## TỔNG QUAN VỀ MULTIMEDIA APIS TRONG ANDROID

### ****Multimedia APIs là gì?****

**Multimedia APIs** trong Android là **tập hợp các giao diện lập trình ứng dụng (API)** cho phép các lập trình viên **xử lý và tương tác với các nội dung đa phương tiện** như **âm thanh, hình ảnh, video và camera** trong ứng dụng của mình.

- Để làm việc vớiMultimedia APIs trong ứng dụng Android, cần:

**- Xin Quyền (Permissions):**

+ Khai báo trong AndroidManifest.xml (ví dụ: CAMERA, RECORD\_AUDIO, READ/WRITE\_EXTERNAL\_STORAGE).

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

<uses-permission android:name="android.permission.RECORD\_AUDIO" />

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

+ Yêu cầu quyền lúc chạy ứng dụng (từ Android 6.0 trở lên).

**- Sử dụng các lớp thường dùng:**

+ MediaPlayer: Dùng để phát nhạc và video từ file, link mạng, hoặc tài nguyên trong ứng dụng.

+ MediaRecorder: Dùng để ghi âm thanh và video.

+ SurfaceView **/** TextureView: Các "màn hình" để hiển thị video hoặc preview từ camera.

**+ Quản lý Vòng đời (Lifecycle):** Rất quan trọng! Phải giải phóng tài nguyên (camera, MediaPlayer,...) khi Activity/Fragment bị tạm dừng (onPause) hoặc hủy (onDestroy) để tránh lỗi.

**1. MediaPlayer**

**- Chức năng:** Phát các tệp âm thanh và video.

Hỗ trợ nhiều định dạng như MP3, AAC, MP4, 3GP, M4A, ...

**- Có thể phát từ:**

+ File cục bộ (internal/external storage)

+ URI từ Internet

+ Resources (như R.raw.music)

**- Cung cấp các phương thức như:**

mediaPlayer.start(); // bắt đầu phát

mediaPlayer.pause(); // tạm dừng

mediaPlayer.stop(); // dừng phát

mediaPlayer.release(); // giải phóng tài nguyên

**Dùng cho:** ứng dụng phát nhạc, phát âm thanh hiệu ứng, xem video đơn giản.

### Phát Video

**\*VideoView (Đơn giản):**

Cách dễ nhất để phát video. Chỉ cần đặt nó vào layout XML và cung cấp đường dẫn video.

Có thể dùng MediaController để thêm các nút điều khiển (play, pause, tua).

// Trong Activity

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

String videoPath = "android.resource://" + getPackageName() + "/" + R.raw.ten\_video\_cua\_ban;

Uri uri = Uri.parse(videoPath);

videoView.setVideoURI(uri);

videoView.setMediaController(new MediaController(this)); // Thêm điều khiển

videoView.start(); // Bắt đầu phát

**\*MediaPlayer **với** SurfaceView**/**TextureView (Linh hoạt hơn):**

Bạn tự quản lý MediaPlayer và liên kết nó với một SurfaceView để hiển thị video.

Cho phép kiểm soát nhiều hơn so với VideoView.

### Phát Âm thanh

**\* MediaPlayer (Phổ biến):**

Dùng để phát các file âm thanh (nhạc mp3, âm báo,...).

Cách dùng tương tự như phát video với MediaPlayer.

// Phát từ thư mục raw

MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.ten\_audio\_cua\_ban);

mediaPlayer.start(); // Bắt đầu phát

mediaPlayer.setOnCompletionListener(mp -> {

mp.release(); // Giải phóng khi phát xong

});

**\*SoundPool (Cho âm thanh ngắn):**

Lý tưởng cho các hiệu ứng âm thanh ngắn, cần phát nhanh (tiếng click, âm thanh trong game).

