



UNIVERSIDAD PRIVADA DE TACNA

FACULTAD DE INGENIERIA

Escuela Profesional de Ingeniería de Sistemas

PRACTICA DE SERVICIOS

Curso: Soluciones Móviles I

Docente: Ing. Alberto Johnatan Flor Rodríguez

Alumno:

RODRIGUEZ MAMANI, Juan Rigoberto

(2017057862)

**Tacna – Perú
2020**



INDICE GENERAL

INTRODUCCIÓN	3
I. OBJETIVOS.....	4
1.1. General.....	4
1.2. Específicos.....	4
II. DESARROLLO.....	4
2.1. Practica de Reproductor de Música	4
2.1.1. Diseño	4
2.1.2. Programación.....	12
2.1.3. Prueba.....	17
2.2. Practica de Calculadora con Servicios	20
2.2.1. Diseño	20
2.2.2. Programación.....	23
2.2.3. Prueba.....	30
2.3. Repositorio GitHub	33
CONCLUSIONES	34
BIBLIOGRAFIA.....	35



INTRODUCCIÓN

El Android Studio es la plataforma libre desarrollada por Google, ampliamente utilizada en multitud de dispositivos como móviles, tabletas, TV, wearables e Internet de las cosas. Su expansión ha sido espectacular, siendo el S.O. más utilizado en la actualidad. Tras realizar este curso conocerás los fundamentos del desarrollo de aplicaciones en Android y podrás realizar sencillas aplicaciones, que incluyan los aspectos más importantes y novedosos de esta plataforma.

Android es un sistema operativo que fue creado especialmente para teléfonos con pantalla táctil, los llamados de nueva generación o los inteligentes, las tablets comunes y las que funcionan con líneas telefónica; entrando en esta gama los relojes inteligentes, televisores y algunos aditamentos de los nuevos automóviles, esta empresa de nombre Android Inc, fue respaldada por Google, que en el año 2005 la misma google se apodera comprando por varios millones de dólares, previendo que esta nueva plataforma en la tecnología de avanzada de los estándares en los dispositivos móviles resultaba un buen negocio a futuro.

I. OBJETIVOS

1.1. General

- Realizar la practica encargada del laboratorio de Servicios.

1.2. Específicos

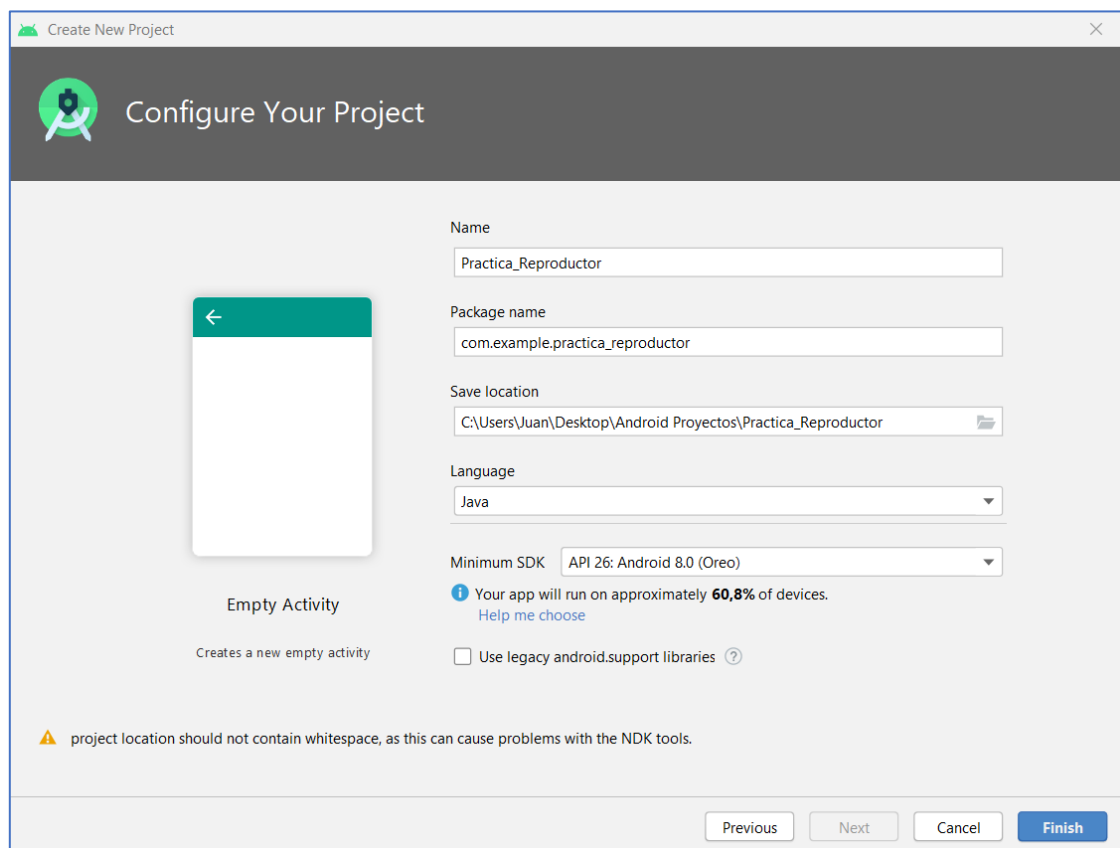
- Investigar la problemática propuesta de la practica encargada.
- Utilizar herramientas de desarrollo en Android Studio.
- Proponer una solución al problema planteado.

II. DESARROLLO

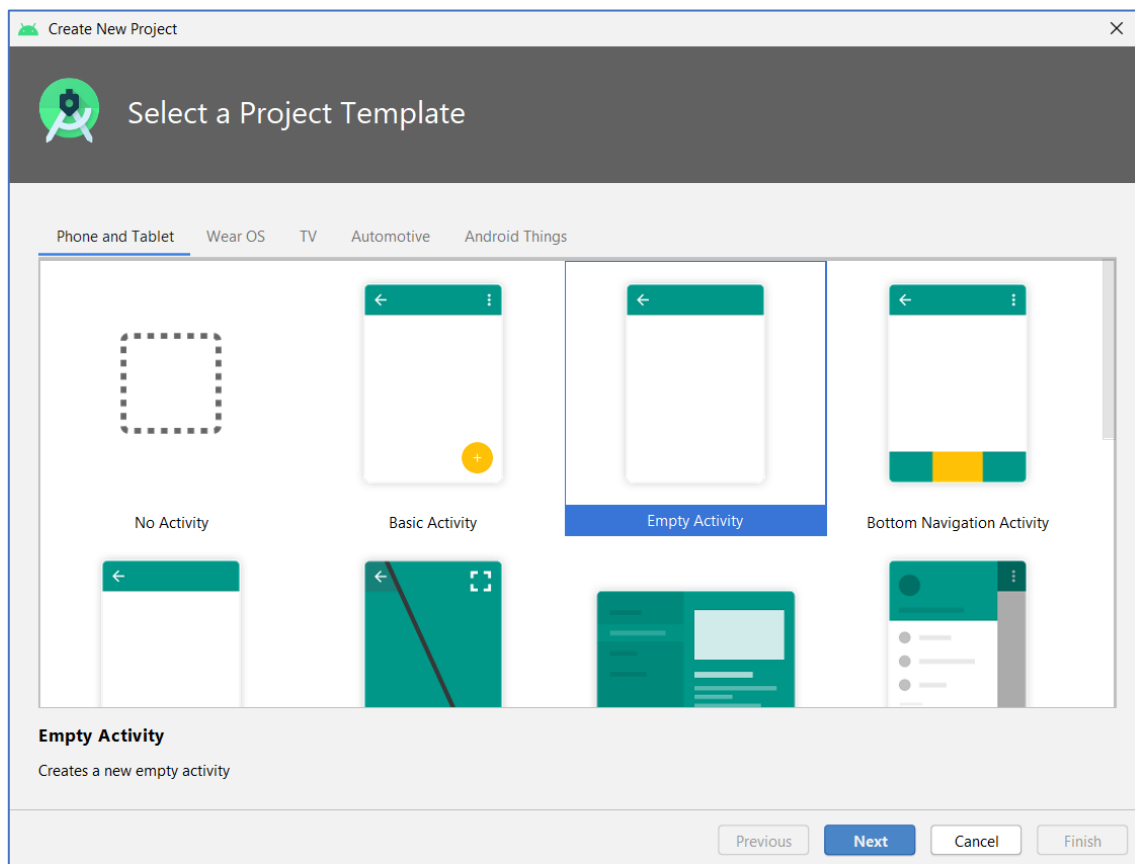
2.1. Practica de Reproductor de Música

2.1.1. Diseño

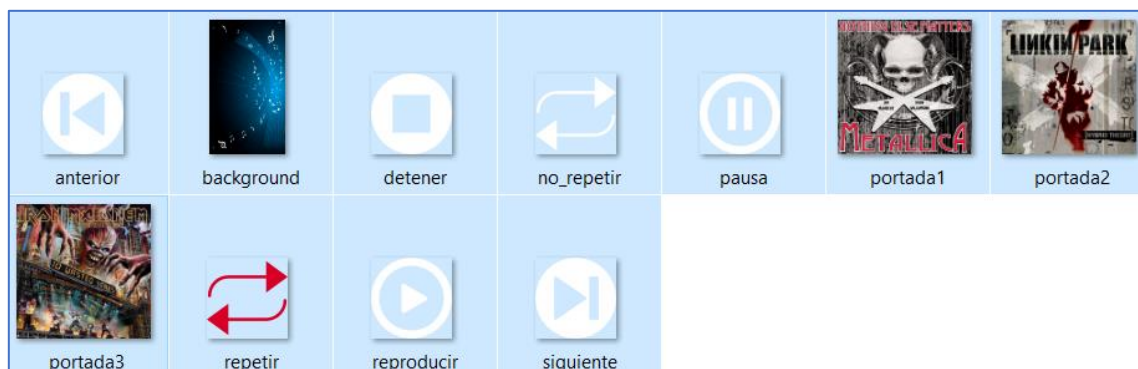
- Lo primero que haremos será crear un nuevo proyecto con el nombre: **Practica_Reproductor** luego elegiremos el Minimum SDK la cual será el **API 26**.



