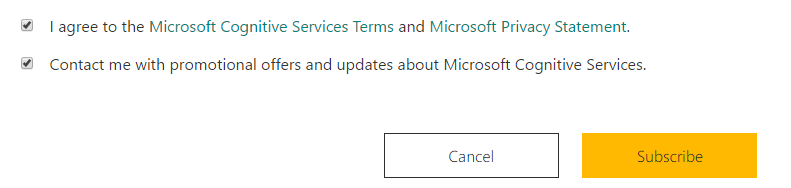
Por favor ingresa a <http://microsoft.com/cognitive>, regístrate dando clic en Get started for free



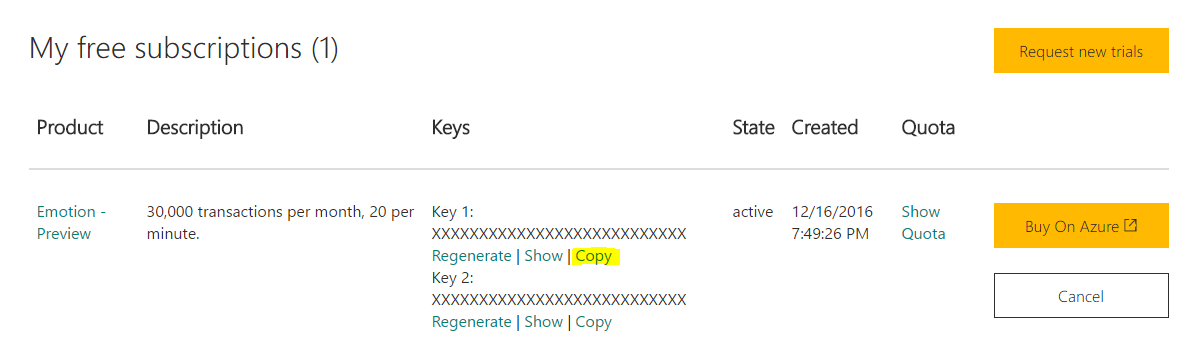


Una vez que iniciaste sesión y **comprobaste** tu correo electrónico, selecciona la opción: Emotion – Preview así como los términos del servicio y da clic en subscribe





Copia el Key 1, el cual utilizaremos en el siguiente paso



A continuación, agrega una nueva clase (*ServiceEmotions.cs)* en el proyecto portable (*Emotions*) inserta el siguiente código remplazando la variable *key* con tu llave de servicio de Emotions de Cognitive Services

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using System.Collections.Generic;

using Microsoft.ProjectOxford.Emotion;

namespace Emotions

{

public class ServiceEmotions

{

static string key = "InsertaTuEmotionsKey";

public static async Task<Dictionary<string, float>> GetEmotions(Stream stream)

{

EmotionServiceClient cliente = new EmotionServiceClient(key);

var emotions = await cliente.RecognizeAsync(stream);

if (emotions == null || emotions.Count() == 0)

return null;

return emotions[0].Scores.ToRankedList().ToDictionary(x => x.Key, x => x.Value);

}

}

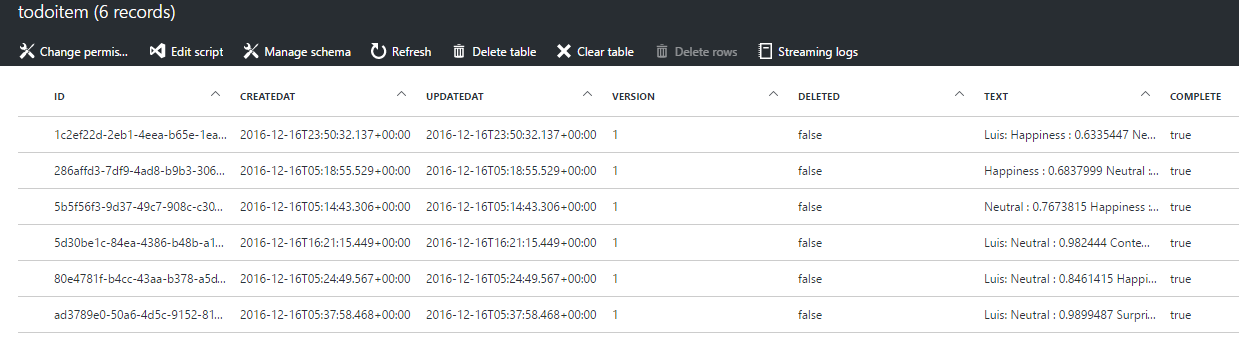
}

¿Para qué sirve la clase anterior? La clase anterior implementa un método que realiza una consulta al servicio de Emotions de Cognitive services y devuelve el resultado del análisis en forma de diccionario. Cada elemento es una emoción y brinda su puntuación correspondiente.

# Azure mobile apps

Adicionalmente a obtener el resultado del análisis de la imagen, enviaremos dicho resultado a una tabla en un Mobile App en Azure, para lo cual necesitamos crear la clase *ItemManager.cs* en el proyecto portable (Emotions)

La clase *ItemManager.cs* nos servirá para agregar un nuevo ítem en una tabla en el siguiente Mobile App, dicho ítem incluirá el resultado del análisis realizado con el Emotions API.