Tải sẵn âm thanh vào bộ nhớ để độ trễ thấp.

|  |  |
| --- | --- |
| **TÊN** | **CODE** |
| AudioPlayMediaPlayerActivity.java | package com.example.multimediaapis;  import android.media.MediaPlayer; import android.os.Bundle; import android.widget.Button; import android.widget.Toast;  import androidx.appcompat.app.AppCompatActivity;  public class AudioPlayMediaPlayerActivity extends AppCompatActivity {   private MediaPlayer mediaPlayer;  private Button btnPlay, btnPause, btnStop;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_audio\_play\_mediaplayer*);   btnPlay = findViewById(R.id.*btnPlay*);  btnPause = findViewById(R.id.*btnPause*);  btnStop = findViewById(R.id.*btnStop*);   btnPlay.setOnClickListener(v -> playAudio());  btnPause.setOnClickListener(v -> pauseAudio());  btnStop.setOnClickListener(v -> stopAudio());  }   private void playAudio() {  if (mediaPlayer == null) {  *// Tạo mediaPlayer từ file trong res/raw* mediaPlayer = MediaPlayer.*create*(this, R.raw.*my\_audio*);  mediaPlayer.setOnCompletionListener(mp -> {  *// Tự động giải phóng khi phát xong* stopAudio();  Toast.*makeText*(this, "Phát xong!", Toast.*LENGTH\_SHORT*).show();  });  }  mediaPlayer.start();  Toast.*makeText*(this, "Đang phát...", Toast.*LENGTH\_SHORT*).show();  }   private void pauseAudio() {  if (mediaPlayer != null && mediaPlayer.isPlaying()) {  mediaPlayer.pause();  Toast.*makeText*(this, "Đã tạm dừng", Toast.*LENGTH\_SHORT*).show();  }  }   private void stopAudio() {  if (mediaPlayer != null) {  mediaPlayer.release();  mediaPlayer = null;  Toast.*makeText*(this, "Đã dừng", Toast.*LENGTH\_SHORT*).show();  }  }   @Override  protected void onStop() {  super.onStop();  *// Đảm bảo giải phóng tài nguyên khi activity không còn hiển thị* stopAudio();  } } |
| activity\_audio\_play\_mediaplayer.xml | *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:gravity="center"  android:orientation="vertical"  tools:context=".AudioPlayMediaPlayerActivity">   <TextView  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Phát Audio với MediaPlayer"  android:textSize="20sp"  android:layout\_marginBottom="20dp"/>   <Button  android:id="@+id/btnPlay"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Phát" />   <Button  android:id="@+id/btnPause"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="10dp"  android:text="Tạm Dừng" />   <Button  android:id="@+id/btnStop"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="10dp"  android:text="Dừng" />  </LinearLayout> |
| **RUN** |  |

**2. MediaRecorder**

**- Chức năng:** Ghi âm hoặc quay video bằng micro và camera.

**- Cho phép cấu hình:**

+ Nguồn âm thanh (MIC)

+ Nguồn hình ảnh (Camera)

+ Định dạng đầu ra (MP4, 3GP, AAC…)

**- Quy trình sử dụng:**

mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP);

mediaRecorder.setOutputFile("/path/to/file.3gp");

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);

mediaRecorder.prepare();

mediaRecorder.start();

**Dùng cho:** app ghi âm, camera quay video đơn giản.

### 2.1. Quay Video

**\*MediaRecorder (Thường dùng cùng CameraX hoặc Camera2):**

Là lớp chính để ghi video.

Khi dùng **CameraX**, VideoCapture sẽ giúp bạn cấu hình MediaRecorder dễ dàng hơn.

**Các bước chính:** Đặt nguồn video (từ camera), nguồn audio (mic), định dạng output, file lưu,... rồi gọi prepare() và start().

