

Reto 6 - Xamarin Championship

Luis Ruvalcaba Sanchez

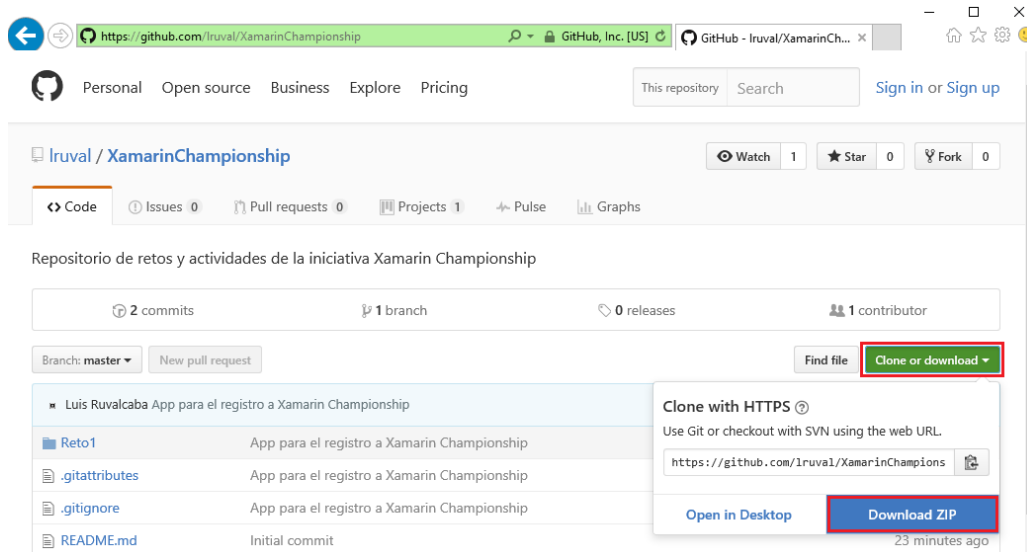
Technical Evangelist – lruval@microsoft.com

Gracias por tu interés en Xamarin Championship, los objetivos del reto 6 son los siguientes:

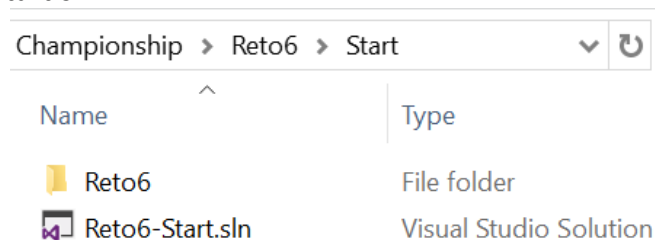
- Actividad: Utilizar el plugin Xam.Plugin.Media para acceder a la cámara de tu dispositivo Android
- Actividad: Analizar una imagen utilizando servicios cognitivos de Azure.
- Reto: Registrar los resultados obtenidos en la base de datos del campeonato.

1. [Descarga](#) el código fuente de Xamarin Championship, el cual incluye el proyecto inicial del reto 6