El código de la clase ItemManager.cs es el siguiente:

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.WindowsAzure.MobileServices;

using Newtonsoft.Json;

namespace Emotions

{

public partial class ItemManager

{

static ItemManager defaultInstance = new ItemManager();

MobileServiceClient client;

IMobileServiceTable<TodoItem> todoTable;

private ItemManager()

{

this.client = new MobileServiceClient(@"https://xamarinmxdev.azurewebsites.net");

this.todoTable = client.GetTable<TodoItem>();

}

public static ItemManager DefaultManager

{

get

{

return defaultInstance;

}

private set

{

defaultInstance = value;

}

}

public MobileServiceClient CurrentClient

{

get { return client; }

}

public async Task SaveTaskAsync(TodoItem item)

{

if (item.Id == null)

{

await todoTable.InsertAsync(item);

}

else

{

await todoTable.UpdateAsync(item);

}

}

}

public class TodoItem

{

string id;

string name;

bool done;

[JsonProperty(PropertyName = "id")]

public string Id

{

get { return id; }

set { id = value; }

}

[JsonProperty(PropertyName = "text")]

public string Name

{

get { return name; }

set { name = value; }

}

[JsonProperty(PropertyName = "complete")]

public bool Done

{

get { return done; }

set { done = value; }

}

[Version]

public string Version { get; set; }

}

}

A continuación procederemos a modificar la clase MainActivity.cs del proyecto Emotions.Droid para integrar toda la funcionalidad restante

1. Agrega los siguientes *usings* a la clase

using Emotions;

using System.IO;

using System.Text;

using Plugin.Connectivity;

using Plugin.Media.Abstractions;

using System.Collections.Generic;

1. Agrega las siguientes *fields* a nivel clase, agrega tu nombre en el *string* de ResultadoEmociones

ItemManager manager;

static Stream streamCopy;

string ResultadoEmociones = "Emotions Nombre: "; //Escribe tu nombre

TextView txtResultado;

Button btnRegistraResultados;

Button btnAnalizaFoto;

1. Modifica el método OnCreate con el siguiente código

protected override void OnCreate(Bundle bundle)

{

base.OnCreate(bundle);

//Mobile App Azure

Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();

manager = ItemManager.DefaultManager;

// Set our view from the "main" layout resource

SetContentView(Resource.Layout.Main);

Button btnCamara = FindViewById<Button>(Resource.Id.btnCamara);

btnAnalizaFoto = FindViewById<Button>(Resource.Id.btnAnalizaFoto);

btnRegistraResultados = FindViewById<Button>(Resource.Id.btnRegistraResultados);

txtResultado = FindViewById<TextView>(Resource.Id.txtOutput);

btnRegistraResultados.Visibility = ViewStates.Invisible;

btnAnalizaFoto.Visibility = ViewStates.Invisible;

btnCamara.Click += BtnCamara\_Click;

btnAnalizaFoto.Click += BtnAnalizaFoto\_Click;

btnRegistraResultados.Click += BtnRegistraResultados\_Click;

}

1. Agrega los siguientes métodos para manejar los eventos de OnClick de los botones que colocamos en la interface de usuario

private async void BtnRegistraResultados\_Click(object sender, EventArgs e)

{

if (CrossConnectivity.Current.IsConnected &&

await CrossConnectivity.Current.IsRemoteReachable("http://www.microsoft.com"))

{

btnRegistraResultados.Visibility = ViewStates.Invisible;

Toast.MakeText(this, "Registrando tus resultados", ToastLength.Short).Show();

await manager.SaveTaskAsync(new TodoItem { Done = true, Name = ResultadoEmociones });

}

else

{

Toast.MakeText(this, "No tienes acceso a internet", ToastLength.Short).Show();

}

}

private async void BtnAnalizaFoto\_Click(object sender, EventArgs e)

{

if (CrossConnectivity.Current.IsConnected &&

await CrossConnectivity.Current.IsRemoteReachable("http://www.microsoft.com"))

{

if (streamCopy != null)

{

btnAnalizaFoto.Visibility = ViewStates.Invisible;

Toast.MakeText(this, "Analizando imagen utilizando Cognitive Services", ToastLength.Short).Show();

Dictionary<string, float> emotions = null;

try

{

streamCopy.Seek(0, SeekOrigin.Begin);

emotions = await ServiceEmotions.GetEmotions(streamCopy);

}

catch (Exception ex)

{

Toast.MakeText(this, "Se ha presentado un error al conectar con los servicios", ToastLength.Short).Show();

return;

}

StringBuilder sb = new StringBuilder();

if (emotions != null)

{

txtResultado.Text = "---Análisis de Emociones---";

sb.AppendLine();

foreach (var item in emotions)

{

string toAdd = item.Key + " : " + item.Value + " ";

sb.Append(toAdd);

}

txtResultado.Text += sb.ToString();

btnRegistraResultados.Visibility = ViewStates.Visible;

}

else txtResultado.Text = "---No se detectó una cara---";

ResultadoEmociones += sb.ToString();

}

else txtResultado.Text = "---No has seleccionado una imagen---";

}

else

{

Toast.MakeText(this, "No tienes acceso a internet", ToastLength.Short).Show();

}

}

private async void BtnCamara\_Click(object sender, EventArgs e)

{

MediaFile file = null;

try

{

file = await ServiceImage.TakePicture(true);

}

catch (Android.OS.OperationCanceledException)

{

}

SetImageToControl(file);

btnAnalizaFoto.Visibility = ViewStates.Visible;

}

private void SetImageToControl(MediaFile file)

{

if (file == null)

{

return;

}

ImageView imgImage = FindViewById<ImageView>(Resource.Id.imageViewFoto);

imgImage.SetImageURI(Android.Net.Uri.Parse(file.Path));

var stream = file.GetStream();

streamCopy = new MemoryStream();

stream.CopyTo(streamCopy);

stream.Seek(0, SeekOrigin.Begin);

file.Dispose();

}

**Referencias**

<https://www.microsoft.com/cognitive-services>

<https://azure.microsoft.com/en-us/marketplace/partners/microsoft/trymobileappsnode/>