Bloque 1 · Tema 1D

Intents y permisos

Ciclo de vida, comunicación entre componentes, permisos en tiempo de ejecución y entorno de desarrollo

2 horas · AppDummy · permisos · RA1 · RA2



Hoja de ruta del tema

6 apartados

1 **Ciclo de vida**
Callbacks de una Activity

2 **Android Manifest**
Componentes, permisos y filtros

3 **Intents**
Explícitos, implícitos y resultados

4 **Permisos**
Normales, peligrosos y en tiempo de ejecución

5 **Entorno**
Android Studio, Logcat, emuladores

6 **Proyecto**
Estructura y Version Catalog

¿Qué vamos a evaluar?

RA1 · RA2

RA1

Aplica tecnologías de desarrollo para dispositivos móviles evaluando sus características y capacidades.

RA2

Desarrolla aplicaciones para dispositivos móviles analizando y empleando las tecnologías y librerías específicas.

RA1-a	Limitaciones de ejecución en el móvil
RA1-c	Instalar, configurar y usar entornos
RA1-d	Configuraciones de los dispositivos
RA1-e	Perfiles dispositivo-aplicación
RA1-f	Estructura de aplicaciones existentes
RA1-g	Modificar aplicaciones existentes
RA1-h	Uso de emuladores
RA2-j	Establecer los permisos requeridos

★ *RA2-j: el criterio que vertebra el apartado de permisos*

01

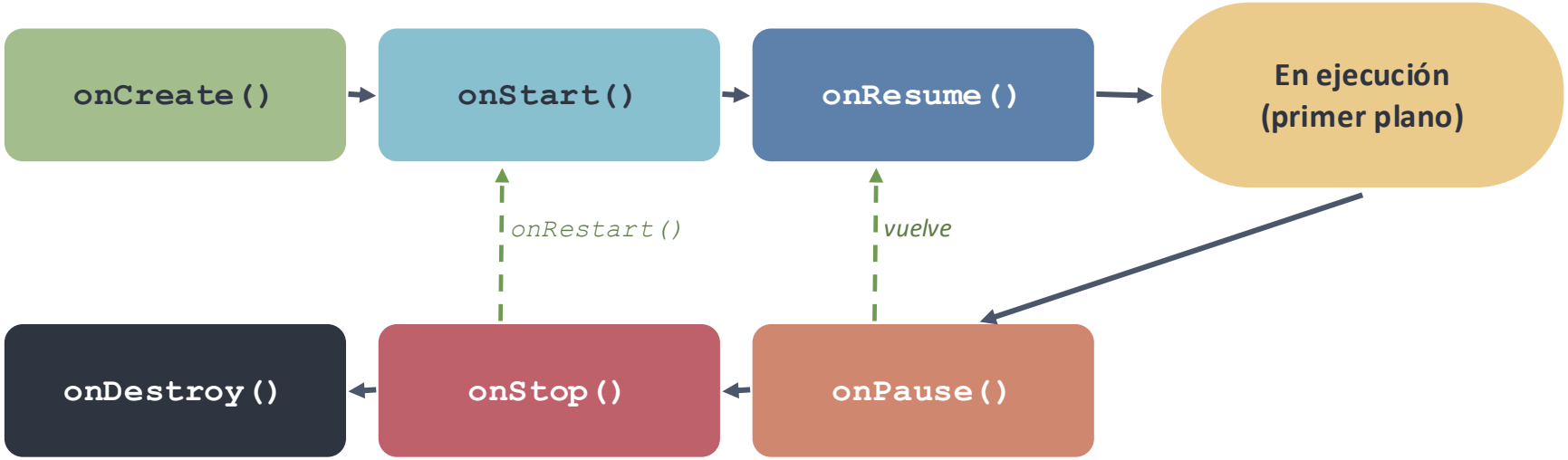
Activity y Manifest

Ciclo de vida y archivo de configuración de la app



Ciclo de vida de una Activity

`Activity`



Escenarios típicos

Pulsar Inicio	`onPause → onStop`
Rotar la pantalla	`onDestroy → onCreate (se recrea)`
Volver a la app	`onRestart → onStart → onResume`

¿Y con Compose?