|  |  |
| --- | --- |
| **TÊN** | **CODE** |
| VideoRecordActivity.java | package com.example.multimediaapis;  import android.Manifest; import android.content.Intent; import android.content.pm.PackageManager; import android.media.MediaPlayer; import android.net.Uri; import android.os.Build; import android.os.Bundle; import android.provider.MediaStore; import android.util.Log; import android.view.SurfaceHolder; import android.view.SurfaceView; import android.widget.Button; import android.widget.Toast;  import androidx.annotation.NonNull; import androidx.annotation.Nullable; import androidx.appcompat.app.AppCompatActivity; import androidx.core.app.ActivityCompat; import androidx.core.content.ContextCompat;  public class VideoRecordActivity extends AppCompatActivity {   private static final int *REQUEST\_VIDEO\_CAPTURE* = 1;  private static final int *REQUEST\_PERMISSIONS* = 101;   private Uri videoUri;  private SurfaceView surfaceView;  private SurfaceHolder surfaceHolder;  private MediaPlayer mediaPlayer;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_video\_record*);   surfaceView = findViewById(R.id.*surfaceView*);  surfaceHolder = surfaceView.getHolder();   Button btnRecord = findViewById(R.id.*btnRecordVideo*);  btnRecord.setOnClickListener(view -> {  if (checkAndRequestPermissions()) {  launchCamera();  }  });  }   private void launchCamera() {  Intent takeVideoIntent = new Intent(MediaStore.*ACTION\_VIDEO\_CAPTURE*);  takeVideoIntent.putExtra(MediaStore.*EXTRA\_DURATION\_LIMIT*, 15);   if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {  startActivityForResult(takeVideoIntent, *REQUEST\_VIDEO\_CAPTURE*);  } else {  Toast.*makeText*(this, "Không tìm thấy ứng dụng camera", Toast.*LENGTH\_SHORT*).show();  }  }   @Override  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  super.onActivityResult(requestCode, resultCode, data);   if (requestCode == *REQUEST\_VIDEO\_CAPTURE* && resultCode == *RESULT\_OK* && data != null) {  videoUri = data.getData();  Log.*d*("VIDEO\_DEBUG", "Video URI: " + videoUri);   playVideoWithMediaPlayer();  }  }   private void playVideoWithMediaPlayer() {  if (videoUri == null) {  Toast.*makeText*(this, "Không có video để phát", Toast.*LENGTH\_SHORT*).show();  return;  }   surfaceHolder.addCallback(new SurfaceHolder.Callback() {  @Override  public void surfaceCreated(SurfaceHolder holder) {  try {  if (mediaPlayer != null) mediaPlayer.release();   mediaPlayer = new MediaPlayer();  mediaPlayer.setDataSource(VideoRecordActivity.this, videoUri);  mediaPlayer.setDisplay(holder);  mediaPlayer.setLooping(true);  mediaPlayer.prepare();  mediaPlayer.start();  } catch (Exception e) {  Log.*e*("VIDEO\_DEBUG", "Lỗi phát video: " + e.getMessage());  Toast.*makeText*(VideoRecordActivity.this, "Lỗi khi phát video", Toast.*LENGTH\_SHORT*).show();  }  }   @Override public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {}  @Override public void surfaceDestroyed(SurfaceHolder holder) {}  });  }   private boolean checkAndRequestPermissions() {  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {  boolean cameraGranted = ContextCompat.*checkSelfPermission*(this, Manifest.permission.*CAMERA*) == PackageManager.*PERMISSION\_GRANTED*;  boolean audioGranted = ContextCompat.*checkSelfPermission*(this, Manifest.permission.*RECORD\_AUDIO*) == PackageManager.*PERMISSION\_GRANTED*;  boolean readGranted = ContextCompat.*checkSelfPermission*(this,  Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU* ?  Manifest.permission.*READ\_MEDIA\_VIDEO* :  Manifest.permission.*READ\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_GRANTED*;   if (cameraGranted && audioGranted && readGranted) {  return true;  }   String[] permissions;  if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*TIRAMISU*) {  permissions = new String[]{  Manifest.permission.*CAMERA*,  Manifest.permission.*RECORD\_AUDIO*,  Manifest.permission.*READ\_MEDIA\_VIDEO* };  } else {  permissions = new String[]{  Manifest.permission.*CAMERA*,  Manifest.permission.*RECORD\_AUDIO*,  Manifest.permission.*READ\_EXTERNAL\_STORAGE* };  }   ActivityCompat.*requestPermissions*(this, permissions, *REQUEST\_PERMISSIONS*);  return false;  }  return true;  }   @Override  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  if (requestCode == *REQUEST\_PERMISSIONS*) {  boolean granted = true;  for (int result : grantResults) {  if (result != PackageManager.*PERMISSION\_GRANTED*) {  granted = false;  break;  }  }   if (granted) {  launchCamera();  } else {  Toast.*makeText*(this, "Bạn cần cấp đủ quyền để quay video", Toast.*LENGTH\_LONG*).show();  }  }  super.onRequestPermissionsResult(requestCode, permissions, grantResults);  }   @Override  protected void onDestroy() {  super.onDestroy();  if (mediaPlayer != null) mediaPlayer.release();  } } |
| activity\_video\_record.xml | <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:orientation="vertical"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:padding="16dp">   <Button  android:id="@+id/btnRecordVideo"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Quay video" />   <SurfaceView  android:id="@+id/surfaceView"  android:layout\_width="match\_parent"  android:layout\_height="300dp"  android:layout\_marginTop="16dp" /> </LinearLayout> |
| **RUN** |  |

### 2.2. Ghi Âm

**\*MediaRecorder(Phổ biến):**

Là lớp chính để ghi âm từ microphone.

