PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES

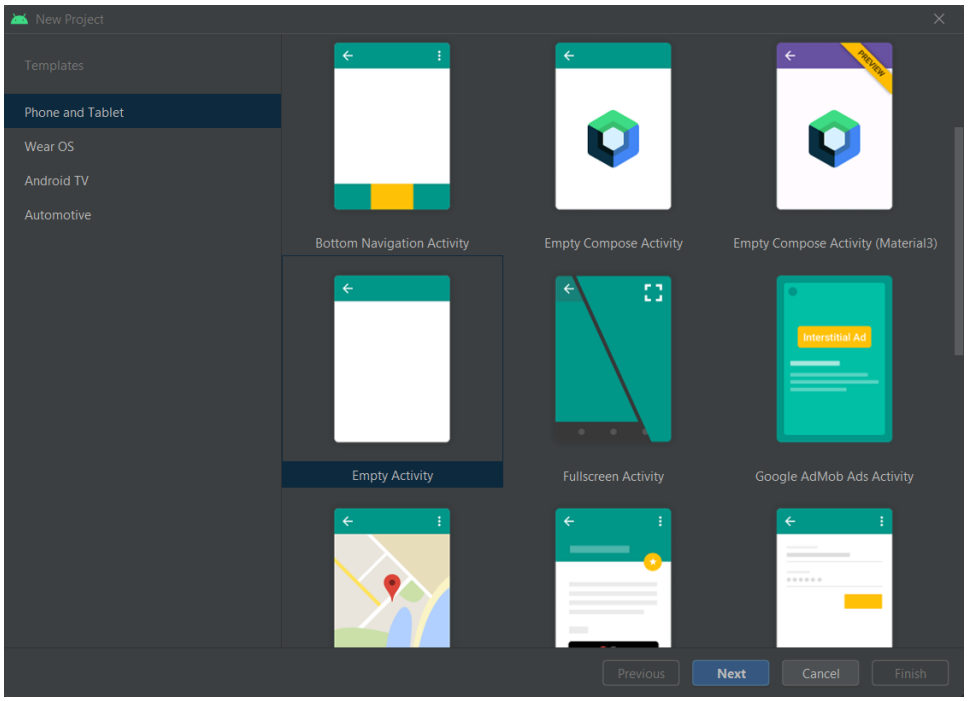
Nayra Infantes Tardón

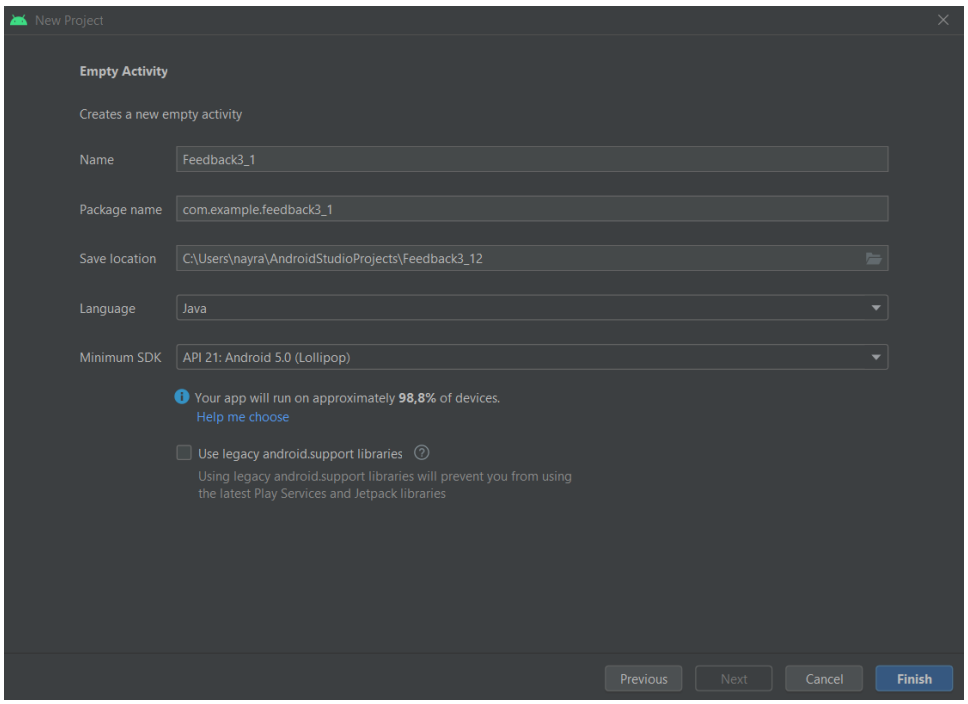
CFP/UAX
Centro de Formación Profesional

Ejercicio feedback unidad 3

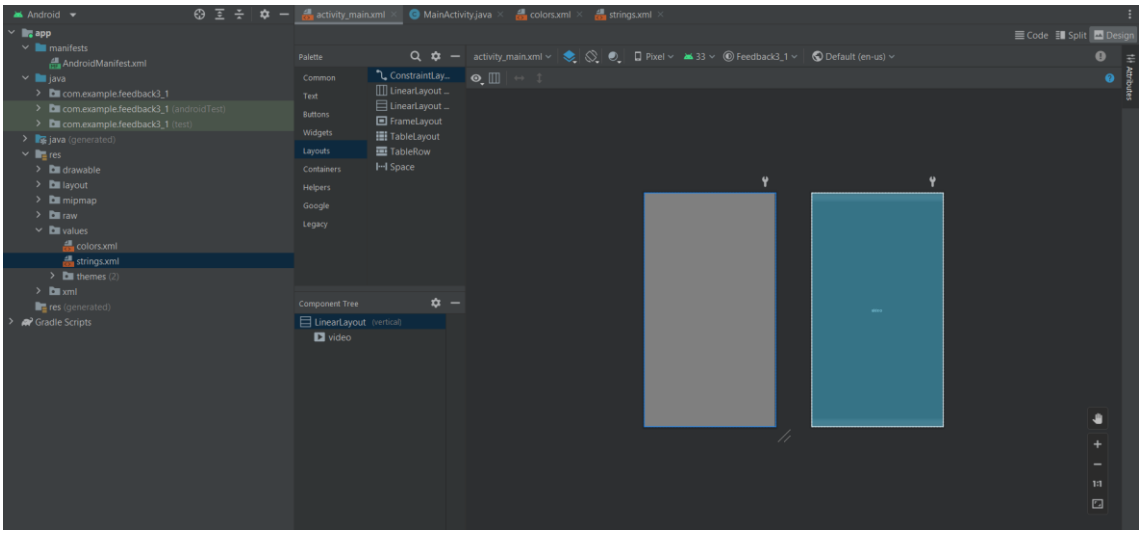
Realizar una app que reproduzca un vídeo, usando para ello el componente VideoView.

