Bloque 1 · Tema 1C

UI avanzada

Material 3, diseño adaptativo, animaciones y Canvas en Jetpack Compose

2 horas

AppDummy · listado de libros

RA1



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

Hoja de ruta del tema

6 apartados

1 **Ecosistema Android**
Arquitectura, limitaciones y API Level

2 **Modifier avanzado**
weight, clickable, BoxWithConstraints

3 **Material 3**
Tema, colores y tipografía

4 **Componentes M3**
Card, NavigationBar, Chips, Progress

5 **Animaciones**
AnimatedVisibility y animate*AsState

6 **Canvas**
Gráficos personalizados

¿Qué vamos a evaluar?

RA1

RA1

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

RA1-b	Identificar las tecnologías de desarrollo móvil
RA1-d	Identificar configuraciones que clasifican los dispositivos móviles
RA1-g	Realizar modificaciones sobre aplicaciones existentes
RA1-h	Usar emuladores para comprobar el funcionamiento

01

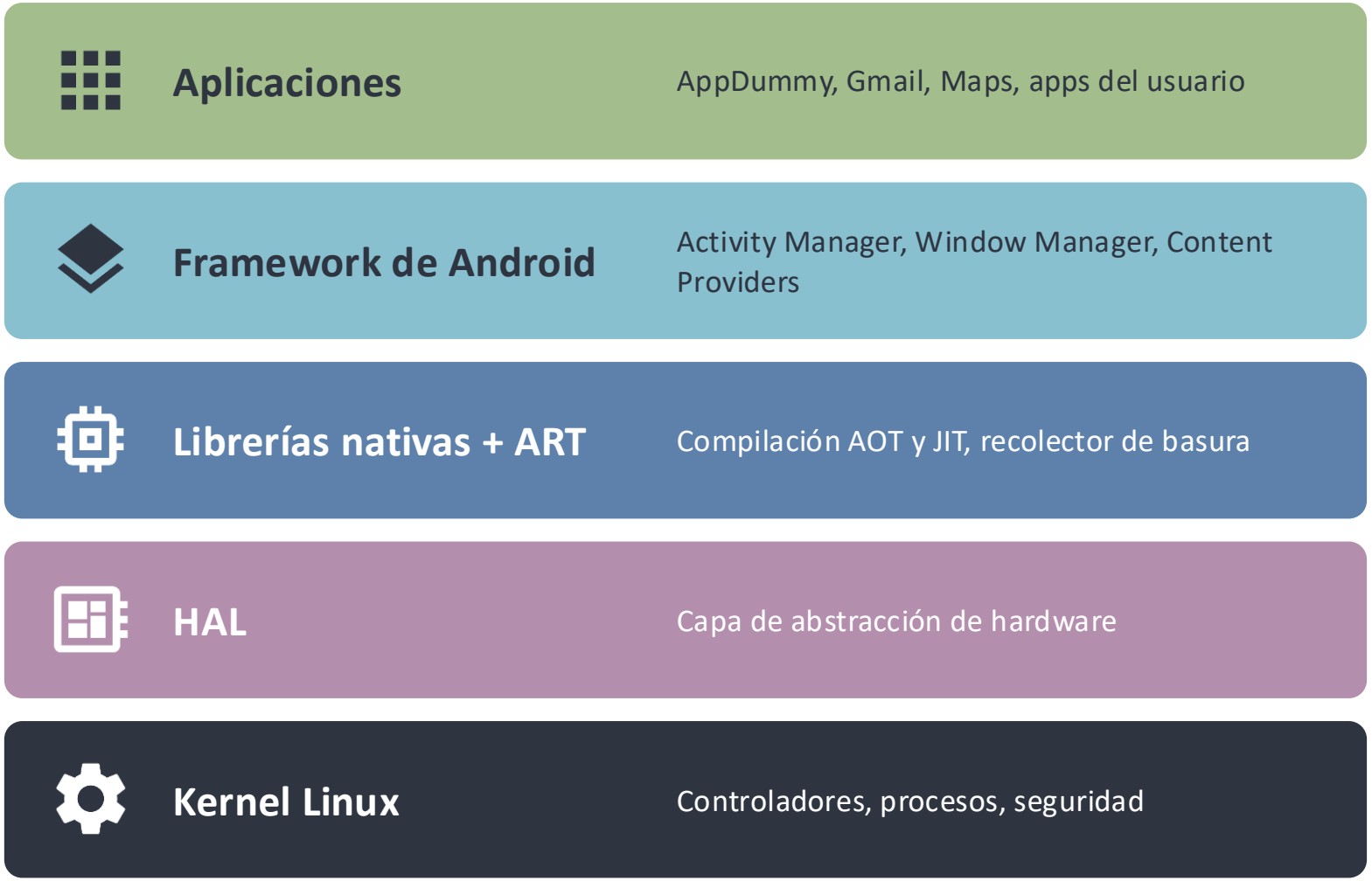
Ecosistema Android

Arquitectura del sistema, limitaciones y compatibilidad



Arquitectura del sistema Android

Android



abstracción

Nuestro código

Vive en la capa superior y usa el framework (API Kotlin/Java). Nunca habla directamente con el hardware.