**Các bước chính:** Đặt nguồn audio (MIC), định dạng output, bộ mã hóa âm thanh, file lưu,... rồi gọi prepare() và start().

MediaRecorder mediaRecorder = new MediaRecorder();

String outputFile = getExternalCacheDir().getAbsolutePath() + "/ghi\_am.3gp";

mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);

mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE\_GPP);

mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR\_NB);

mediaRecorder.setOutputFile(outputFile);

try {

mediaRecorder.prepare();

mediaRecorder.start();

// Để dừng: mediaRecorder.stop(); mediaRecorder.release();

} catch (IOException e) {

// Xử lý lỗi

}

**\*AudioRecord(Nâng cao):**

Cho phép đọc dữ liệu audio "thô" trực tiếp từ microphone để xử lý ở mức thấp. Phức tạp hơn nhưng linh hoạt cho các tác vụ như phân tích âm thanh thời gian thực.

|  |  |
| --- | --- |
| **TÊN** | **CODE** |
| VideoRecordActivity.java | package com.example.multimediaapis;  import android.Manifest; import android.content.pm.PackageManager; import android.media.MediaRecorder; import android.os.Bundle; import android.util.Log; import android.widget.Button; import android.widget.TextView; import android.widget.Toast;  import androidx.annotation.NonNull; import androidx.appcompat.app.AppCompatActivity; import androidx.core.app.ActivityCompat; import androidx.core.content.ContextCompat;  import java.io.IOException;  public class AudioRecordActivity extends AppCompatActivity {   private static final String *TAG* = "AudioRecordActivity";  private static final int *REQUEST\_CODE\_PERMISSIONS\_AUDIO* = 20;  private static final String[] *REQUIRED\_PERMISSIONS\_AUDIO* = new String[]{Manifest.permission.*RECORD\_AUDIO*};   private Button btnStartRecord, btnStopRecord;  private TextView tvStatus;   private MediaRecorder mediaRecorder;  private String outputFile;   @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_audio\_record*);   btnStartRecord = findViewById(R.id.*btnStartRecord*);  btnStopRecord = findViewById(R.id.*btnStopRecord*);  tvStatus = findViewById(R.id.*tvStatus*);   *// Đường dẫn lưu file ghi âm* outputFile = getExternalCacheDir().getAbsolutePath() + "/myaudio.3gp";   btnStartRecord.setOnClickListener(v -> {  if (checkPermissions()) {  startRecording();  } else {  requestPermissions();  }  });   btnStopRecord.setOnClickListener(v -> stopRecording());  }   private void startRecording() {  mediaRecorder = new MediaRecorder();  mediaRecorder.setAudioSource(MediaRecorder.AudioSource.*MIC*);  mediaRecorder.setOutputFormat(MediaRecorder.OutputFormat.*THREE\_GPP*);  mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.*AMR\_NB*);  mediaRecorder.setOutputFile(outputFile);   try {  mediaRecorder.prepare();  mediaRecorder.start();  tvStatus.setText("Đang ghi âm...");  btnStartRecord.setEnabled(false);  btnStopRecord.setEnabled(true);  Toast.*makeText*(this, "Bắt đầu ghi âm", Toast.*LENGTH\_SHORT*).show();  } catch (IOException e) {  Log.*e*(*TAG*, "prepare() failed", e);  Toast.*makeText*(this, "Ghi âm thất bại", Toast.*LENGTH\_SHORT*).show();  }  }   private void stopRecording() {  if (mediaRecorder != null) {  mediaRecorder.stop();  mediaRecorder.release();  mediaRecorder = null;  tvStatus.setText("Đã dừng. File được lưu tại:\n" + outputFile);  btnStopRecord.setEnabled(false);  btnStartRecord.setEnabled(true);  Toast.*makeText*(this, "Đã dừng ghi âm", Toast.*LENGTH\_SHORT*).show();  }  }   private boolean checkPermissions() {  return ContextCompat.*checkSelfPermission*(this, Manifest.permission.*RECORD\_AUDIO*) == PackageManager.*PERMISSION\_GRANTED*;  }   private void requestPermissions() {  ActivityCompat.*requestPermissions*(this, *REQUIRED\_PERMISSIONS\_AUDIO*, *REQUEST\_CODE\_PERMISSIONS\_AUDIO*);  }   @Override  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  super.onRequestPermissionsResult(requestCode, permissions, grantResults);  if (requestCode == *REQUEST\_CODE\_PERMISSIONS\_AUDIO*) {  if (grantResults.length > 0 && grantResults[0] == PackageManager.*PERMISSION\_GRANTED*) {  startRecording();  } else {  Toast.*makeText*(this, "Bạn phải cấp quyền để ghi âm.", Toast.*LENGTH\_SHORT*).show();  }  }  }   @Override  protected void onStop() {  super.onStop();  if (mediaRecorder != null) {  mediaRecorder.release();  mediaRecorder = null;  }  } } |
| activity\_audio\_record.xml | *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:gravity="center"  android:orientation="vertical"  tools:context=".AudioRecordActivity">   <TextView  android:id="@+id/tvStatus"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Sẵn sàng ghi âm"  android:textSize="18sp"  android:layout\_marginBottom="20dp"/>   <Button  android:id="@+id/btnStartRecord"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Bắt đầu Ghi Âm" />   <Button  android:id="@+id/btnStopRecord"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginTop="10dp"  android:enabled="false"  android:text="Dừng Ghi Âm" />  </LinearLayout> |
| **RUN** |  |

