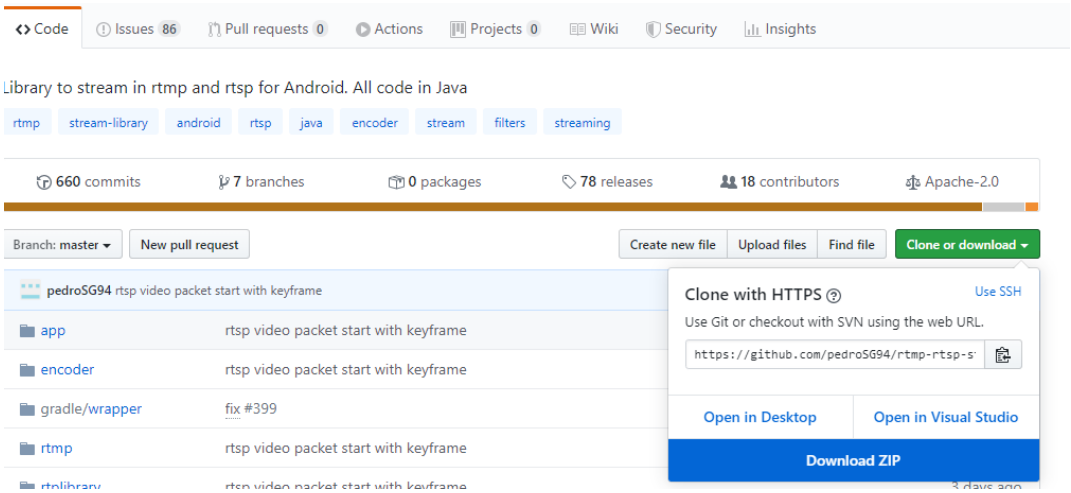
Descargar librería de Github

1. Abre el repositorio https://github.com/pedroSG94/rtmp-rtsp-stream-client-java

2.-Descargar la librería rtmp-rtsp-stream-client-java alojada en el repositorio de github.



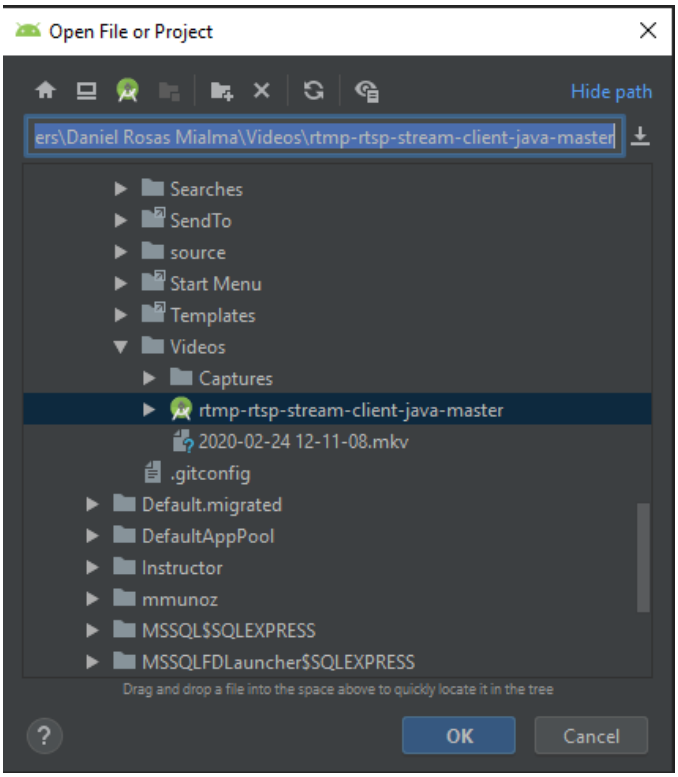
Abrir la librería rtmp-rtsp-stream-client-java con Android Studio.

Iniciaremos abriendo la librería utilizando el IDE de Android Studio

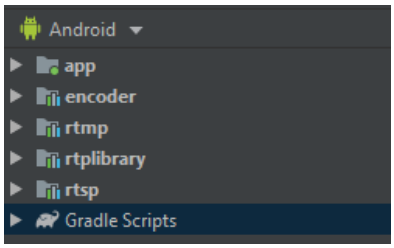
1. Abre Android Studio bajo el contexto del Administrador.

2. En la ventana de inicio selecciona la opción **Open an existing Android Studio Project**

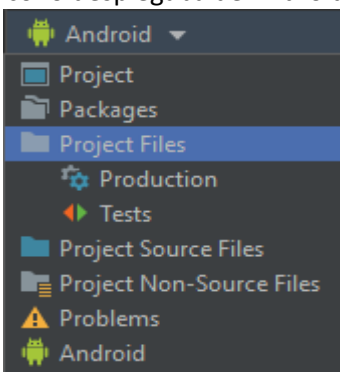
3. En la ventana **Open File or Project** selecciona la librería **rtmp-rtsp-stream-client-java** que fue descargada del repositorio de github



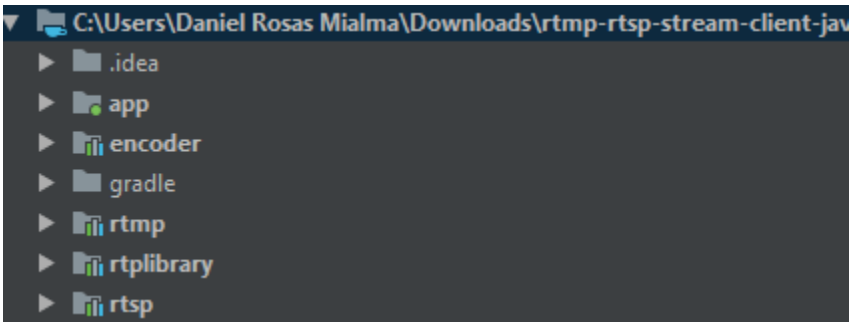
Al finalizar de abrir el proyecto se mostrará una estructura similar a la siguiente.



4. nos situaremos en el Icono desplegaba de Android, para seleccionar **el icono de Project Files**.



Para que nos muestra una estructura similar a la siguiente, con la cual podemos manipular mejor las carpetas.

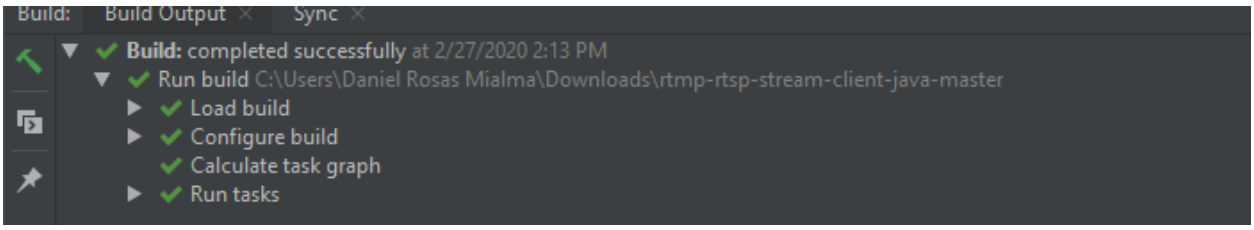


El proyecto contiene las siguientes 6 carpetas:

- **.idea.** La carpeta idea contienen información del proyecto y también información propia de la maquina en la que fue creada el proyecto
- **App.** La carpeta app contiene todo lo relacionado a los ejemplos proporcionados por el autor que se pueden realizar con la librería.
- **Encoder.** La carpeta encoder contiene todas las clases que permiten realizar el proceso de encoding.
- **Grandle.** En esta carpeta se almacenan una serie de ficheros Gradle que permiten compilar y construir la aplicación.
- **Rtmp.** La carpeta Rtmp contiene todas las clases que permiten crear paquetes de RTMP de audio y video para ser encapsulado y poder ser enviados al servidor
- **Rtplibrary.** La carpeta RtpLibrary contienen las clases que permiten construir la forma de como se obtiene el audio y video para que posteriormente pueda ser codificado y enviado al servidor.
- **Rtsp.** La carpeta Rtsp contiene todas las clases que permiten crear paquetes de RTP de audio y video para ser encapsulado y poder ser enviado al servidor

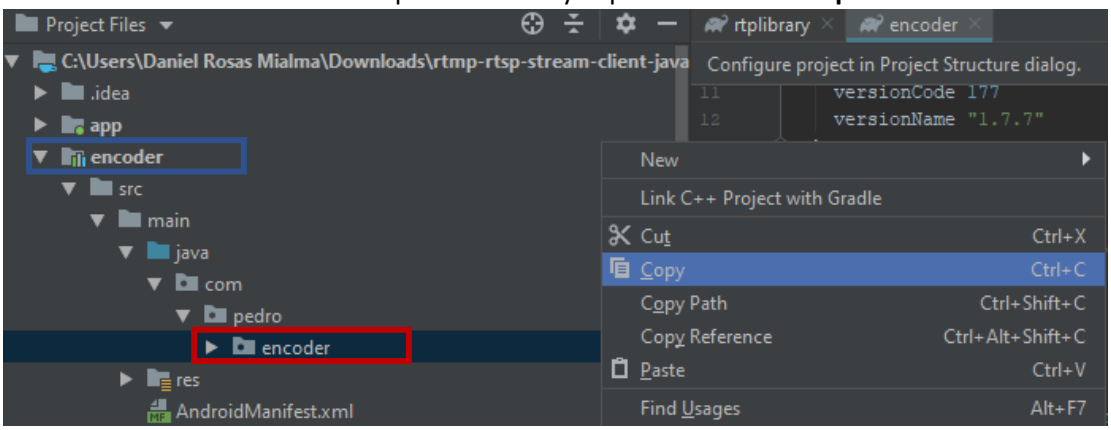
7.- Compilar el proyecto dando click al icono de martillo de color verde.