Limitaciones del dispositivo móvil

RA1-a



Memoria RAM

El sistema puede matar procesos

Liberar recursos al pausar · LazyColumn



CPU

ARM, núcleos limitados

No bloquear el hilo principal · Dispatchers.IO



Batería

Recurso crítico para el usuario

Minimizar trabajo en segundo plano



Almacenamiento

Limitado y compartido

No guardar lo innecesario · comprimir



Conectividad

Puede perderse en cualquier momento

Offline-first (Bloque 3)



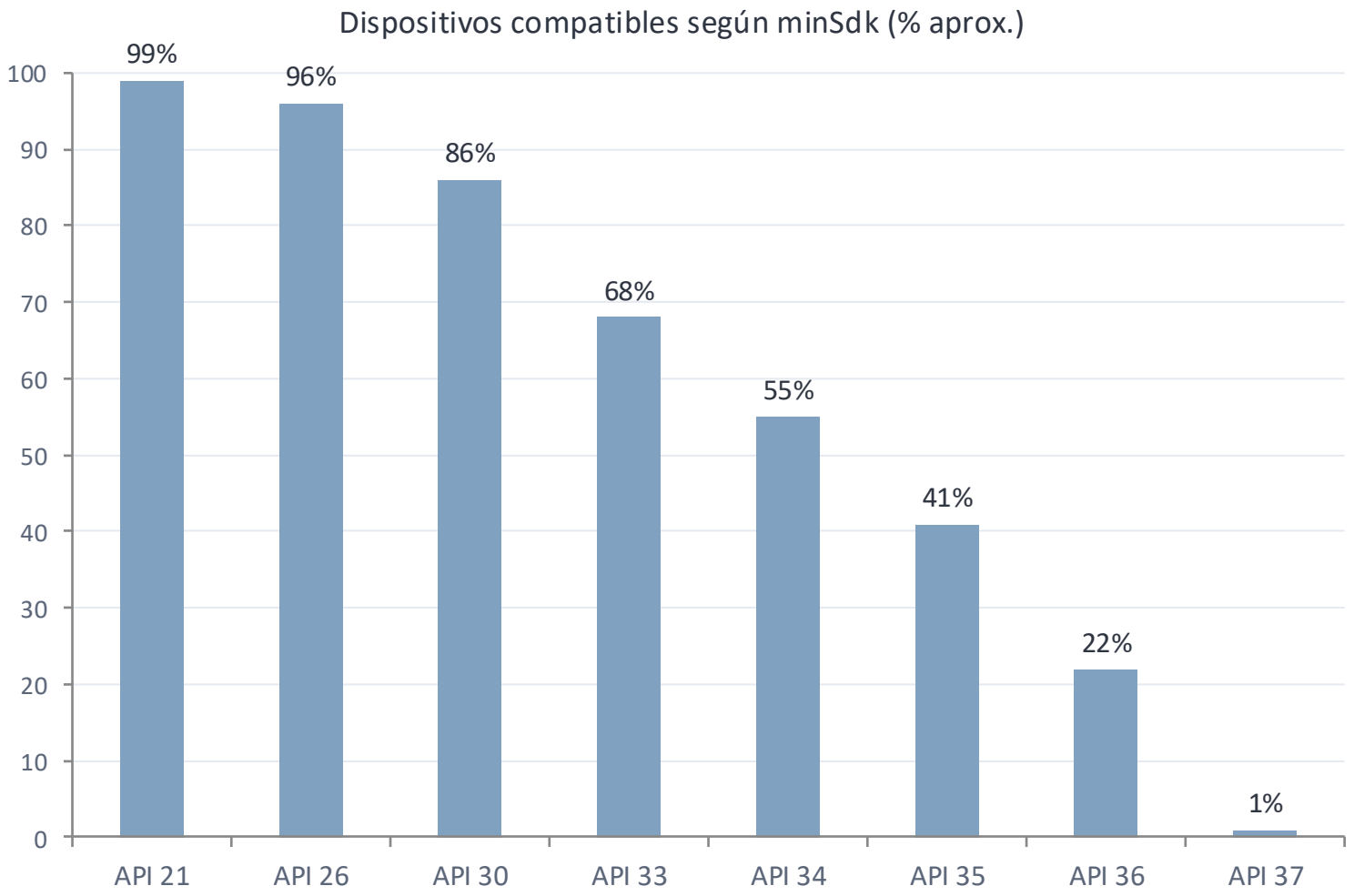
Pantalla

De 4'' a 13'' y distintas densidades

dp/sp en lugar de px · layouts adaptativos

API Level y compatibilidad

`minSdk = 30`



`minSdk = 30`

Android 11 · criterio del curso. Cubre más del 80 % de los dispositivos activos.

Más API Level

Más funciones disponibles

Menos dispositivos

Menor cuota de mercado

Cuotas estimadas: varían según la fuente y el momento (apilevels.com).