**3. Camera2 API**

**- Chức năng:** Giao tiếp trực tiếp với phần cứng Camera, kiểm soát nâng cao.

**- Cung cấp khả năng:**

+ Kiểm soát lấy nét tự động, phơi sáng, ISO, chế độ flash

+ Chụp ảnh RAW

+ Lấy luồng ảnh dạng byte để xử lý AI

+ Phức tạp hơn Camera API cũ (đã deprecated)

**- Cần tạo:**

CameraManager → CameraDevice → CaptureSession

**- Dùng cho**: ứng dụng camera chuyên nghiệp (máy ảnh ảo, chụp RAW, AR…).

Cách hiện đại và được khuyên dùng nhất là **CameraX**:

**\*CameraX (Khuyên dùng):** Là thư viện Jetpack giúp việc lập trình camera dễ dàng hơn nhiều. Nó tự xử lý nhiều vấn đề phức tạp trên các dòng máy Android khác nhau.

**Các thành phần chính:**

Preview: Hiển thị hình ảnh trực tiếp từ camera.

ImageCapture: Chụp ảnh tĩnh.

VideoCapture (hoặc Recorder): Quay video.

**Luồng làm việc cơ bản:** Xin quyền -> Lấy ProcessCameraProvider -> Chọn camera (trước/sau) -> Gắn các "use case" (Preview, ImageCapture,...) vào vòng đời của Activity/Fragment.