<https://github.com/lruval/XamarinChampionship>

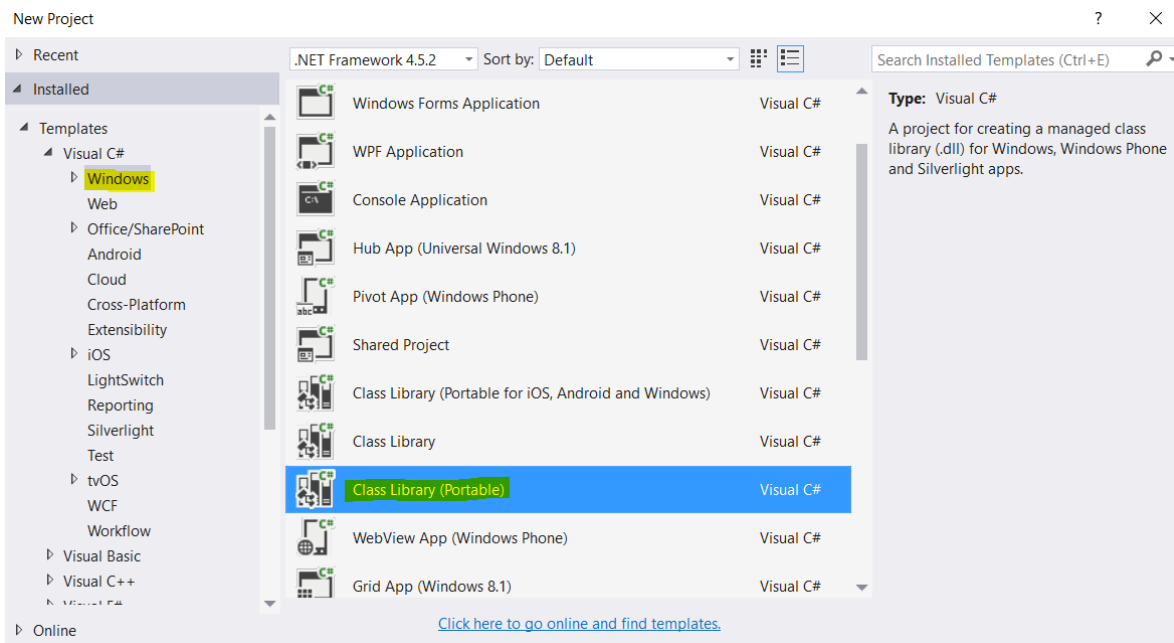


2. Extrae el archivo comprimido, navega a la carpeta “Reto6\Start” y selecciona el archivo Reto6-Start.sln

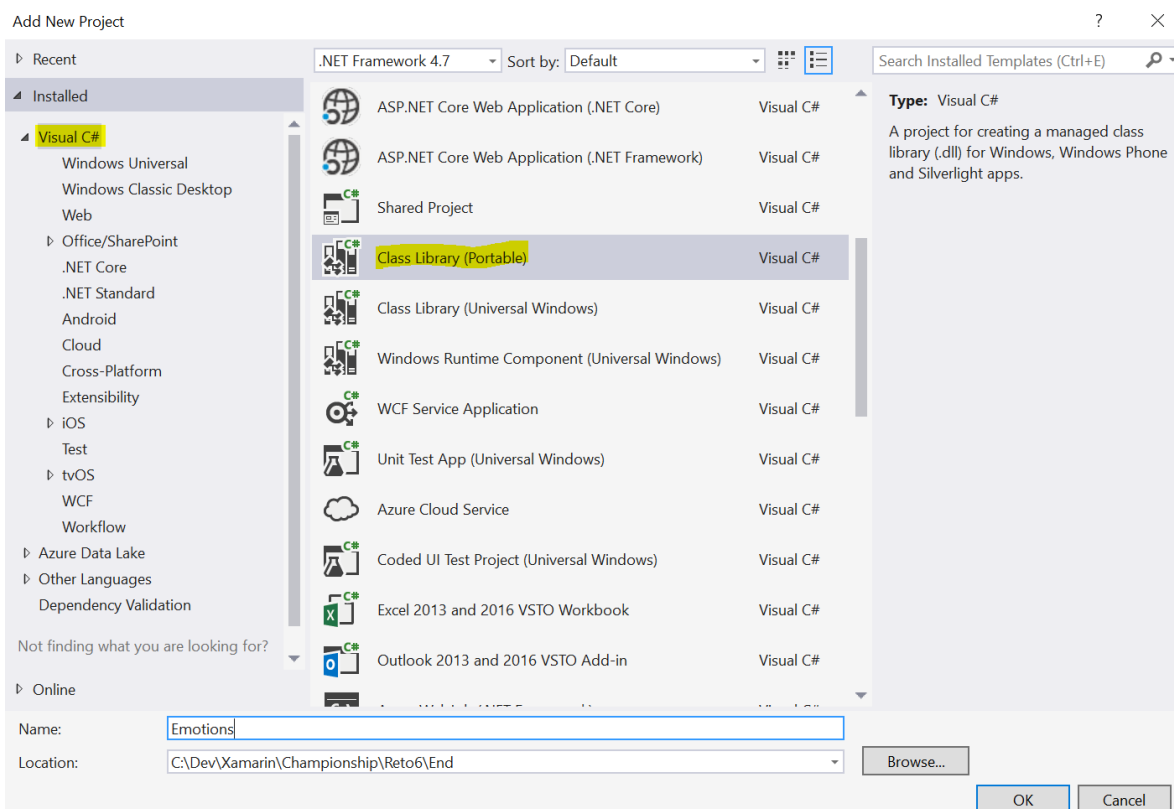


3. Agregaremos un nuevo proyecto a la solución actual, el tipo de proyecto es *Portable Class Library*, para completar esta actividad deberás seleccionar la solución actual, dar clic derecho y seleccionar “Add new Project”