02

Modifier avanzado

Distribución proporcional, interacción y diseño adaptativo



weight: reparto proporcional del espacio

weight



weight(2f) · 2/3

weight(1f) · 1/3

DistribucionProporcional.kt

```
Row(Modifier.fillMaxWidth().height(60.dp)) {
    Box(Modifier.weight(2f).fillMaxHeight()
        .background(MaterialTheme.colorScheme.primary))
    { Text("Título", color = Color.White) }
    Box(Modifier.weight(1f).fillMaxHeight()
        .background(MaterialTheme.colorScheme.secondary))
    { Text("★ 8.5", color = Color.White) }
}
```

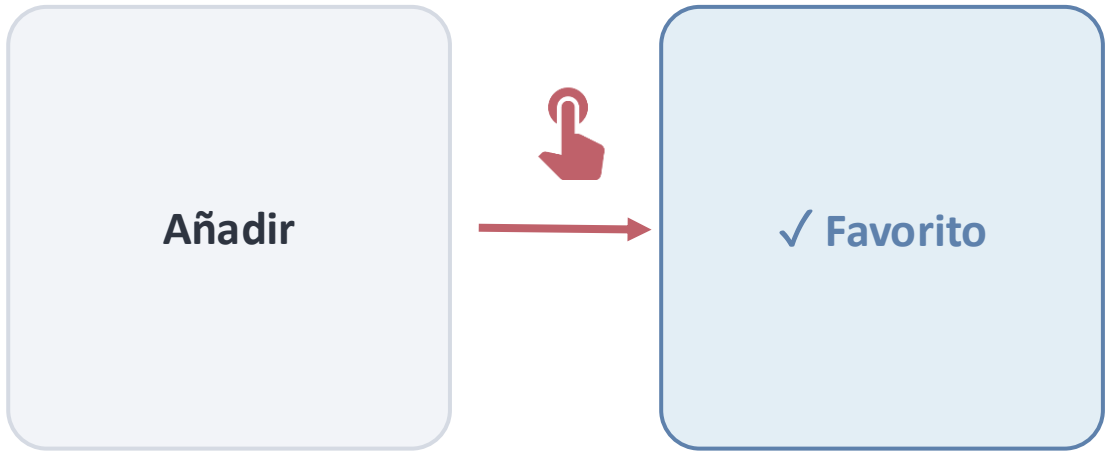
¿Cómo se calcula?

ancho = peso / suma de pesos

2 / (2 + 1) = 2/3 del espacio libre. Solo existe dentro de Row y Column.

Modificadores de interacción

clickable



`seleccionado = !seleccionado`

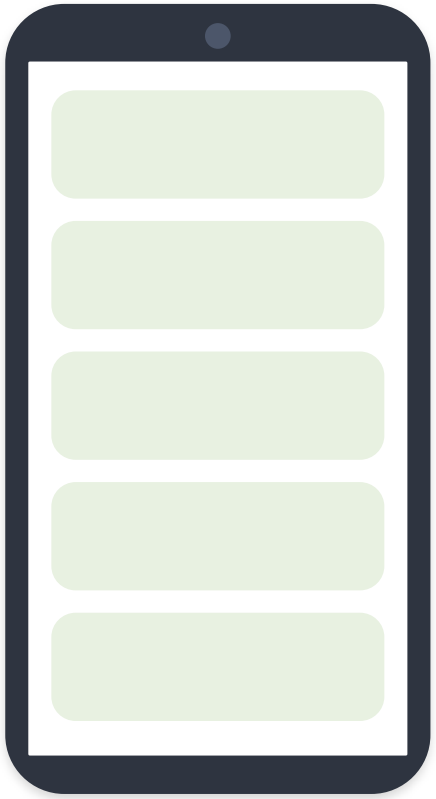
clickable añade el efecto ripple, accesibilidad y foco. El orden con clip y padding decide el área pulsable.

EjemploInteraccion.kt

```
var seleccionado by remember {
    mutableStateOf(false) }
Box(
    modifier = Modifier
        .size(120.dp)
        .clip(RoundedCornerShape(12.dp))
        .background(
            if (seleccionado)
              MaterialTheme.colorScheme.primaryContainer
            else
              MaterialTheme.colorScheme.surfaceVariant)
        .clickable { seleccionado = !seleccionado }
        .padding(16.dp),
    contentAlignment = Alignment.Center
) { Text(if (seleccionado) "✓ Favorito" else "Añadir") }
```

