Mídias e Outros Recursos

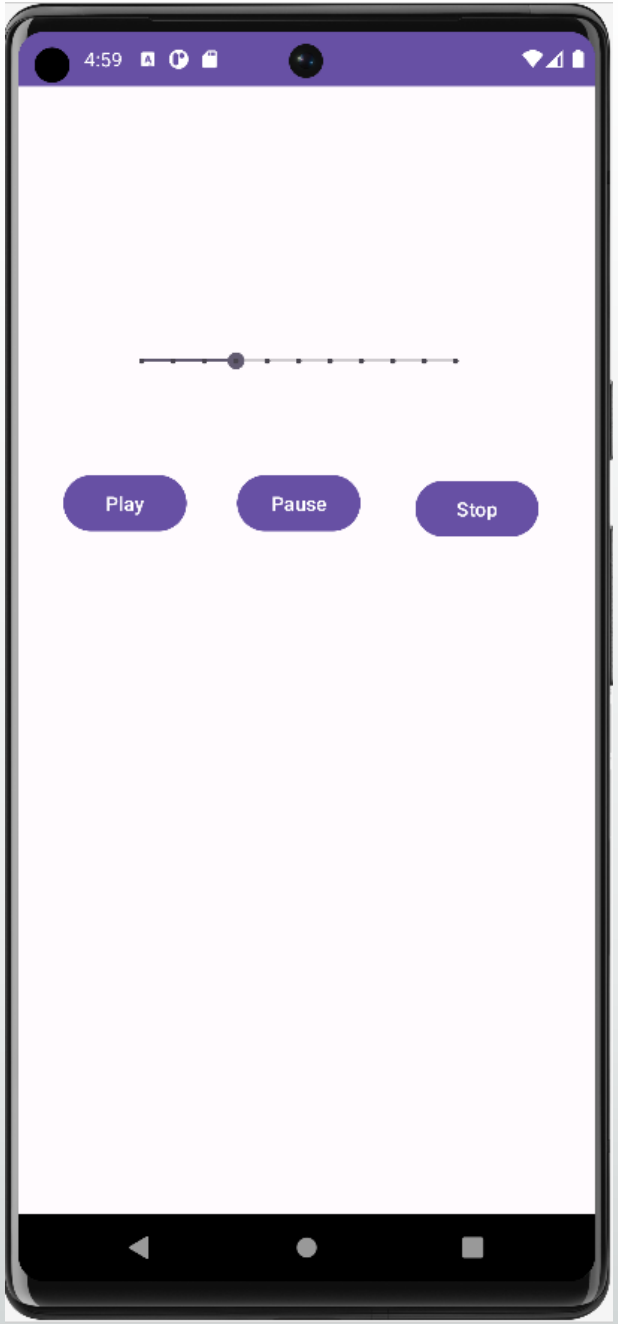
Dispositivos Móveis

Prof. Edson Mota, Ph.D, MSc, PMP

Executando Sons

Visão Geral

- O processo de execução de sons no Android Studio envolve o uso de uma biblioteca de áudio nativa, que fornece classes e métodos para manipular e reproduzir áudios.
- Durante a execução de sons, é importante lidar com eventos como a conclusão da reprodução, pausas ou volume
- O processo de execução de sons no Android Studio também pode envolver o uso de recursos adicionais, como permissões do sistema para acessar o áudio do dispositivo, especialmente se você estiver trabalhando com a gravação de áudio ou reprodução de som em segundo plano
- Para melhorar o desempenho e evitar problemas relacionados à execução de sons, é importante gerenciar corretamente os recursos de áudio, liberando-os quando não estiverem mais em uso



```java
public class MainActivity extends AppCompatActivity {

    2 usages
    private Button btPlay, btPause, btStop;
    1 usage
    private SeekBar skbVolume;

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

        btPlay = findViewById(R.id.btPlay);
        btPause = findViewById(R.id.btPause);
        btStop = findViewById(R.id.btStop);
        skbVolume = findViewById(R.id.skbVolume);

        btPlay.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                ExecutaSom();
            }
        });
    }

    1 usage
    private void ExecutaSom() {

    }
}
```