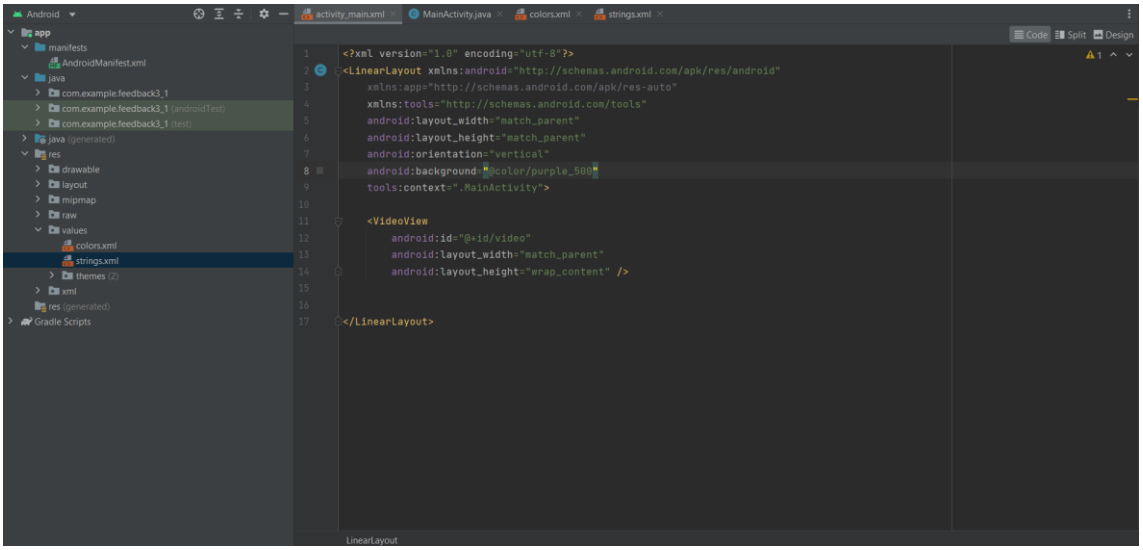
Para poder diseñar nuestra aplicación lo primero que debemos hacer es crear un nuevo proyecto, en nuestro caso se llamará Feedback3_1, partiremos de una actividad vacía para poder tener una base previa, y el lenguaje a utilizar será Java.





