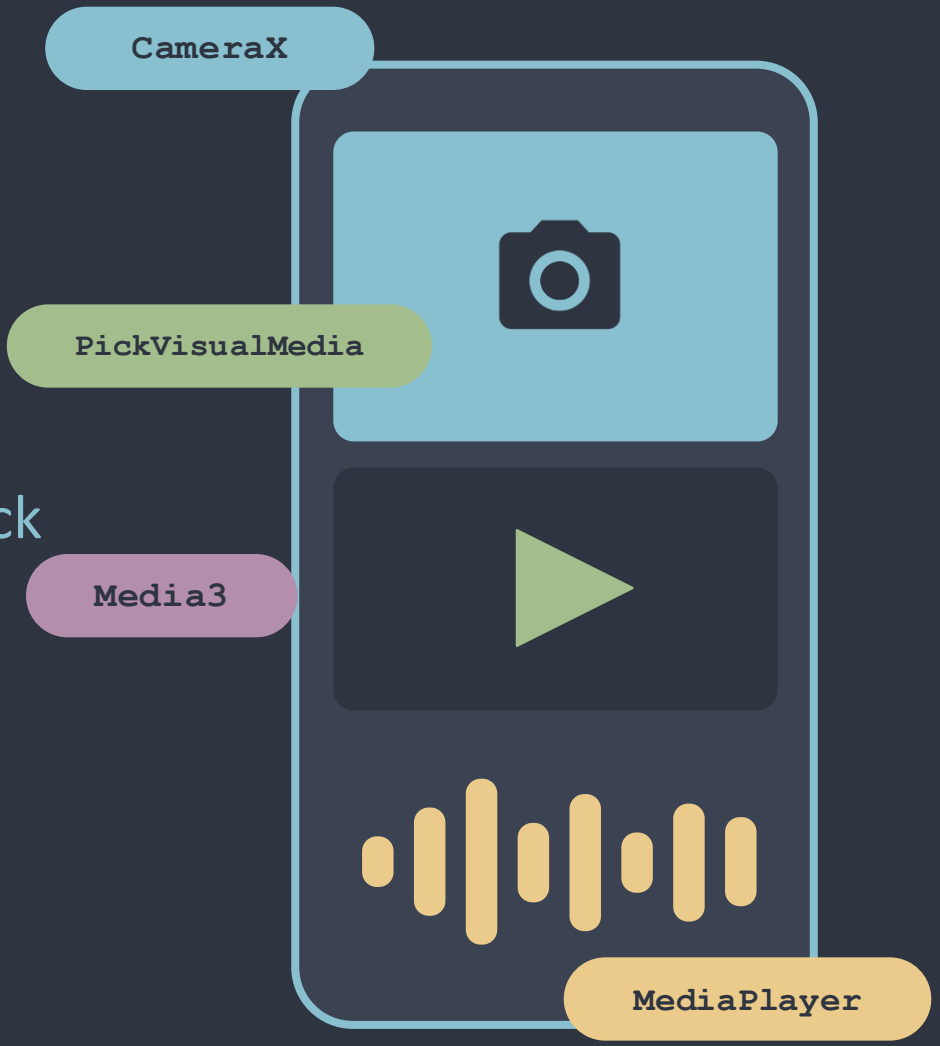
Bloque 4 · Tema 7

Multimedia

Cámara, galería, audio y vídeo en Android con Jetpack Compose

9 horas · AppDummy v8 → v9 · RA3



Programación Multimedia y Dispositivos Móviles (0489) · 2º CFGS DAM

¿Qué añadimos a AppDummy?

`v8 → T7`

Portada propia

Foto con la cámara o imagen de la galería que sustituye a la de Open Library.

`CameraX · PickVisualMedia`

Muestra de audio

Reproductor con barra de progreso para un fragmento sonoro.

`MediaPlayer`

Reproductor de vídeo

Vídeo en streaming con controles estándar.

`Media3 · ExoPlayer`

Hilo conductor: la arquitectura no cambia



Hoja de ruta del tema

10 apartados

1

Dependencias

CameraX y Media3

2

Permisos

En tiempo de ejecución

3

Almacenamiento

filesDir/portadas/

4

Room

Columna portada_local

5

CameraX

Previsualizar y fotografiar

6

Selector

PickVisualMedia

7

Integración

Navegar y devolver resultado

8

Audio

MediaPlayer

9

Vídeo

Media3 · ExoPlayer

10

Grabación

VideoCapture (ampliación)

¿Qué vamos a evaluar?

RA3



RA3

Desarrolla programas que integran contenidos multimedia analizando y empleando las tecnologías y librerías específicas.

