

Programação Para Dispositivos Móveis I

CAMERA E MEDIA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt

Índice

- Camera;
- Media Recorder;
- Leitura Adicional.

CameraX

CameraX é um novo componente e faz parte das bibliotecas do Android Jetpack, entre os principais objetivos encontramos:

- Facilidade de uso;
- Consistência através de vários dispositivos.

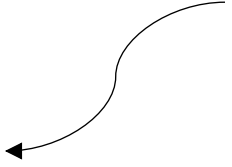
Serão apenas introduzidos aspetos iniciais desta API, mas a documentação pode ser consultada em <https://developer.android.com/training/camerax>

Câmara

Configuração [build.gradle (Project)]

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

Verificar a inclusão dos
repositórios da Google



Câmara

Configuração [build.gradle (Module)]

```
dependencies {  
  
    def camerax_version = '1.4.2'  
  
    implementation "androidx.camera:camera-core:$camerax_version"  
    implementation "androidx.camera:camera-camera2:$camerax_version"  
    implementation "androidx.camera:camera-lifecycle:$camerax_version"  
    implementation "androidx.camera:camera-video:$camerax_version"  
    implementation "androidx.camera:camera-view:$camerax_version"  
    implementation "androidx.camera:camera-extensions:$camerax_version"  
  
    ...  
}
```

Câmara

Permissões [AndroidManifest.xml]

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

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

Câmara

Permissões [MainActivity.java]

```
private static final int REQUEST_CAMERA_FEATURES = 200;

private static final String[] REQUIRED_PERMISSIONS =
    new String[]{
        Manifest.permission.CAMERA,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
    };
```

Lista de Permissões
necessárias

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

    if(!hasCameraPermission() || !hasStoragePermission()){
        requestPermissions();
    }
}
```

Se o utilizador não deu
permissões de câmara ou
de armazenamento externo,
solicitamos essas
permissões

Câmara

Permissões [MainActivity.java]

```
private boolean hasCameraPermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.CAMERA  
    ) == PackageManager.PERMISSION_GRANTED;  
}
```

Verificação de autorização de
utilização da câmara

```
private boolean hasStoragePermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.WRITE_EXTERNAL_STORAGE  
    ) == PackageManager.PERMISSION_GRANTED;  
}
```

Verificação de autorização de escrita
em armazenamento externo

```
private void requestPermissions(){  
    ActivityCompat.requestPermissions(this,  
        REQUIRED_PERMISSIONS, REQUEST_CAMERA_FEATURES);  
}
```


Câmara

Activity Camara [activity_camera.xml]

```
<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"
    tools:context=".CameraActivity">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/container">

        <androidx.camera.view.PreviewView
            android:id="@+id/preview_view"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
    </FrameLayout>

    <ImageButton
        android:id="@+id/btn_picture"
        android:layout_width="72dp"
        android:layout_height="72dp"
        android:layout_margin="24dp"
        android:background="?android:attr/selectableItemBackground"
        app:srcCompat="@android:drawable/ic_menu_camera"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

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

Elemento PreviewView que vai ser utilizado para mostrar a captura de imagem atual pela câmara

Câmara

[CameraActivity.java]

```
private PreviewView previewView;  
private ImageButton btnPicture;  
private ListenableFuture<ProcessCameraProvider> cameraProviderFuture;  
private ImageCapture imageCapture;  
  
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_camera);  
  
    previewView = findViewById(R.id.preview_view);  
    btnPicture = findViewById(R.id.btn_picture);  
    btnPicture.setOnClickListener(this::onClick);  
  
    cameraProviderFuture = ProcessCameraProvider.getInstance(this);  
    cameraProviderFuture.addListener((Runnable) this::run, ContextCompat.getMainExecutor(this));  
}
```

Adição do listener, e
respetivo Executor

Câmara

[CameraActivity.java]

```
private void run() {  
    try {  
        ProcessCameraProvider cameraProvider = cameraProviderFuture.get();  
        bindPreview(cameraProvider);  
    } catch (ExecutionException | InterruptedException exception) {  
        exception.printStackTrace();  
    }  
}
```

Método a ser executado pelo listener

Instanciação do cameraProvider

Método para gerir o preview da câmara

Câmara

[CameraActivity.java]

```
private void bindPreview(ProcessCameraProvider cameraProvider) {  
    Preview preview = new Preview.Builder()  
                      .build();  
  
    CameraSelector cameraSelector = new CameraSelector.Builder()  
                                    .requireLensFacing(CameraSelector.LENS_FACING_BACK)  
                                    .build();  
  
    preview.setSurfaceProvider(previewView.getSurfaceProvider());  
    cameraProvider.bindToLifecycle((LifecycleOwner) this, cameraSelector, preview);  
}
```