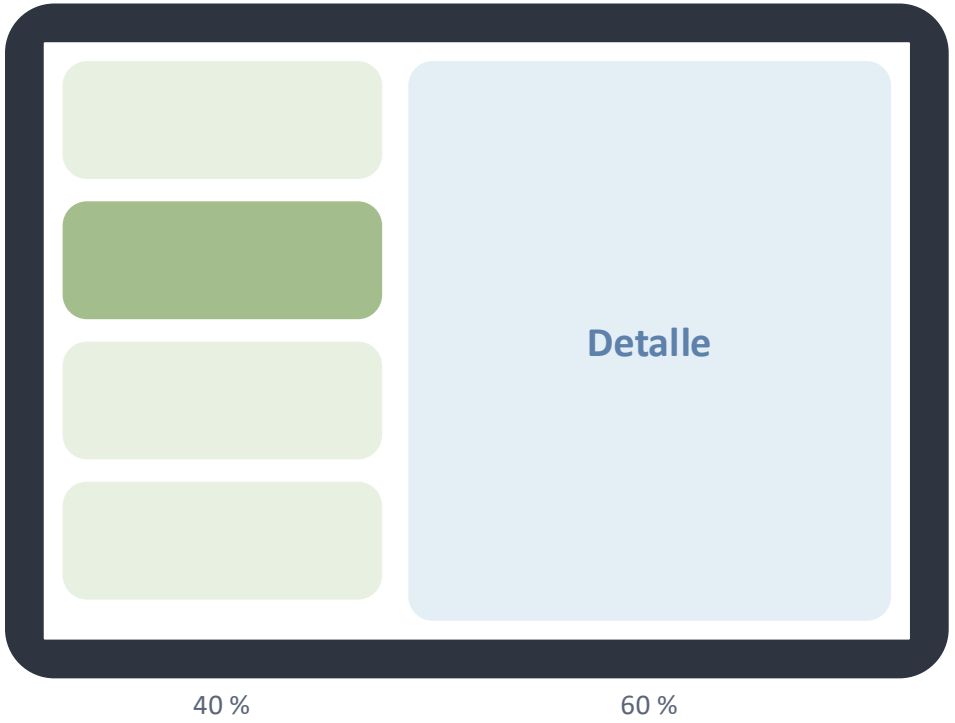
Diseño adaptativo con BoxWithConstraints

Adaptativo



`maxWidth < 600.dp`

Móvil: lista vertical



`maxWidth ≥ 600.dp`

Tablet: maestro-detalle

```
BoxWithConstraints(
  Modifier.fillMaxSize()
) {
  if (maxWidth < 600.dp) {
    Column {
      ContenidoMovil() }
  } else {
    Row {
      Box(Modifier
        .weight(0.4f)) { Lista() }
      Box(Modifier
        .weight(0.6f)) { Detalle()
}
    }
  }
}
```

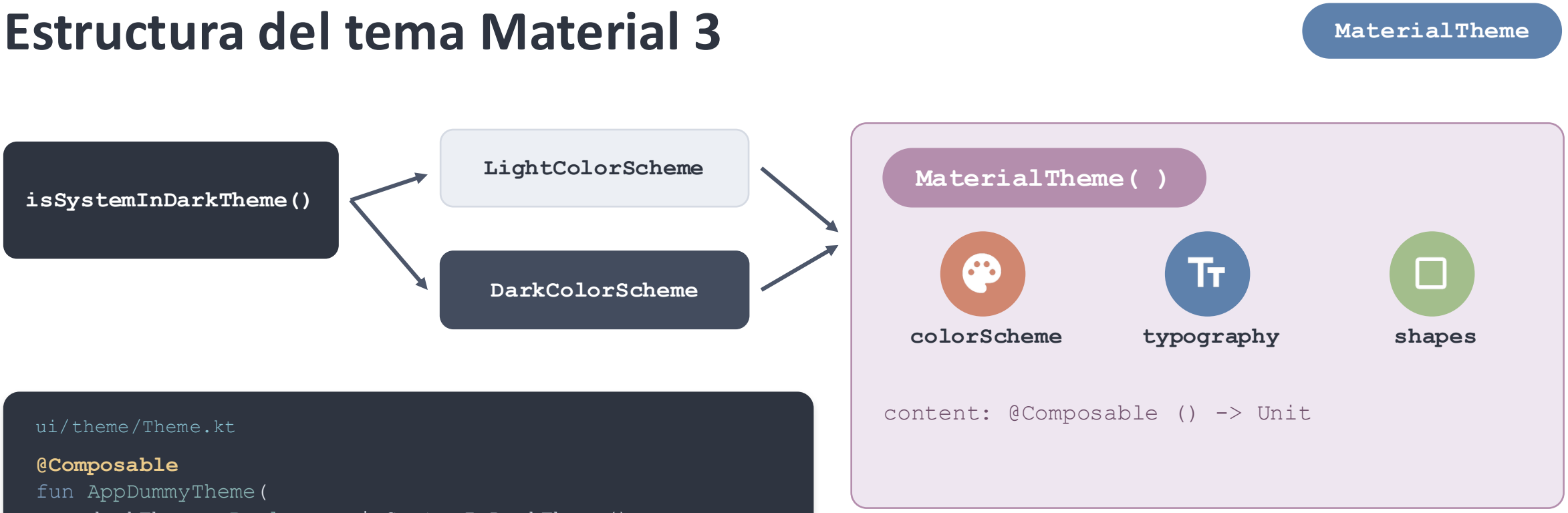
03

Material 3

Tema, roles de color y tipografía



Estructura del tema Material 3



ui/theme/Theme.kt

```
@Composable
fun AppDummyTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    content: @Composable () -> Unit
) {
    val colorScheme =
        if (darkTheme) DarkColorScheme else
LightColorScheme
    MaterialTheme(colorScheme = colorScheme,
        typography = Typography, content = content)
}
```

Generado por Android Studio

Todo lo que se dibuja dentro de AppDummyTheme lee sus colores y estilos con MaterialTheme.colorScheme y MaterialTheme.typography.