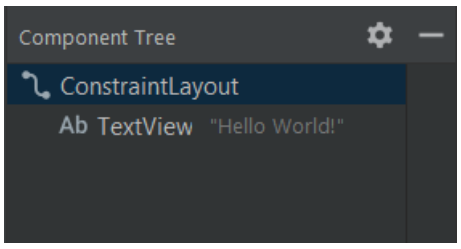
Gracias a la interfaz que nos proporciona Android Studio podemos crear nuestro VideoView desde de una parte gráfica en la que funciona todo de una manera muy intuitiva o directamente a través de código en nuestro archivo activity_main.xml.



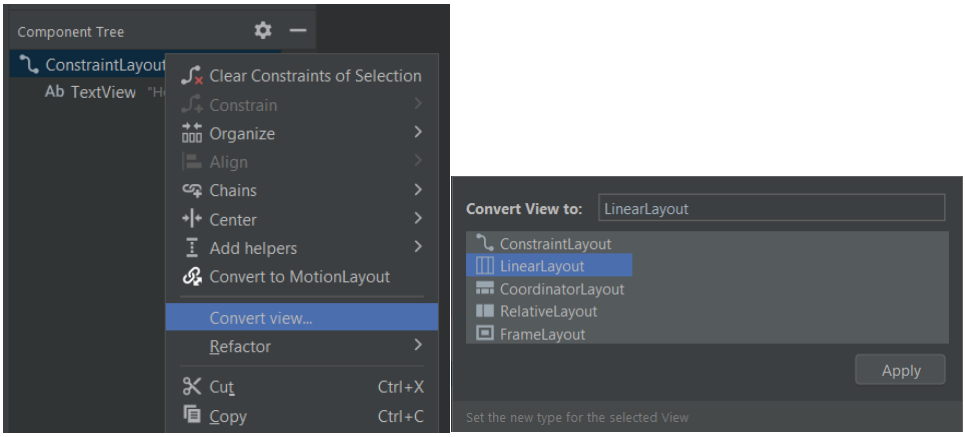


```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/purple_500"
    tools:context=".MainActivity">

    <VideoView
        android:id="@+id/video"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />


</LinearLayout>
```

Por defecto toda nuestra aplicación estará contenida en un ConstrainLayout, que nos permitirte posicionar y redimensionar views de forma flexible a partir de una gran variedad de reglas de restricción.



Cambiaremos esta selección a un LinearLayout, que es un grupo de vistas que alinea todos los elementos secundarios en una única dirección, de manera vertical u horizontal, esto nos ayudará a simplificar nuestro diseño, ya que lo ideal para la vista de nuestra aplicación, es que primero se muestre el vídeo y bajo este, sus controles.