Vamos separar as funções do evento click criando um método EmitirSom();

```
2 usages
private Button btPlay, btPause, btStop;
1 usage
private SeekBar skbVolume;
3 usages
private MediaPlayer mediaPlayer;

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

    btPlay = findViewById(R.id.btPlay);
    btPause = findViewById(R.id.btPause);
    btStop = findViewById(R.id.btStop);
    skbVolume = findViewById(R.id.skbVolume);

    mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.musica);

    btPlay.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            ExecutaSom();
        }
    });
}
```

```
1 usage
private void ExecutaSom() {

    if (mediaPlayer != null){
        mediaPlayer.start();
    }
}
```

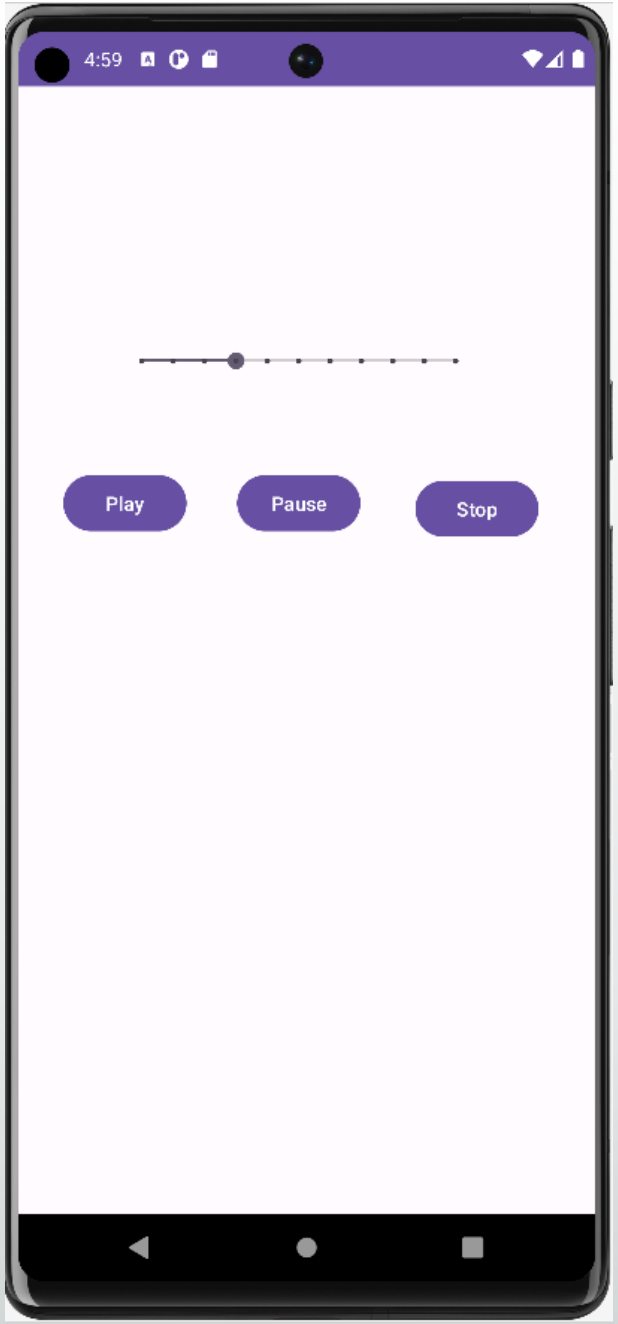
Com isso já conseguimos executar o som.

Pausando o som

- A configuração do “pause” deve suspender a execução do áudio e retomar do ponto no qual parou

- “stop”, implica em encerrar a execução do áudio

Vamos configurar esses recursos



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

    btPlay = findViewById(R.id.btPlay);
    btPause = findViewById(R.id.btPause);
    btStop = findViewById(R.id.btStop);
    skbVolume = findViewById(R.id.skbVolume);

    mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.musica);

    btPlay.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {ExecutaSom();}
    });

    btPause.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {PausaSom();}
    });

    btStop.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {PararSom();}
    });
```

Temos agora três métodos com suas respectivas funcionalidades associadas

1 usage

```
private void ExecutaSom() {
    if (mediaPlayer != null){
        mediaPlayer.start();
    }
}
```

1 usage

```
private void PausaSom(){
    if (mediaPlayer != null){
        mediaPlayer.pause();
    }
}
```

1 usage

```
private void ParaSom() {
    if (mediaPlayer != null){
        mediaPlayer.stop();
        mediaPlayer = MediaPlayer.create(getApplicationContext(), R.raw.musica);
    }
}
```

Ao acionar o método stop, a referência é perdida, assim é preciso reconfigurar.

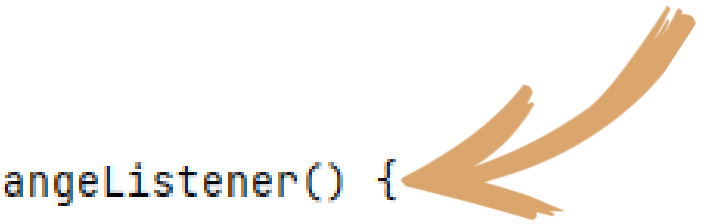
Recurso Volume

```
1 usage
private void InicializaSeekBar(){

    skbVolume = findViewById(R.id.skbVolume);

    //Configurar o audio Manager
    audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE );

    // Obtendo o estado do volume do aparelho
    int volumeMaximo = audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
    int volumeAtual = audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);

    // Integrando o volume do dispositivo com a seekbar
    skbVolume.setMax(volumeMaximo);
    skbVolume.setProgress(volumeAtual);
```

Vamos criar um novo método que inicializa o componente SeekBar, associando-o com o volume do dispositivo.

No método onCreate, devemos ter uma chamada para esse método.

Agora, basta configurar o evento



```
// Integrando o volume do dispositivo com a seekbar
skbVolume.setMax(volumeMaximo);
skbVolume.setProgress(volumeAtual);

skbVolume.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
    @Override
    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
        audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,progress, flags: 0);
    }

    @Override
    public void onStartTrackingTouch(SeekBar seekBar) {}

    @Override
    public void onStopTrackingTouch(SeekBar seekBar) {}
});
```

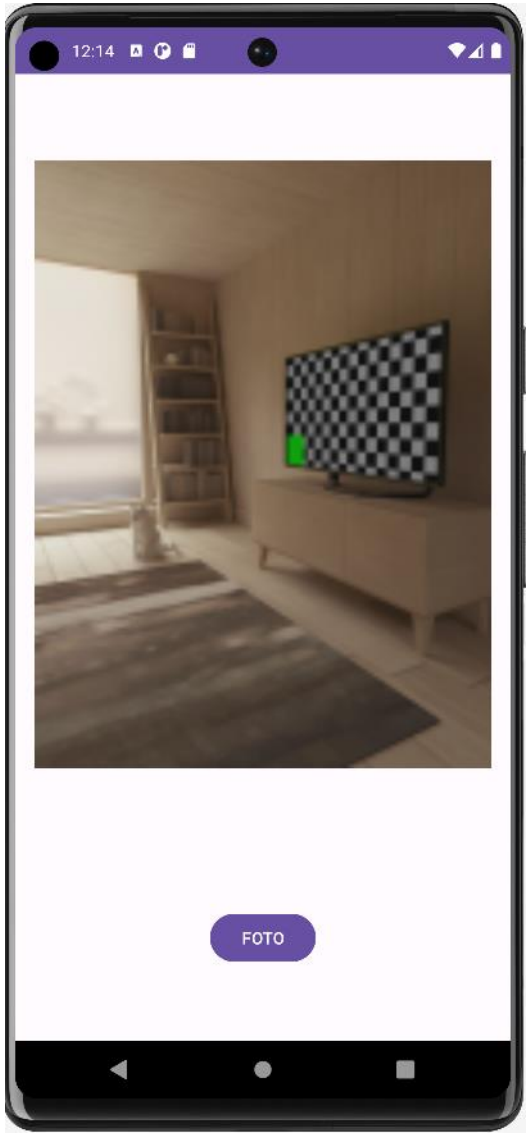
Recursos de Controle

```java
@Override
protected void onStop() {
    super.onStop();
    if (mediaPlayer != null){
        mediaPlayer.pause();
    }
}
```

Quando a activity for destruída, os recursos de memória devem ser liberados.