8. verificar que la compilación no tenga errores.

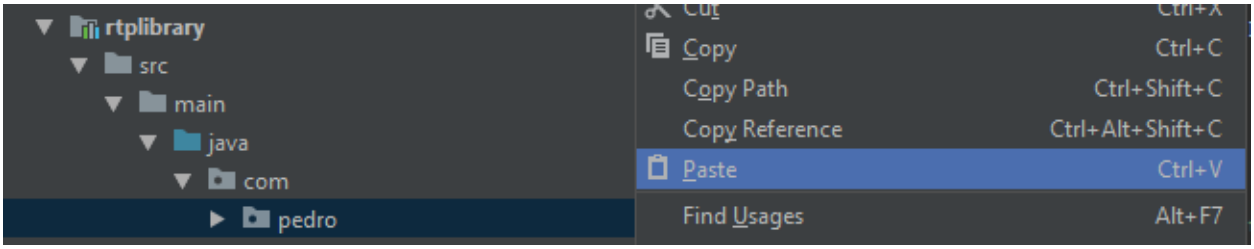


Agrupar librería Encoder dentro de Rtplibrary.

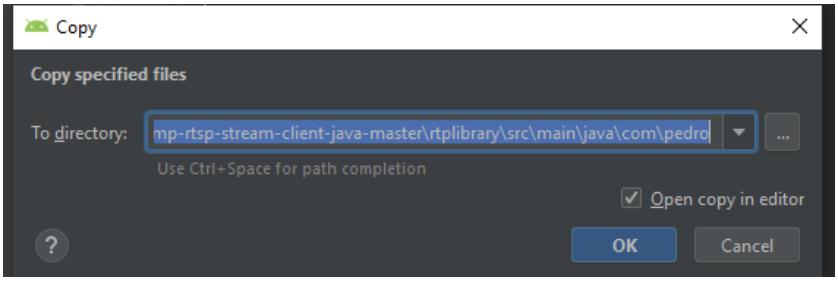
1. Primero nos situaremos en la carpeta **encoder** y copiaremos la **subcarpeta encoder.**



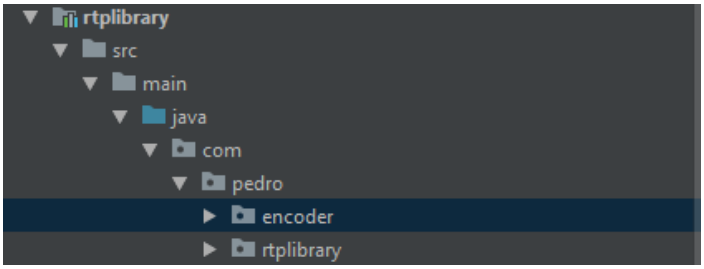
2. Ahora nos dirigimos hacia la carpeta **rtplibrary** y nos situaremos en la subcarpeta **pedro** y daremos click derecho y posteriormente Pegar.



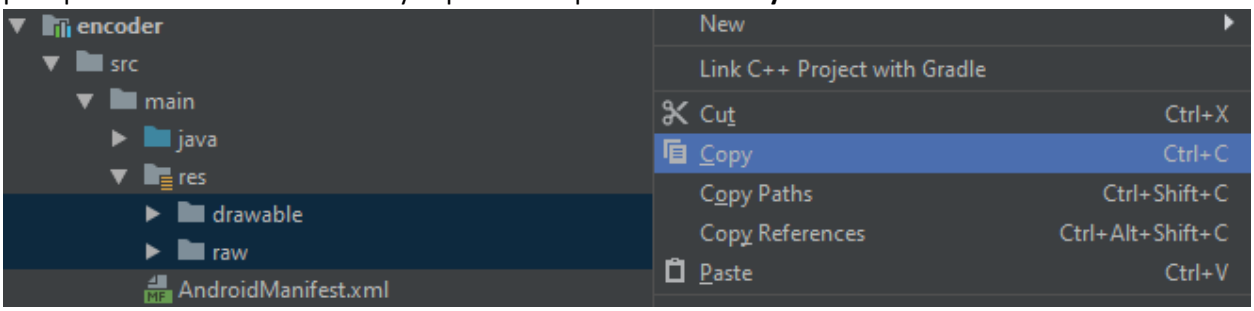
3. Nos mostrara una ventana emergente, le daremos ok y empezara a copiar la carpeta **encoder** hacia la **subcarpeta pedro.**



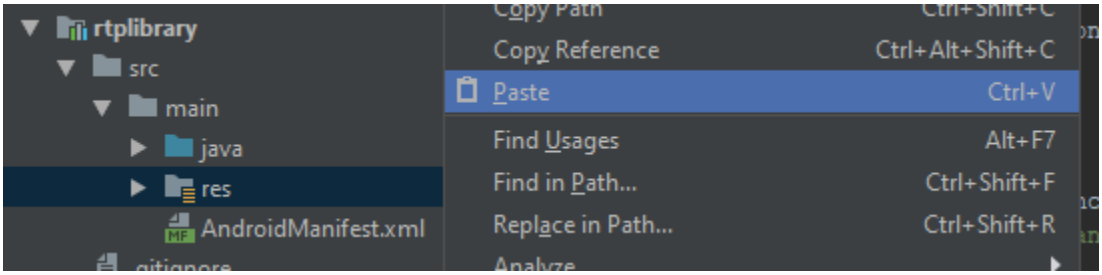
4. Al finalizar se podrá notar que en la estructura de la carpeta rtplibrary se encuentra también la carpeta **encoder**



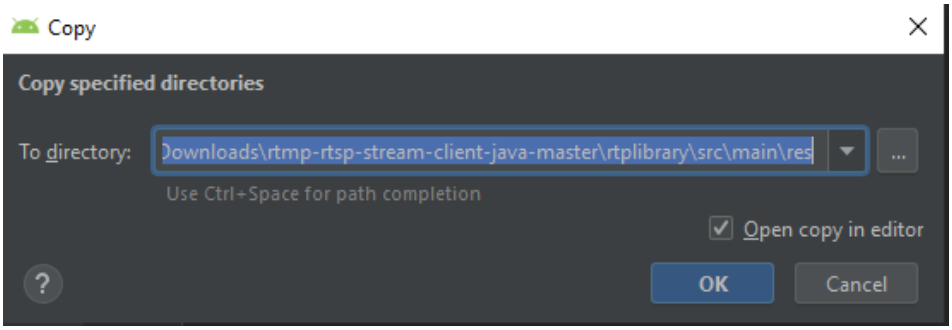
5. Nos volveremos a situar en la carpeta **encoder**, pero ahora nos dirigiremos hacia la carpeta res, para posteriormente seleccionar y copiar las carpetas **drawable y raw.**



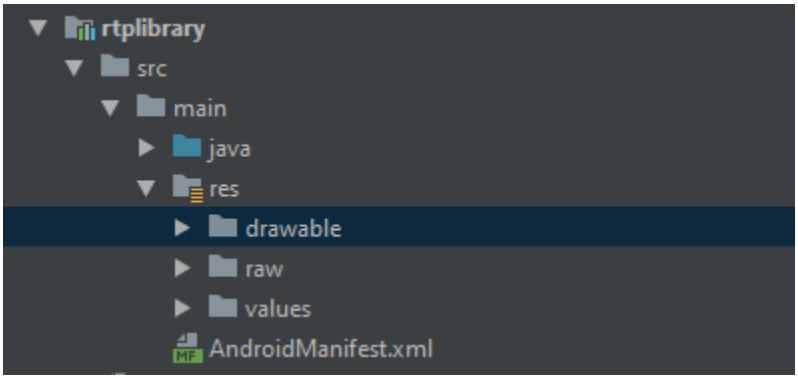
6. Ahora abriremos la carpeta rtplibrary, pero nos situaremos en la subcarpeta **res** y daremos click derecho y posteriormente Pegar.