La MainActivity raramente sobrescribe estos callbacks: Compose y el ViewModel gestionan el ciclo de vida. Conocerlos explica qué pasa al ir a segundo plano o rotar.

¿Qué hacer en cada callback?

Activity



onCreate

Inicialización única: setContent, datos iniciales



onStart / onStop

Iniciar o detener animaciones y sensores de bajo consumo



onResume / onPause

Cámara, audio en primer plano, UI en tiempo real



onDestroy

Liberar recursos que no gestionan su ciclo de vida

MainActivity.kt

```
class MainActivity : ComponentActivity() {
    override fun onCreate(
        savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContent {
            AppDummyTheme { AppDummyApp() }
        }
    }
    // Con Compose + ViewModel, raramente
    // se sobrescriben onPause/onStop
}
```

Bloque 4: los recursos multimedia (cámara, audio) se enlazan al ciclo de vida para liberarlos al pausar.

AndroidManifest.xml: la ficha de la app

Manifest

app/src/main/AndroidManifest.xml

```
<manifest xmlns:android="…">
  <uses-permission
      android:name="android.permission.INTERNET" />
  <application
      android:name=".AppDummyApplication"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/Theme.AppDummy">
    <activity
        android:name=".MainActivity"
        android:exported="true">
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name=
            "android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

uses-permission

Permisos que la app necesita

application

Icono, nombre, tema y clase Application

activity · exported

Componente visible para otras apps

MAIN + LAUNCHER

La convierte en pantalla de inicio

Compose + Navigation: una sola Activity

02

Intents

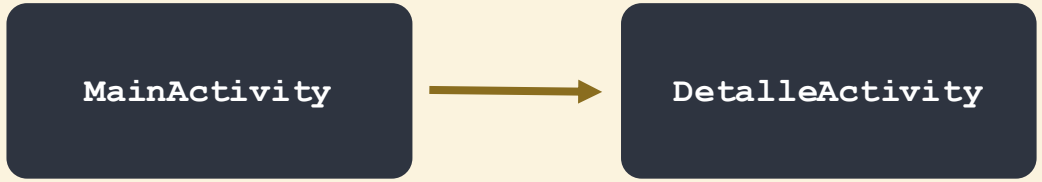
Mensajes para comunicar componentes y apps



Dos tipos de Intent

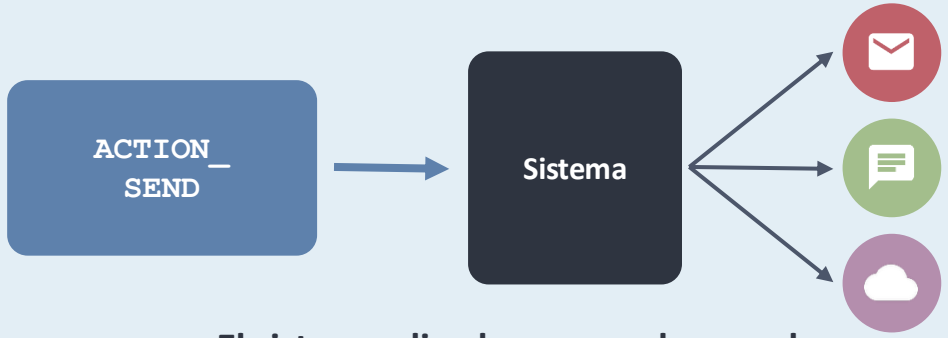
`Intent`

Explícito Nombra el componente exacto



Navegar dentro de la misma app

Implícito Declara una acción



El sistema elige la app que la resuelve

Tipo	Uso habitual
Explícito	Integrar componentes concretos (en Compose, la navegación interna será Navigation Compose)
Implícito	Abrir una URL, compartir contenido, enviar un correo, marcar un teléfono

Intent explícito: enviar y recibir datos

Explícito

Activity de origen

```
val intent = Intent(this,
    DetalleActivity::class.java).apply {
    putExtra("PELICULA_ID", 42)
    putExtra("PELICULA_TITULO", "Dune: Parte 2")
}
startActivity(intent)
```

DetalleActivity.onCreate()