Roles de color: semántica, no códigos

`colorScheme`

primary
`texto: onPrimary`
Acción principal

primaryContainer
`texto: onPrimaryContainer`
Contenedores destacados

surface
`texto: onSurface`
Fondo de Card y BottomSheet

error
`texto: onError`
Mensajes de error

`background: fondo general de la pantalla`



Así sí

```
.background(MaterialTheme
    .colorScheme.primaryContainer)
```



Evitar

```
.background(Color(0xFF6200EE))
```

No respeta el tema: el modo oscuro deja de funcionar.

Escala tipográfica de Material 3

`typography`

Display — `typography.displayLarge`

Headline — `typography.headlineLarge`

Title — `typography.titleLarge`

Body: texto de lectura — `typography.bodyLarge`

LABEL · BOTONES — `typography.labelLarge`

Cada grupo tiene variantes Large · Medium · Small

04

Componentes Material 3

Card, NavigationBar, Chips e indicadores de progreso



Card y Chips

Card · Chip

Proyecto Hail Mary ★ 8.5

Ciencia ficción

`Card(onClick) · elevation 4.dp · shape 12.dp · AssistChip`

FilterChip en una LazyRow

✓ Todos | Acción | Drama | Sci-Fi

`selected = genero == generoSeleccionado`

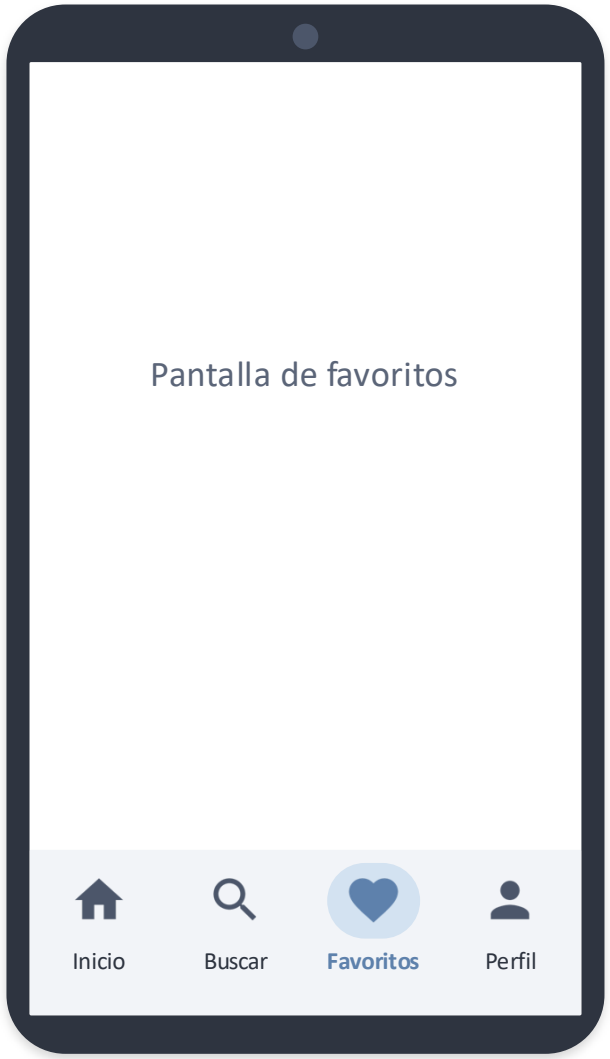
AssistChip: acción contextual · FilterChip: selección con check

FiltrosPorGenero.kt

```
LazyRow(horizontalArrangement =
    Arrangement.spacedBy(8.dp)) {
  items(generos) { genero ->
    FilterChip(
      selected = genero == generoSeleccionado,
      onClick = { generoSeleccionado = genero },
      label = { Text(genero) },
      leadingIcon = if (genero ==
          generoSeleccionado) {
        { Icon(Icons.Default.Check,
            contentDescription = null) }
      } else null
    )
  }
}
```

NavigationBar: secciones principales

NavigationBar



seccionActual

estado con remember

↓

NavigationBarItem

selected + onClick

↓

when (seccionActual)

decide la pantalla

En el Bloque 2: Navigation Compose con NavHost

AppDummyConNavegacion.kt

```
Scaffold(bottomBar = {
  NavigationBar {
    secciones.forEach { seccion ->
      NavigationBarItem(
        selected = seccionActual
            == seccion.ruta,
        onClick = {
          seccionActual = seccion.ruta },
        icon = { Icon(seccion.icono,
          seccion.etiqueta) },
        label = { Text(seccion.etiqueta) })
    }
  }
}) { padding -> … }
```

