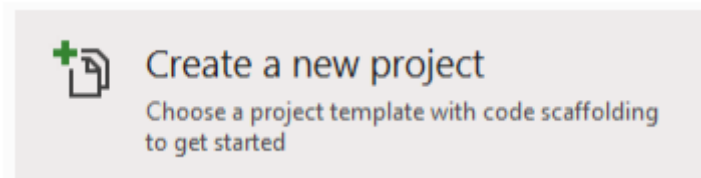


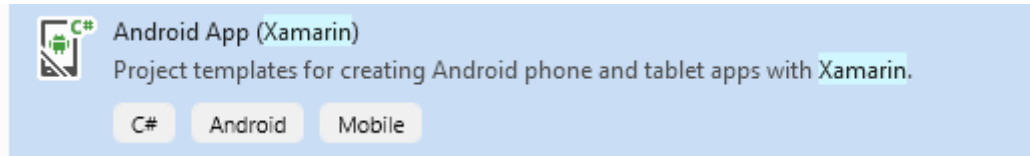
Usar la librería RtmpAndroidClient en un proyecto Xamarin Android.

Empecemos por crear la aplicación utilizando la plantilla Xamarin Android

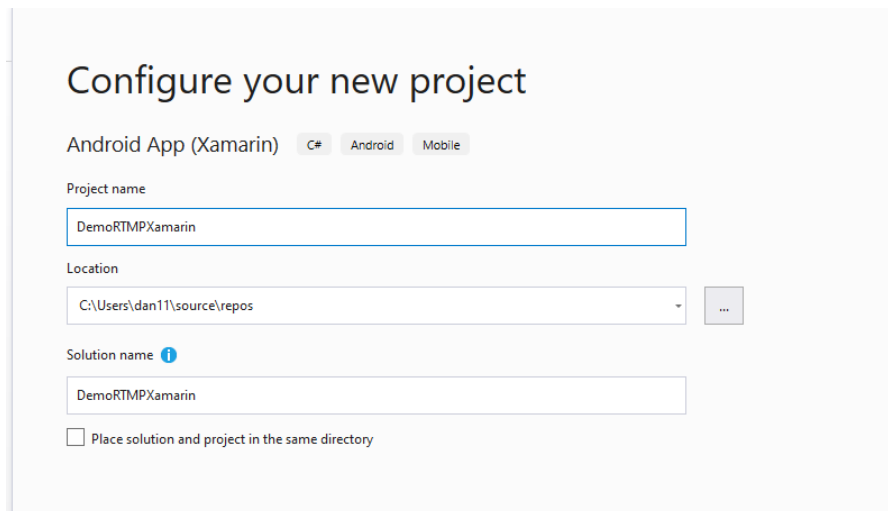
1. Abre Visual Studio bajo el contexto del Administrador.
2. En la ventana de inicio selecciona la opción **Create a new project**.



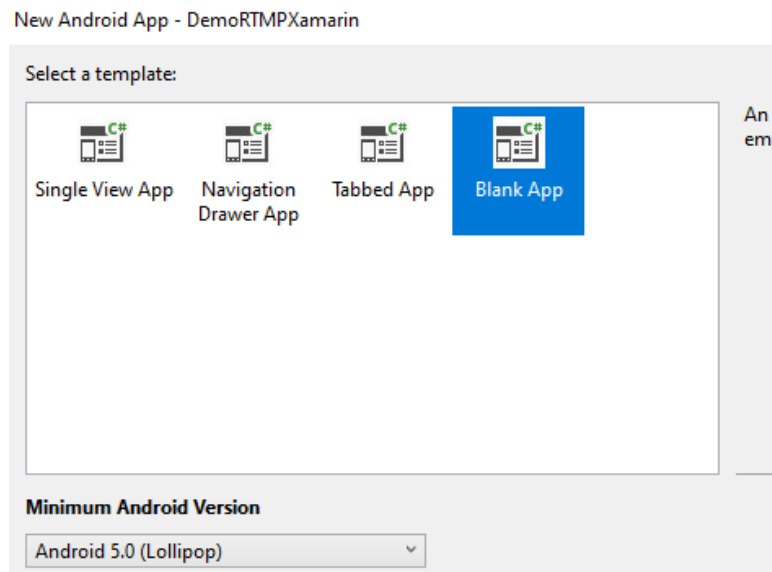
3. En la ventana **Create a new project** selecciona la plantilla **Android App (Xamarin)**



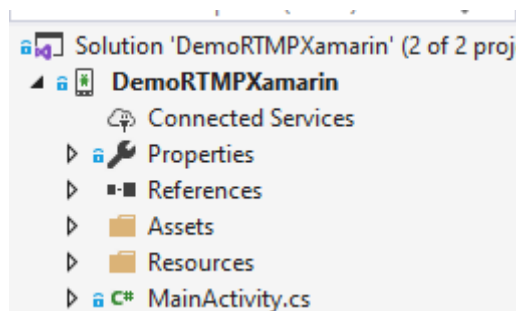
4. En la ventana **Configure your new project** proporciona **DemoRTMPXamarin** como el nombre del proyecto, selecciona la ubicación y haz clic en Create para continuar.



5. Seleccionar una Plantilla en Blanco y como **Minimun Android Version** selecciona Android 5.0 (Lollipop)



Al finalizar la creación de la solución se mostrará una estructura similar a la siguiente.



Modificar el Manifiesto de Android

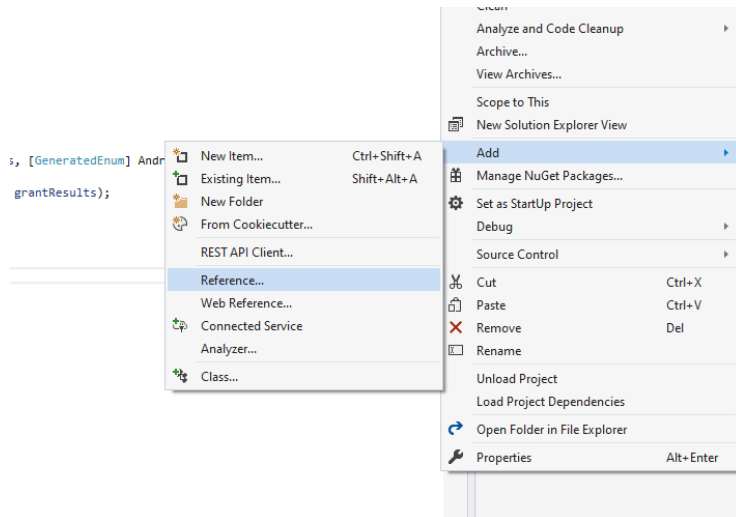
1. Nos situaremos en el proyecto **DemoRTMPXamarin** abriremos el archivo **Properties/AndroidManifest.xml** y agrega el siguiente código al manifiesto de android para tener los permisos de Cámara y Grabación de audio.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" android:versionCode="1" andri
<uses-sdk android:minSdkVersion="21" android:targetSdkVersion="28" />
<application android:allowBackup="true" android:icon="@mipmap/ic_launcher" android:label="@stri
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

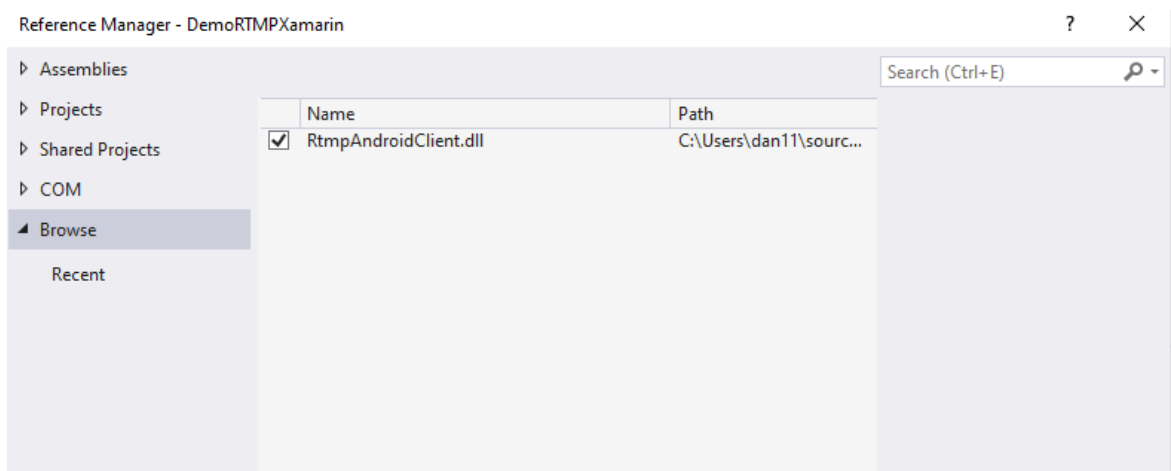
Agregar al proyecto la referencia de la Librería RtmpAndroidCliente.

La Librería puede ser descargada en el siguiente repositorio <https://github.com/NetDanydrago/RtmpAndroidClient>

1. Dar Click derecho en el nombre del proyecto -> add -> references.



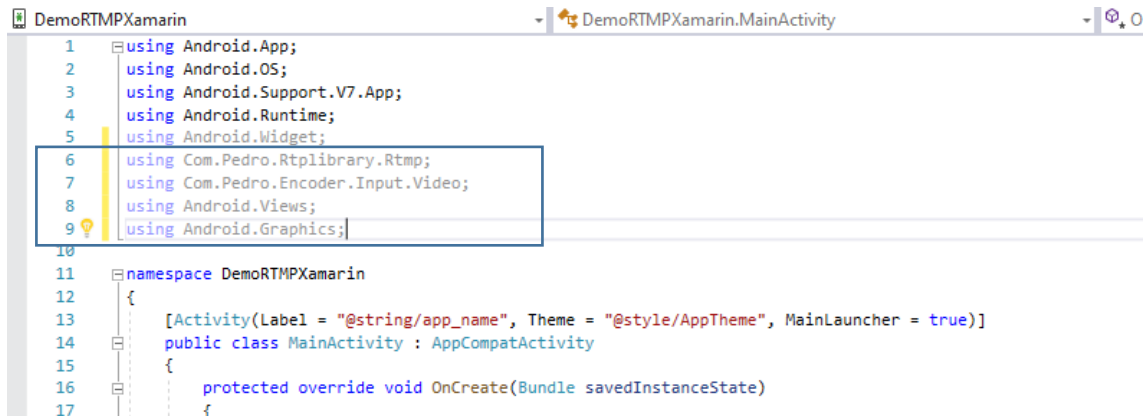
2. Selecciona el componente RtpmAndroidClient.



3. Dar click en Ok

Agregar los espacios de nombre de la librería RtmpAndroidClient.

1. Abre el archivo de clase MainActivity.cs
2. Agrega los siguientes espacios de nombre, al inicio de la clase MainActivity.

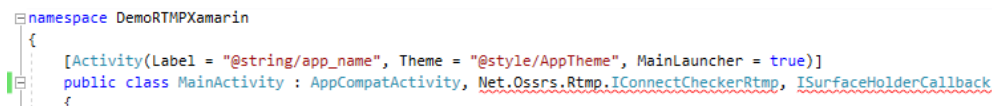


```

1  using Android.App;
2  using Android.OS;
3  using Android.Support.V7.App;
4  using Android.Runtime;
5  using Android.Widget;
6  using Com.Pedro.Rtplibrary.Rtmp;
7  using Com.Pedro.Encoder.Input.Video;
8  using Android.Views;
9  using Android.Graphics;
10
11 namespace DemoRTMPXamarin
12 {
13     [Activity(Label = "@string/app_name", Theme = "@style/AppTheme", MainLauncher = true)]
14     public class MainActivity : AppCompatActivity
15     {
16         protected override void OnCreate(Bundle savedInstanceState)
17         {

```

3. En la definición de la Clase **MainActivity** agregamos las interfaces **IconnectCheckerRtmp** e **ISurfaceHolderCallback** para poder implementarlas posteriormente.



```

namespace DemoRTMPXamarin
{
    [Activity(Label = "@string/app_name", Theme = "@style/AppTheme", MainLauncher = true)]
    public class MainActivity : AppCompatActivity, IconnectCheckerRtmp, ISurfaceHolderCallback
    {

```

Implementar la Interface IConectChecketRtmp

Su principal función es notificar a la clase cuando Inicia o se detiene un stream.