7. Nos mostrara una ventana emergente, le daremos ok y empezara a copiar las carpetas **raw y drawable** hacia la subcarpeta **res.**



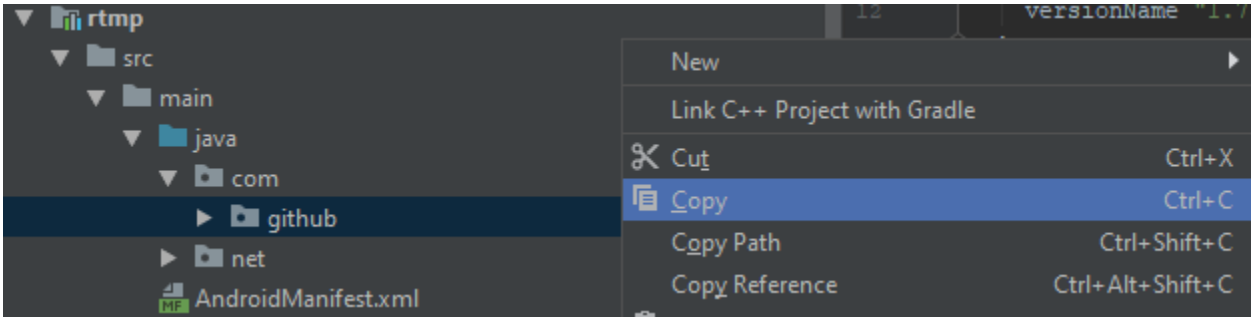
8. Al finalizar se podrá notar que en nuestra estructura de la carpeta **rtplibrary** se encuentra también las carpetas **raw y drawable**



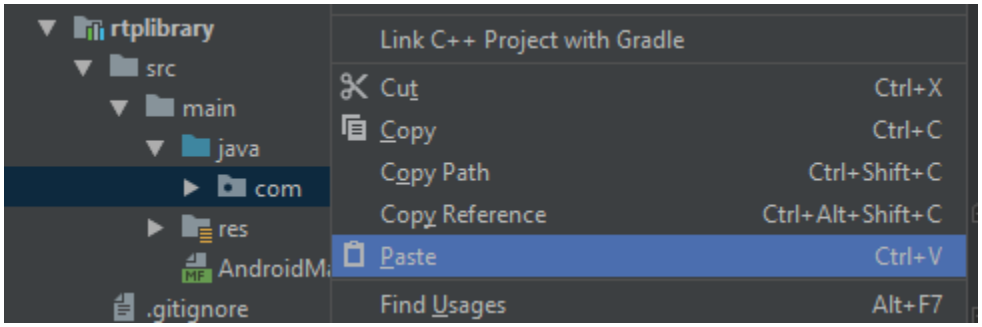
Una vez terminado de agrupar la carpeta **encoder,** podremos continuar con la carpeta **rtmp**

Agrupar librería rtmp dentro de rtplibrary

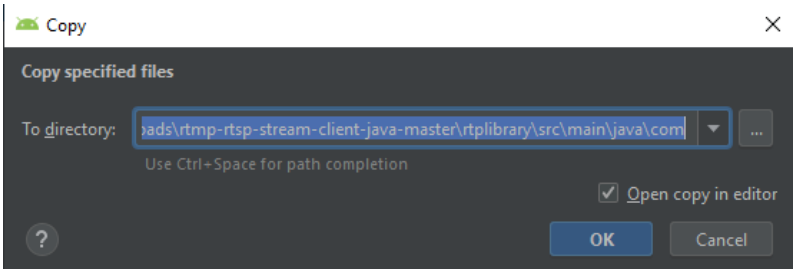
1. Nos situaremos en la carpeta **rtmp** y copiaremos la **subcarpeta github.**



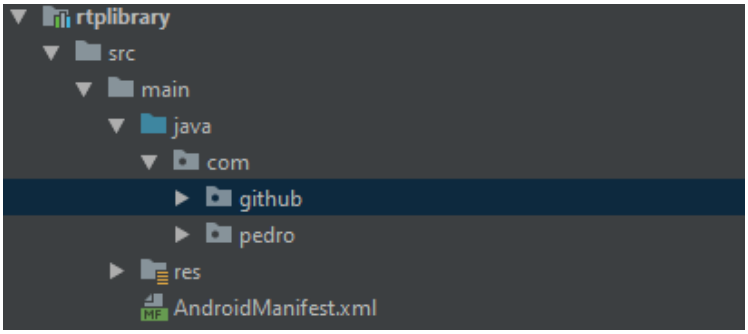
2. Ahora nos dirigimos hacia la **carpeta rtplibrary** y nos situaremos en la **subcarpeta com** y daremos clic derecho y posteriormente Pegar.



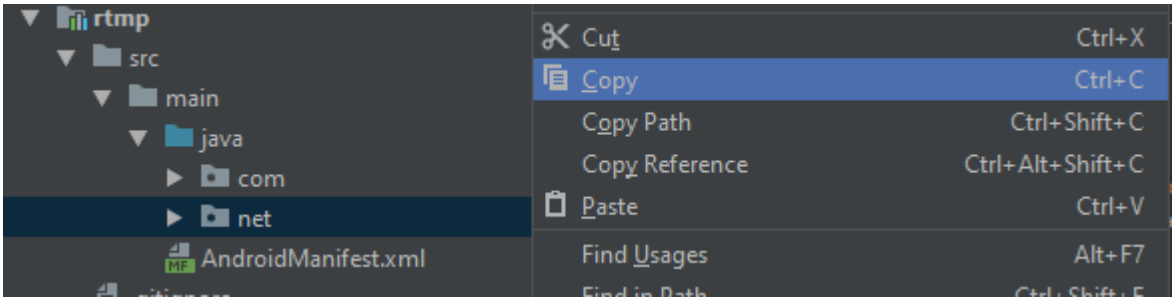
3. Nos mostrara una ventana emergente, le daremos ok y empezara a copiar la **carpeta github** hacia la **subcarpeta com.**



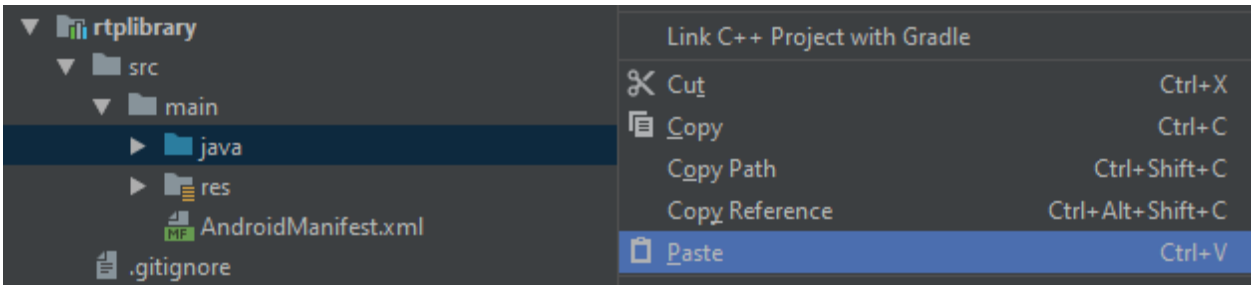
4. Al finalizar se podrá notar que en nuestra estructura de la carpeta **rtplibrary** se encuentra también la carpeta **github dentro de la subcarpeta com.**



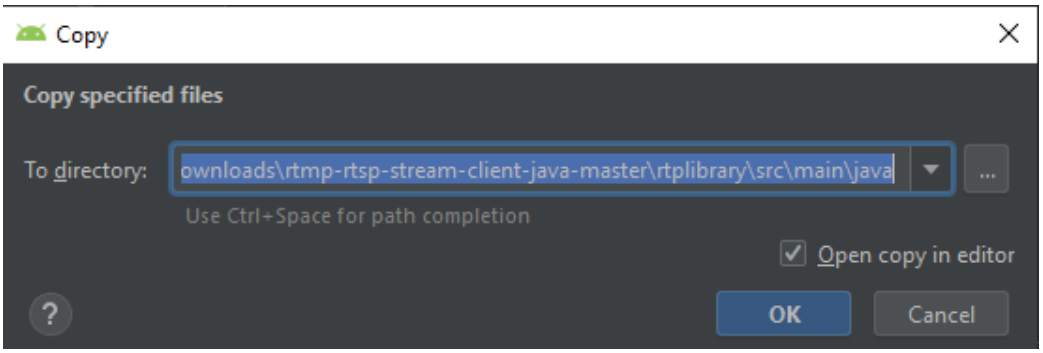
5. Nos volveremos a situar dentro en la **carpeta rtmp** y copiaremos la **subcarpeta net.**



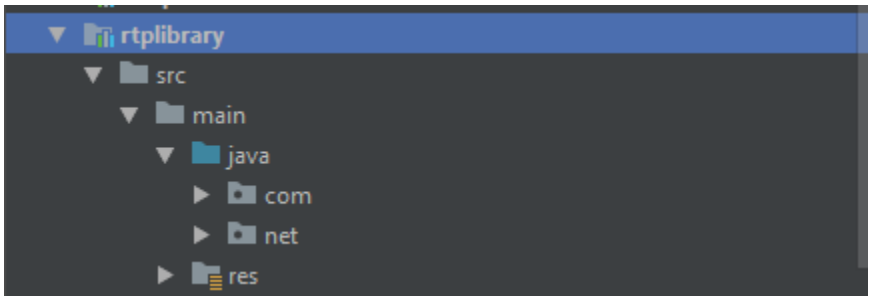
6. Ahora nos dirigimos hacia la car**peta rtplibrary** y nos situaremos en la **subcarpeta java** y daremos clic derecho y posteriormente Pegar.