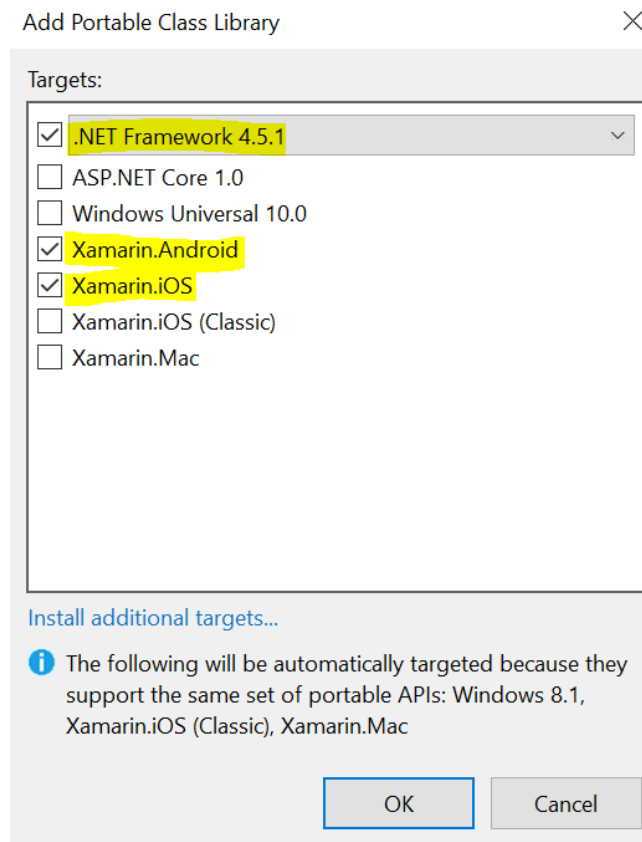
Si utilizas Visual Studio 2015 deberás de seleccionar la plantilla Visual C# -> Windows -> Class Library (Portable) y crear un nuevo proyecto llamado “Emotions”



Si utilizas Visual Studio 2017 deberás seleccionar la plantilla Visual C# -> Class Library (Portable) y crear un nuevo proyecto llamado “Emotions”

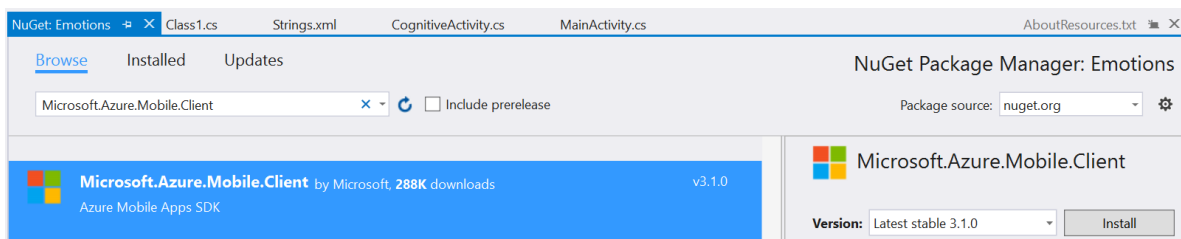


4. Seleccionar las siguientes plataformas para que el nuevo proyecto compartido sea compatible con .Net Framework, Xamarin.Android y Xamarin.iOS

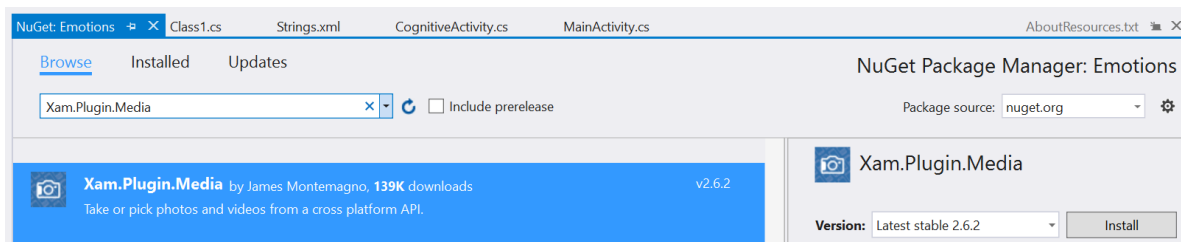


5. Selecciona el proyecto Emotions, da clic derecho y selecciona *Manage Nuget Packages* y agrega los siguientes paquetes