CE: RA3-a, b, c, e, g, h

RA3-a

Tecnologías multimedia en dispositivos móviles

RA3-b

Clases de captura, procesamiento y almacenamiento

RA3-c

Conversión de datos multimedia entre formatos

RA3-e

Eventos, tipos de media y excepciones

RA3-g

Extraer, transformar y almacenar datos multimedia

RA3-h

Depurar y documentar los programas

★ *RA3-b vertebra el tema: capturar (CameraX), procesar y almacenar (filesDir + Room)*

01

Preparación

Dependencias, manifiesto y permisos en tiempo de ejecución



Tres tecnologías, tres dependencias

Gradle KTS

CameraX

5 artefactos, misma versión — `1.6.2`

Media3

exoplayer + ui — `1.11.0`

MediaPlayer

Sin dependencias — `framework`

camera-camera2 es obligatorio
Es la implementación real: sin él la cámara no arranca.

gradle/libs.versions.toml

```
[versions]
cameraX = "1.6.2"
media3 = "1.11.0"
```

app/build.gradle.kts

```
implementation(libs.androidx.camera.core)
implementation(libs.androidx.camera.camera2)
implementation(libs.androidx.camera.lifecycle)
implementation(libs.androidx.camera.view)   // PreviewView
implementation(libs.androidx.camera.video)
implementation(libs.androidx.media3.exoplayer)
implementation(libs.androidx.media3.ui)     // PlayerView
```

Permisos en el manifiesto

AndroidManifest

AndroidManifest.xml

```
<uses-feature
    android:name="android.hardware.camera"
    android:required="false" />

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



required="false"
La app se puede instalar en dispositivos sin cámara: la función se desactiva, no la app entera.

INTERNET

Se concede al instalar — Normal

CAMERA

Ya presente desde v3 — Peligroso

RECORD_AUDIO

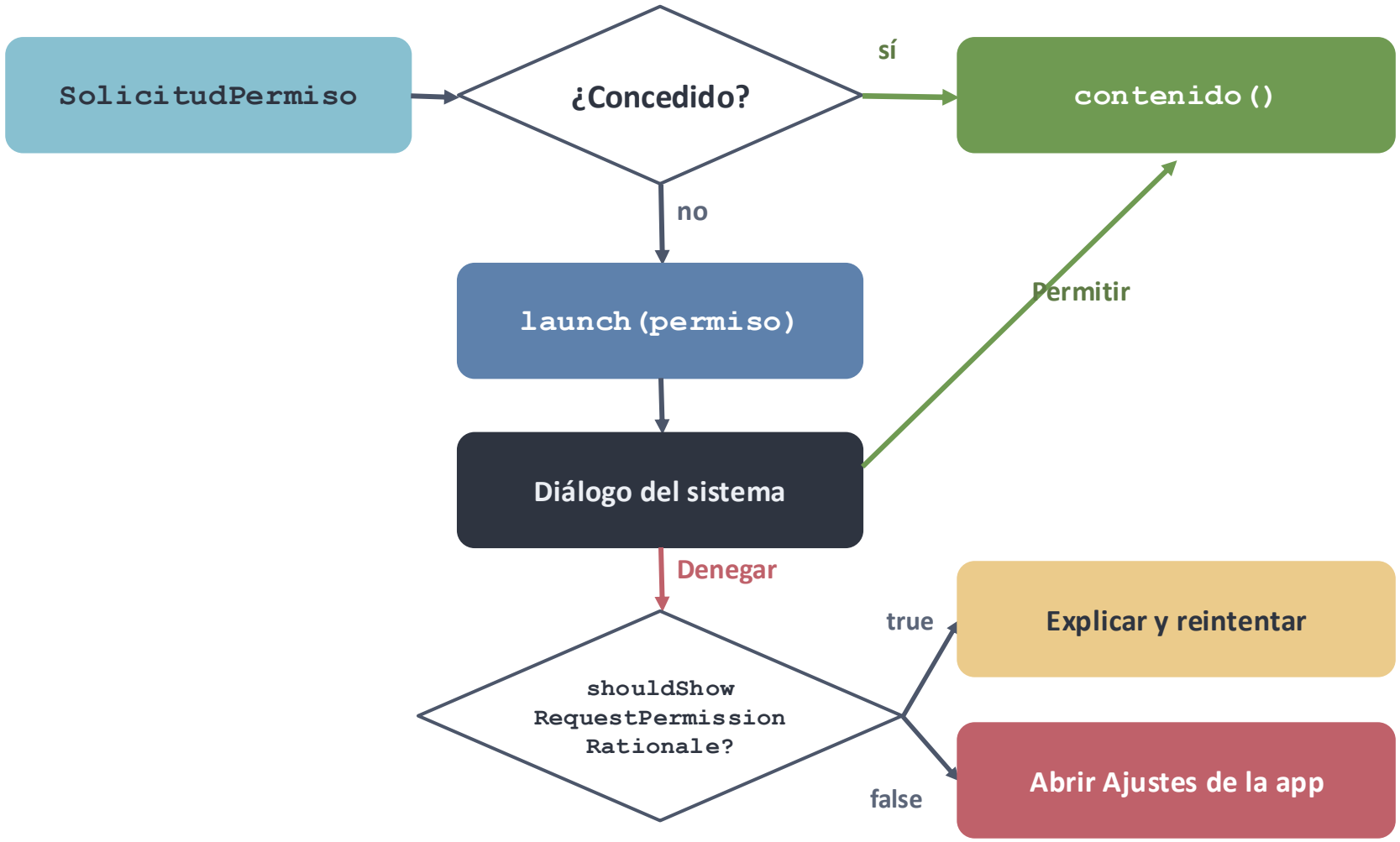
Nuevo: grabar vídeo con sonido — Peligroso

Almacenamiento

filesDir y selector de fotos — No se necesita

Permisos peligrosos en tiempo de ejecución

`Activity Result API`



Concedido

Se muestra el contenido protegido.

Primera petición

Botón que lanza el contrato RequestPermission.

Denegado

Reintentar o abrir los Ajustes de la app.

SolicitudPermiso: un envoltorio reutilizable

Compose

ui/componentes/SolicitudPermiso.kt (simplificado)