7. Nos mostrara una ventana emergente, le daremos ok y empezara a copiar la **carpeta net** hacia la **subcarpeta java.**

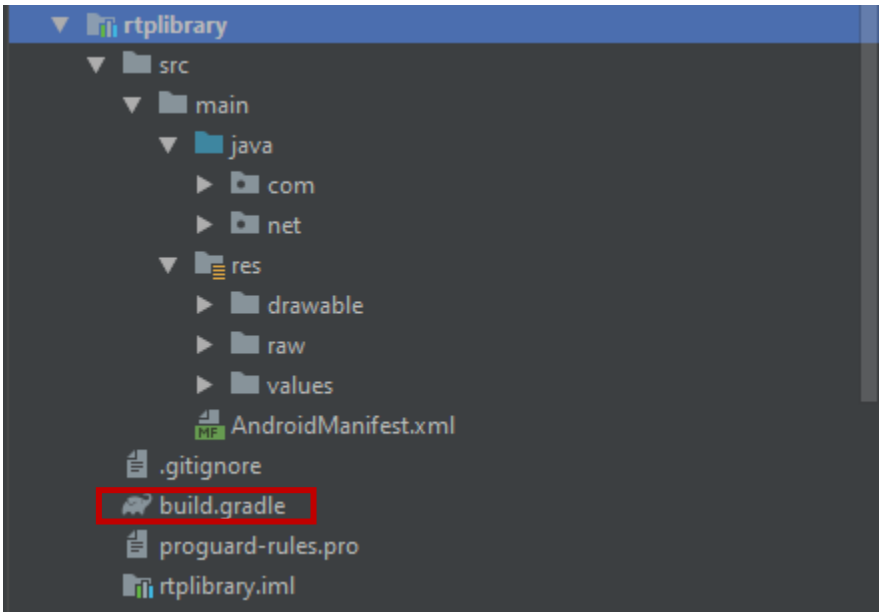


8. Al finalizar se podrá notar que en nuestra estructura de la carpeta maestra **rtplibrary** se encuentra también la carpeta net dentro de la subcarpeta **java.**



Modificar el archivo Build Grandle

Realizaremos algunos cambios en el archivo **build grandle (**Es un archivo de configuración funciona con el paquete de herramientas de compilación a fin de proporcionar procesos y ajustes configurables específicos para la compilación y prueba de apps para Android.)



1. Abre el archivo build.grandle
2. Al final del archivo localiza el apartado de *dependencies.*

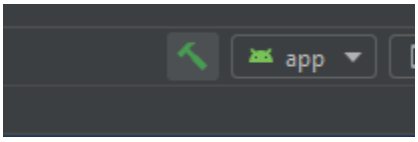
```
dependencies {
    api project(':encoder')
    api project(':rtmp')
    api project(':rtsp')
}
```

3. Remplaza el contenido del apartado *dependencies* con el siguiente para poder quitar las dependencias de **las carpetas que agrupamos en el proyecto rtplibrary.**

```
}

dependencies {
    api 'androidx.annotation:annotation:1.1.0'
    api project(':rtsp')
}
```

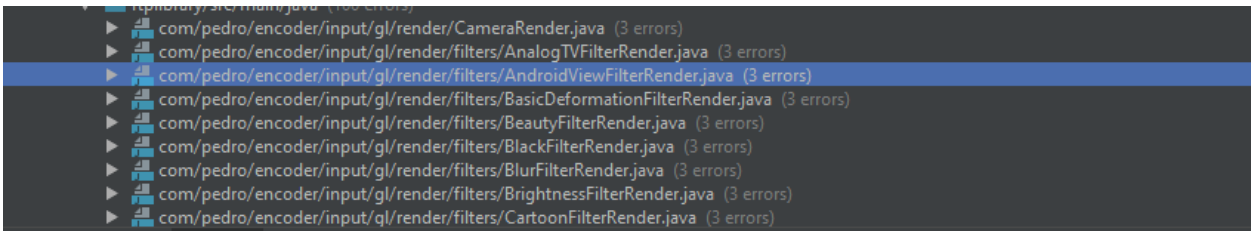
(**Observación** podrás notar que aún se encuentra una dependencia al proyecto rtsp, en nuestro caso como no vamos a hacer uso del protocolo **RTP**, no necesitamos agrupar esa librera, a menos que se quiera utilizar se realizaría el mismo proceso que con las anteriores y se quitaría esa dependencia.)

4. Compila el proyecto dando click en el icono de martillo de color verde.



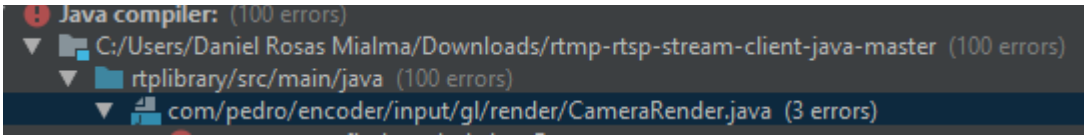
Solucionar problemas de compilación.

Al termina de compilar si realizaste los pasos anteriores de manera adecuada se mostrará un listado de errores, los cuales se generaron dentro de las clases de la carpeta encoder. Todos los errores corresponden a un espacio de nombre que ya no puede ser encontrado, porque agrupamos todas las dependencias dentro del proyecto rtplibrary.

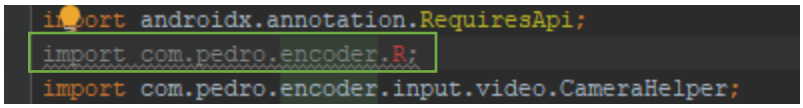


Realiza los siguientes pasos para solucionar los errores de compilación.

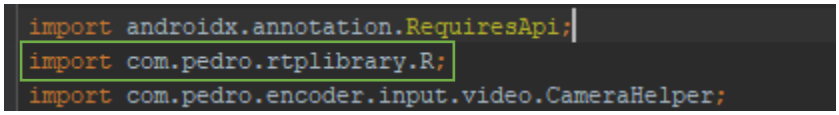
1. Daremos doble click a un error.



2. Se abrirá un archivo que contiene una clase mostrando el error que corresponde a un espacio de nombre.

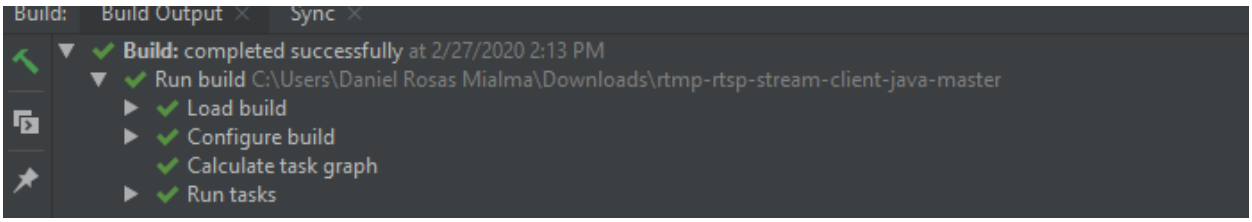


3. Para solucionarlo Modifica la línea **import com.pedro.encoder.R** por la siguiente.



4. La línea agregada permitirá tener acceso a los recursos del espacio de nombre que no encontraba.

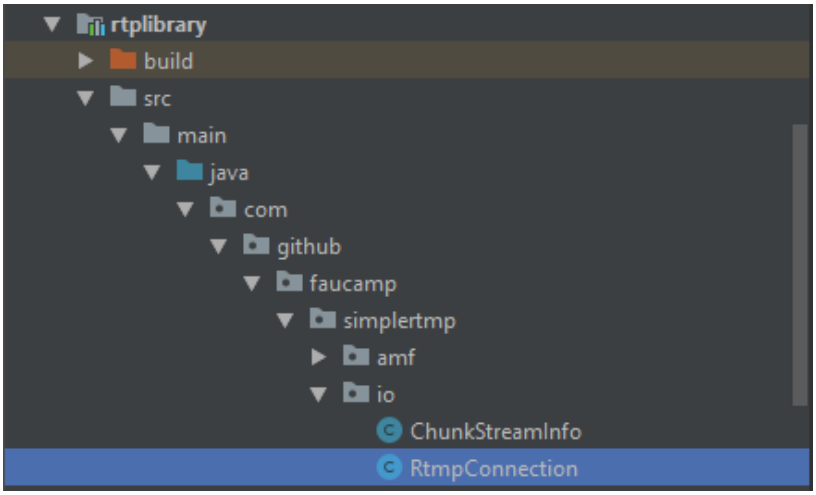
5. Compilaremos el proyecto y podremos notar que uno de los errores de la lista se solucionó, para arreglar los demás realizaremos el mismo proceso. Hasta que nuestro proyecto compile sin errores.



Refactorizar el código para aceptar URLs provenientes de Azure

En esta parte modificaremos la clase RtmpConection que nos permitirá conectarnos tanto en el servicio de **Azure Media Services** como hacia otros servidores de streaming tradicionales.

1. Abre el archivo **RtmpConnection**



2. Modifica el contenido de la variable **rtmpUrlPattern de la clase RtmpConnection** por lo siguiente **"^rtmps?://([^/:]+)(?::(\\d+))/?([^*]*)$*/([^/]+)".**

```
/**
 * Main RTMP connection implementation class
 *
 * @author francois, leoma, pedro
 */
public class RtmpConnection implements RtmpPublisher {

  private static final String TAG = "RtmpConnection";
  private static final Pattern rtmpUrlPattern =
      Pattern.compile("^rtmps?://([^/:]+)(?::(\\d+))/?([^*]*)$*/([^/]+)");
```

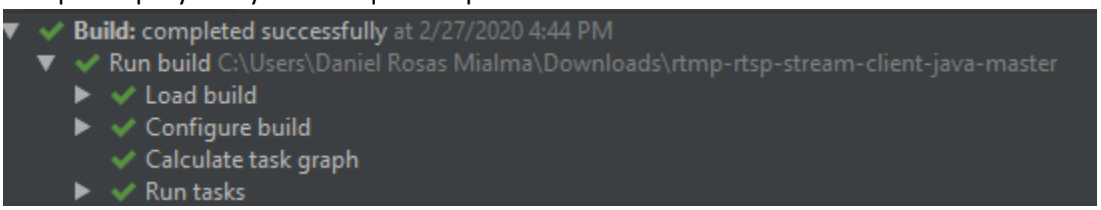
Se modifico la expresión regular para aceptar URLs provenientes de Azure.

3. Modifica la asignación de la variable **tcUrl** de la clase **RtmpConnection** por lo siguiente

“rtmpMatcher.group(0).substring(0,rtmpMatcher.group(0).length()-(streamName.length()+1));”

```
String portStr = rtmpMatcher.group(2);
port = portStr != null ? Integer.parseInt(portStr) : 1935;
appName = rtmpMatcher.group(3);
streamName = rtmpMatcher.group(4);
tcUrl = rtmpMatcher.group(0).substring(0,rtmpMatcher.group(0).length()-(streamName.length()+1));
```

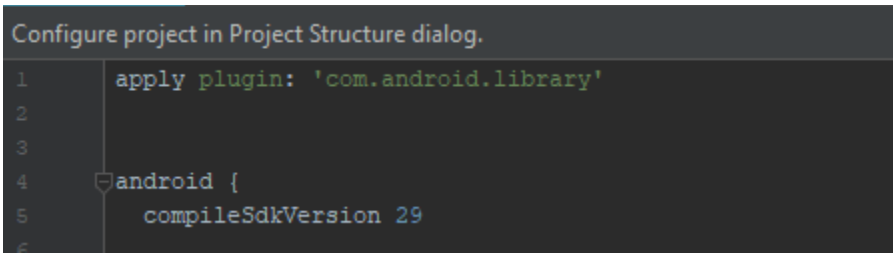
4. Compila el proyecto y verifica que compile correctamente.



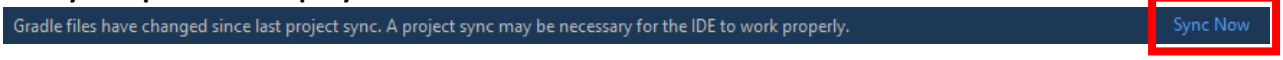
Generar un AAR de Rtplibrary

Como podemos suponer, nuestra librería esta lista para generar un archivo **AAR** en términos de C# un AAR vendría siendo una **DLL.**

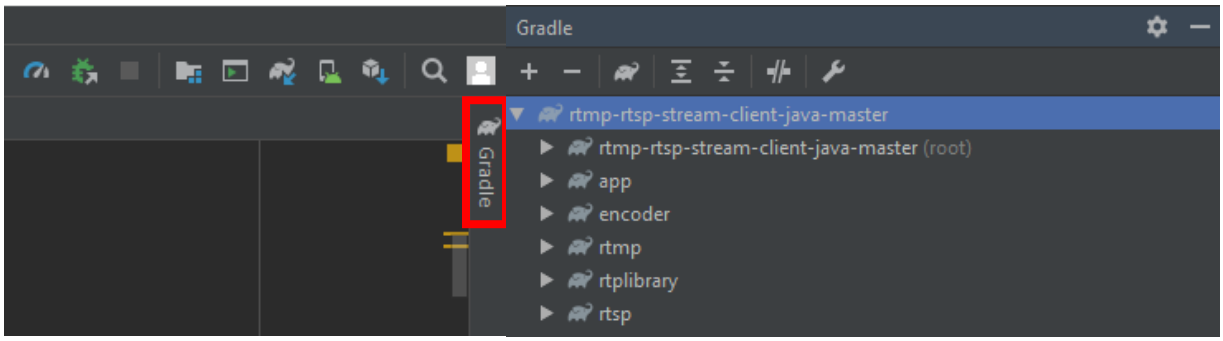
1. Abre el archivo build.grandle de la **carpeta RtpLibrary** y modificar su contenido en la parte superior para que sea similar al siguiente.



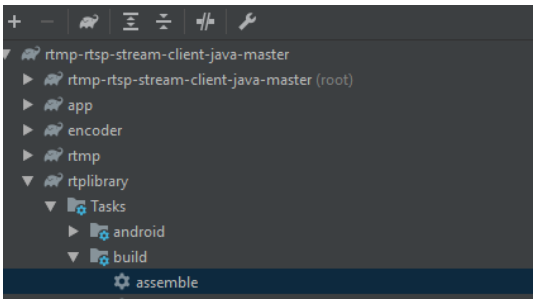
2. **Se mostrará una notificación de que habido cambios en el proyecto, daremos click en Sync Now y Compilaremos el proyecto**



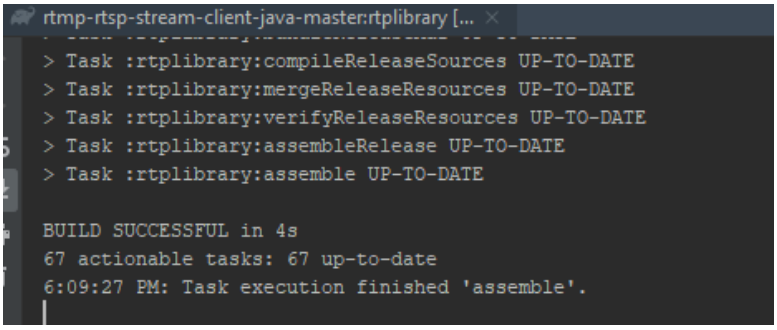
3. Haz Click en la pestaña Grandle, se mostrará una estructura similar a la siguiente.