```
val id = intent.getIntExtra("PELICULA_ID", -1)
val titulo = intent
    .getStringExtra("PELICULA_TITULO") ?: ""
setContent {
    PantallaDetalle(id = id, titulo = titulo)
}
```



Bloque 2: NavController.navigate(...) sustituye este patrón entre pantallas de la misma app

Intents implícitos: delegar en otras apps

Implícito

Compartir texto

ACTION_SEND

```
type = "text/plain"
EXTRA_TEXT + createChooser()
```

Abrir una URL

ACTION_VIEW

```
Uri.parse(url)
```

Enviar un correo

ACTION_SENDTO

```
data = Uri.parse("mailto:")
EXTRA_EMAIL, EXTRA_SUBJECT
```

Marcar un teléfono

ACTION_DIAL

```
Uri.parse("tel:+34600000000")
```

En un Composable

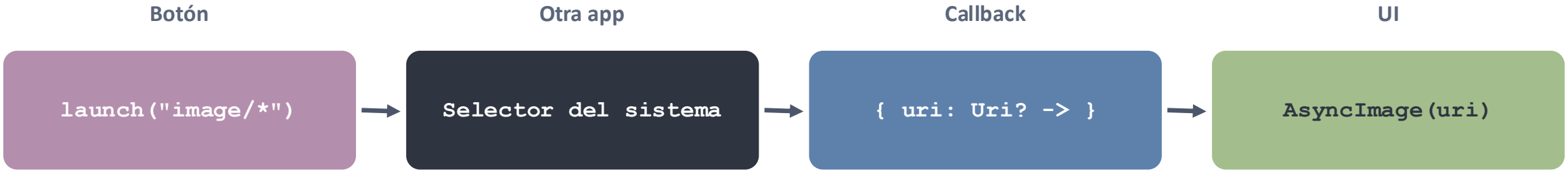
```
val context = LocalContext.current
Button(onClick = {
  val intent = Intent(
    Intent.ACTION_VIEW,
    url.toUri())
  context.startActivity(intent)
}) { Text("Ver en web") }
```

Android 11+ y resolveActivity

Sin <queries> en el Manifest devuelve null aunque haya apps. Alternativa: try/catch de ActivityNotFoundException.

Recibir un resultado de otra app

Resultado



uri es null si el usuario cancela

SeleccionarImagen.kt

```
var imagenUri by remember {
    mutableStateOf<Uri?>(null) }
val seleccionadorImagen =
    rememberLauncherForActivityResult(
        ActivityResultContracts.GetContent()
    ) { uri: Uri? -> imagenUri = uri }

Button(onClick = {
    seleccionadorImagen.launch("image/*") }) { … }
```

Contrato

Define qué se lanza y qué tipo se devuelve: GetContent, RequestPermission, TakePicture…

Ampliación: PickVisualMedia

Selector de fotos del sistema: no requiere permisos de almacenamiento.

03

Permisos

Proteger los recursos sensibles del usuario



Tipos de permisos

Permisos



Normales

Bajo riesgo: se conceden al instalar.

`INTERNET`

`VIBRATE`

Solo Manifest



Peligrosos

Datos sensibles o hardware.

`CAMERA`

`READ_CONTACTS`

`ACCESS_FINE_LOCATION`

Manifest + diálogo en ejecución



De firma

Solo para apps firmadas con el mismo certificado.

`protectionLevel="signature"`

Sin diálogo

Desde Android 6.0 (API 23) los permisos peligrosos se solicitan en tiempo de ejecución. Todos se declaran en el Manifest.

Declarar permisos en el Manifest

Manifest

AndroidManifest.xml