```kotlin
@Composable
fun SolicitudPermiso(
    permiso: String,
    contenido: @Composable () -> Unit
) {
    val context = LocalContext.current
    var concedido by remember { mutableStateOf(
        ContextCompat.checkSelfPermission(context, permiso)
            == PackageManager.PERMISSION_GRANTED) }

    val launcher = rememberLauncherForActivityResult(
        ActivityResultContracts.RequestPermission()
    ) { ok -> concedido = ok }

    if (concedido) contenido()
    else Button(onClick = { launcher.launch(permiso) }) {
        Text("Conceder permiso")
    }
}
```

RequestPermission

Un permiso → callback con Boolean

```
Uso: SolicitudPermiso(CAMERA) {
PantallaCamara() }
```

RequestMultiplePermissions

Varios permisos → Map<String, Boolean>

Cámara + micrófono para grabar vídeo con sonido.

02

Portada propia

Almacenamiento, Room, CameraX y selector de fotos



¿Dónde guardamos la imagen?

Almacenamiento



filesDir

Elegido

`filesDir/portadas/`

- ✓ Privado de la app
- ✓ No requiere permisos
- ✓ Rutas estables y controladas
- ✗ Invisible en la galería
- ✗ Se borra al desinstalar



MediaStore

`Pictures/… (almacenamiento compartido)`

- ✓ Visible en la galería
- ✓ Sobrevive a la desinstalación
- ✗ Contenido compartido con otras apps
- ✗ Gestión de Uri y ContentValues
- ✗ Más complejo para una portada

AlmacenPortadas: un fichero nuevo cada vez

`filesDir`

data/datasource/local/AlmacenPortadas.kt (simplificado)

```
object AlmacenPortadas {
    private fun carpeta(ctx: Context) =
        File(ctx.filesDir, "portadas").apply { mkdirs() }

    fun nuevoFicheroPara(ctx: Context, id: String) =
        File(carpeta(ctx),
            "${id.substringAfterLast('/')}_" +
            "${System.currentTimeMillis()}.jpg")

    fun copiarDesdeUri(ctx: Context, uri: Uri, destino: File) {
        ctx.contentResolver.openInputStream(uri)?.use { input ->
            destino.outputStream().use { input.copyTo(it) }
        }   // use {} cierra los flujos aunque haya excepción
    }

    fun limpiarAnteriores(ctx: Context, id: String, actual: File)
    // borra el resto de portadas de ese libro
}
```

```
files/
└ portadas/
    └ OL45883W_1730622001.jpg
    └ OL45883W_1730622950.jpg
    └ OL27482W_1730623110.jpg
```



¿Por qué la marca de tiempo?

Ruta distinta = StateFlow, Compose y la caché de Coil detectan el cambio y repintan.

Room: la ruta, no la imagen

Room · v3

id	titulo	cover	portada_local
OL45883W	Hail Mary	https://…	/data/…/OL45883W_….jpg
OL27482W	Juego de…	https://…	null

```kotlin
// Libro.kt
@ColumnInfo(name = "portada_local")
val portadaLocal: String? = null   // solo local

// LibrosDao.kt
@Query("UPDATE libros SET portada_local = :ruta
        WHERE id = :id")
suspend fun guardarPortadaLocal(id: String, ruta: String?)
```



version = 3

Destrucción de tablas: fallbackToDestructiveMigration



Excluir de la sincronización

Como esFavorito: la red no debe pisarla

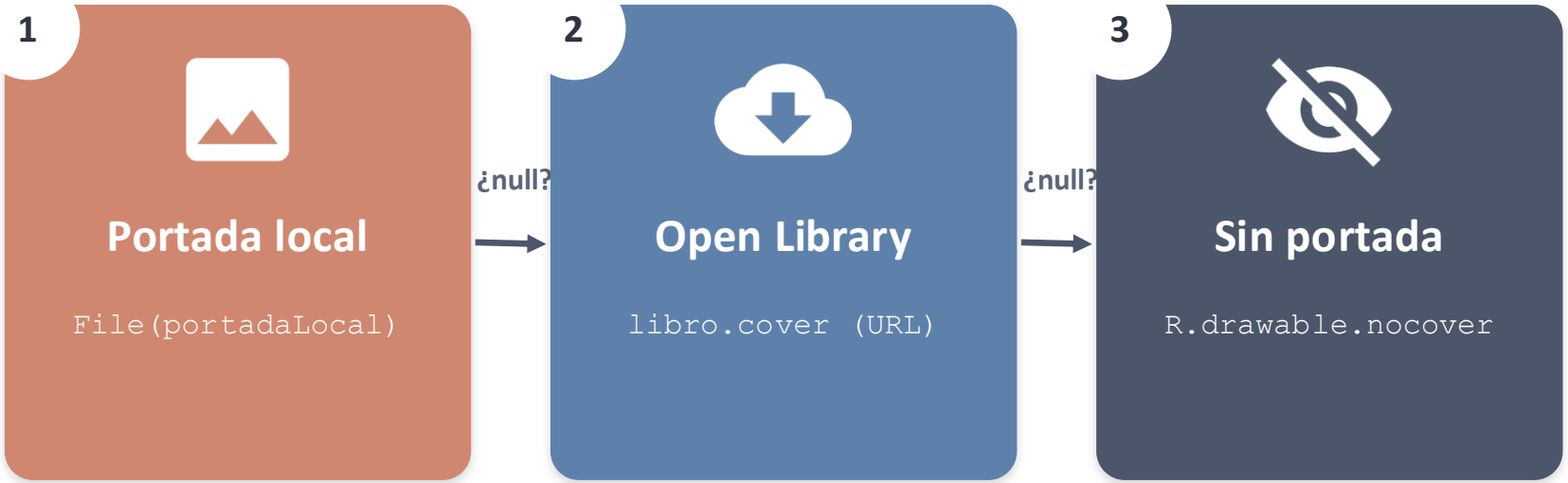


Mismo flujo de capas

DAO → LocalDataSource → Repository

CaratulaLibro: ¿qué imagen se muestra?

Coil



ui/componentes/CaratulaLibro.kt (simplificado)

```
val modelo: Any = when {
    libro.portadaLocal != null -> File(libro.portadaLocal)
    libro.cover != null        -> libro.cover
    else                       -> R.drawable.nocover
}
AsyncImage(model = modelo, contentDescription = libro.titulo)
```

Caché de Coil

Coil cachea por la clave del modelo. Si se sobrescribe el mismo fichero, puede mostrar la imagen antigua.

Solución del tema: un nombre de fichero nuevo en cada captura.

DetalleViewModel pasa a AndroidViewModel

ViewModel

Galería `asignarPortadaLocal(uri)`

Copiar → guardar ruta → limpiar → recargar

Cámara `asignarPortadaCapturada(ruta)`

Fichero ya escrito: solo guardar ruta

Quitar `quitarPortadaLocal()`

Ruta a null y borrar ficheros

No unificar galería y cámara
Copiar un fichero que ya está en filesDir es trabajo inútil.

DetalleViewModel.kt (simplificado)

```
class DetalleViewModel(
    app: Application,
    private val repo: LibrosRepository,
    private val id: String
) : AndroidViewModel(app) {

  fun asignarPortadaLocal(uri: Uri) =
    viewModelScope.launch {
      val ctx = getApplication<Application>()
      val f = AlmacenPortadas
          .nuevoFicheroPara(ctx, id)
      AlmacenPortadas
          .copiarDesdeUri(ctx, uri, f)
      repo.guardarPortadaLocal(id,
          f.absolutePath)
      AlmacenPortadas
          .limpiarAnteriores(ctx, id, f)
    }
}
```

CameraX: la cámara por casos de uso

CameraX

Preview

Imagen en directo en un PreviewView

ImageCapture

Hacer fotografías

ImageAnalysis

Procesar fotogramas (ML, QR…)

VideoCapture

Grabar vídeo con un Recorder

Preview + ImageCapture = combinación garantizada

Se usa en T7

awaitInstance()

ProcessCameraProvider

→

unbindAll()

Libera usos previos

→

bindToLifecycle()

owner + selector + casos

→

Ciclo de vida

Abre y cierra la cámara solo

PreviewView dentro de Compose

AndroidView

ui/camara/PantallaCamara.kt (simplificado)

```
val preview = remember { Preview.Builder().build() }
val imageCapture = remember { ImageCapture.Builder().build() }
val owner = LocalLifecycleOwner.current

LaunchedEffect(selector) {          // trasera / frontal
    val provider = ProcessCameraProvider.awaitInstance(context)
    provider.unbindAll()
    provider.bindToLifecycle(owner, selector, preview, imageCapture)
}

AndroidView(
    factory = { ctx ->              // se ejecuta UNA vez
        PreviewView(ctx).apply {
            scaleType = PreviewView.ScaleType.FILL_CENTER
        }
    },
    update = { view ->              // en cada recomposición
        preview.setSurfaceProvider(view.surfaceProvider)
    }
)
```

factory Crea la View clásica una sola vez

update Sincroniza la View con el estado actual



Ampliación

camera-compose ofrece CameraXViewfinder, un visor nativo de Compose sin AndroidView.

Hacer la foto: takePicture

ImageCapture

Pulsar disparador

Estado: capturando

nuevoFicheroPara()

filesDir/portadas/...

takePicture()

OutputFileOptions(File)

onImageSaved

Ruta al detalle

```
val fichero = AlmacenPortadas.nuevoFicheroPara(context, libroId)
val opciones = ImageCapture.OutputFileOptions.Builder(fichero).build()

imageCapture.takePicture(opciones, ContextCompat.getMainExecutor(context),
    object : ImageCapture.OnImageSavedCallback {
        override fun onImageSaved(r: ImageCapture.OutputFileResults) =
            onFotoHecha(fichero.absolutePath)
        override fun onError(e: ImageCaptureException) = onError(e)
    })
```

RA3-e

La captura puede fallar: ImageCaptureException llega en onError y la UI debe informar al usuario.

Galería: el selector de fotos del sistema

`PickVisualMedia`

```kotlin
// ui/detalle/SelectorPortada.kt
val selector = rememberLauncherForActivityResult(
    ActivityResultContracts.PickVisualMedia()
) { uri: Uri? ->
    uri?.let { viewModel.asignarPortadaLocal(it) }
}

selector.launch(PickVisualMediaRequest(
    ActivityResultContracts.PickVisualMedia.ImageOnly))
```

	`PickVisualMedia`	`ACTION_GET_CONTENT`
Permisos	Ninguno	Según proveedor
Acceso	Solo lo elegido	Depende de la app
Interfaz	Del sistema	Variable



La Uri es temporal

`content://media/picker/…`



Hoy — Se ve la portada

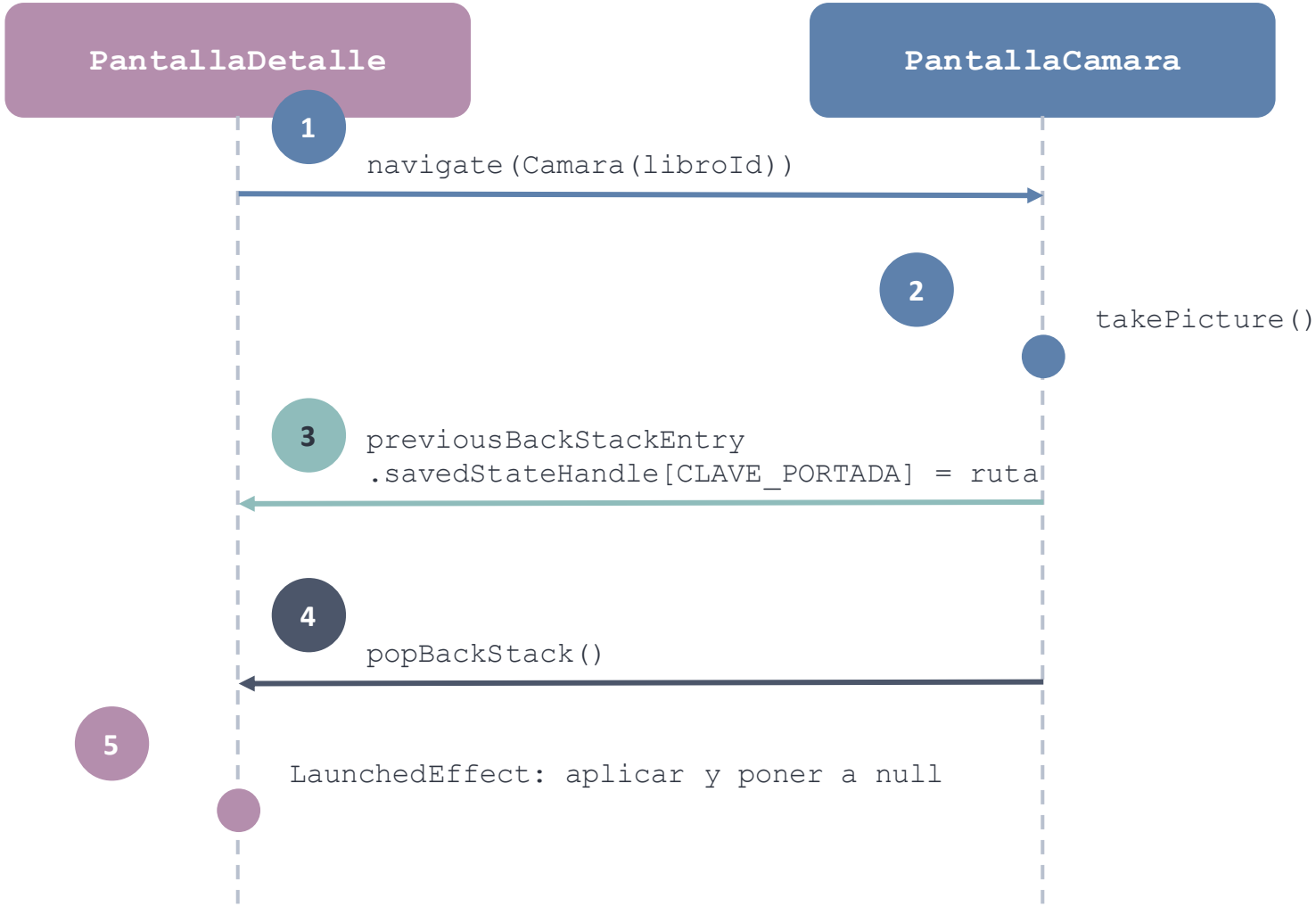


Mañana — Permiso caducado → sin imagen

Por eso se copia a un fichero propio y en Room se guarda esa ruta.

Devolver la foto al detalle

savedStateHandle



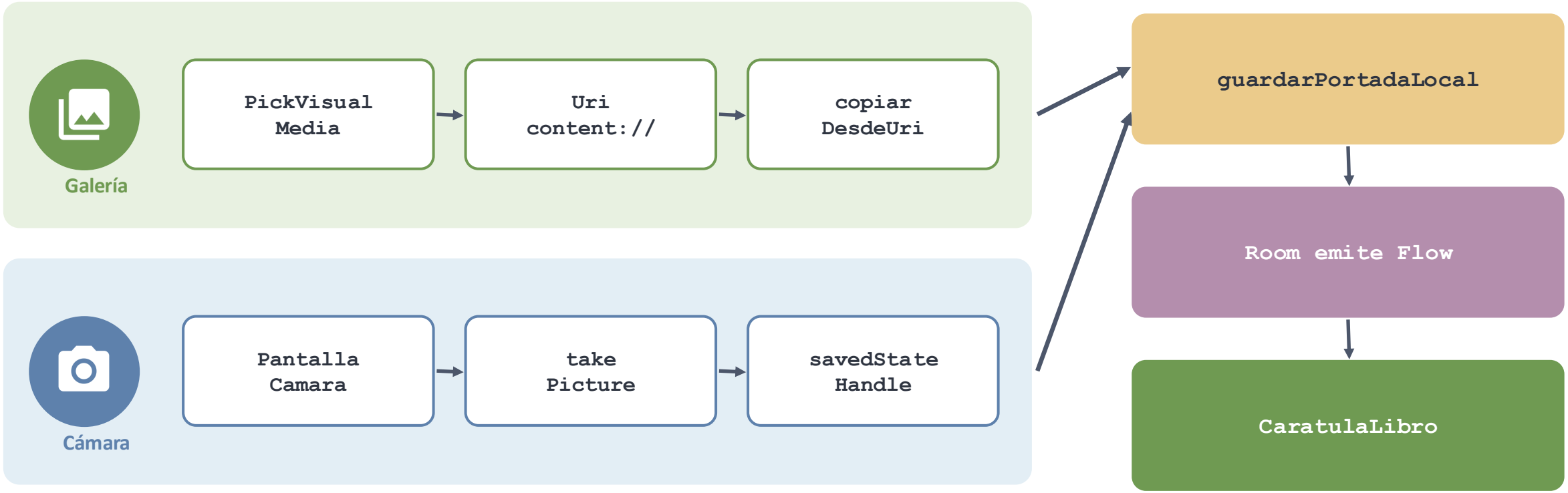