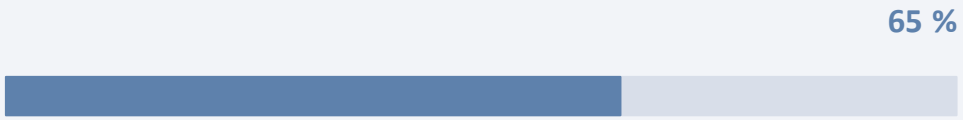
Indicadores de progreso

Progress



CircularProgressIndicator

Espera de duración desconocida



LinearProgressIndicator

Progreso conocido: progress = { 0.65f }

```
if (cargando) CircularProgressIndicator(Modifier.size(48.dp))
else LinearProgressIndicator(progress = { progreso }, modifier = Modifier.fillMaxWidth())
```

05

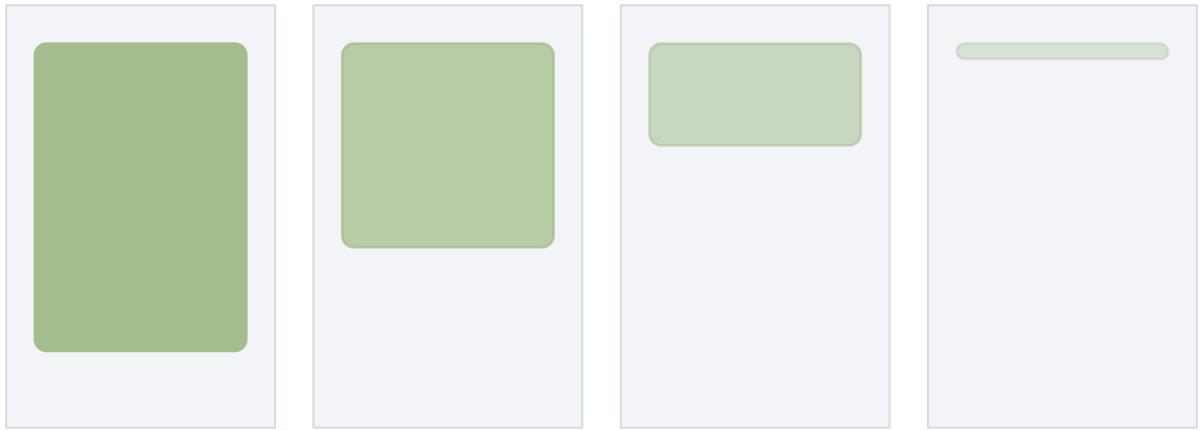
Animaciones y Canvas

Movimiento declarativo y gráficos personalizados



AnimatedVisibility: aparecer y desaparecer

Animación



`exit = fadeOut() + shrinkVertically()`

`enter = fadeIn() + expandVertically()`

Las transiciones se combinan con + y basta con cambiar el valor de visible.

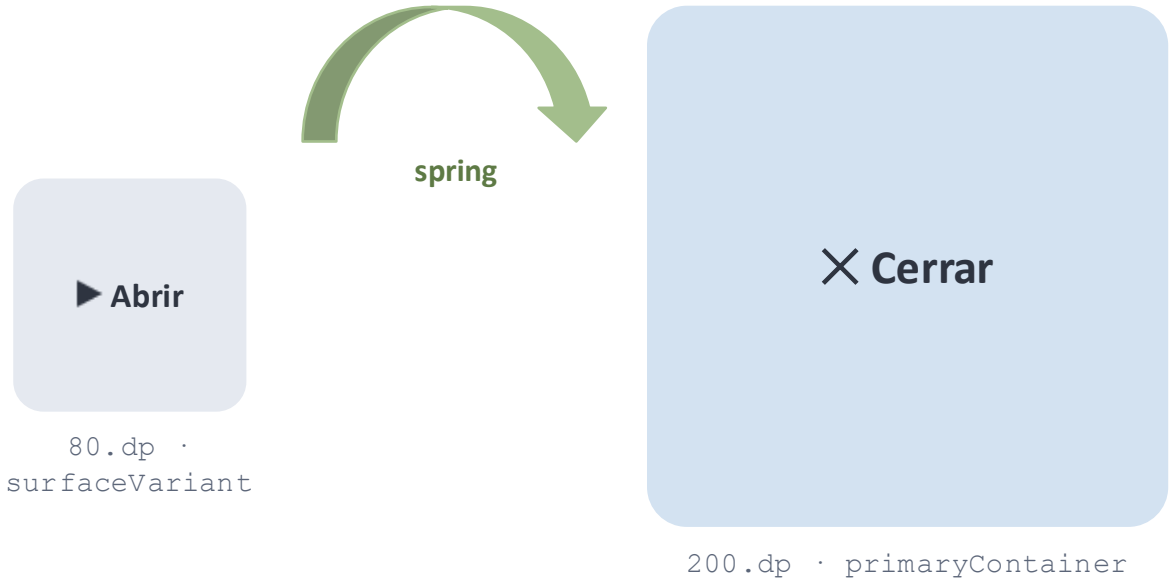
EjemploAnimatedVisibility.kt

```
var visible by remember {
    mutableStateOf(true) }

Button(onClick = { visible = !visible }) {
    Text(if (visible) "Ocultar"
         else "Mostrar")
}
AnimatedVisibility(
    visible = visible,
    enter = fadeIn() + expandVertically(),
    exit = fadeOut() + shrinkVertically()
) {
    Card { Text("Contenido animado") }
}
```