**\*Camera2 API (Nâng cao):** Cho phép kiểm soát camera chi tiết hơn nhưng phức tạp hơn.

**\*Camera API (Cũ - Không nên dùng):** Đã lỗi thời.

|  |  |
| --- | --- |
| **TÊN** | **CODE** |
| CameraXActivity.java | package com.example.multimediaapis;  import android.Manifest;  import android.content.Intent;  import android.content.pm.PackageManager;  import android.media.MediaPlayer;  import android.net.Uri;  import android.os.Build;  import android.os.Bundle;  import android.provider.MediaStore;  import android.util.Log;  import android.view.SurfaceHolder;  import android.view.SurfaceView;  import android.widget.Button;  import android.widget.Toast;  import androidx.annotation.NonNull;  import androidx.annotation.Nullable;  import androidx.appcompat.app.AppCompatActivity;  import androidx.core.app.ActivityCompat;  import androidx.core.content.ContextCompat;  public class VideoRecordActivity extends AppCompatActivity {  private static final int REQUEST\_VIDEO\_CAPTURE = 1;  private static final int REQUEST\_PERMISSIONS = 101;  private Uri videoUri;  private SurfaceView surfaceView;  private SurfaceHolder surfaceHolder;  private MediaPlayer mediaPlayer;  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.activity\_video\_record);  surfaceView = findViewById(R.id.surfaceView);  surfaceHolder = surfaceView.getHolder();  Button btnRecord = findViewById(R.id.btnRecordVideo);  btnRecord.setOnClickListener(view -> {  if (checkAndRequestPermissions()) {  launchCamera();  }  });  }  private void launchCamera() {  Intent takeVideoIntent = new Intent(MediaStore.ACTION\_VIDEO\_CAPTURE);  takeVideoIntent.putExtra(MediaStore.EXTRA\_DURATION\_LIMIT, 15);  if (takeVideoIntent.resolveActivity(getPackageManager()) != null) {  startActivityForResult(takeVideoIntent, REQUEST\_VIDEO\_CAPTURE);  } else {  Toast.makeText(this, "Không tìm thấy ứng dụng camera", Toast.LENGTH\_SHORT).show();  }  }  @Override  protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {  super.onActivityResult(requestCode, resultCode, data);  if (requestCode == REQUEST\_VIDEO\_CAPTURE && resultCode == RESULT\_OK && data != null) {  videoUri = data.getData();  Log.d("VIDEO\_DEBUG", "Video URI: " + videoUri);  playVideoWithMediaPlayer();  }  }  private void playVideoWithMediaPlayer() {  if (videoUri == null) {  Toast.makeText(this, "Không có video để phát", Toast.LENGTH\_SHORT).show();  return;  }  surfaceHolder.addCallback(new SurfaceHolder.Callback() {  @Override  public void surfaceCreated(SurfaceHolder holder) {  try {  if (mediaPlayer != null) mediaPlayer.release();  mediaPlayer = new MediaPlayer();  mediaPlayer.setDataSource(VideoRecordActivity.this, videoUri);  mediaPlayer.setDisplay(holder);  mediaPlayer.setLooping(true);  mediaPlayer.prepare();  mediaPlayer.start();  } catch (Exception e) {  Log.e("VIDEO\_DEBUG", "Lỗi phát video: " + e.getMessage());  Toast.makeText(VideoRecordActivity.this, "Lỗi khi phát video", Toast.LENGTH\_SHORT).show();  }  }  @Override public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {}  @Override public void surfaceDestroyed(SurfaceHolder holder) {}  });  }  private boolean checkAndRequestPermissions() {  if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.M) {  boolean cameraGranted = ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) == PackageManager.PERMISSION\_GRANTED;  boolean audioGranted = ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD\_AUDIO) == PackageManager.PERMISSION\_GRANTED;  boolean readGranted = ContextCompat.checkSelfPermission(this,  Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU ?  Manifest.permission.READ\_MEDIA\_VIDEO :  Manifest.permission.READ\_EXTERNAL\_STORAGE) == PackageManager.PERMISSION\_GRANTED;  if (cameraGranted && audioGranted && readGranted) {  return true;  }  String[] permissions;  if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.TIRAMISU) {  permissions = new String[]{  Manifest.permission.CAMERA,  Manifest.permission.RECORD\_AUDIO,  Manifest.permission.READ\_MEDIA\_VIDEO  };  } else {  permissions = new String[]{  Manifest.permission.CAMERA,  Manifest.permission.RECORD\_AUDIO,  Manifest.permission.READ\_EXTERNAL\_STORAGE  };  }  ActivityCompat.requestPermissions(this, permissions, REQUEST\_PERMISSIONS);  return false;  }  return true;  }  @Override  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  if (requestCode == REQUEST\_PERMISSIONS) {  boolean granted = true;  for (int result : grantResults) {  if (result != PackageManager.PERMISSION\_GRANTED) {  granted = false;  break;  }  }  if (granted) {  launchCamera();  } else {  Toast.makeText(this, "Bạn cần cấp đủ quyền để quay video", Toast.LENGTH\_LONG).show();  }  }  super.onRequestPermissionsResult(requestCode, permissions, grantResults);  }  @Override  protected void onDestroy() {  super.onDestroy();  if (mediaPlayer != null) mediaPlayer.release();  }  } |
| activity\_camerax.xml | *<?*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"  tools:context=".CameraXActivity">   <androidx.camera.view.PreviewView  android:id="@+id/previewView"  android:layout\_width="match\_parent"  android:layout\_height="0dp"  android:layout\_weight="1"  android:layout\_margin="8dp"  android:layout\_gravity="center"  app:implementationMode="performance" />   <Button  android:id="@+id/btnCapture"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:layout\_marginBottom="50dp"  android:text="Chụp Ảnh"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintEnd\_toEndOf="parent"  app:layout\_constraintStart\_toStartOf="parent" />  </androidx.constraintlayout.widget.ConstraintLayout> |
| **RUN** | Screenshot 2025-06-07 123301 |