```
PantallaDetalle (simplificado)

val h = entrada.savedStateHandle
val ruta by h
  .getStateFlow<String?>(
     CLAVE_PORTADA, null)
  .collectAsStateWithLifecycle()

LaunchedEffect(ruta) {
  ruta?.let {
    viewModel
     .asignarPortadaCapturada(it)
    h[CLAVE_PORTADA] = null
  }
}
```

Recorrido completo: dos caminos, un destino

Integración



Comprobaciones manuales

✓ Foto nueva	✓ Galería	✓ Reemplazar	✓ Quitar portada
✓ Girar pantalla	✓ Reiniciar app	✓ Denegar permiso	✓ Device Explorer

03

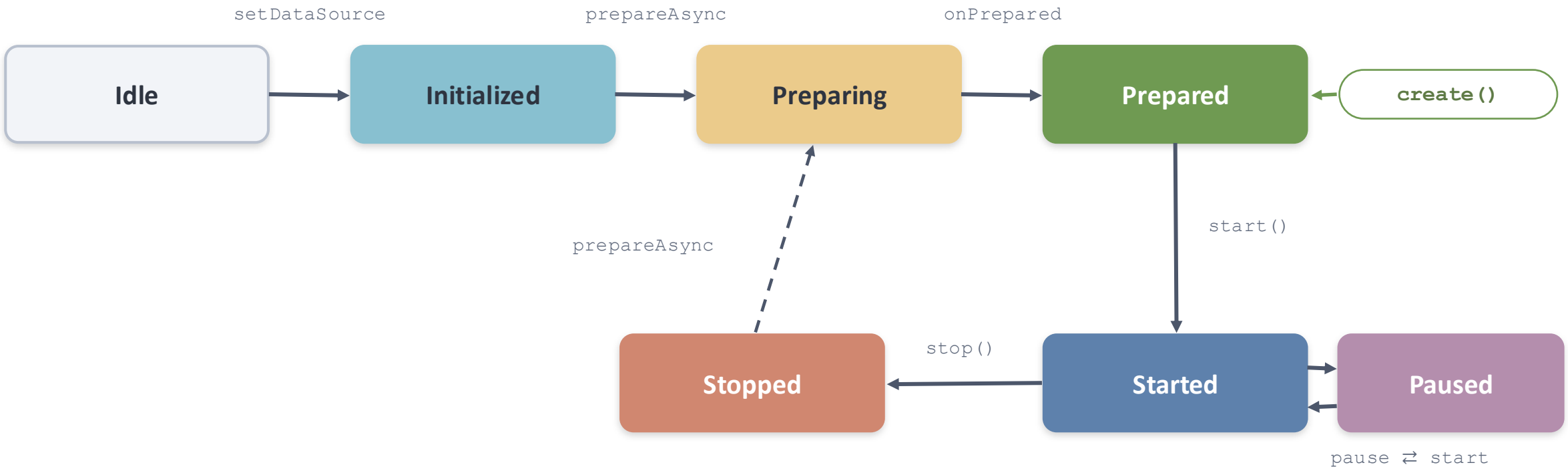
Audio con MediaPlayer

Una máquina de estados con la que no se juega



MediaPlayer es una máquina de estados

MediaPlayer



End ← `release()` desde cualquier estado

Método en el estado incorrecto
→ `IllegalStateException`

AudioViewModel: local o remoto

MediaPlayer

res/raw

MediaPlayer.create()

Devuelve el reproductor ya preparado.

URL remota

prepareAsync()

prepare() bloquearía el hilo principal mientras descarga.

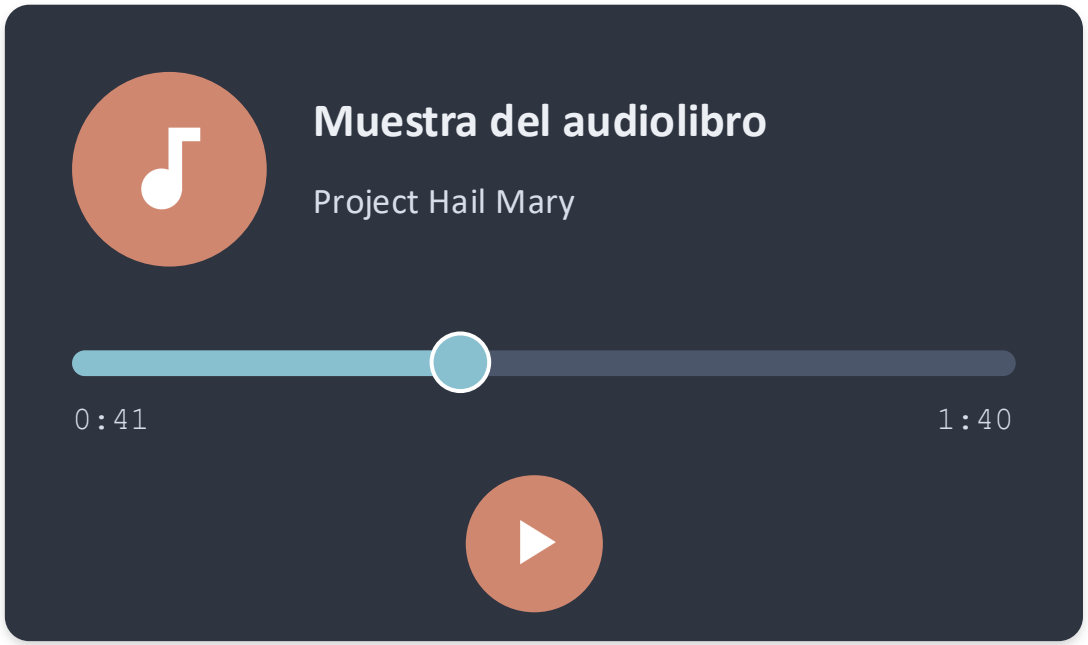
ui/audio/AudioViewModel.kt (simplificado)

```
class AudioViewModel(app: Application) : AndroidViewModel(app) {
    private var player: MediaPlayer? = null

    fun cargarRemoto(url: String) {
        player = MediaPlayer().apply {
            setAudioAttributes(AudioAttributes.Builder()
                .setUsage(AudioAttributes.USAGE_MEDIA).build())
            setDataSource(url)
            setOnPreparedListener { _listo.value = true }
            prepareAsync()
        }
    }
    private fun seguirProgreso() = viewModelScope.launch {
        while (isActive && player?.isPlaying == true) {
            _posicion.value = player!!.currentPosition
            delay(250)
        }
    }
    override fun onCleared() { player?.release(); player = null }
}
```

ReproductorAudio: la UI del slider

Compose




```
fun formatearTiempo(ms: Int): String {
    val s = ms / 1000
    return "%d:%02d".format(s / 60, s % 60)
}
```

1 **Slider**

value = posición actual / duración

2 **Arrastre**

Estado local: no se mueve con el progreso

3 **Al soltar**

onValueChangeFinished → seekTo()

4 **formatearTiempo**

Función pura 41000 ms → "0:41"

04

Vídeo con Media3

ExoPlayer y PlayerView dentro de Compose



MediaPlayer frente a Media3 (ExoPlayer)

Media3

	MediaPlayer	Media3 · ExoPlayer
Origen	Framework Android	Biblioteca Jetpack
Streaming adaptativo	Limitado	HLS · DASH
DRM	No	Sí
Personalización	Baja	Alta
Actualizaciones	Con la versión de Android	Independientes
Uso en T7	Audio local sencillo	Vídeo en streaming



ExoPlayer 2 está obsoleto

`exoplayer2`
↓
`androidx.media3`

ExoPlayer en el ViewModel, PlayerView en la UI

Media3

VideoViewModel
posee el ExoPlayer

player →

ReproductorVideo
PlayerView (AndroidView)

onCleared()