animate*AsState: animar un valor

`animateDpAsState`




```kotlin
EjemploAnimacionValor.kt
var expandido by remember {
    mutableStateOf(false) }
val tamanyo by animateDpAsState(
    targetValue =
        if (expandido) 200.dp else 80.dp,
    animationSpec = spring(dampingRatio =
        Spring.DampingRatioMediumBouncy),
    label = "tamaño")
val color by animateColorAsState(…)

Box(Modifier.size(tamanyo)
    .background(color, RoundedCornerShape(12.dp))
    .clickable { expandido = !expandido })
```

`targetValue`	Valor destino según el estado
`animationSpec`	spring, tween… cómo se llega
`label`	Nombre en el inspector de animaciones

Canvas: dibujar un indicador de puntuación

Canvas



≥ 8 · verde ≥ 6 · naranja < 6 · rojo

`startAngle = 135f` Empieza abajo a la izquierda

`sweepAngle = 270f * progreso` Recorre hasta 3/4 de vuelta

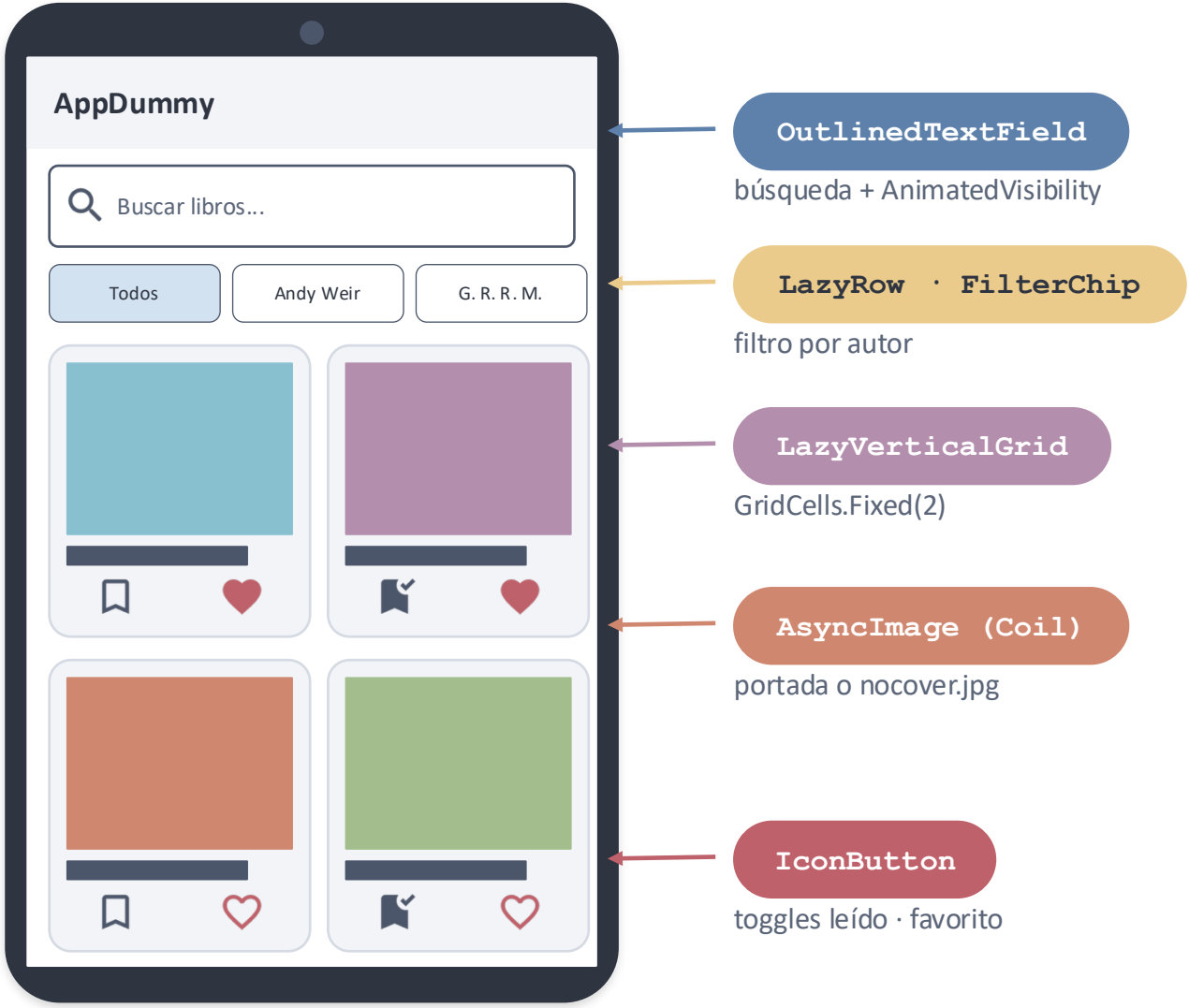
`Stroke(cap = Round)` Trazo con extremos redondeados