```
<!-- Normal -->
<uses-permission android:name=
    "android.permission.INTERNET" />
<!-- Peligrosos -->
<uses-permission android:name=
    "android.permission.CAMERA" />
<uses-permission android:name=
    "android.permission.READ_MEDIA_IMAGES" />
<uses-permission android:name=
    "android.permission.READ_EXTERNAL_STORAGE"
    android:maxSdkVersion="32" />
<uses-feature android:name="android.hardware.camera"
    android:required="false" />
```

Acceso a imágenes según la versión



READ_EXTERNAL_STORAGE

Con maxSdkVersion = 32

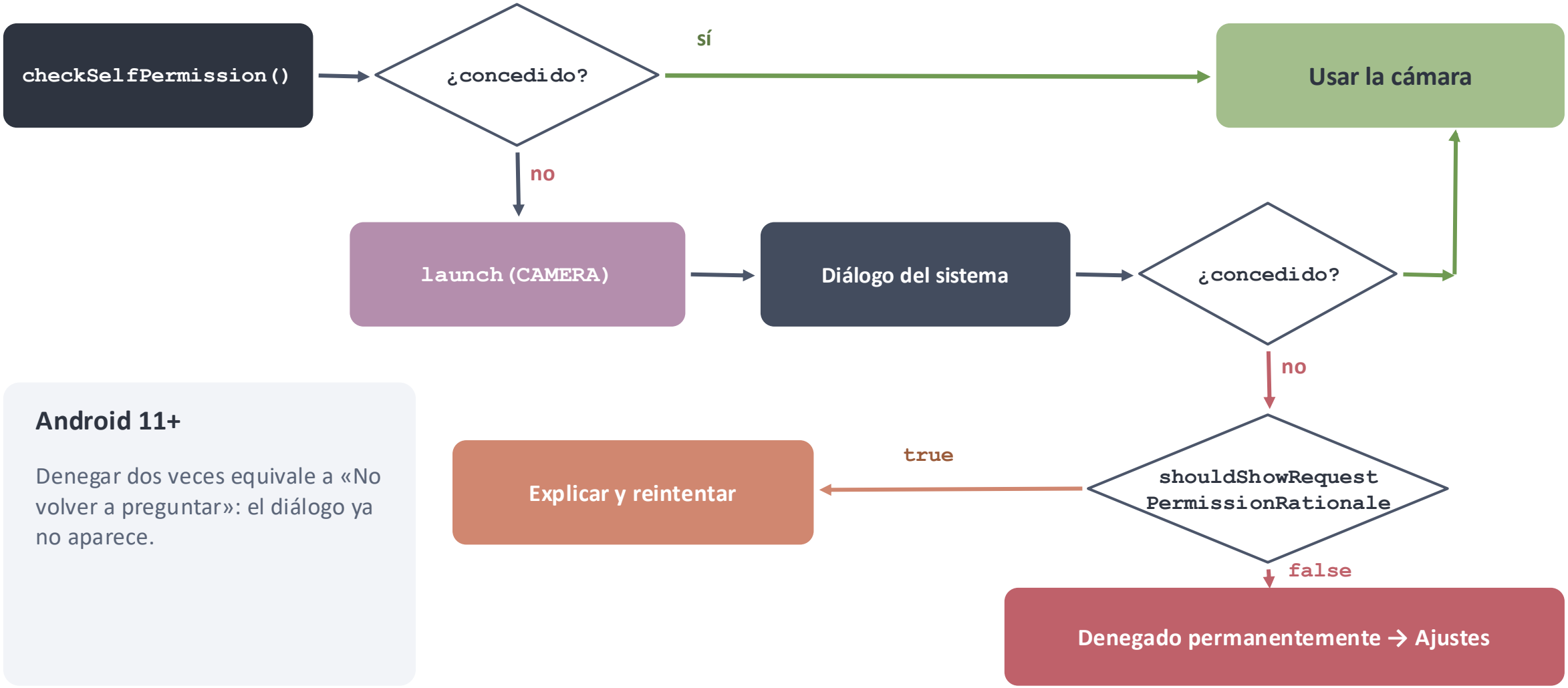
READ_MEDIA_IMAGES / _VIDEO

Permisos separados por tipo de medio

uses-feature required="false": la app se puede instalar en dispositivos sin cámara; hay que comprobarlo en código.

Flujo de una solicitud en tiempo de ejecución

Runtime



Android 11+

Denegar dos veces equivale a «No volver a preguntar»: el diálogo ya no aparece.

Solicitar un permiso desde Compose

`RequestPermission`

PantallaConPermisoCamara.kt