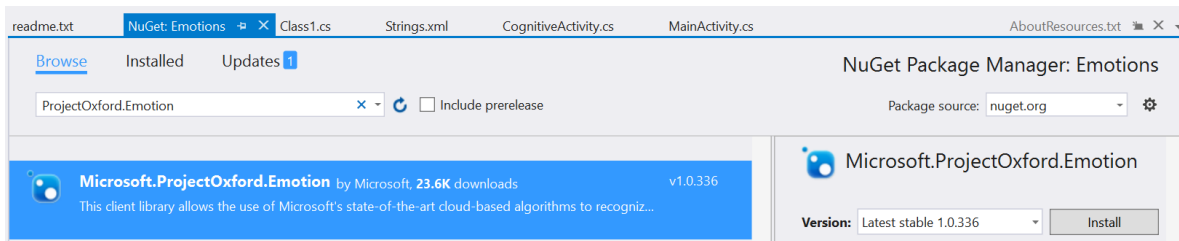
Microsoft.Azure.Mobile.Client



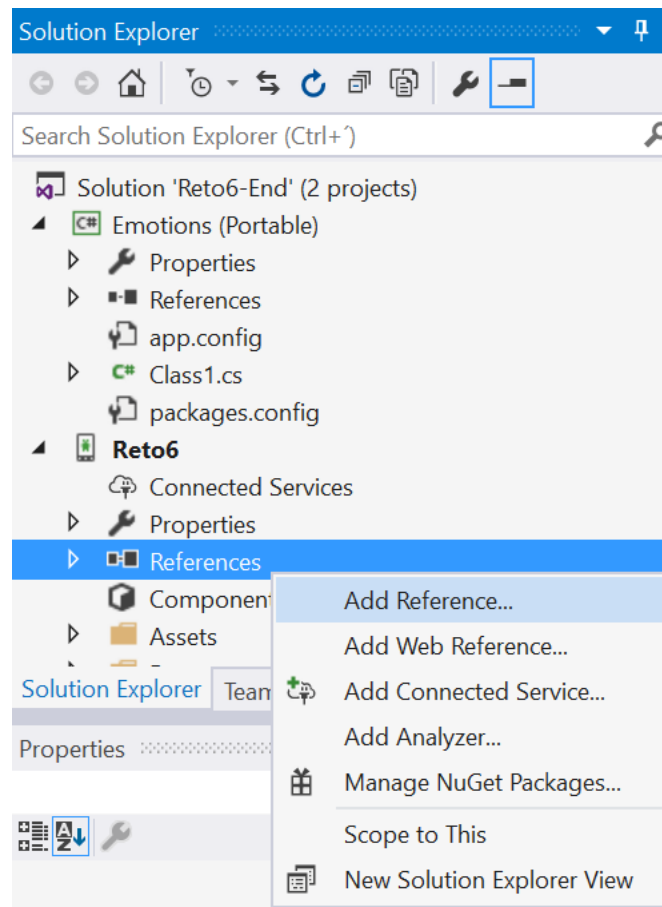
Xam.Plugin.Media * También deberás de instalar este paquete en el proyecto Reto 6 *