**4. ExoPlayer**

**- Chức năng:** Phát video/audio nâng cao, stream từ mạng, hỗ trợ DASH, HLS, v.v.

Google phát triển, hỗ trợ tốt hơn MediaPlayer

- Hỗ trợ:

Phụ đề, Playlist, Streaming online, Tùy biến UI,...

**- Cấu hình đơn giản:**

ExoPlayer player = new ExoPlayer.Builder(context).build();

player.setMediaItem(MediaItem.fromUri(uri));

player.prepare();

player.play();

**Dùng cho**: YouTube-like app, video stream, app nghe nhạc online.

**\*ExoPlayer (Khuyên dùng cho hầu hết trường hợp):**

Thư viện của Google, mạnh mẽ và linh hoạt hơn MediaPlayer.

Rất tốt cho streaming, video có định dạng phức tạp, tùy chỉnh giao diện.

Cần thêm thư viện ExoPlayer vào dự án.

|  |  |
| --- | --- |
| **TÊN** | **CODE** |
| VideoPlayExoPlayerActivity.java | package com.example.multimediaapis;  import android.annotation.SuppressLint; import android.net.Uri; import android.os.Bundle;  import androidx.appcompat.app.AppCompatActivity;  import com.google.android.exoplayer2.ExoPlayer; import com.google.android.exoplayer2.MediaItem; import com.google.android.exoplayer2.Player; import com.google.android.exoplayer2.ui.PlayerView;   public class VideoPlayExoPlayerActivity extends AppCompatActivity {   private PlayerView playerView;  private ExoPlayer exoPlayer;   @SuppressLint("MissingInflatedId")  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_video\_play\_exoplayer*);   playerView = findViewById(R.id.*playerView*);  }   private void initializePlayer() {  exoPlayer = new ExoPlayer.Builder(this).build();  playerView.setPlayer(exoPlayer);   *// Đường dẫn đến file video trong thư mục res/raw* Uri videoUri = Uri.*parse*("android.resource://" + getPackageName() + "/" + R.raw.*my\_video*);  MediaItem mediaItem = MediaItem.*fromUri*(videoUri);   exoPlayer.setMediaItem(mediaItem);  exoPlayer.setRepeatMode(Player.*REPEAT\_MODE\_ONE*); *// Lặp video (test thử)* exoPlayer.setPlayWhenReady(true);  exoPlayer.prepare();   }   private void releasePlayer() {  if (exoPlayer != null) {  exoPlayer.release();  exoPlayer = null;  }  }   @Override  protected void onStart() {  super.onStart();  *// API level 24+ thì initialize ở onStart, release ở onStop* initializePlayer();  }   @Override  protected void onResume() {  super.onResume();  if (exoPlayer == null) {  *// API level < 24 thì initialize ở onResume, release ở onPause* initializePlayer();  }  }   @Override  protected void onPause() {  super.onPause();  *// API level < 24* releasePlayer();  }   @Override  protected void onStop() {  super.onStop();  *// API level 24+* releasePlayer();  } } |
| activity\_video\_play\_exoplayer.xml | *<?*xml version="1.0" encoding="utf-8"*?>* <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:context=".VideoPlayExoPlayerActivity">   <com.google.android.exoplayer2.ui.PlayerView  android:id="@+id/playerView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" />  </androidx.constraintlayout.widget.ConstraintLayout> |
| **RUN** |  |