```
var tienePermiso by remember { mutableStateOf(
    ContextCompat.checkSelfPermission(context,
        Manifest.permission.CAMERA)
        == PackageManager.PERMISSION_GRANTED) }
var denegadoPermanente by remember {
    mutableStateOf(false) }

val solicitarPermiso =
  rememberLauncherForActivityResult(
    ActivityResultContracts.RequestPermission()
  ) { concedido ->
    tienePermiso = concedido
    if (!concedido && activity != null)
      denegadoPermanente = !ActivityCompat
        .shouldShowRequestPermissionRationale(
          activity, Manifest.permission.CAMERA)
  }
```

when en la UI



`tienePermiso`

Icono y vista de cámara (Bloque 4)



`denegadoPermanente`

Botón «Abrir Ajustes»



`else`

Explicación + «Conceder permiso»

Extensión auxiliar

```
fun Context.findActivity(): Activity? =
  when (this) { is Activity -> this
    is ContextWrapper ->
      baseContext.findActivity()
    else -> null }
```

Solicitar varios permisos a la vez

Múltiples

Build.VERSION.SDK_INT ≥ TIRAMISU (33)

READ_MEDIA_IMAGES

READ_MEDIA_VIDEO

si no: READ_EXTERNAL_STORAGE

↓

RequestMultiplePermissions()

↓

Map<String, Boolean>

✓ IMAGES → true

✗ VIDEO → false

SolicitarPermisosMultimedia.kt

```
val permisos = if (Build.VERSION.SDK_INT >=
        Build.VERSION_CODES.TIRAMISU)
    arrayOf(Manifest.permission.READ_MEDIA_IMAGES,
            Manifest.permission.READ_MEDIA_VIDEO)
else arrayOf(
    Manifest.permission.READ_EXTERNAL_STORAGE)

val solicitarPermisos =
  rememberLauncherForActivityResult(
    ActivityResultContracts
      .RequestMultiplePermissions()
  ) { resultados -> permisosEstado = resultados }

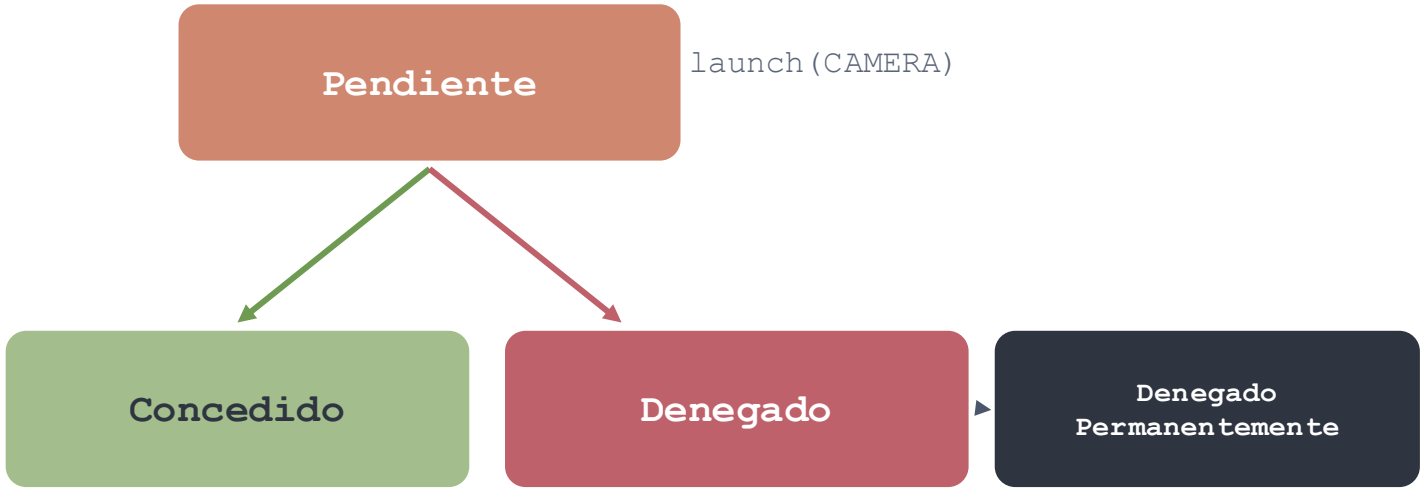
val todosConcedidos =
    permisosEstado.all { it.value }
```

Desarrollo guiado: estados del permiso

AppDummy v3