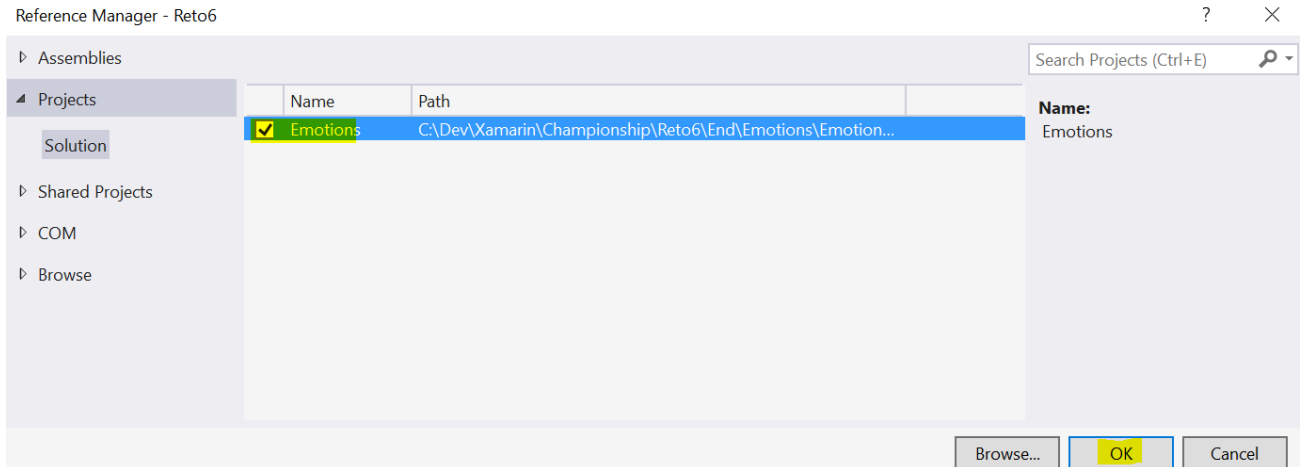


ProjectOxford.Emotion



6. Selecciona el proyecto “Reto6” y agrega como referencia el proyecto de clase portable “Emotions”





7. Selecciona el archivo “Reto6\Resources\layout\Cognitive.axml” y modifica la interface de usuario para agregar los siguientes elementos

- 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 debería de verse de la siguiente manera:



Si tienes alguna duda de cómo crear la interface de usuario, puedes utilizar el siguiente código XML como referencia

```
Cognitive.xml  + X
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:orientation="vertical"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:minWidth="25px"
7      android:minHeight="25px">
8      <ImageView
9          android:src="@android:drawable/ic_menu_gallery"
10         android:layout_width="match_parent"
11         android:layout_height="200dp"
12         android:id="@+id/imageViewFoto" />
13     <Button
14         android:text="Toma una foto"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content"
17         android:id="@+id/btnCamara" />
18     <Button
19         android:text="Analizar foto"
20         android:layout_width="match_parent"
21         android:layout_height="wrap_content"
22         android:id="@+id/btnAnalizaFoto" />
23     <Button
24         android:text="Registra resultados"
25         android:layout_width="match_parent"
26         android:layout_height="wrap_content"
27         android:id="@+id/btnRegistraResultados" />
28     <TextView
29         android:text="Output: "
30         android:textAppearance="?android:attr/textAppearanceMedium"
31         android:layout_width="match_parent"
32         android:layout_height="wrap_content"
33         android:id="@+id/txtOutput" />
34 </LinearLayout>
```

8. Selecciona el proyecto portable (Emotions) y agrega una nueva clase llamada ServiceImage.cs y agrega el siguiente código:

```
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 = "Championship",
        Name = "Reto6_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.



The screenshot displays the 'Emotion APIs' interface. On the left, a photo of a man and a young child is shown. Both faces are enclosed in bounding boxes, and the word 'Surprise' is written above each box. On the right, the 'Face detection' section shows the bounding box coordinates for the detected faces. Below that, the 'Emotion scores' section lists the confidence scores for various emotions.

Face detection

```

"faceRectangle": {
  "width": 193,
  "height": 193,
  "left": 326,
  "top": 204
}

```

Emotion scores