← Iniciar o preview da imagem

← Selecionar a câmara traseira

Câmara

Captura de Imagem [CameraActivity.java]

```
private void bindPreview(ProcessCameraProvider cameraProvider) {  
    ImageAnalysis imageAnalysis = new ImageAnalysis.Builder()  
        .setBackpressureStrategy(ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST)  
        .build();  
  
    imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(this), image -> {  
        image.close();  
    });  
  
    Preview preview = new Preview.Builder().build();  
  
    CameraSelector cameraSelector = new CameraSelector.Builder()  
        .requireLensFacing(CameraSelector.LENS_FACING_BACK)  
        .build();  
  
    imageCapture = new ImageCapture.Builder()  
        .setTargetRotation(this.getDisplay().getRotation())  
        .setFlashMode(ImageCapture.FLASH_MODE_AUTO)  
        .setCaptureMode(ImageCapture.CAPTURE_MODE_MAXIMIZE_QUALITY)  
        .build();  
  
    preview.setSurfaceProvider(previewView.getSurfaceProvider());  
    cameraProvider.bindToLifecycle((LifecycleOwner) this, cameraSelector, imageCapture, imageAnalysis, preview);  
}
```

Câmara

Captura de Imagem [CameraActivity.java]

```
private void onClick(View view) {  
    if(view == btnPicture){  
        takePicture();  
    }  
}  
  
private void takePicture() {  
    File cameraOutputPath = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM),  
        "IMG_" + System.currentTimeMillis() + ".jpg");  
  
    ImageCapture.OutputFileOptions cameraOutput = new ImageCapture.OutputFileOptions.Builder(cameraOutputPath).build();  
  
    imageCapture.takePicture(cameraOutput, ContextCompat.getMainExecutor(this), new ImageCapture.OnImageSavedCallback() {  
        @Override  
        public void onImageSaved(@NonNull ImageCapture.OutputFileResults outputFileResults) {  
            Toast.makeText(getApplicationContext(), "Picture saved!!", Toast.LENGTH_SHORT).show();  
        }  
  
        @Override  
        public void onError(@NonNull ImageCaptureException exception) {  
        }  
    }  
    );  
}
```

Path para guardar as fotografias tiradas

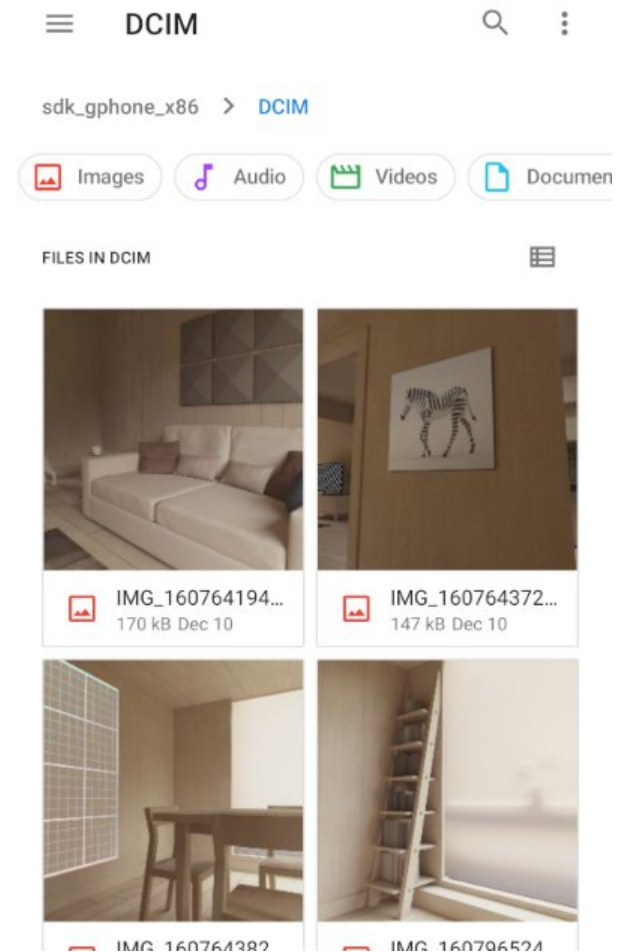
Obtenção do tempo atual em milissegundos para garantir nomes de ficheiros únicos

Câmara

Resultado



Captura de fotografia



Media

O Audio e Video, nomeadamente a sua captura e reprodução, pode ser feita através dos componentes Media para Android.

Entre outros, podemos encontrar nestes componentes

- MediaRecorder – gravação de vídeo e áudio;
- MediaPlayer – leitura de vídeo e áudio;

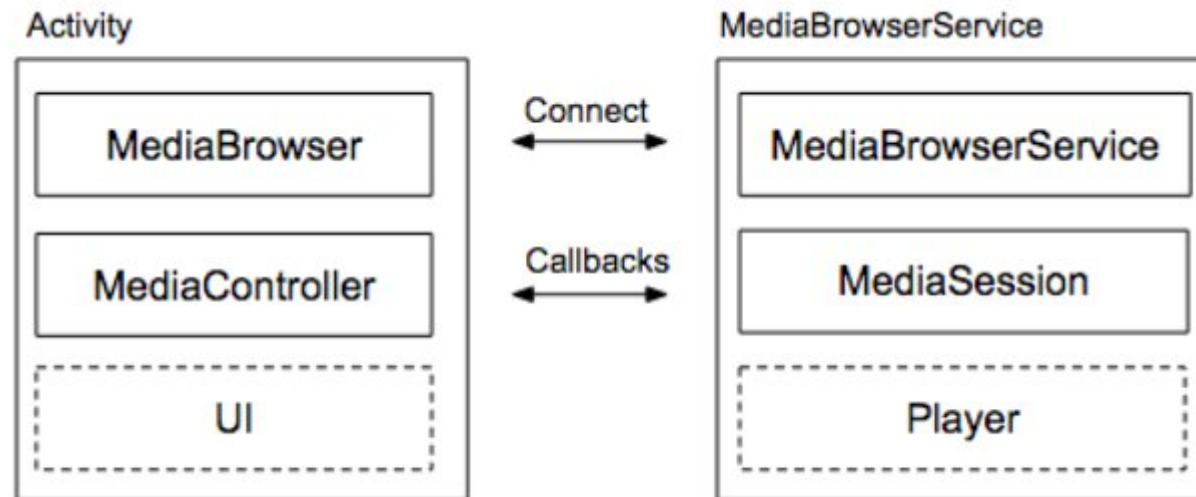
Os seguintes exemplos são focados na captura sonora, documentação e casos de uso adicionais podem ser encontrados em

<https://developer.android.com/guide/topics/media>

Media

Arquiteturas de Media Apps em Android

A gravação e leitura de Media é feita num serviço separado da Interface



Captura de Audio

Permissões [AndroidManifest.xml]

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

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

Captura de Audio

Permissões [MainActivity.java]

```
private static final int REQUEST_AUDIO_RECORD = 300;

private static final String[] REQUIRED_PERMISSIONS =
    new String[]{
        Manifest.permission.RECORD_AUDIO,
        Manifest.permission.WRITE_EXTERNAL_STORAGE
    };
```

Lista de Permissões
necessárias

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

    if(!hasRecordingPermission() || !hasStoragePermission()){
        requestPermissions();
    }
}
```

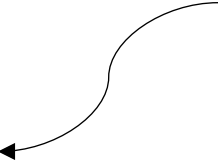
Se o utilizador não deu
permissões de captura de
áudio ou de armazenamento
externo, solicitamos essas
permissões

Captura de Audio

Permissões [MainActivity.java]


```
private boolean hasRecordingPermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.RECORD_AUDIO  
    ) == PackageManager.PERMISSION_GRANTED;  
}
```

Verificação de autorização
de utilização da captura de
áudio



```
private boolean hasStoragePermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.WRITE_EXTERNAL_STORAGE  
    ) == PackageManager.PERMISSION_GRANTED;  
}
```

Verificação de autorização
de escrita em
armazenamento externo



```
private void requestPermissions(){  
    ActivityCompat.requestPermissions(this,  
        REQUIRED_PERMISSIONS, REQUEST_AUDIO_RECORD);  
}
```

Captura de Audio

[MainActivity.java]

```
private void startRecording() {
```

```
    File recordingOutputPath = new File(Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DCIM),  
        "recording_" + System.currentTimeMillis() + ".3gp");
```

Path para guardar as
capturas de áudio

Obtenção do tempo atual em milissegundos para garantir nomes de ficheiros únicos

```
    mediaRecorder = new MediaRecorder();  
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);  
    mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);  
    if (Build.VERSION.SDK_INT ≥ Build.VERSION_CODES.O) {  
        mediaRecorder.setOutputFile(recordingOutputPath);  
    }  
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

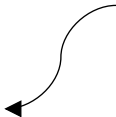
Capturar a partir do
microfone

```
    try {  
        mediaRecorder.prepare();  
    } catch (IOException ioException) {  
        ioException.printStackTrace();  
    } finally {  
        mediaRecorder.start();  
    }  
}
```

Captura de Audio

[MainActivity.java]

Tanto este método como o
startRecording estão
associados a botões



```
private void stopRecording(){  
  
    mediaRecorder.stop();  
    mediaRecorder.release();  
  
}
```