```
EstadoPermiso (simplificado)

sealed class EstadoPermiso {
  data object Concedido
  data object Pendiente
  data object Denegado
  data object
    DenegadoPermanentemente
}
```



Concedido

«Puedes usar la cámara»

Pendiente

Botón «Conceder permiso»

Denegado / Permanente

Botón «Abrir Ajustes»

```
Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS) con data = Uri.fromParts("package", packageName, null)
```

04

Entorno de desarrollo

Android Studio, Logcat, emuladores y estructura del proyecto



Android Studio: tres herramientas clave

Android Studio

SDK Manager

Tools → SDK Manager

- Android 14 (API 34)
- Android 15 (API 35)
- Android Emulator
- SDK Platform-Tools

Device Manager

Tools → Device Manager

- Crea emuladores (AVD)
- Recomendado: Pixel 6
- API 36 · x86_64

Logcat

View → Tool Windows → Logcat

- Logs en tiempo real
- Filtrar por tag, nivel o texto
- Imprescindible para depurar

Logging con Logcat

Log

V	`Log.v`	Verbose	Detalle exhaustivo
D	`Log.d`	Debug	Información de depuración
I	`Log.i`	Info	Evento importante
W	`Log.w`	Warning	Situación anómala
E	`Log.e`	Error	Fallo grave + excepción

severidad ↓

PeliculasRepository.kt

```kt
class PeliculasRepository {
  companion object {
    private const val TAG = "PeliculasRepo"
  }
  suspend fun cargarPeliculas(): List<String> {
    Log.d(TAG, "Iniciando carga")
    return try {
      val r = listOf("Dune", "Inception")
      Log.i(TAG, "Cargadas ${r.size}")
      r
    } catch (e: Exception) {
      Log.e(TAG, "Error al cargar", e)
      emptyList()
    }
  }
}
```

Emulador frente a dispositivo real

RA1-h

Emulador (AVD)

- Sin dispositivo físico
- Fácil de configurar
- Simula red, batería o GPS
- Más lento
- Sin NFC ni Bluetooth real

Dispositivo real

1. **Opciones de desarrollador**
   Ajustes → Acerca del teléfono → 7 toques en «Número de compilación»
2. **Depuración USB**
   Opciones de desarrollador → Depuración USB
3. **ADB inalámbrico**
   Android 11+: Depuración inalámbrica

Estructura de un proyecto Android

Proyecto

```
MiApp/
├─ app/src/main/
│   ├─ java/…/MainActivity.kt
│   ├─ ui/theme · ui/components          pantallas y tema
│   ├─ data/                             Bloques 2-3
│   ├─ res/drawable · mipmap · values    recursos
│   ├─ AndroidManifest.xml
├─ app/src/test · androidTest            JUnit · UI Test
├─ app/build.gradle.kts                  módulo app
gradle/libs.versions.toml                versiones
build.gradle.kts · settings.gradle.kts   proyecto
```

- Punto de entrada
- Código de la app
- Recursos y configuración
- Pruebas
- Gradle y dependencias

Vista «Android» de Android Studio

Agrupa manifests, kotlin+java, res y Gradle Scripts: no es la estructura real de carpetas en disco.

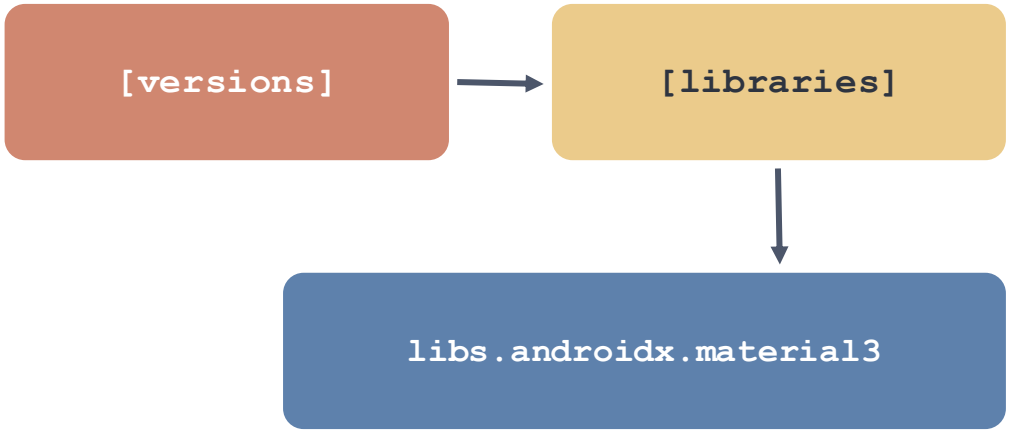
Version Catalog: dependencias centralizadas

`libs.versions.toml`