IndicadorPuntuacion.kt

```
Canvas(modifier.size(80.dp)) {
  val grosor = size.minDimension * 0.1f
  // arco de fondo
  drawArc(Color.LightGray,
    startAngle = 135f, sweepAngle = 270f,
    useCenter = false,
    style = Stroke(grosor,
        cap = StrokeCap.Round))
  // arco de progreso
  drawArc(colorArco,
    startAngle = 135f,
    sweepAngle = 270f * progreso,
    useCenter = false,
    style = Stroke(grosor,
        cap = StrokeCap.Round))
}
```

Desarrollo guiado: pantalla de listado

AppDummy v2




```
LibroUI (modelo provisional)
data class LibroUI(
  val id: Int = 0,
  val titulo: String = "",
  val autor: String = "",
  val year: Int? = 1900,
  val isbn: String = "",
  val cover: String = "",
  val esFavorito: Boolean = false,
  val leido: Boolean = false)
```

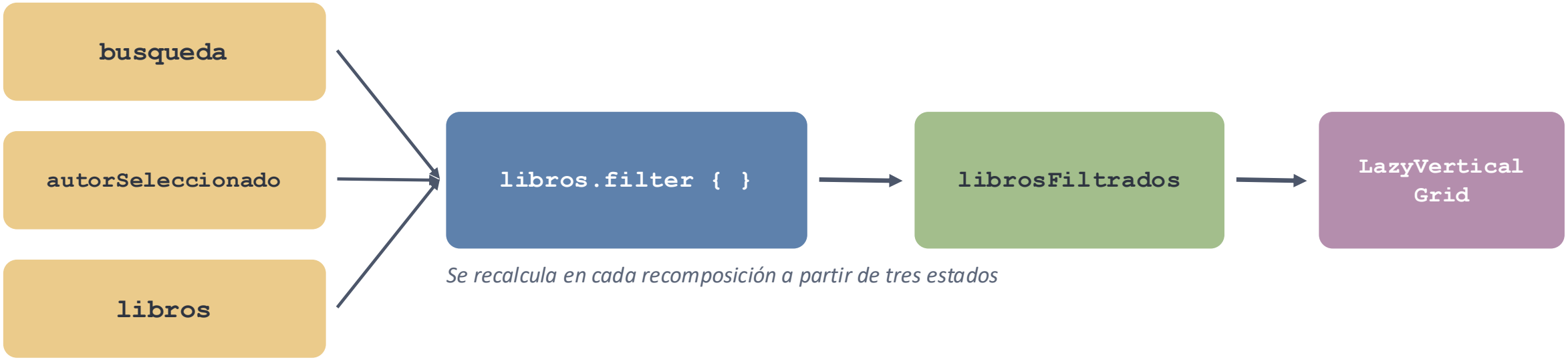
Dependencias y permiso

```
io.coil-kt.coil3:coil-compose:3.5.0
io.coil-kt.coil3:coil-network-okhttp:3.5.0
```

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

Filtrado reactivo y estado inmutable

AppDummy v2



Se recalcula en cada recomposición a partir de tres estados

```
Actualizar sin mutar
onToggleFavorito = { id ->
    libros = libros.map {
        if (it.id == id)
            it.copy(esFavorito = !it.esFavorito)
        else it
    }
}
```

Lista nueva = recomposición

map + copy crean una lista nueva; al asignarla al estado, Compose detecta el cambio. ItemLibro solo recibe datos y lambdas.

Ideas clave



Diseña para el móvil

RAM, batería, red y pantallas variables.



minSdk = 30

Equilibrio entre funciones y cuota.



weight y constraints

Reparto proporcional y layouts adaptativos.



Colores del tema

Roles semánticos: el modo oscuro funciona solo.



Animar es declarativo

Cambia el estado; Compose interpola.



Canvas

drawArc y compañía para lo que no existe.

Referencias

Fuentes

Layouts en Compose
developer.android.com/develop/ui/compose/layouts/basics

Componentes Material 3 en Compose
developer.android.com/develop/ui/compose/components

Animaciones en Compose
developer.android.com/develop/ui/compose/animation/introduction

Diseño adaptativo en Compose
developer.android.com/develop/ui/compose/layouts/adaptive

Canvas en Compose
developer.android.com/develop/ui/compose/graphics/draw/overview

API Levels
apilevels.com

Coil — carga de imágenes
coil-kt.github.io/coil

Tema 1C. UI avanzada en Compose (apuntes)
documentation.javiercarrasco.es/pmdm/B1/B1-T1c_UI_Avanzada