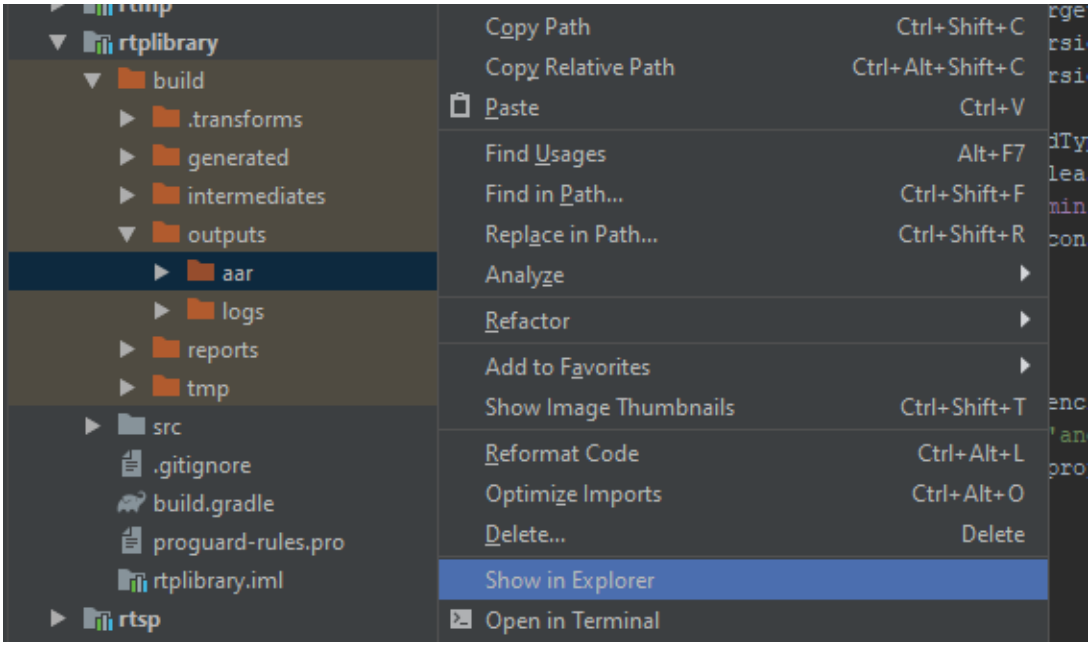
4. Nos situaremos en el apartado de RtpLibrary, posteriormente abriremos los directorios Task → Build y daremos click en **Assemble.**



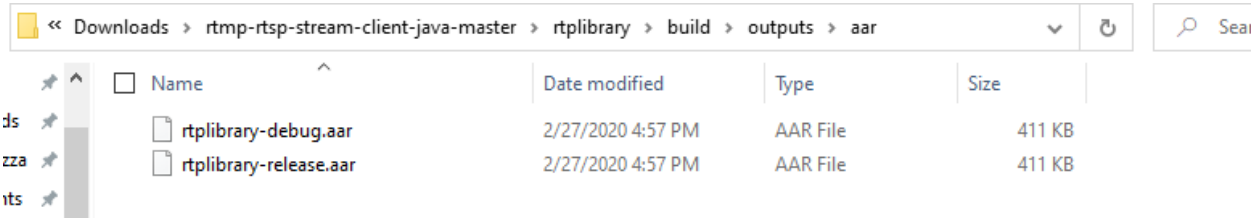
5. Si nuestro ensamblado se generó correctamente nos mostrar una pantalla similar a la siguiente en la parte inferior de Android Studio.



6. Para localizar donde se genero el AAR de la librería, nos situaremos en la carpeta **rtplibrary** abriremos los directorios **build -> outputs** y daremos click derecho al directoria **aar** a continuación daremos click en **Show in Explore.**



7. Se abrirá el explorador de Windows a continuación daremos click en la carpeta **AAR** y podremos encontrar el **AAR** la versión debug y reléase de **rtplibrary**

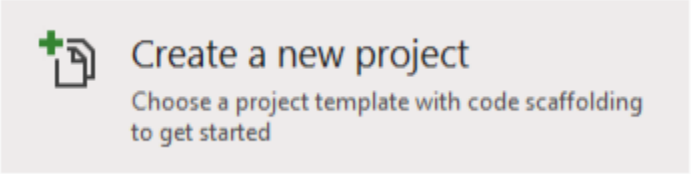


Ahora podremos encapsular el AAR en una Librería Xamarin Android Bidings y hacer uso de este componente en un Proyecto Xamarin Android.

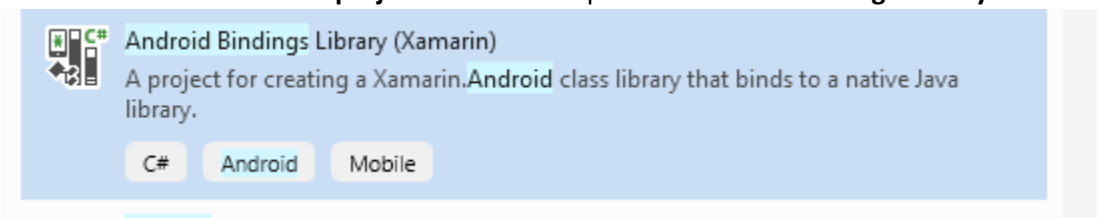
Encapsular AAR RtpLibrary en una Librería Xamarin Android Bindings Library.

Empecemos por crear la aplicación utilizando la plantilla Android Bindings Libray.

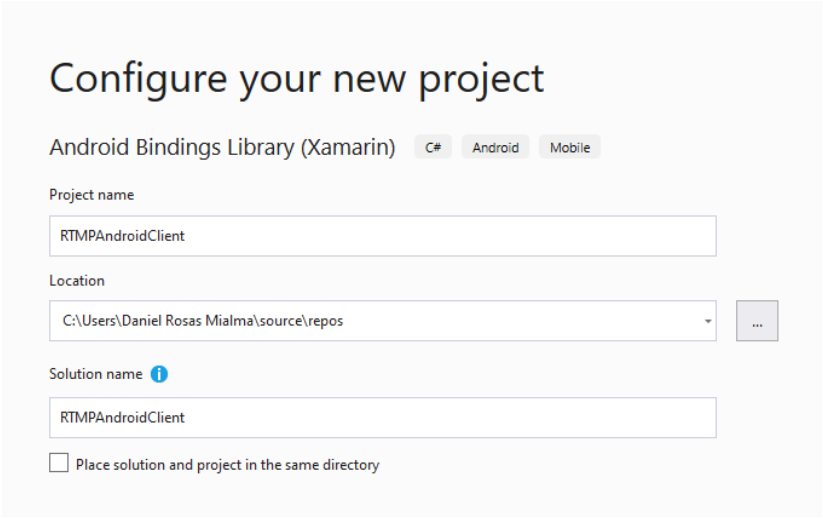
1. Abre Visual Studio bajo el contexto del Administrador.
2. En la ventana de inicio selecciona la opción **Create a new project.**



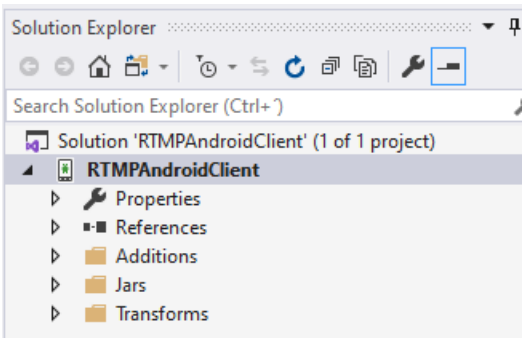
3. En la ventana **Create a new project** selecciona la plantilla **Android Bindings Library**



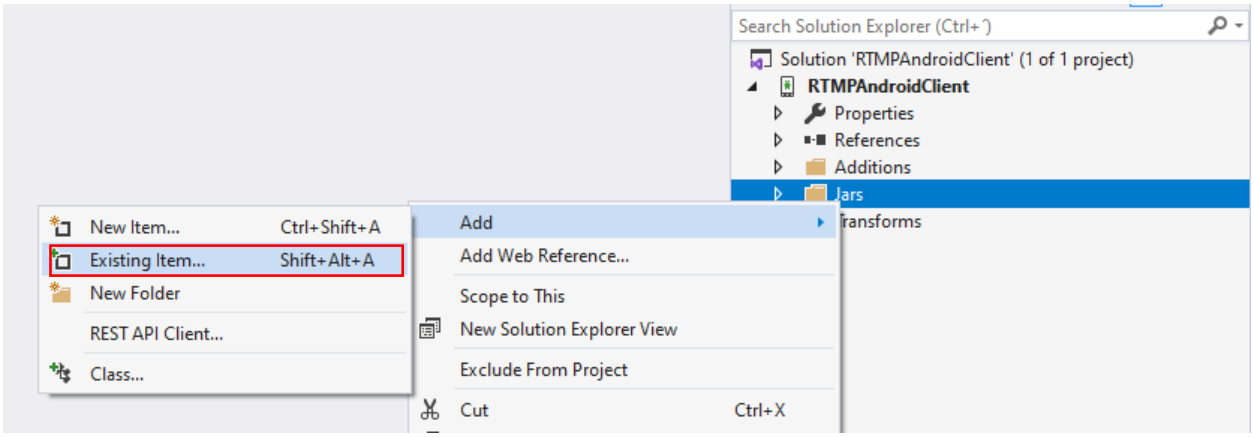
4. En la ventana **Configure your new project** proporciona **RTMPAndroidClient** como el nombre del proyecto, selecciona la ubicación y haz clic en Create para continuar.