Tal y como indica el enunciado debemos utilizar el componente VideoView que permite reproducir dentro de él un vídeo que se asignará en el archivo MainActivity.java.
Esta etiqueta tendrá un id que será video; un layout width o ancho del grupo, en este caso match parent, es decir, la vista se expande lo máximo posible dentro del Linear Layout; y, además, un layout height, que define el alto, en el que indicaremos a la vista que establezca el tamaño necesario para ajustar el contenido gracias a wrap content.

Como detalles opcionales, en nuestro caso hemos decidido añadir un fondo con color al LinearLayout que contiene todo lo demás con “android:background" para proporcionarle un mejor aspecto a la aplicación.

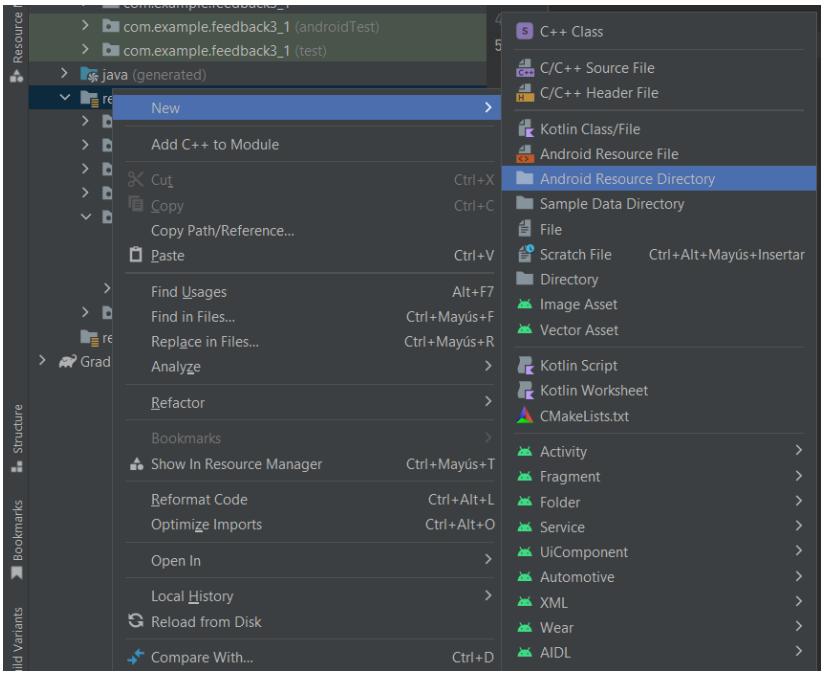
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/purple_500"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <VideoView
        android:id="@+id/video"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

También hemos modificado algunos valores por defecto, como los colores de la parte de la app que se ubica en la zona superior de nuestra pantalla y el nombre de la app para que muestre la palabra “CACHORRO”, estos cambios se realizan en los archivos strings.xml y colors.xml, ubicados dentro de la carpeta Values que a su vez se encuentra en la carpeta res.

```xml
<resources>
    <string name="app_name">CACHORRO</string>
</resources>
```

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFFFFF</color>
    <color name="purple_500">#494949</color>
    <color name="purple_700">#000000</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#016C87</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

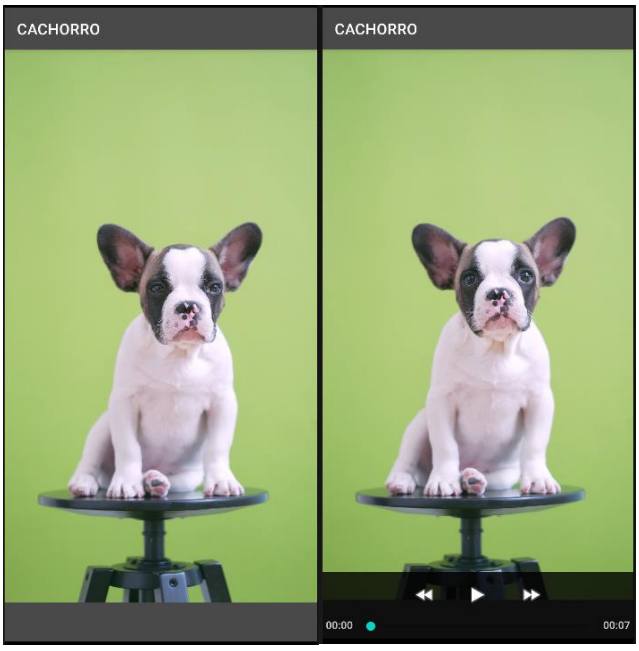
Cuando tengamos, por así decirlo, la parte más visual terminada, debemos añadir a nuestra carpeta res una nueva carpeta, esta carpeta debe ser de tipo raw, la nombraremos de igual forma para evitar posibles errores, y va a ser la encargada de almacenar nuestro vídeo en formato .mp4.