```
player.release()
```

onRelease

```
view.player = null
```

Liberar no es desvincular: la View suelta el player, el ViewModel lo destruye.

¿Cuándo pausar?

RESUMED
Reproduce

→

ON_PAUSE
Multiventana: sigue visible

→

ON_STOP
Ya no visible → pause()

```
VideoViewModel · ReproductorVideo
val player = ExoPlayer.Builder(app)
  .build().apply {
    setMediaItem(
      MediaItem.fromUri(URL_VIDEO))
    prepare()
  }

// En el Composable
DisposableEffect(owner) {
  val obs = LifecycleEventObserver
  { _, e -> if (e ==
      Lifecycle.Event.ON_STOP)
      player.pause() }
  owner.lifecycle.addObserver(obs)
  onDispose { owner.lifecycle
      .removeObserver(obs) }
}
```

Ampliación: reproductor 100 % Compose

media3-ui-compose

media3-ui-compose

Estados observables del reproductor para construir controles propios.

media3-ui-compose-material3

Componentes listos: Player y MiniController con Material 3.

Elimina AndroidView: el reproductor se compone como cualquier otro elemento.

Media3 1.11 (ejemplo oficial)

```
// media3-ui-compose-material3
Player(
    player = player,
    topControls = {
        PlayerDefaults.TopControls(player) },
    centerControls = {
        PlayerDefaults.CenterControls(player) },
    bottomControls = {
        PlayerDefaults.BottomControls(player) },
)
```

05

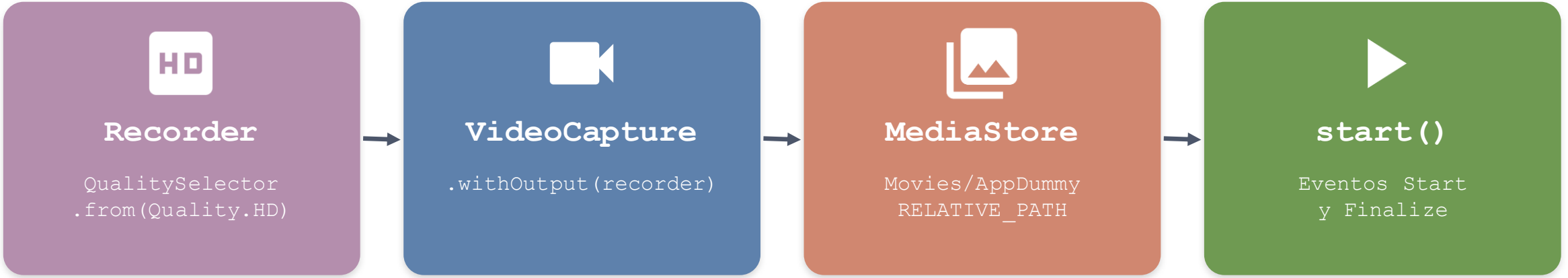
Grabar vídeo

VideoCapture y MediaStore · ampliación



VideoCapture: grabar en la galería

Ampliación




```
val salida = MediaStoreOutputOptions.Builder(contentResolver,
        MediaStore.Video.Media.EXTERNAL_CONTENT_URI)
    .setContentValues(valores)    // DISPLAY_NAME + RELATIVE_PATH
    .build()
grabacion = videoCapture.output
    .prepareRecording(context, salida)
    .withAudioEnabled()           // ¡requiere RECORD_AUDIO!
    .start(ContextCompat.getMainExecutor(context)) { evento -> … }
```

Sin RECORD_AUDIO concedido, withAudioEnabled() falla

Aquí sí MediaStore: el vídeo debe verse en la galería (API 29+)

Ideas clave

Permisos en ejecución

CAMERA y RECORD_AUDIO se piden; si se deniegan, a Ajustes.

Fichero propio + ruta

filesDir para la imagen; Room solo guarda la ruta.

CameraX por casos de uso

unbindAll() y bindToLifecycle(): la cámara sigue al ciclo de vida.

Uri content:// temporal

PickVisualMedia sin permisos, pero se copia la imagen.

MediaPlayer = estados

prepareAsync para red; release() en onCleared.

Media3 para vídeo

ExoPlayer en el ViewModel; pausar en ON_STOP.

Referencias

Fuentes

Solicitar permisos de ejecución
developer.android.com/training/permissions/requesting

CameraX: descripción general
developer.android.com/media/camera/camerax

Selector de fotos
developer.android.com/training/data-storage/shared/photopicker

MediaPlayer: descripción general
developer.android.com/media/platform/mediaplayer

Jetpack Media3
developer.android.com/media/media3

Versiones de CameraX
developer.android.com/jetpack/androidx/releases/camera

Tema 7. Multimedia (apuntes)
documentation.javiercarrasco.es/pmdm/B4/B4-T7_Multimedia