Reprodução de Audio

Permissões [AndroidManifest.xml]

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

Reprodução de Audio

Permissões [MainActivity.java]

```
private static final int REQUEST_AUDIO_PLAYBACK = 400;

private static final String[] REQUIRED_PERMISSIONS =
    new String[]{
        Manifest.permission.READ_EXTERNAL_STORAGE
    };
```

Lista de Permissões
necessárias

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

    if(!hasPlaybackPermission() || ){
        requestPermissions();
    }
}
```

Se o utilizador não deu
permissões de leitura de
armazenamento externo,
solicitamos essas
permissões

Reprodução de Audio

Permissões [MainActivity.java]

```
private boolean hasPlaybackPermission() {  
    return ContextCompat.checkSelfPermission(  
        this,  
        Manifest.permission.READ_EXTERNAL_STORAGE  
    ) == PackageManager.PERMISSION_GRANTED;  
}
```

Verificação de autorização
de leitura de
armazenamento externo

```
private void requestPermissions(){  
    ActivityCompat.requestPermissions(this,  
        REQUIRED_PERMISSIONS, REQUEST_AUDIO_RECORD);  
}
```

Reprodução de Audio

Serviço de Reprodução [MusicPlayerService.java]

```
public class MusicPlayerService extends Service {  
  
    public static final String EXTRA_MUSIC = "song_name";  
    private MediaPlayer mediaPlayer;  
  
    @Override  
    public void onCreate() {  
        super.onCreate();  
    }  
  
    @Nullable  
    @Override  
    public IBinder onBind(Intent intent) {  
        return null;  
    }  
}
```

Reprodução de Audio

Serviço de Reprodução [MusicPlayService.java]

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
    String resourcesPath = "android.resource://pt.ipp.estg.t_media/";
    mediaPlayer = new MediaPlayer();

    try {
        mediaPlayer.setDataSource(getApplicationContext(),
                                   Uri.parse(resourcesPath + intent.getIntExtra(this.EXTRA_MUSIC, 0)));
        mediaPlayer.setLooping(false);
        mediaPlayer.prepare();
    } catch (IOException ioException) {
        ioException.printStackTrace();
    } finally {
        mediaPlayer.start();
    }
    return START_NOT_STICKY;
}
```

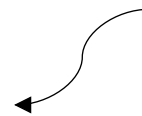
Acesso à pasta de resources. O Path deve ser substituído pelo path do projeto Android

Reprodução de Audio

Serviço de Reprodução [MusicPlayService.java]

```
@Override  
public void onDestroy() {  
    super.onDestroy();  
    mediaPlayer.release();  
}
```

Terminar a reprodução se o serviço for destruído.



Reprodução de Audio

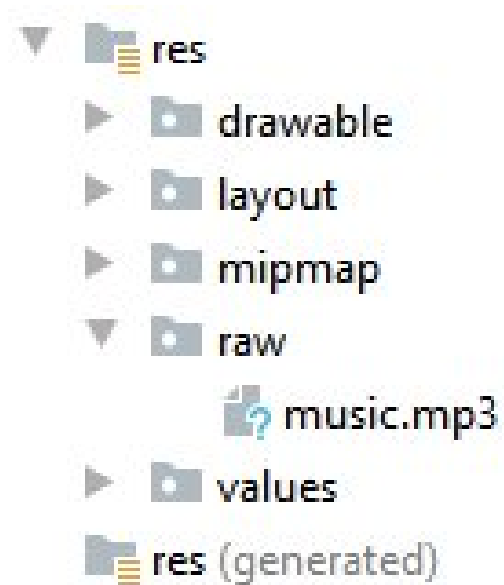
Reproduzir Música [MainActivity.java]

Este procedimento está
associado a um botão

```
private void startPlayer() {  
    intent = new Intent(this, MediaPlayerService.class);  
  
    intent.putExtra(MediaPlayerService.EXTRA_MUSIC, R.raw.music);  
    startService(intent);  
}
```

Reprodução de Audio

Localização dos ficheiros de media



Para efeitos de teste,
podemos adicionar o
ficheiro que queremos
reproduzir na pasta raw
(deve ser criada)

Leitura Recomendada

CameraX:

<https://developer.android.com/training/camerax>

Media Recorder:

<https://developer.android.com/guide/topics/media/mediarecorder>

Programação Para Dispositivos Móveis I

CAMERA E MEDIA

2024/_25 CTeSP – Desenvolvimento para a Web e Dispositivos Móveis

Ricardo Barbosa , rmb@estg.ipp.pt

Carlos Aldeias, cfpa@estg.ipp.pt

Adaptação do conteúdo dos slides de João Ramos jrmr@estg.ipp.pt e Fábio Silva fas@estg.ipp.pt