Una vez completados los pasos anteriores comenzaremos a trabajar en nuestro archivo ActivityMain.java.
Debemos crear la variable VideoView del mismo tipo e importar la clase para poder referenciar nuestro contenedor de vídeo, dado que queremos mostrar nuestro video en un “recuadro” haremos la referencia al layout con VideoView a través del id con el que antes lo hemos nombrado, que es video.
Una vez referenciado el VideoView, enlazaremos el recurso, es decir, nuestro archivo de vídeo.
La clase para poder añadir unos controles básicos al vídeo es MediaController, debemos importarla y asignarle el VideoView que hemos creado.

```java
package com.example.feedback3_1;

import androidx.appcompat.app.AppCompatActivity;

import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class MainActivity extends AppCompatActivity {

    VideoView videoView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        videoView = (VideoView) findViewById(R.id.video);

        videoView.setVideoURI(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.cachorro));

        MediaController mediaController = new MediaController( context: this);
        videoView.setMediaController(mediaController);

        videoView.start();
    }
}
```

Nuestra aplicación debe mostrar pues, un video y sus controles de la siguiente forma:



Realizar una app que grabe audio. La interfaz debe tener al menos dos botones: uno para iniciar la grabación y otro para parar.

Para la realización de este ejercicio utilizaremos un ConstrainLayout, así podremos posicionar de mejor manera nuestros botones.
Necesitaremos crear un cronómetro para que el usuario tenga constancia de que efectivamente se está grabando su pista de audio, un botón dinámico que utilizaremos tanto para iniciar la grabación como para finalizarla, y por último un botón que sirva para iniciar la reproducción del audio. Esto último podremos hacerlo mediante la creación de los respectivos métodos en nuestro archivo MainActivity.java y referenciándolos con la propiedad “android:onClick” en nuestro archivo activity_main.xml.
Como ya hicimos en el anterior feedback, en la ruta Res > Drawable añadiremos las imágenes de los botones y del background, y además modificaremos tanto los colores, como el icono y nombre de la aplicación para proporcionarle un mejor aspecto.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout 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"
    android:background="@drawable/background"
    tools:context=".MainActivity" >

    <Chronometer
        android:id="@+id/crono"
        android:layout_width="163dp"
        android:layout_height="54dp"
        android:fontFamily="sans-serif-black"
        android:textAlignment="center"
        android:textColor="#636363"
        android:textSize="34sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />


    <Button
        android:id="@+id/botonrec"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:background="@drawable/rec"
        android:onClick="grabar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.302"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.64" />

    <Button
        android:id="@+id/botonplay"
        android:layout_width="80dp"
        android:layout_height="80dp"
        android:background="@drawable/play"
        android:onClick="reproducir"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.682"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.64" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

En nuestro fichero ActivityMain.java crearemos los permisos necesarios para que nuestra aplicación funcione de manera correcta, será el usuario quien decida si quiere o no que nuestra aplicación interactúe con su sistema de archivos y el sensor del micrófono, es muy importante nombrarlos también en AndroidManifest.xml.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

La clase MediaRecorder captura audio del micrófono de un dispositivo, lo guarda y lo reproduce posteriormente con ayuda de la clase MediaPlayer. Haremos uso de ambas en este ejercicio.
Necesitamos también crear los botones, el cronómetro y un string que almacene la ruta del audio que grabaremos.
Como siempre, con el comando alt+intro importaremos las clases que necesitemos para la correcta ejecución de nuestra aplicación.