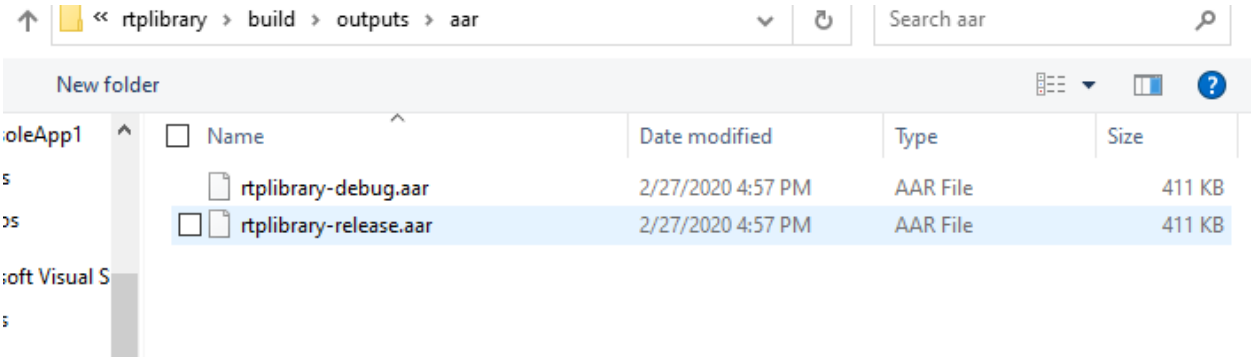
Al finalizar la creación de la solución se mostrará una estructura similar a la siguiente.



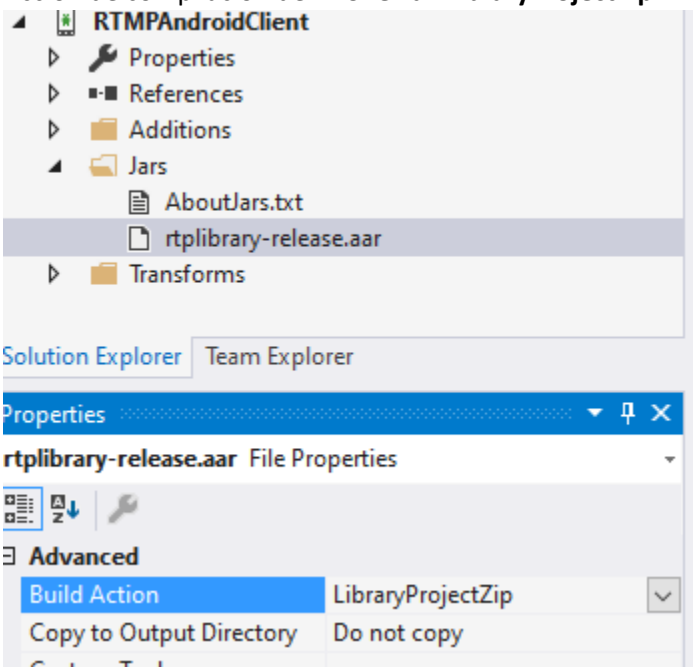
5. En el proyecto RTMPAndroidClient nos situaremos en el directorio Jars a continuación, daremos click derecho -> add -> Existing Item.



6. Se mostrar el explorador de Windows y agregaremos el **AAR** rtplibrary que generamos anteriormente.

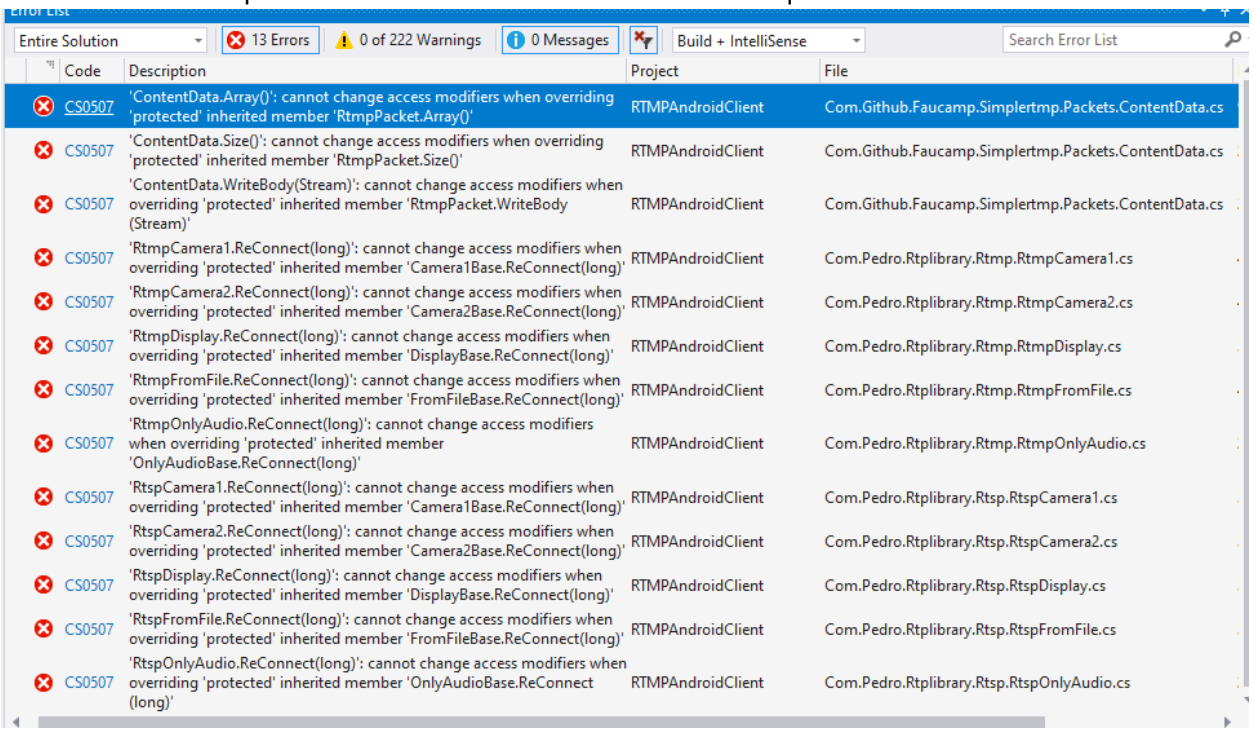


7. Daremos un clic en la librería que acabamos de agregar a nuestro proyecto y cambiaremos su Acción de compilación de **“None” a “LibraryProjectZip”**



8. Compilamos nuestra librería.

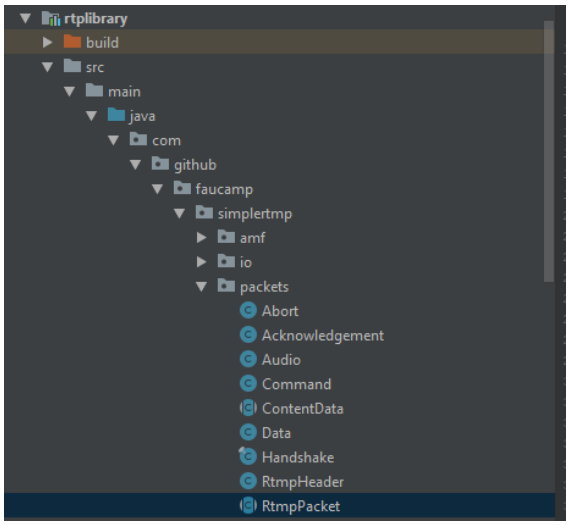
9. Al momento de compilar nuestra librería nos daremos cuenta de que tiene en listado 13 Errores.



Estos errores se generan porque la librería nativa tiene métodos con un acceso protect, en mi caso por cuestiones de simplicidad para solucionar este error, regresare a Android Studio para modificar cada una de las clases que marcan error y cambiare el tipo de acceso de los métodos por **Public.**

Solucionar los errores que se generan al encapsular la librería nativa

1. Nos dirigiremos a nuestro proyecto en Android Studio.
2. Abriremos el archivo **RtmpPacket** alojado en el directorio **packets** de rtplibrary.



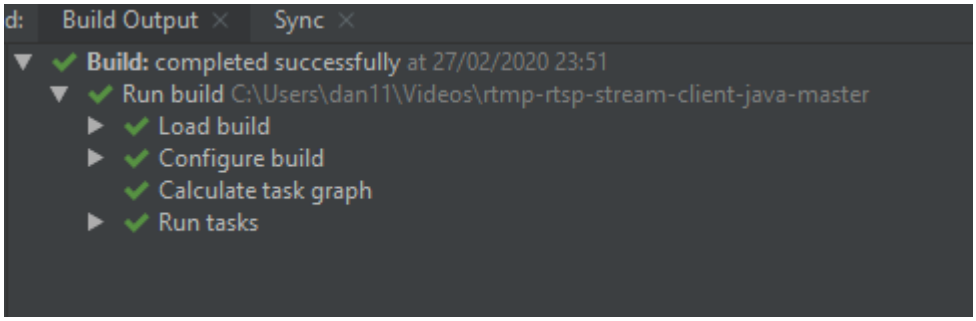
3. Localizaremos los métodos ***writeBody, array(), y size()*** podremos notar que estos 3 métodos son los mismo que ocasionan que la encapsulación de nuestra librería genere un error al compilar.

```
protected abstract void readBody(InputStream in) throws IOException;

protected abstract void writeBody(OutputStream out) throws IOException;

protected abstract byte[] array();
```

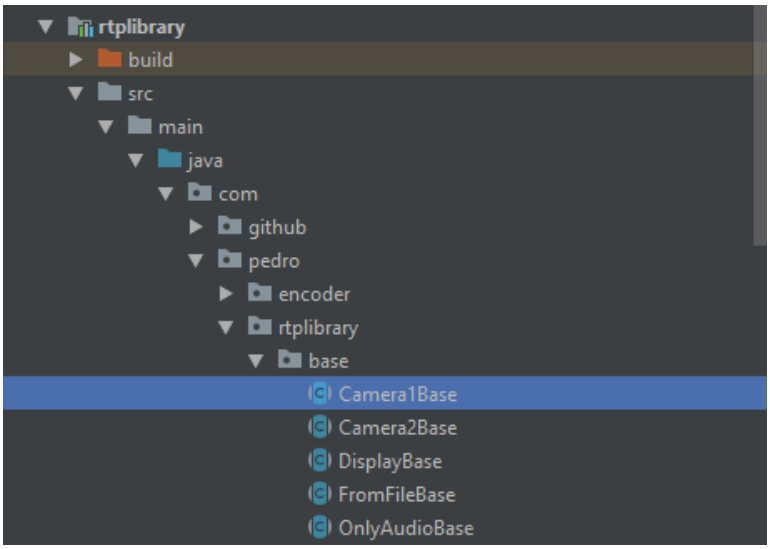
4. modificaremos el tipo de acceso protected por public para evitar los errores que se generan al encapsular la librería.

```
public abstract void readBody(InputStream in) throws IOException;

public abstract void writeBody(OutputStream out) throws IOException;

public abstract byte[] array();
```

5. guardaremos cambios y realizaremos los mismos pasos con los siguientes archivos alojados en el mismo directorio.
    - WindowAckSize
    - User Control
    - SetPeerBandWitdth
    - SetChunkSize.
    - Data
    - ContentData
    - Command
    - Acknowledgement
    - Abort
6. Compilaremos el proyecto y verificamos que no tenga errores.



7. Para solucionar el problema con el Método **ReConnect** abriremos el archivo **Camera1Base** alojado en el directorio **base** de **rtplibrary.**



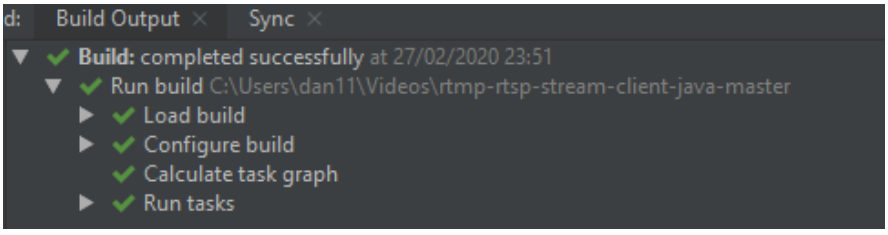
8. Localizaremos el método **ReConnect**

```
547  public abstract boolean shouldRetry(String reason);
548
549  public abstract void setReTries(int reTries);
550
551  protected abstract void reConnect(long delay);
552
```

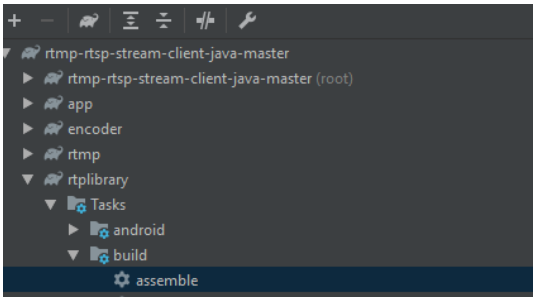
9. Modificaremos el tipo de acceso protected por public para evitar los errores al encapsular la librería.

```
549  public abstract void setReTries(int reTries);
550
551  public abstract void reConnect(long delay);
552
553  //cache control
```

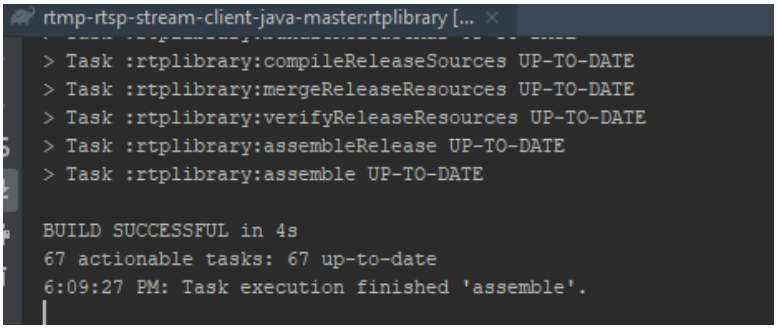
10. guardaremos cambios y realizaremos los mismos pasos con los siguientes archivos alojados en el mismo directorio.
    - Camera2Base
    - DisplayBase
    - FromFileBase
    - OnlyAudioBase.
11. Compilaremos el proyecto y verificamos que no tenga errores.



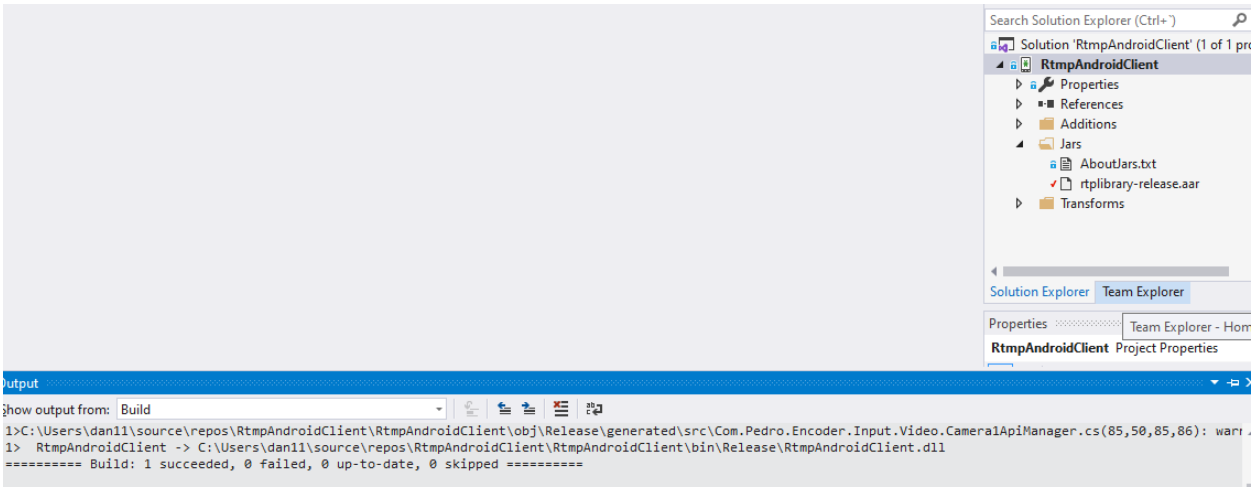
12. Una vez terminado de modificar las clases anteriores volveremos a generar el AAR
13. Abriremos la pestaña Gradle a continuación nos situaremos en el apartado de RtpLibrary, posteriormente abriremos los directorios Task → Build y daremos click en **Assemble.**



14. Si nuestro ensamblado se generó correctamente nos mostrar una pantalla similar a la siguiente en la parte inferior de Android Studio.



15. Regresar a nuestro proyecto dentro de visual Studio.
16. Remplazar el viejo AAR por el nuevo.
17. Cambiaremos su Acción de compilación de **“None” a “LibraryProjectZip”**
18. Limpiaremos la Solución y compilaremos, si los anteriores pasos se hicieron de manera correcta nuestra librería habrá compilado exitosamente.



Con esto terminamos el encapsulamiento de nuestra librería Xamarin Android Bindings ahora podremos utilizar nuestro componente en cualquier proyecto Xamarin Android y poder generar un Streaming.