```toml
# gradle/libs.versions.toml
[versions]
agp = "9.2.1"
kotlin = "2.4.0"
composeBom = "2026.06.01"
activityCompose = "1.13.0"

[libraries]
androidx-compose-bom = { group = "androidx.compose",
    name = "compose-bom", version.ref = "composeBom" }
androidx-material3 = { group =
    "androidx.compose.material3", name = "material3" }

[plugins]
kotlin-compose = { id =
    "org.jetbrains.kotlin.plugin.compose",
    version.ref = "kotlin" }
```



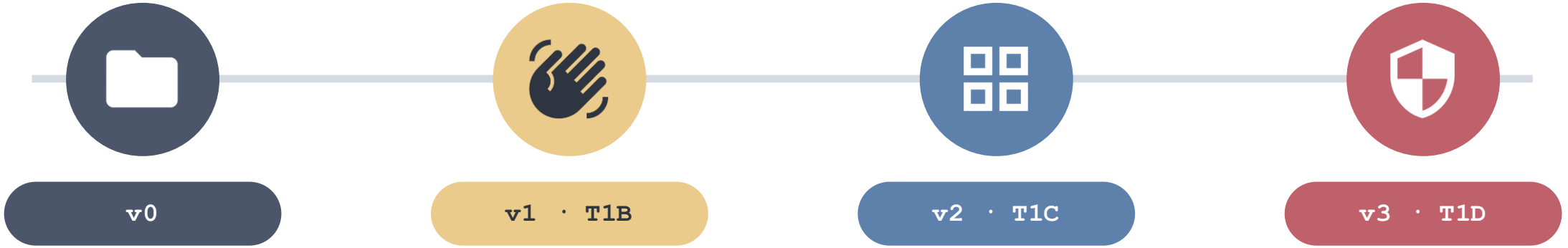
guiones → puntos

```gradle
build.gradle.kts (app)

dependencies {
    implementation(
        platform(libs.androidx.compose.bom))
    implementation(libs.androidx.material3)
    implementation(
        libs.androidx.activity.compose)
}
```

AppDummy al terminar el Bloque 1

DPG 1



v0

Proyecto inicial

Empty Activity + Git

v1 · T1B

Bienvenida

Campo de nombre y botón habilitado condicionalmente

v2 · T1C

Listado

Búsqueda, filtro por autor y toggles de favorito y leído

v3 · T1D

Permisos

Gestión de al menos un permiso

Bloque 2: el estado pasa al ViewModel (MVVM) y la navegación a Navigation Compose.

Ideas clave

Ciclo de vida

Rotar recrea la Activity; Inicio la detiene.

Manifest

Componentes, permisos y punto de entrada.

Intents

Explícito nombra; implícito declara una acción.

Permisos peligrosos

Manifest + solicitud en tiempo de ejecución.

Logcat

TAG y nivel para depurar con criterio.

Version Catalog

Versiones en un solo lugar: libs.*

Referencias

Fuentes



Intents y filtros de intents
developer.android.com/guide/components/intents-filters



Permisos en Android
developer.android.com/guide/topics/permissions/overview



Ciclo de vida de una Activity
developer.android.com/guide/components/activities/activity-lifecycle



ActivityResultContracts
developer.android.com/reference/androidx/activity/result/contract/ActivityResultContracts



Visibilidad de paquetes (Android 11+)
developer.android.com/training/package-visibility



Android Manifest — referencia
developer.android.com/guide/topics/manifest/manifest-intro



Tema 1D. Intents, permisos y entorno (apuntes)
documentation.javiercarrasco.es/pmdm/B1/B1-T1d_Intents_Permisos