```

"scores": {
  "anger": 5.182241e-8,
  "contempt": 0.0000242813,
  "disgust": 5.621025e-7,
  "fear": 0.00115027453,
  "happiness": 1.06114619e-8,
  "neutral": 0.003540177,
  "sadness": 9.30888746e-7,
  "surprise": 0.9952837
}

```

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

Get started for free

My account 

Get started for free

Subscribe in seconds

Sign up for free API keys using one of the accounts below



Microsoft account



GitHub



LinkedIn

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



Hello, Luis Ruvalcaba!

lruval-dev@outlook.com

Request new trials

Product Name	Description
<input type="checkbox"/> Recommendations - Preview	10,000 transactions per month.
<input type="checkbox"/> Text Analytics - Preview	5,000 transactions per month.
<input type="checkbox"/> Academic - Preview	10,000 transactions per month, 3 per second for interpret, 1 per second for evaluate, 6 per minute for calcHistogram.
<input type="checkbox"/> Computer Vision - Preview	5,000 transactions per month, 20 per minute.
<input checked="" type="checkbox"/> Emotion - Preview	30,000 transactions per month, 20 per minute.

- ☒ I agree to the [Microsoft Cognitive Services Terms](#) and [Microsoft Privacy Statement](#).
- ☒ Contact me with promotional offers and updates about Microsoft Cognitive Services.


Cancel

Subscribe

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

My free subscriptions (1)

Request new trials

Product	Description	Keys	State	Created	Quota	
Emotion - Preview	30,000 transactions per month, 20 per minute.	Key 1: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy Key 2: XXXXXXXXXXXXXXXXXXXXXXXXXXXX Regenerate Show Copy	active	12/16/2016 7:49:26 PM	Show Quota	Buy On Azure  <div>Cancel</div>

A continuación, agrega una nueva clase (*ServiceEmotions.cs*) en el proyecto portable (*Emotions*) inserta el siguiente código reemplazando 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<TorneoItem> todoTable;

        private ItemManager()
        {
            this.client = new
MobileServiceClient(@"https://xamarinchampions.azurewebsites.net/");
            this.todoTable = client.GetTable<TorneoItem>();
        }

        public static ItemManager DefaultManager
        {
            get
            {
                return defaultInstance;
            }
            private set
            {
                defaultInstance = value;
            }
        }

        public MobileServiceClient CurrentClient
        {
            get { return client; }
        }

        public async Task SaveTaskAsync(TorneoItem item)
        {
            if (item.Id == null)
            {
                await todoTable.InsertAsync(item);
            }
        }
    }
}
```

```

        else
        {
            await todoTable.UpdateAsync(item);
        }
    }
}

public class TorneoItem
{
    private string _id;
    private string _email;

    [JsonProperty(PropertyName = "id")]
    public string Id
    {
        get { return _id; }
        set
        {
            _id = value;
        }
    }

    [JsonProperty(PropertyName = "Email")]
    public string Email
    {
        get { return _email; }
        set
        {
            _email = value;
        }
    }

    public string Reto { get; set; }
    public string DeviceId { get; set; }

    [Version]
    public string Version { get; set; }
}
}

```

Consumiendo la clase portable

A continuación, procederemos a modificar la clase CognitiveActivity.cs del proyecto Reto6 para integrar toda la funcionalidad restante

1. Modifica el encabezado de usings con los siguientes valores

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Android.App;
using Android.Content;
using Android.OS;
using Android.Runtime;
using Android.Views;
using Android.Widget;
using Emotions;
using System.IO;
using Plugin.Media.Abstractions;

```

2. Agrega las siguientes *fields* a nivel clase, agrega tu **nombre** y **código de país** en el *string* de *ResultadoEmociones*

```
ItemManager manager;
static Stream streamCopy;
string ResultadoEmociones = "Reto6 + CountryCode + Nombre: ";
TextView txtResultado;
Button btnRegistraResultados;
Button btnAnalizaFoto;
```

3. Modifica el método *OnCreate* con el siguiente código

```
protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);
    Microsoft.WindowsAzure.MobileServices.CurrentPlatform.Init();
    manager = ItemManager.DefaultManager;
    SetContentView(Resource.Layout.Cognitive);

    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;
}
```

4. 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)
{
    btnRegistraResultados.Visibility = ViewStates.Invisible;
    Toast.MakeText(this, "Registrando tus resultados",
ToastLength.Short).Show();
    TorneoItem registro = new TorneoItem();
    registro.DeviceId =
Android.Provider.Settings.Secure.GetString(ContentResolver,
Android.Provider.Settings.Secure.AndroidId);
    registro.Email = "";
    registro.Reto = ResultadoEmociones;
}
private async void BtnAnalizaFoto_Click(object sender, EventArgs e)
{
    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---";
}

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();
}

```

Prueba la aplicación, deberías de poder tomar una fotografía y analizarla con cognitive services. El **último paso** para completar este reto es que revises el código fuente del método *BtnRegistraResultados_Click* y agregues tu correo electrónico en la variable **registro.Email** y por último deberás de enviar tu registro al backend, como tip puedes darle una revisa al código fuente de la clase **ItemManager**