```
@Override
protected void onStop() {
    super.onStop();
    if (mediaPlayer != null){
        mediaPlayer.pause();
        mediaPlayer.release();
        mediaPlayer = null;
    }
}
```

Recursos da Câmera


12:14
FOTO

```
2 usages
public class MainActivity extends AppCompatActivity {

    2 usages
    private Button btFoto;
    1 usage
    private ImageView imgFoto;

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

        btFoto = findViewById(R.id.btFoto);
        imgFoto = findViewById(R.id.imgFoto);

        btFoto.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TirarFoto();
            }
        });
    }
    1 usage
    private void TirarFoto(){

    }
}
```

Permite que o aplicativo solicite o uso do recurso da câmera do dispositivo.

AndroidManifest.xml | activity_main.xml | MainActivity.java

```
4
5       <uses-feature
6           android:name="android.hardware.camera"
7           android:required="false" />
8       <uses-permission android:name="android.permission.CAMERA"> </uses-permission>
9
10      <application
11          android:allowBackup="true"
12          android:dataExtractionRules="@xml/data_extraction_rules"
```

Solicitando permissão ao usuário

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

    btFoto = findViewById(R.id.btFoto);
    imgFoto = findViewById(R.id.imgFoto);

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions( activity: this, new String[] {Manifest.permission.CAMERA}, requestCode: 0);
    }
```

Capturando Imagens

- O acesso ao recurso é realizando a partir do objeto Intent.
  - Expressa uma "intenção" do que se deseja realizar no APP
    - **`ACTION_IMAGE_CAPTURE`**, por exemplo, refere-se a obter os dados de imagem da câmera.
- A intenção é passada para um agente que configura o acesso e preparar a infraestrutura do dispositivo para realizar a operação
- Um contrato define um padrão para uma determinada ação e seus resultados esperados. (funciona como uma interface anônima, que indica funcionalidades e comportamentos esperados

Criando o objeto de acesso a câmera

```
1 usage
private ActivityResultLauncher<Intent> objCamera = registerForActivityResult(
            new ActivityResultContracts.StartActivityForResult(), new ActivityResultCallback<ActivityResult>() {
    @Override
    public void onActivityResult(ActivityResult result) {
        if (result != null && result.getResultCode() == RESULT_OK){

            Bundle extra = result.getData().getExtras();
            Bitmap imagem = (Bitmap) extra.get("data");

            imgFoto.setImageBitmap(imagem);

        }
    }
});
```

Agora, basta passar o método o objeto Intent para o registerForActivityResult() através do método launh()

```
1 usage
private void TirarFoto(){
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    objCamera.launch(intent);
}
```

QR-CODE

Para implementar a leitura do QR-CODE, vamos utilizar o IntentIntegrator, que é usado para integrar a funcionalidade de leitura de código QR em um aplicativo Android.

GradleModule

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    implementation 'com.google.zxing:core:3.5.1'
    implementation 'com.journeyapps:zxing-android-embedded:4.2.0'

}
```

Permite que o aplicativo solicite o uso do recurso da câmera do dispositivo.

AndroidManifest.xml | activity_main.xml | MainActivity.java

```
4
5       <uses-feature
6           android:name="android.hardware.camera"
7           android:required="false" />
8       <uses-permission android:name="android.permission.CAMERA"> </uses-permission>
9
10      <application
11          android:allowBackup="true"
12          android:dataExtractionRules="@xml/data_extraction_rules"
```

```java
public class MainActivity extends AppCompatActivity {

    2 usages
    private TextView txtGetQRCode;
    2 usages
    private Button btScanner;

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

        txtGetQRCode = findViewById(R.id.txtGetQRCode);
        btScanner = findViewById(R.id.btScanner);

        btScanner.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

            }
        });
    }
```

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

    txtGetQRCode = findViewById(R.id.txtGetQRCode);
    btScanner = findViewById(R.id.btScanner);

    btScanner.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            IntentIntegrator intent_integrator = new IntentIntegrator( activity: MainActivity.this);
            intent_integrator.setOrientationLocked(true);
            intent_integrator.setPrompt("Scan QRCode");
            intent_integrator.setDesiredBarcodeFormats(intent_integrator.QR_CODE);
            intent_integrator.initiateScan();
        }
    });
}
```

O método onActivityResult faz o papel de um Callbask. Ele é chamado automaticamente quando uma atividade é finalizada e retorna um resultado da leitura

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    IntentResult intentResult = IntentIntegrator.parseActivityResult(requestCode, resultCode,data);
    if (intentResult != null) {
        String content = intentResult.getContents();
        if (content != null) {
            txtGetQRCode.setText(intentResult.getContents());
        }
    }
}
```

Bons Estudos!