1. Agrega el siguiente código para implementar la interfaz **IConectChecketRtmp** dentro de la clase **MainActivity** debajo del método OnCreate.



```

public void OnAuthErrorRtmp()
{
    RunOnUiThread(() => { Toast.MakeText(this, "Auth Error", ToastLength.Long).Show(); });
}

public void OnAuthSuccessRtmp()
{
    RunOnUiThread(() => { Toast.MakeText(this, "Auth Success", ToastLength.Long).Show(); });
}

public void OnConnectionFailedRtmp(string reason)
{
    RunOnUiThread(() => { Toast.MakeText(this, "Failed", ToastLength.Long).Show(); });
}

public void OnConnectionSuccessRtmp()
{
    RunOnUiThread(() => { Toast.MakeText(this, "Success", ToastLength.Long).Show(); });
}

public void OnDisconnectRtmp()
{
    RunOnUiThread(() => { Toast.MakeText(this, "Disconnect", ToastLength.Long).Show(); });
}

public void OnNewBitrateRtmp(long bitrate)
{
}

```

Crear una interfaz de usuario sencilla.

Ahora agregaremos el código axml para definir una interfaz en nuestra aplicación.

1. Abre el Archivo **Resources/Layout/activity_main.xml**
2. Agrega el siguiente código para definir una interfaz de usuario dentro del archivo **activity_main.xml**.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <SurfaceView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/mysurfaceview" />
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <Button
            android:layout_width="128.5dp"
            android:layout_height="41.0dp"
            android:text="Switch Camera"
            android:id="@+id/ButtonSwitch"
        />
        <Button
            android:layout_width="129.0dp"
            android:layout_height="41.0dp"
            android:text="Start Stream"
            android:id="@+id/ButtonStart"
        />
        <Button
            android:layout_width="129.0dp"
            android:layout_height="41.0dp"
            android:text="Stop Stream"
            android:id="@+id/ButtonStop"
        />
    </LinearLayout>
    <EditText
        android:gravity="center"
        android:inputType="textEmailAddress"
        android:textAlignment="center"
        android:layout_marginTop="480dp"
        android:layout_width="match_parent"
        android:text="rtmp://dotnetpuebla.com:1935/TestSingle"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:id="@+id/TextRtmp" />
</RelativeLayout>
```

Uso del componente RtmpAndroidClient para generar un streaming.

1. Abrir el Archivo de clase **ActivityMain.cs**
2. definir una variable privada de la clase **RtmpCamera2**

```
private RtmpCamera2 RtmpCamera2;
```
3. Agrega el siguiente código dentro del método **OnCreate** para hacer uso de los elementos de la interfaz de usuario que definimos anteriormente

```
private RtmpCamera2 RtmpCamera2;

protected override void OnCreate(Bundle savedInstanceState)
{
    base.OnCreate(savedInstanceState);
    Xamarin.Essentials.Platform.Init(this, savedInstanceState);
    // Set our view from the "main" layout resource
    SetContentView(Resource.Layout.activity_main);

    SurfaceView surfaceView = FindViewById<SurfaceView>(Resource.Id.mysurfaceview);
    var SwitchButton = FindViewById<Button>(Resource.Id.ButtonSwitch);
    var StartButton = FindViewById<Button>(Resource.Id.ButtonStart);
    var StopButon = FindViewById<Button>(Resource.Id.ButtonStop);
    var RtmpText = FindViewById<EditText>(Resource.Id.TextRtmp);
    RtmpCamera2 = new RtmpCamera2(surfaceView, this);
    RtmpCamera2.SetRetries(10);
    surfaceView.Holder.AddCallback(this);
}
```

4. Agrega el siguiente código dentro del método **OnCreate** para definir las funciones de los botones que nos permitirá Iniciar un stream, detener un stream y cambiar la cámara frontal a la trasera y viceversa.

```
SwitchButton.Click += (sender, e) =>
{
    try
    {
        RtmpCamera2.SwitchCamera();
    }
    catch (CameraOpenException ext)
    {
        Toast.MakeText(this, ext.Message, ToastLength.Long).Show();
    }
};

StartButton.Click += (sender, e) =>
{
    if (RtmpCamera2.PrepareAudio() && RtmpCamera2.PrepareVideo())
    {
        if (string.IsNullOrEmpty(RtmpText.Text))
        {
            RtmpCamera2.StartStream("rtmp://dotnetpuebla.com:8088:1935/TestSingle/Test");
        }
        else
        {
            RtmpCamera2.StartStream(RtmpText.Text);
        }
    }
};

StopButon.Click += (sender, e) =>
{
    RtmpCamera2.StopStream();
};
```

Implementar las Interface ISurfaceHolderCallback,

La cual nos permitirá visualizar en la pantalla de nuestro dispositivo lo que capturara nuestro stream.

1. Agrega el siguiente código para implementar la interfaz **ISurfaceHolderCallback** dentro de la clase **MainActivity** debajo del método **OnCreate**.

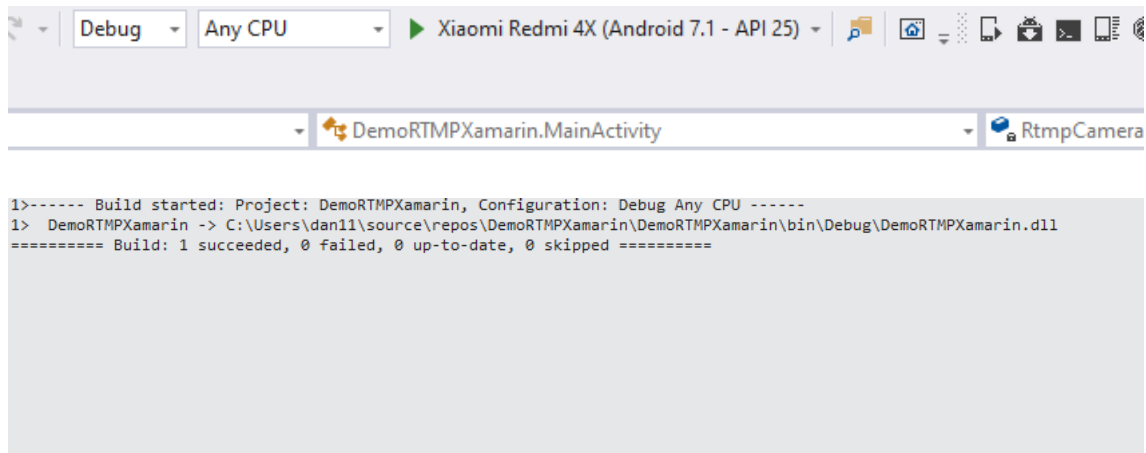
```
public void SurfaceChanged(ISurfaceHolder holder, [GeneratedEnum] Format format, int width, int height)
{
    RtmpCamera2.StartPreview();
}

public void SurfaceCreated(ISurfaceHolder holder)
{
}

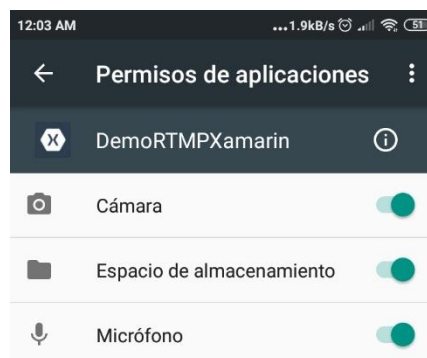
public void SurfaceDestroyed(ISurfaceHolder holder)
{
    if (RtmpCamera2.IsStreaming)
    {
        RtmpCamera2.StopStream();
    }
    RtmpCamera2.StopPreview();
}
```

Probar la aplicación

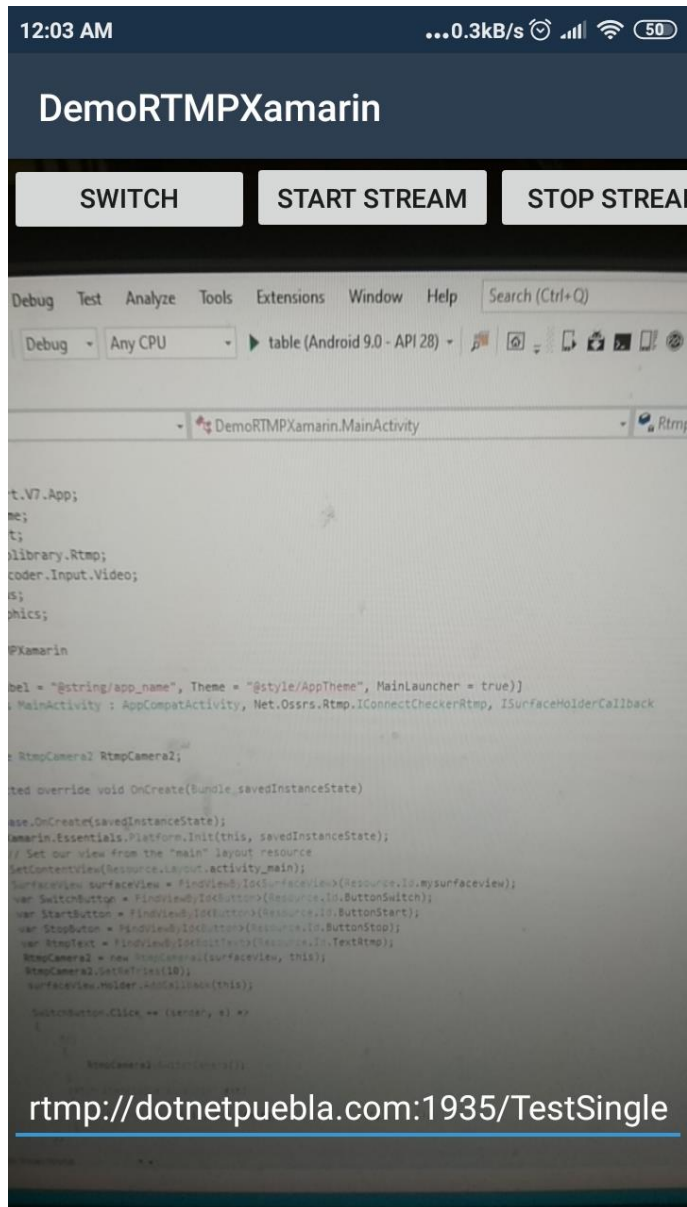
1. Compila el proyecto y ejecútala en un dispositivo móvil.



2. Una vez instalada en nuestro dispositivo antes de abrirla, activaremos los permisos de nuestra aplicación para que pueda funcionar correctamente.



3. Una vez configurado los permisos abriremos nuestra aplicación y se mostrara una pantalla similar a la siguiente.
4. Presiona el botón Start Stream para iniciar a transmitir al servidor que tenemos escrito por default en la parte inferior o también puedes cambiarla para transmitir hacia un servidor diferente.



Como observacion adicional los servidores de streaming tienen un formato por default en la estructura de la url.

Rtmp://Host:Puerto/AppName/StreamingName

Con esto finalizamos la demostración de cómo generar un streaming con Xamarin Android.

La creación de este demo fue posible gracias a la librería nativa en **java rtmp-rtsp-stream-client-java**, la cual se puede encontrar en el siguiente repositorio de **GitHub**.

<https://github.com/pedroSG94/rtmp-rtsp-stream-client-java>