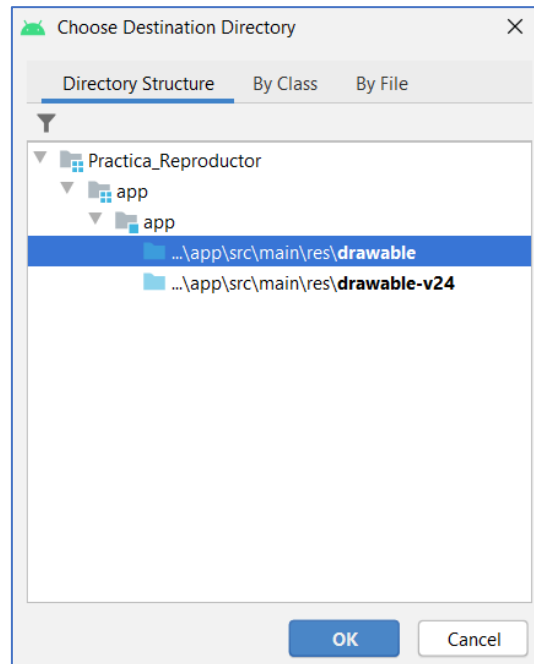
- Pero antes de eso elegiremos un **Empty Activity**.



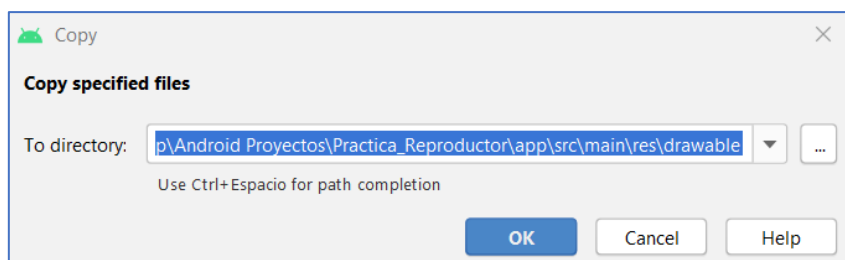
- Ahora para este paso es necesario tener algunas imágenes en nuestra computadora ya que los usaremos como recursos en nuestra aplicación (imágenes de botones, portadas de canciones).



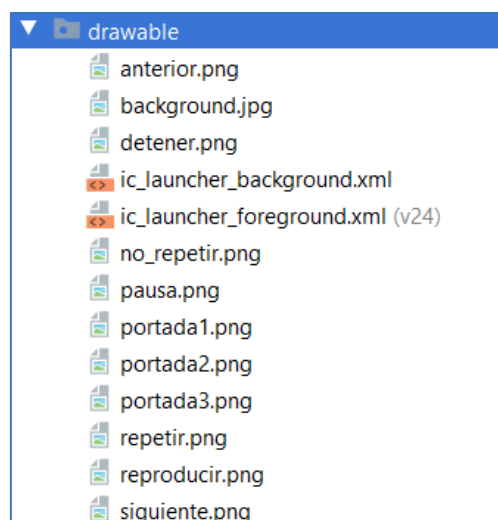
- Copiaremos dichas imágenes y lo pegaremos dentro de la carpeta drawable que se encuentra dentro de nuestro proyecto.



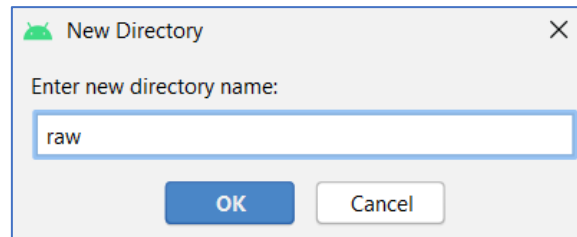
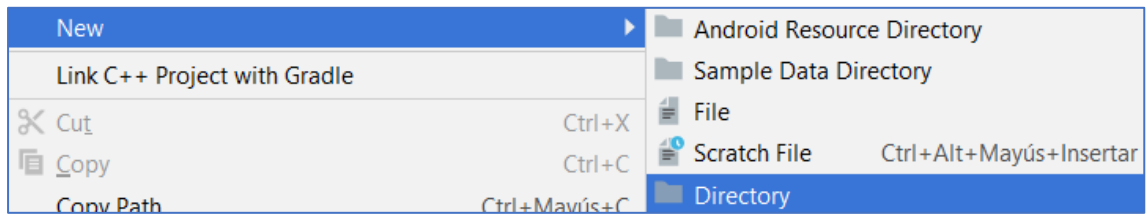
- Nos aparecerá el siguiente recuadro y le daremos clic en OK.



- Así es como debería quedarnos dicha carpeta.



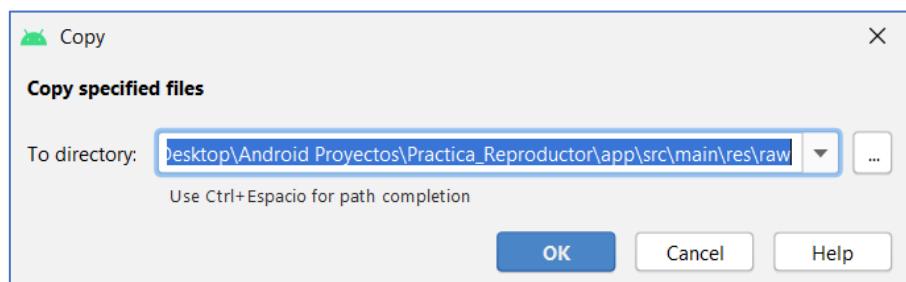
- Seguidamente crearemos un nuevo directorio en nuestro proyecto al cual llamaremos **raw**.



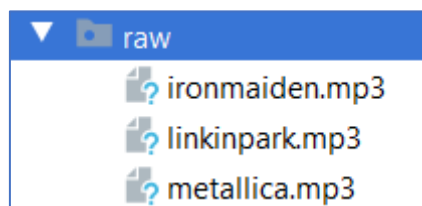
- Dentro de dicho directorio pegaremos las canciones que utilizaremos en nuestra aplicación. Para eso primero ubicaremos las canciones en nuestra computadora, le daremos Ctrl+C.



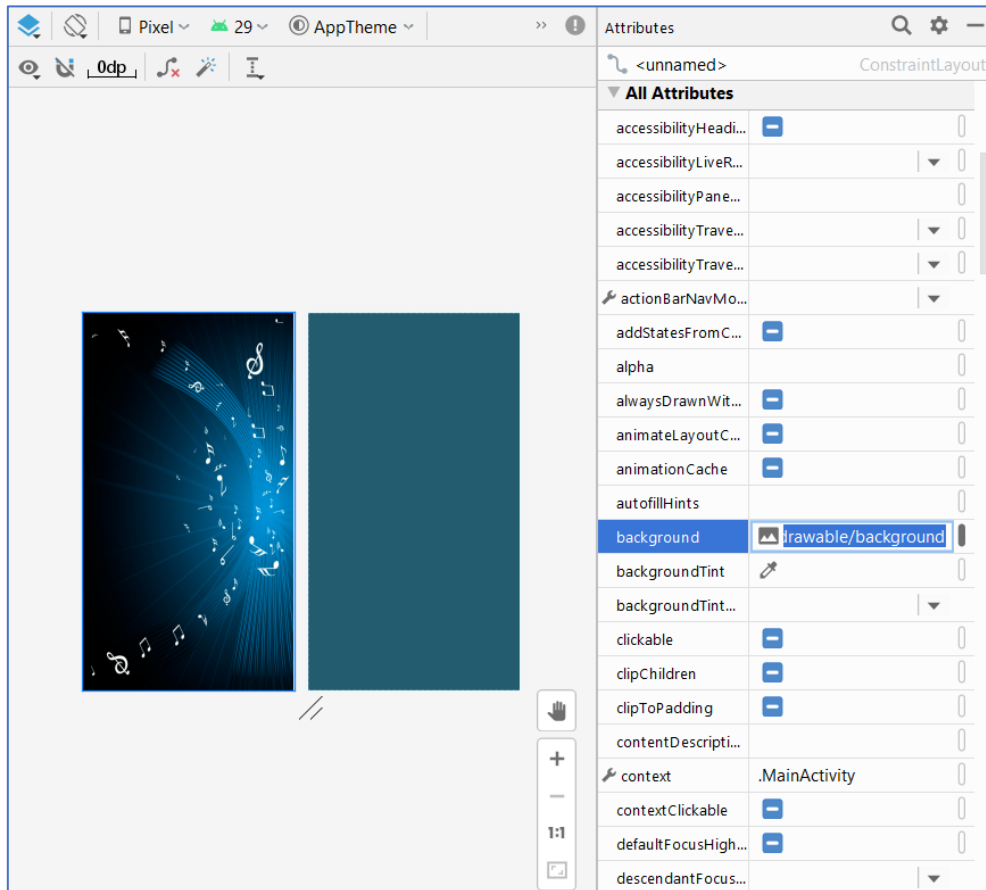
- Nos dirigiremos a nuestra carpeta recién creada llamada **raw** y pegaremos nuestros archivos Ctrl+V.



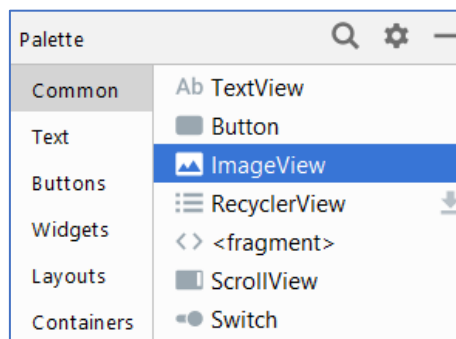
- Nuestra carpeta raw debería quedar de la siguiente forma.



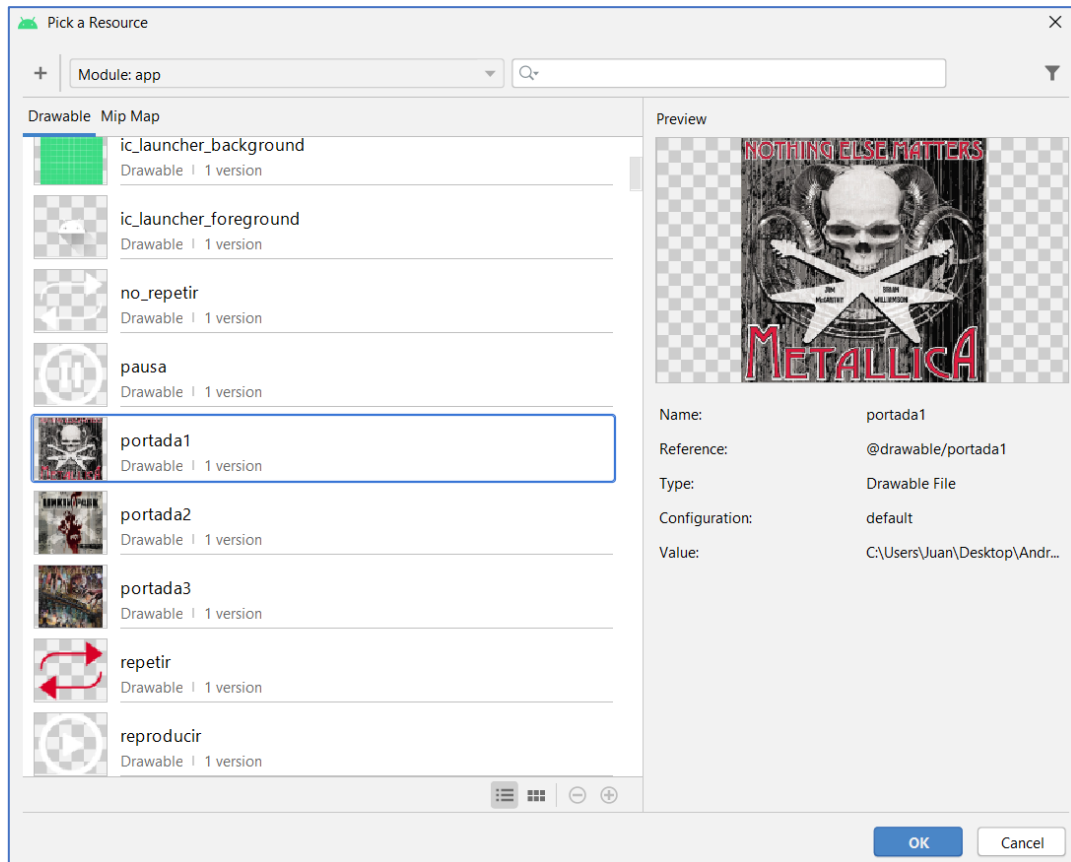
- Acto seguido, vamos a elaborar el diseño de nuestra aplicación, para eso nos dirigimos a nuestro activity y nos dirigimos a los **atributes** de la aplicación y en el apartado llamado **background** indicaremos la ruta de nuestra imagen que se encuentra en nuestro drawable.



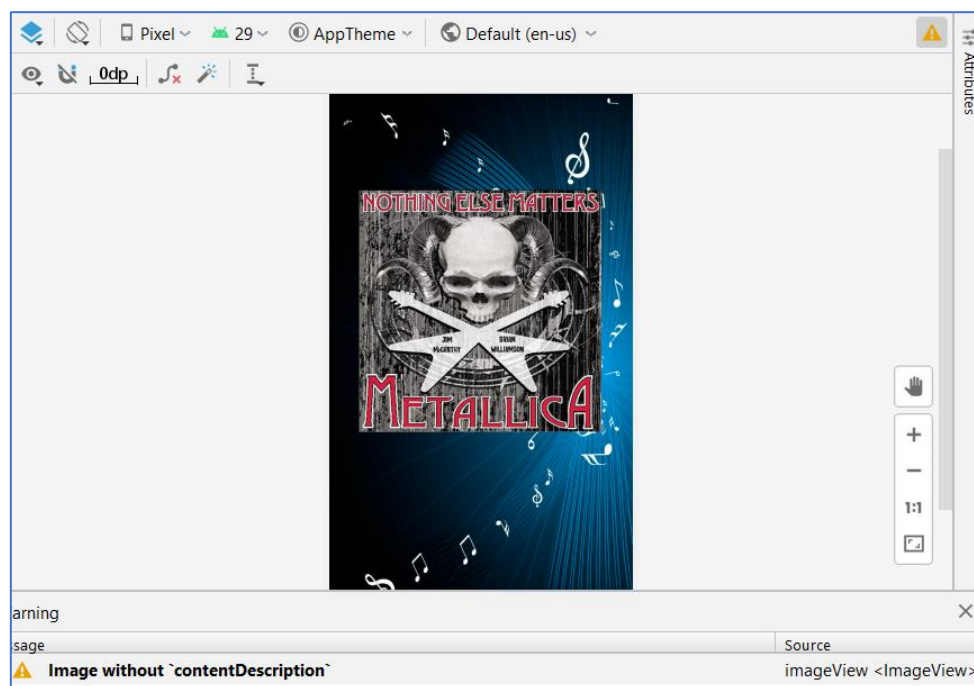
- Luego agregaremos una imagen la cual mostrará la portada de la música en reproducción. Para nos vamos a las herramientas y arrastraremos el componente llamado **ImageView**.



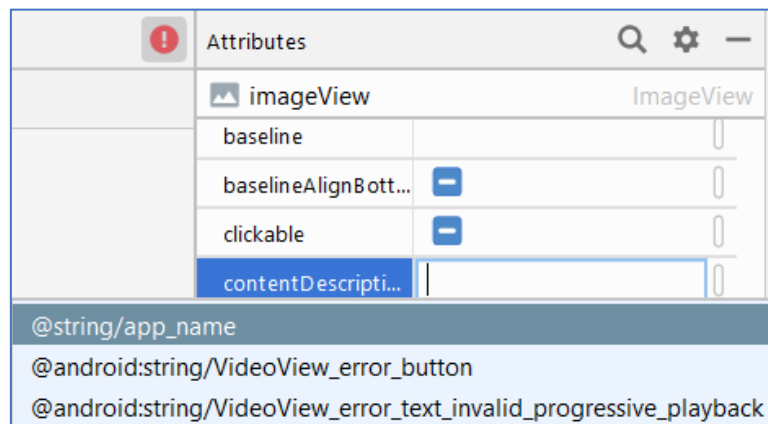
- Ahora elegiremos la imagen que se mostrará en nuestro **ImageView**, en este caso elegiremos la imagen de la **portada1**.



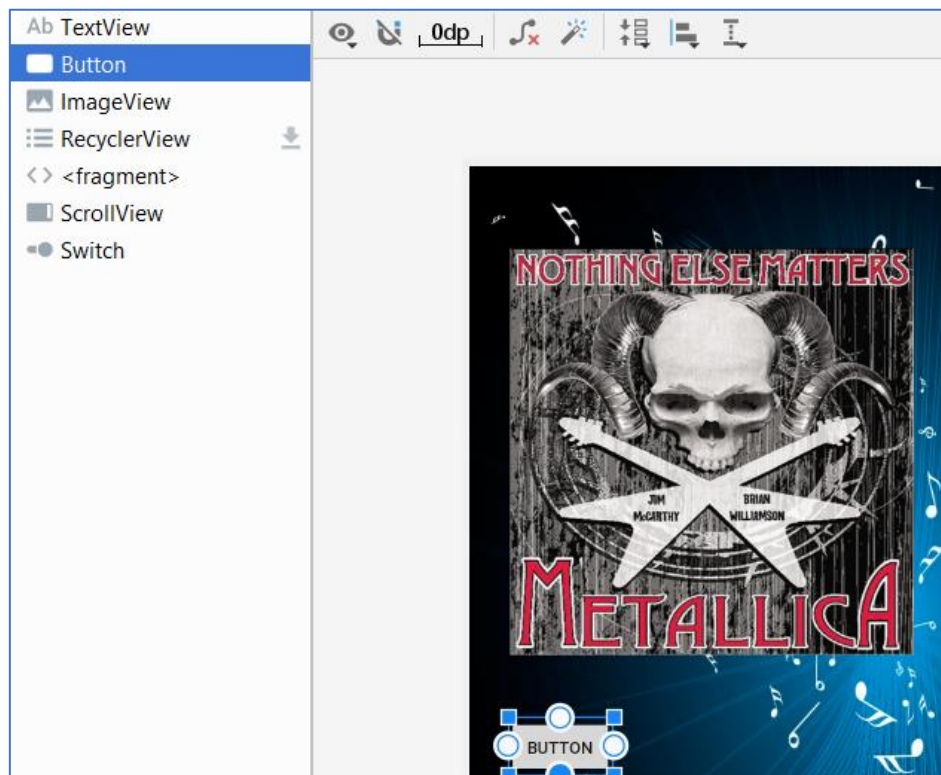
- Luego acomodaremos la posición de la imagen de tal manera que nos quede así.



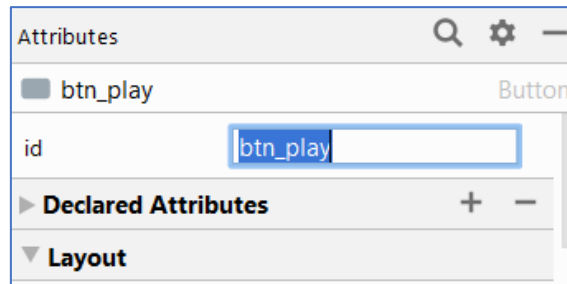
- Luego de haber hecho esto, lo más probable es que Android nos muestre una advertencia de diseño, esto lo solucionaremos agregando una referencia de tipo String en su atributo **contentDescription**. Para esto debemos ir a los **attributes** y en el apartado de contentDescription y presionaremos **Ctrl+Space** luego nos aparecerán varias referencias del propio Android en el cual nosotros elegiremos la que se llama **@string/app_name** y con esto habremos solucionado dicha advertencia.



- Una vez que hayamos agregado la imagen de la portada, lo que haremos ahora será agregar los botones de nuestro reproductor. Para esto nos vamos a las herramientas y buscaremos el componente llamado **Button** y lo arrastraremos a nuestro diseño.



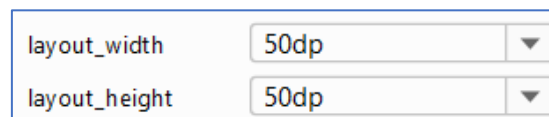
- Seleccionaremos nuestro **button** y en el apartado de **attributes** cambiaremos si **id** por **btn_play**.



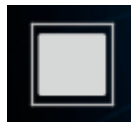
- Acto seguido modificaremos el aspecto de nuestro botón, para eso en attributes nos dirigiremos a su layout.



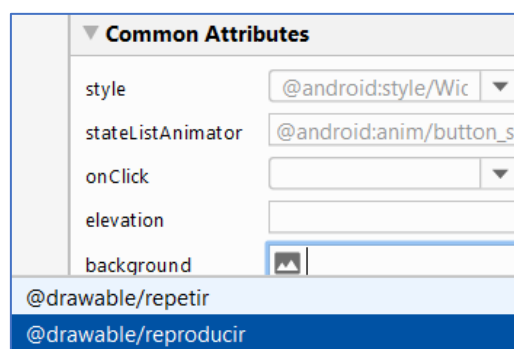
- Y cambiaremos sus valores por estos nuevos.



- De tal forma que el nuevo diseño del botón será el siguiente.



- Ahora vamos a agregar una imagen para eso nos vamos al **background** del botón.



- Y elegiremos la imagen de nuestro drawable llamado **reproducir**.



- Debemos repetir el mismo proceso para los otros botones, no debemos olvidar de cambiar el **id** de los nuevos botones por los siguientes.

btn_play, **btn_stop**, **btn_repetir**, **btn_anterior** y **btn_siguiente**.



- Y con esto habremos terminado de diseñar nuestra aplicación, la cuál se encuentra lista para ser programada.

2.1.2. Programación

- En este paso comenzaremos a programar nuestra aplicación, como primer paso nos dirigiremos a nuestro **MainActivity** y declararemos las siguientes variables.

```
Button play_pause, btn_repetir;  
MediaPlayer mp;  
ImageView iv;  
int repetir = 2, posicion = 0;  
  
MediaPlayer vectormp [] = new MediaPlayer [3];
```

- Dentro de nuestro **onCreate** agregaremos el siguiente código. Básicamente lo que hace es enlazar los botones de nuestro diseño con las respectivas variables antes creadas, además estamos estableciendo las posiciones de las músicas con las que vamos a trabajar.

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

    play_pause = (Button)findViewById(R.id.btn_play);
    btn_repetir = (Button)findViewById(R.id.btn_repetir);
    iv = (ImageView)findViewById(R.id.imageView);

    vectormp[0] = MediaPlayer.create( context: this, R.raw.metallica);
    vectormp[1] = MediaPlayer.create( context: this, R.raw.Linkinpark);
    vectormp[2] = MediaPlayer.create( context: this, R.raw.ironmaiden);
}
```

- Seguidamente crearemos nuestros métodos que darán funcionalidad a nuestra aplicación.
- **Método PlayPause.** Con este método lo que estamos diciendo es que cambie el estado del botón a **Pausa** si se está reproduciendo una música o que cambie a **Play** si la música está en pausa.

```
//Metodo para PlayPause
public void PlayPause(View view){
    if(vectormp[posicion].isPlaying()){
        vectormp[posicion].pause();
        play_pause.setBackgroundResource(R.drawable.reproducir);
        Toast.makeText( context: this, text: "Pausa", Toast.LENGTH_SHORT).show();
    }else{
        vectormp[posicion].start();
        play_pause.setBackgroundResource(R.drawable.pausa);
        Toast.makeText( context: this, text: "Play", Toast.LENGTH_SHORT).show();
    }
}
```

- **Método Stop.** Este método lo que hará será indicar a la aplicación que deje de reproducir la pista actual, además cambia la apariencia del botón de Stop a Play como también de la imagen que se está mostrando.

```
//Metodo Stop
public void Stop(View view){
    if(vectormp[posicion] != null){
        vectormp[posicion].stop();
        vectormp[0] = MediaPlayer.create( context: this, R.raw.metallica);
        vectormp[1] = MediaPlayer.create( context: this, R.raw.Linkinpark);
        vectormp[2] = MediaPlayer.create( context: this, R.raw.ironmaiden);
        posicion = 0;
        play_pause.setBackgroundResource(R.drawable.reproducir);
        iv.setImageResource(R.drawable.portada1);
        Toast.makeText( context: this, text: "Stop", Toast.LENGTH_SHORT).show();
    }
}
```

- **Método Repetir.** Este método lo que hará será repetir la pista actual además de cambiar la imagen de los botones Repetir y No Repetir.

```
//Metodo Repetir
public void Repetir(View view){
    if(repetir == 1){
        btn_repetir.setBackgroundResource(R.drawable.no_repetir);
        Toast.makeText( context: this, text: "No repetir", Toast.LENGTH_SHORT).show();
        vectormp[posicion].setLooping(false);
        repetir = 2;
    }else{
        btn_repetir.setBackgroundResource(R.drawable.repetir);
        Toast.makeText( context: this, text: "Repetir", Toast.LENGTH_SHORT).show();
        vectormp[posicion].setLooping(true);
        repetir = 1;
    }
}
```

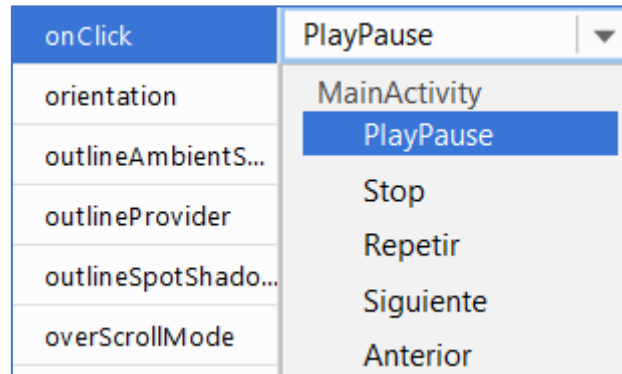
- **Método Siguiente.** Con este método lograremos que la aplicación vea la posición actual de la canción y pase a la siguiente, además de cambiar la imagen de la portada según la posición de la canción que se está reproduciendo.

```
//Metodo Siguiente
public void Siguiente(View view){
    if(posicion < vectormp.length -1){
        if(vectormp[posicion].isPlaying()){
            vectormp[posicion].stop();
            posicion++;
            vectormp[posicion].start();
            if(posicion == 0){
                iv.setImageResource(R.drawable.portada1);
            }else if(posicion == 1){
                iv.setImageResource(R.drawable.portada2);
            }else if(posicion == 2){
                iv.setImageResource(R.drawable.portada3);
            }
        }else{
            posicion++;
            if(posicion == 0){
                iv.setImageResource(R.drawable.portada1);
            }else if(posicion == 1){
                iv.setImageResource(R.drawable.portada2);
            }else if(posicion == 2){
                iv.setImageResource(R.drawable.portada3);
            }
        }
    }else{
        Toast.makeText( context: this, text: "No hay mas canciones", Toast.LENGTH_SHORT).show();
    }
}
```

- **Método Anterior.** Este método es similar a la anterior con la única diferencia que, en lugar de sumar posiciones, vamos a restarlas.

```
//Metodo Anterior
public void Anterior(View view){
    if(posicion >= 1){
        if(vectormp[posicion].isPlaying()){
            vectormp[posicion].stop();
            vectormp[0] = MediaPlayer.create( context: this, R.raw.metallica);
            vectormp[1] = MediaPlayer.create( context: this, R.raw.Linkinpark);
            vectormp[2] = MediaPlayer.create( context: this, R.raw.ironmaiden);
            posicion--;
            if(posicion == 0){
                iv.setImageResource(R.drawable.portada1);
            }else if(posicion == 1){
                iv.setImageResource(R.drawable.portada2);
            }else if(posicion == 2){
                iv.setImageResource(R.drawable.portada3);
            }
            vectormp[posicion].start();
        }else{
            posicion--;
            if(posicion == 0){
                iv.setImageResource(R.drawable.portada1);
            }else if(posicion == 1){
                iv.setImageResource(R.drawable.portada2);
            }else if(posicion == 2){
                iv.setImageResource(R.drawable.portada3);
            }
        }
    }else{
        Toast.makeText( context: this, text: "No hay mas canciones", Toast.LENGTH_SHORT).show();
    }
}
```


- Para finalizar el desarrollo de la aplicación ya solo nos quedará asignar los métodos creados con los botones antes diseñados. Para esto nos dirigiremos al diseño de nuestro **activity** y haremos click sobre el botón Play, y en el apartado de **attributes** buscaremos la opción de **onClick** y elegiremos el método **PlayPause**.



- Haremos los mismos pasos para los demás botones y luego de haber hecho esto, podremos compilar nuestra aplicación.



- ¡Y listo! Ahora podremos disfrutar de nuestra aplicación de música y también podemos probar que todos los botones se encuentran totalmente funcionales.

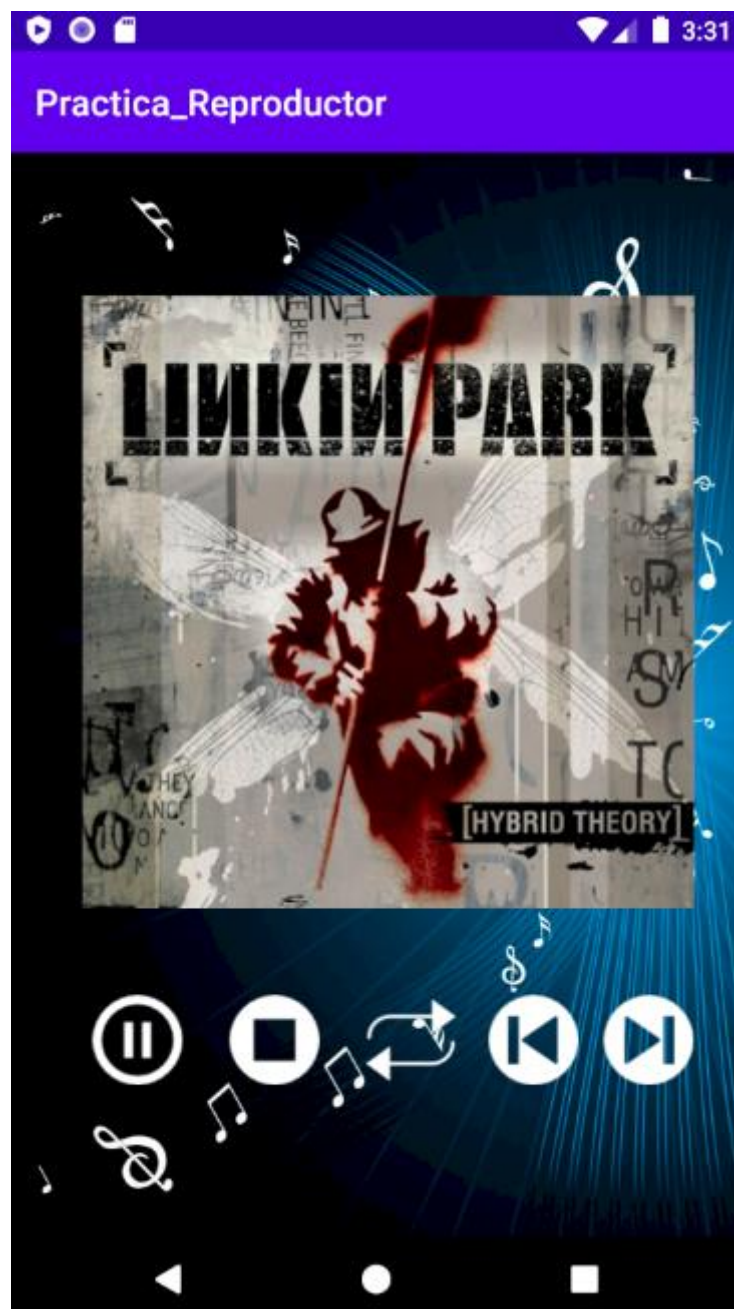
2.1.3. Prueba

- Canción 1: Metallica - Nothing Else Matters.



Publicación	20 de abril de 1992
Formato	Sencillo en CD · casete · vinilo
Grabación	30 de mayo de 1991 <ul style="list-style-type: none">• One on One Studios (North Hollywood, California, E.E. U.U.)
Género(s)	Heavy metal · Power ballad
Duración	6:29

- Canción 2: Linkin Park - In The End.



Publicación	Worldwide: December 2000 UK: August 2001
Formato	Sencillo en CD, Sencillo en DVD, Descarga digital, Streaming
Grabación	1998 - 1999: (Nueva Orleans, Estados Unidos)
Género(s)	Nu metal, ¹ rap rock, rock alternativo, hard rock
Duración	3:37

- Canción 3: Iron Maiden - Wasted Years.

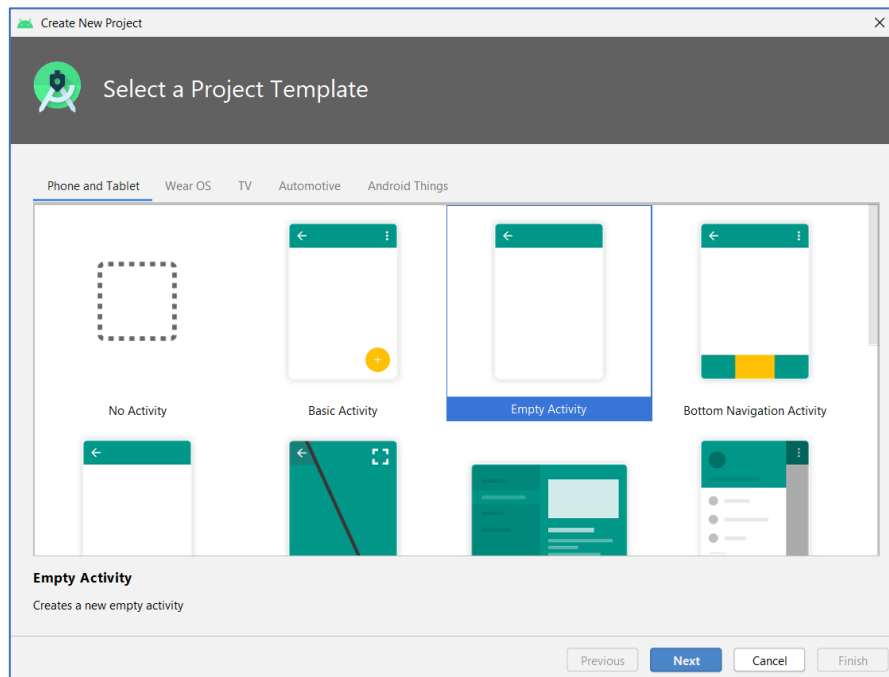


Publicación	6 de septiembre 1986
Formato	Vinilo de 7", 12" y CD
Género(s)	Heavy metal
Duración	5:07
Discográfica	EMI
Autor(es)	Adrian Smith
Productor(es)	Martin Birch

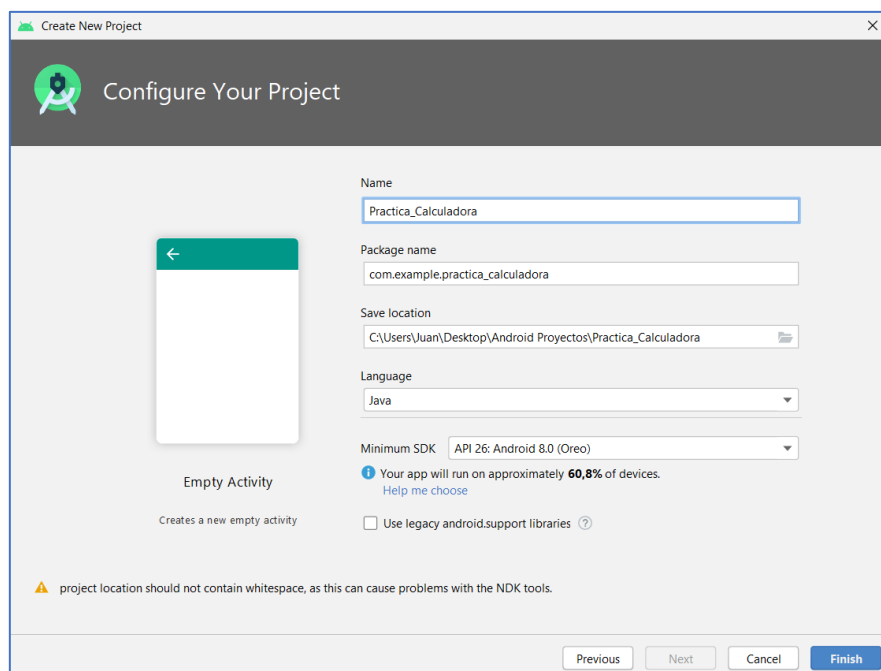
2.2. Practica de Calculadora con Servicios

2.2.1. Diseño

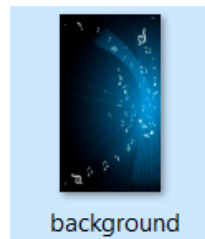
- Lo primero que haremos será crear un nuevo proyecto, para eso primero elegiremos un **Empty Activity** y le damos en **NEXT**.



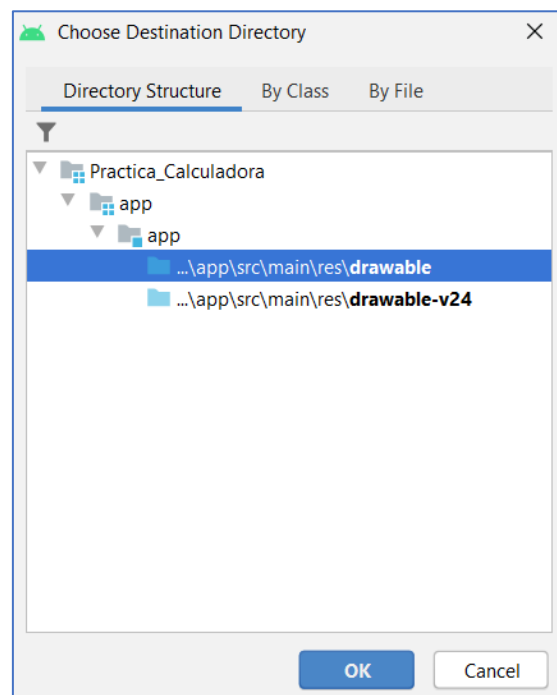
- Seguidamente a nuestro nuevo proyecto le asignaremos el nombre de: **Practica_Calculadora** luego elegiremos el Minimum SDK la cual será el **API 26**.



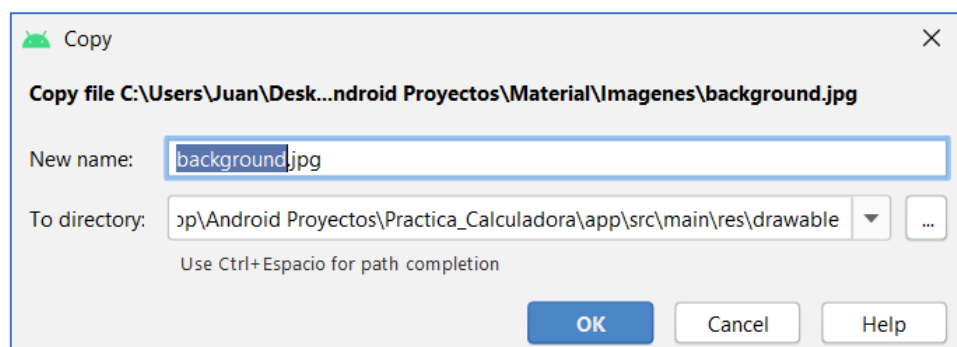
- Ahora le daremos algo de estilo a nuestro formulario, para eso buscaremos una imagen para colocarlo como fondo de nuestro formulario.



- Luego pegaremos dicha imagen dentro de nuestro proyecto, específicamente dentro de la carpeta **drawable**.



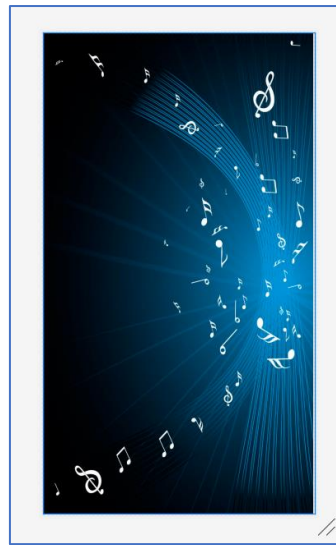
- Nos aparecerá el siguiente recuadro de confirmación, nosotros le daremos clic en **OK**.



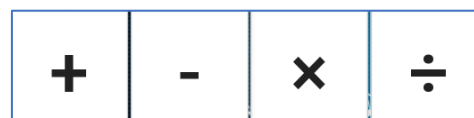
- Ahora nos dirigiremos al diseñador de nuestro **activity_main.xml** y en el apartado de **Attributes** buscaremos el apartado que dice **background** y ahí presionando Ctrl+Space nos aparecerán varias opciones de personalización, nosotros buscaremos la ruta de nuestra imagen y lo seleccionaremos.



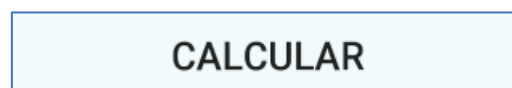
- Nuestro formulario quedará de la siguiente manera.



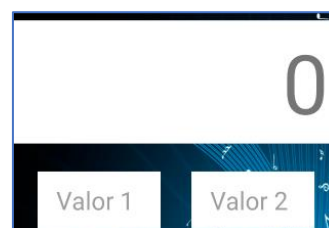
- Seguidamente agregaremos 4 **button** que representarán nuestra Suma, Resta, Multiplicación y División.
- ID: **btn_Suma**, **btn_Resta**, **btn_Multiplicacion**, **btn_División**



- ID: **btn_Calcular**



- Luego agregaremos los **TextView** ID: **valor_1**, **valor_2** y **ver_Resultado**

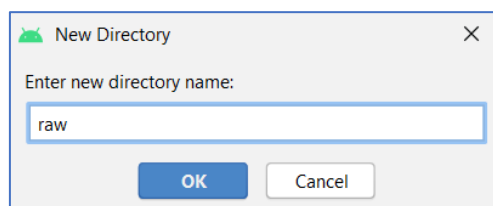
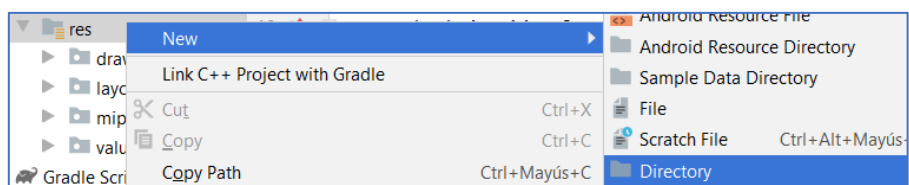


- Luego de haber realizado todos los pasos, nuestro diseño debería tener el siguiente aspecto.

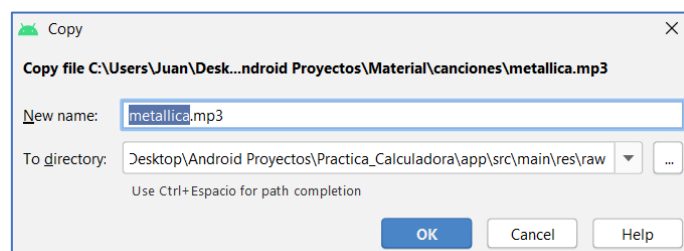


2.2.2. Programación

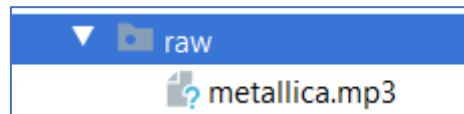
- Como primer paso, dentro de la carpeta **res** vamos a crear un **directory** al cual llamaremos **raw**.



- Como segundo paso, vamos a buscar alguna canción en formato mp3 y vamos a pegarlo dentro de nuestra carpeta recién creada.



- Verificaremos que nuestro archivo de música se encuentra dentro de la carpeta **raw**. Esta pista de audio la utilizaremos como música de fondo en nuestra aplicación.



- Ahora crearemos un servicio al cual llamaremos **ServicioMusica**. Para esto nos dirigiremos a **java, com.example.practica_calculadora** le daremos clic derecho a este último y daremos clic en **New, Java Class**.

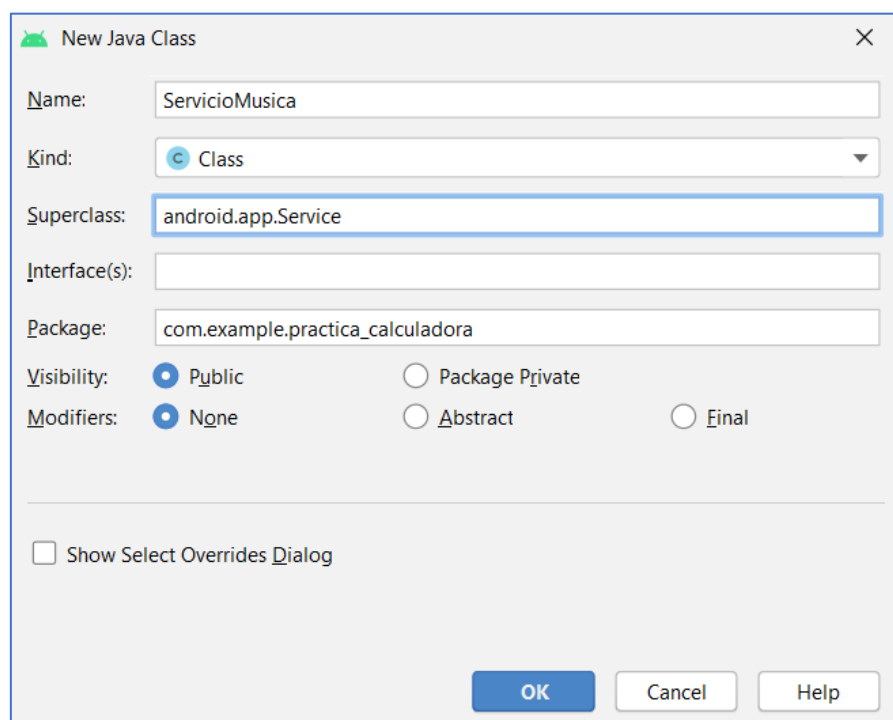


- Nos aparecerá un cuadro donde colocaremos lo siguiente.

Name: **ServicioMusica**

Kind: **Class**

Superclass: **Android.app.Service**



New Java Class

Name:

Kind:

Superclass:

Interface(s):

Package:

Visibility: ☒ Public ☐ Package Private

Modifiers: ☒ None ☐ Abstract ☐ Final

☐ Show Select Overrides Dialog

OK Cancel Help

- Le daremos clic en OK y se habrá creado nuestro servicio llamado ServicioMusica.

- Dentro de nuestro **ServicioMusica.java** declararemos la siguiente variable. No nos olvidemos de importar las librerías correspondientes.

```
MediaPlayer reproductor;
```

- En nuestro **onCreate** agregaremos el siguiente código. Lo que hará básicamente será que al arrancar el servicio, este cargue nuestra pista de audio.

```
@Override  
public void onCreate() {  
    reproductor = MediaPlayer.create(getApplicationContext(),R.raw.metallica);  
}
```

- Crearemos un método que reproducirá nuestra pista de audio.

```
@Override  
public int onStartCommand(Intent intent, int flags, int startId){  
    reproductor.start();  
    return START_STICKY;  
}
```

- Crearemos otro método que detendrá la reproducción de nuestra pista de audio.

```
@Override  
public void onDestroy(){  
    reproductor.stop();  
}
```

- Y en el **AndroidManifest.xml** debemos agregar el servicio recién creado.

```
<application  
    android:allowBackup="true"  
    android:icon="@mipmap/ic_launcher"  
    android:label="Practica_Calculadora"  
    android:roundIcon="@mipmap/ic_launcher_round"  
    android:supportsRtl="true"  
    android:theme="@style/AppTheme">  
    <activity android:name=".MainActivity">  
        <intent-filter>  
            <action android:name="android.intent.action.MAIN" />  
  
            <category android:name="android.intent.category.LAUNCHER" />  
        </intent-filter>  
    </activity>  
    <service android:name=".ServicioMusica"/>  
</application>
```

- Nos dirigiremos a nuestro **MainActivity.java** y declararemos las siguientes variables.

```
private boolean Suma = false;
private boolean Resta = false;
private boolean Multiplicacion = false;
private boolean Division = false;
private boolean Calcular = false;
private String indicador = "";
private int valor1 = 0;
private int valor2 = 0;
private double resultado = 0;
```

- Y no nos olvidemos de los demás componentes.

```
Button btn_Calcular;
Button btn_Suma;
Button btn_Resta;
Button btn_Multiplicacion;
Button btn_Division;
EditText etValor1;
EditText etValor2;
TextView tvResultado;
```

```
Button Comando;
```

- Ahora agregaremos el siguiente código. Lo que Haremos con esto será el de no admitir valores en blanco.

```
private boolean VerificarTextos() {
    etValor1 = findViewById(R.id.valor_1);
    etValor2 = findViewById(R.id.valor_2);

    String e1 = etValor1.getText().toString();
    String e2 = etValor2.getText().toString();

    if (e1.equals("") && e2.equals("")) {
        return false;
    } else {
        return true;
    }
}
```

- Seguidamente Crearemos nuestro Habilitar e Inhabilitar botones, para así evitar empezar la operación si es que no tenemos valores con que trabajar.

```
private void HabilitarControles() {  
    Suma = true; Resta = true; Multiplicacion = true; Division = true;  
}  
  
private void InhabilitarBotones() {  
    btn_Suma.setEnabled(Suma);  
    btn_Resta.setEnabled(Resta);  
    btn_Multiplicacion.setEnabled(Multiplicacion);  
    btn_Division.setEnabled(Division);  
    btn_Calcular.setEnabled(Calcular);  
}
```

- Resolución, este método hará el trabajo de calcular la operación solicitada.

```
private void Resolucion(int opc) {  
    VerificarEstado();  
    etValor1 = (EditText) findViewById(R.id.valor_1);  
    etValor2 = (EditText) findViewById(R.id.valor_2);  
    Valor1 = Integer.parseInt(etValor1.getText().toString());  
    Valor2 = Integer.parseInt(etValor2.getText().toString());  
  
    switch (opc) {  
        case 1: //Primer Caso Suma  
            Comando = (Button) findViewById(R.id.btn_Suma);  
            Indicador = Comando.getText().toString();  
            Resultado = Valor1 + Valor2;  
            Toast.makeText(getApplicationContext(), text: "SUMA", Toast.LENGTH_SHORT).show();  
            break;  
  
        case 2: //Segundo Caso Resta  
            Comando = (Button) findViewById(R.id.btn_Resta);  
            Indicador = Comando.getText().toString();  
            Resultado = Valor1 - Valor2;  
            Toast.makeText(getApplicationContext(), text: "RESTA", Toast.LENGTH_SHORT).show();  
            break;  
  
        case 3: //Tercer Caso Division  
            Comando = (Button) findViewById(R.id.btn_Division);  
            Indicador = Comando.getText().toString();  
            double dnum1 = Double.parseDouble(etValor1.getText().toString());  
            double dnum2 = Double.parseDouble(etValor2.getText().toString());  
            Resultado = dnum1 / dnum2;  
            Toast.makeText(getApplicationContext(), text: "DIVISION", Toast.LENGTH_SHORT).show();  
            break;  
  
        case 4: //Cuarto Caso Multiplicacion  
            Comando = (Button) findViewById(R.id.btn_Multiplicacion);  
            Indicador = Comando.getText().toString();  
            Resultado = Valor1 * Valor2;  
            Toast.makeText(getApplicationContext(), text: "MULTIPLICACION", Toast.LENGTH_SHORT).show();  
            break;  
  
        default:  
            Toast.makeText(getApplicationContext(), text: "Debe seleccionar una operacion", Toast.LENGTH_SHORT).show();  
            break;  
    }  
}
```

- Dentro de nuestro **onCreate** vamos a configurar los 5 botones para que realizar la actividad solicitada, según el botón elegido Suma, Resta, Multiplicación, División y el de Calcular. Pero antes de eso agregaremos la línea de código que inicializará nuestro servicio de música.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    startService(new Intent(getApplicationContext(), ServicioMusica.class));
}
```

- Dentro de nuestro **onCreate** agregaremos el código del botón Calcular.

```
//Boton Calcular
btn_Calcular = (Button)findViewById(R.id.btn_Calcular);
btn_Calcular.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        tvResultado = (TextView) findViewById(R.id.ver_Resultado);

        if (VerificarTextos()) {
            tvResultado.setText(String.valueOf(Resultado));
            HabilitarControles();
        } else {
            Toast.makeText(getApplicationContext(), text: "No ha ingresado numeros", Toast.LENGTH_SHORT).show();
        }
    }
});
```

- Continuando en el **onCreate**, ahora agregaremos el código del botón Suma.

```
//Boton Suma
btn_Suma = (Button)findViewById(R.id.btn_Suma);
btn_Suma.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (VerificarTextos() == true) {
            Resolucion( opc: 1);
        } else {
            Toast.makeText(getApplicationContext(), text: "No hay numeros", Toast.LENGTH_SHORT).show();
        }
    }
});
```

- Seguidamente agregaremos el código del botón Resta.

```
//Boton Resta
btn_Resta = (Button)findViewById(R.id.btn_Resta);
btn_Resta.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (VerificarTextos() == true) {
            Resolucion( opc: 2);
        } else {
            Toast.makeText(getApplicationContext(), text: "No hay numeros", Toast.LENGTH_SHORT).show();
        }
    }
});
```

- Luego agregaremos el código del botón División.

```
//Boton Division
btn_Division = (Button)findViewById(R.id.btn_Division);
btn_Division.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (VerificarTextos() == true) {
            Resolucion( opc: 3);
        } else {
            Toast.makeText(getApplicationContext(), text: "No hay numeros", Toast.LENGTH_SHORT).show();
        }
    }
});
```

- Para terminar, dentro de **onCreate** agregaremos el código del botón Multiplicación.

```
//Boton Multiplicacion
btn_Multiplicacion = (Button)findViewById(R.id.btn_Multiplicacion);
btn_Multiplicacion.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (VerificarTextos() == true) {
            Resolucion( opc: 4);
        } else {
            Toast.makeText(getApplicationContext(), text: "No hay numeros", Toast.LENGTH_SHORT).show();
        }
    }
});

VerificarEstado();
```

- Ah sí, y no olvidemos de agregar nuestros típicos **onPause**, **onStop** y por último **onDestroy**.

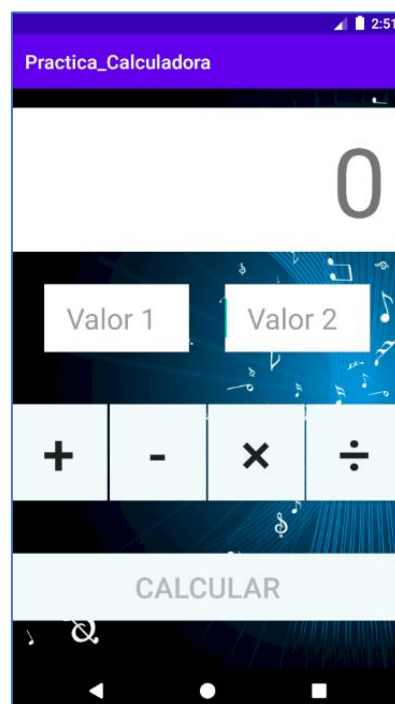
```
@Override
protected void onPause() {
    super.onPause();
    startService(new Intent(getApplicationContext(),ServicioMusica.class));
}

@Override
protected void onStop() {
    super.onStop();
    stopService(new Intent(getApplicationContext(),ServicioMusica.class));
}

@Override
protected void onDestroy() {
    super.onDestroy();
    stopService(new Intent(getApplicationContext(),ServicioMusica.class));
}
```

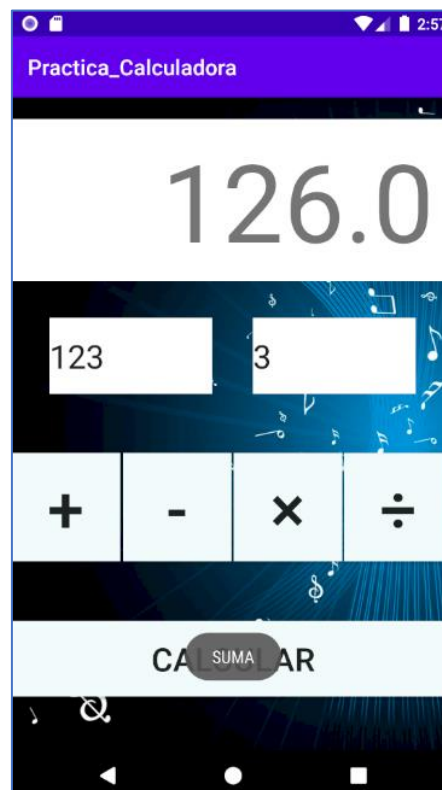
2.2.3. Prueba

- Ahora vamos a probar nuestra aplicación, crucemos los dedos esperando a que este funcione y que no adopte una inteligencia artificial que se revele contra nosotros y destruya al mundo.

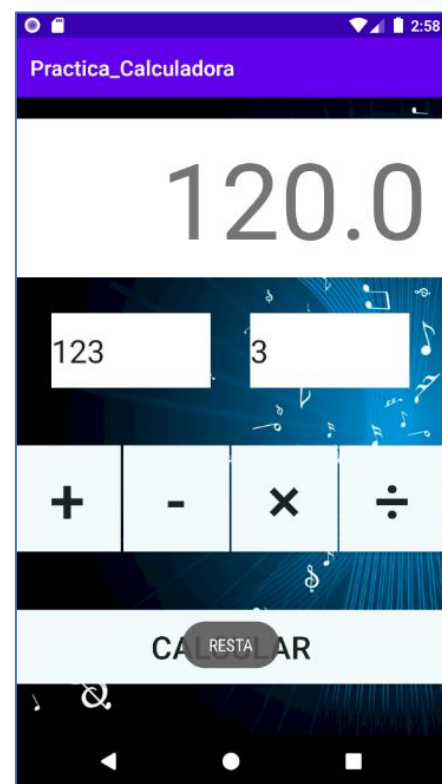


- Apenas se inicie la aplicación, sonará una música de fondo, la cual se trata del grupo de rock llamado **Metallica** con la canción **Nothing else matters**.

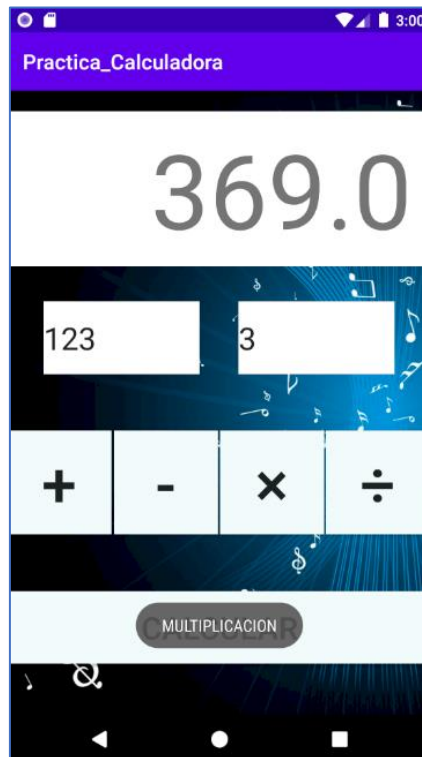
- Operación Sumar.



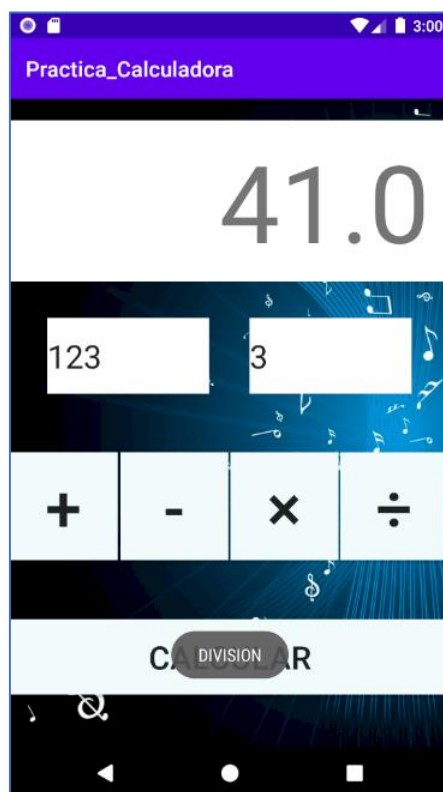
- Operación Restar.



- Operación Multiplicar.



- Operación Restar.



2.3. Repositorio GitHub

Repositorio Reproductor de Música

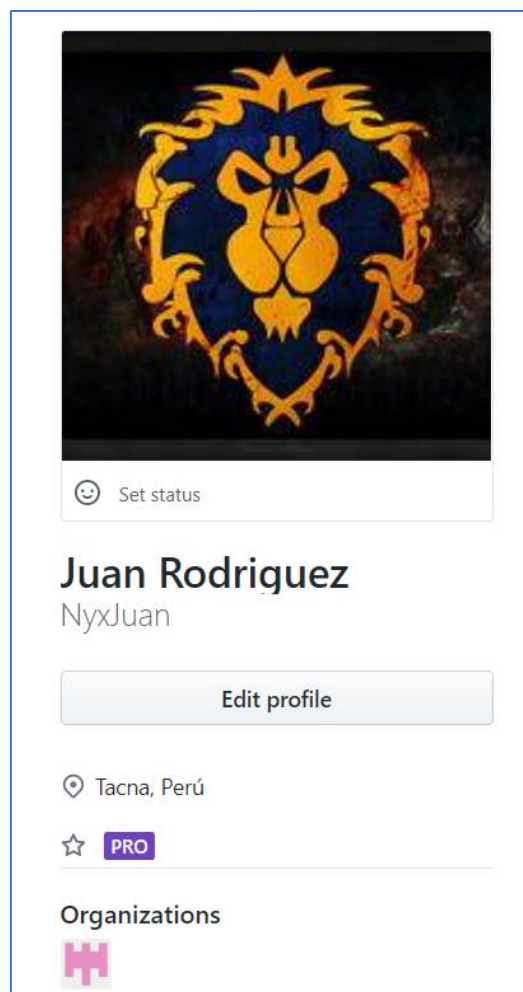
https://github.com/NyxJuan/Moviles_I_Practica_V_ReproductorMusica

Repositorio Calculadora

https://github.com/NyxJuan/Moviles_I_Practica_V_Calculadora

NOTA:

- En caso de que falle la aplicación, intentar limpiar el Emulador dándole clic en **WIPE DATA** o intentar probar con otro emulador.
- Importar la aplicación, intentar otorgar permisos a la solución.
- Intentar correr en el emulador utilizando la **API 26** en adelante.





CONCLUSIONES

- Se logró realizar la practica encargada del laboratorio de Servicios.
- Se pudo Investigar la problemática propuesta de la practica encargada, con la finalidad de poder proponer posibles soluciones.
- Se utilizó varias herramientas para el desarrollo de la aplicación en Android Studio, los cuales fueron útiles para el desarrollo de la solución.
- Se pudo proponer la solución del problema planteado, esto de acuerdo con la problemática planteada del laboratorio de servicios.

BIBLIOGRAFIA

- [1]. Gutiérrez, G. (2019). *Android*. Recuperado el 11 de mayo de 2020, de Concepto Definición. Sitio web: <https://conceptodefinicion.de/android/>
- [2]. Google. (2019). *Google Cloud Platform*. Recuperado el 11 de junio de 2020, de Google. Sitio web: <https://console.cloud.google.com/>
- [3]. ShareIcon. (2018). *Share Icon*. Recuperado el 11 de junio de 2020, de ShareIcon. Sitio web: <https://www.shareicon.net/>
- [4]. La Geekipedia De Ernesto. (2018). Curso Android desde cero #40 | Reproductor de música. Recuperado el 12 de junio de 2020, de YouTube. Sitio web: <https://www.youtube.com/watch?v=IAfGGzZoMwk>
- [5]. Yo Androide. (2019). Android Studio aplicación calculadora desde cero bien explicado. Recuperado el 15 de junio de 2020, de YouTube. Sitio web: <https://www.youtube.com/watch?v=shUMeK4cB58>
- [6]. Yo Androide. (2019). Android Studio aplicación calculadora desde cero bien explicado. Recuperado el 15 de junio de 2020, de YouTube. Sitio web: <https://developer.android.com/guide/components/services>