Debemos crear nuestros métodos para grabar y reproducir ya que onCreate es un método en sí mismo.

En el método grabar, primero comprobaremos si se está grabando o no, si no se estuviera grabando generaríamos la ruta de donde vamos a guardar la pista de audio.
Con la variable rec que hemos declarado antes pero no inicializado, debemos crear un objeto MediaRecorder llamado rec. Primero indicaremos que recurso queremos utilizar, en nuestro caso el micrófono, otorgaremos un formato a la pista de audio y configuraremos el codificador.
Haremos también un tratamiento de excepciones, en el try preparamos y arrancamos el archivo, mientras que en el catch mostramos un mensaje de error.
Le indicaremos al botón que imagen debe mostrar en este caso e iniciaremos el crono en cero segundos, cero minutos.

Por el contrario, si está grabando primero tenemos que parar, y finalizar la grabación; indicaremos la imagen que debe mostrar nuestro botón al finalizar; pararemos el temporizador y lo inicializaremos a cero de nuevo para que no haya lugar a dudas y además, igualaremos de nuevo la variable a null para que pueda volver a entrar por nuestro primer condicional en caso de querer hacer otra grabación.

En el método reproducir, crearemos el objeto MediaPlayer, controlaremos nuestras excepciones con un try, en el que tomaremos la ruta del archivo y lo prepararemos; en con nuestro catch mostraremos un mensaje de error. Iniciaremos una vez realizados todos estos pasos la reproducción del audio.

```java
package com.example.feedback3_2;

import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;
import androidx.core.content.ContextCompat;
import android.Manifest;
import android.content.pm.PackageManager;
import android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.os.Bundle;
import android.os.Environment;
import android.os.SystemClock;
import android.view.View;
import android.widget.Button;
import android.widget.Chronometer;
import java.io.IOException;

public class MainActivity extends AppCompatActivity {

    MediaRecorder rec;
    String archivoSalida = null;
    Button botonRec;
    Chronometer crono;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        crono = (Chronometer) findViewById(R.id.crono);
        botonRec = (Button) findViewById(R.id.botonrec);
        if (ContextCompat.checkSelfPermission(getApplicationContext(),
                Manifest.permission.WRITE_EXTERNAL_STORAGE)!= PackageManager.PERMISSION_GRANTED &&
                ActivityCompat.checkSelfPermission(getApplicationContext(),
                        Manifest.permission.RECORD_AUDIO)!= PackageManager.PERMISSION_GRANTED)
        {ActivityCompat.requestPermissions( activity: MainActivity.this, new String[]
                {Manifest.permission.WRITE_EXTERNAL_STORAGE,
                Manifest.permission.RECORD_AUDIO}, requestCode: 1000);}
    }
    public void grabar (View view){
        if (rec == null){
            archivoSalida = getExternalFilesDir( type: null).getAbsolutePath() + "/Grabacion.mp3";
            rec = new MediaRecorder();
            rec.setAudioSource(MediaRecorder.AudioSource.MIC);
            rec.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
            rec.setAudioEncoder(MediaRecorder.AudioEncoder.DEFAULT);
            rec.setOutputFile(archivoSalida);
        try {
            rec.prepare();
            rec.start();
        } catch(IOException e) {
            System.out.println("Error en la grabación del audio");
        }
        botonRec.setBackgroundResource(R.drawable.rec);
            crono.setBase(SystemClock.elapsedRealtime());
            crono.start();
        }else if(rec != null){
            rec.stop();
            rec.release();
            rec = null;
            botonRec.setBackgroundResource(R.drawable.stop_rec);
            crono.stop();
            crono.setBase(SystemClock.elapsedRealtime());
        }
    }
    public void reproducir(View view){
        MediaPlayer play = new MediaPlayer();
        try{
            play.setDataSource(archivoSalida);
            play.prepare();
        }catch (IOException e){
            System.out.println("Error en la reproducción del audio");
        }
            play.start();
    }

}
```

Nuestra aplicación debe se mostrar pues de la siguiente forma: