Bloque 2 · Tema 3

Navigation

Navegación entre pantallas con Jetpack Compose y rutas tipadas

6 horas · AppDummy · catálogo de libros · RA2



Hoja de ruta del tema

8 apartados

1 **Una sola Activity**

Múltiples pantallas en un NavHost

2 **Componentes**

NavController, NavHost y la pila

3 **Rutas tipadas**

@Serializable y toRoute<T>()

4 **Callbacks**

Las pantallas reciben lambdas

5 **Grafo de AppDummy**

Rutas.kt y AppNavigation

6 **Bottom Navigation**

Catálogo y Favoritos

7 **Gestión de la pila**

navigateUp, popBackStack, opciones

8 **ViewModel**

Por destino y compartido

Desarrollo práctico guiado sobre AppDummy

¿Qué vamos a evaluar?

RA2

RA2

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



Navigation Compose es una de esas librerías específicas

RA2-a	Se ha generado la estructura de clases necesaria para la aplicación.
RA2-c	Se han utilizado las clases necesarias para la conexión y comunicación con dispositivos inalámbricos.
RA2-g	Se han realizado pruebas de interacción usuario-aplicación para optimizar las aplicaciones a partir de emuladores.
RA2-i	Se han documentado los procesos necesarios para el desarrollo de las aplicaciones.

01

Fundamentos

Una sola Activity, NavController, NavHost y la pila de navegación



Una sola Activity, múltiples pantallas

`Single Activity`

Enfoque clásico



- ✗ Una Activity por pantalla en el manifest
- ✗ Datos en los extras del Intent (Bundle)
- ✗ Un ciclo de vida por pantalla

Single Activity + NavHost



- ✓ Navegación declarativa
- ✓ Botón Atrás gestionado por la pila
- ✓ Argumentos tipados y animaciones

Las tres piezas de Navigation Compose

Componentes



NavController

El conductor: decide a dónde ir y mantiene el estado de la navegación.

`rememberNavController()`



NavHost

El escenario: muestra el composable del destino actual.

`startDestination = Listado`



Back stack

La memoria: pila de destinos visitados; la cima es lo que se ve.

`navigate()` · `navigateUp()`

Flujo: la pantalla avisa → el NavController navega → el NavHost pinta la cima de la pila

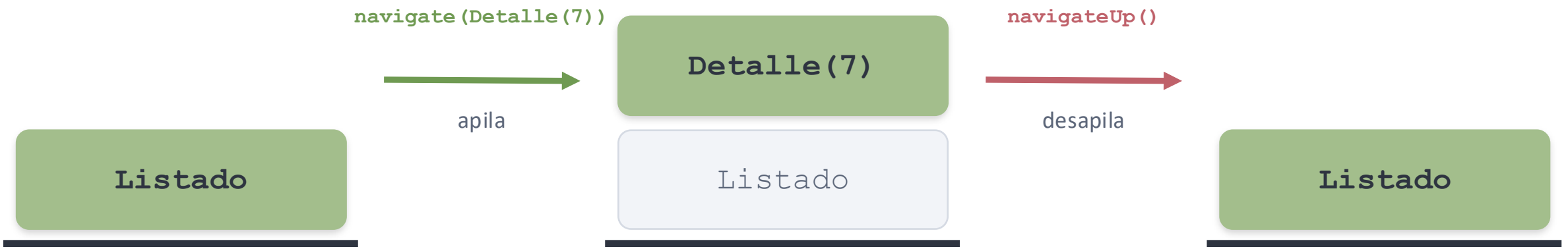
La pila de navegación en acción

Back stack

1 · Al abrir la app

2 · Tras pulsar un libro

3 · Tras volver



La pantalla visible es siempre la cima de la pila (en verde). Atrás = sacar la cima.

Dependencias necesarias

Gradle KTS

gradle/libs.versions.toml

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

serialization (plugin)	Misma versión que Kotlin
navigation-compose	2.9.8 · API tipada desde 2.8.0
kotlinx-serialization-json	1.11.0 · rutas como datos

Punto de partida: AppDummy v4 (Tema 2)

build.gradle.kts

```
// build.gradle.kts (proyecto)
plugins {
    alias(libs.plugins.kotlin.serialization) apply false
}

// build.gradle.kts (módulo app)
plugins {
    alias(libs.plugins.kotlin.serialization)
}

dependencies {
    implementation(
        "androidx.navigation:navigation-compose:2.9.8")
    implementation(
        "org.jetbrains.kotlinx:kotlinx-serialization-json:1.11.0")
}
```

02

Rutas tipadas

@Serializable, reglas de definición y toRoute<T>()



De rutas de texto a rutas tipadas

`@Serializable`

API antigua · rutas String

```
composable("detalle/{id}") { entry ->
    val id = entry.arguments
        ?.getString("id")?.toInt()
    PantallaDetalle(id)
}

// Navegar
navController.navigate("detalle/$id")
```

API tipada · navigation 2.8.0+

```
@Serializable
data class Detalle(val id: Int)

composable<Detalle> { entry ->
    val ruta = entry.toRoute<Detalle>()
    PantallaDetalle(ruta.id)
}

// Navegar
navController.navigate(Detalle(id))
```

Una errata en el texto falla al ejecutar

El compilador comprueba rutas y tipos

Cómo se define una ruta

Rutas.kt

Sin argumentos

data object

```
@Serializable
data object Listado
```

Inicio, Listado, Favoritos

Con argumentos

data class

```
@Serializable
data class Detalle(
    val id: Int
)
```

Detalle(id: Int)

Opcionales

nullable / defecto

```
@Serializable
data class Busqueda(
    val texto: String? = null,
    val pagina: Int = 1
)
```

Ejemplo ilustrativo

navegacion/Rutas.kt · Tipos básicos (Int, Long, Float, Boolean, String…) de serie; un tipo propio necesita un NavType

Leer los argumentos con toRoute<T>()

toRoute

1 · Se navega

```
navigate(
Detalle(42))
```

2 · Entrada en la pila

```
Detalle
(id = 42)
```

3a · NavHost `entry.toRoute<Detalle>().id`

3b · ViewModel `savedStateHandle.toRoute<Detalle>().id`

DetalleViewModel.kt (simplificado)

```
class DetalleViewModel(
    savedStateHandle: SavedStateHandle
) : ViewModel() {
    // El id llega dentro de la ruta:
    // la pantalla no tiene que pasarlo
    private val id =
        savedStateHandle.toRoute<Detalle>().id
}
```



SavedStateHandle

Ya contiene los argumentos del destino. El ViewModel obtiene el id y pide el libro a la capa de datos: encaja con MVVM.

03

Arquitectura de navegación

Callbacks, grafo de AppDummy y AppNavigation



Las pantallas reciben lambdas, no el NavController

Callbacks

Antipatrón

```
@Composable
fun PantallaListado(
    navController: NavController
) {
    // ...

navController.navigate(Detalle(libro.id))
}
```

Patrón recomendado

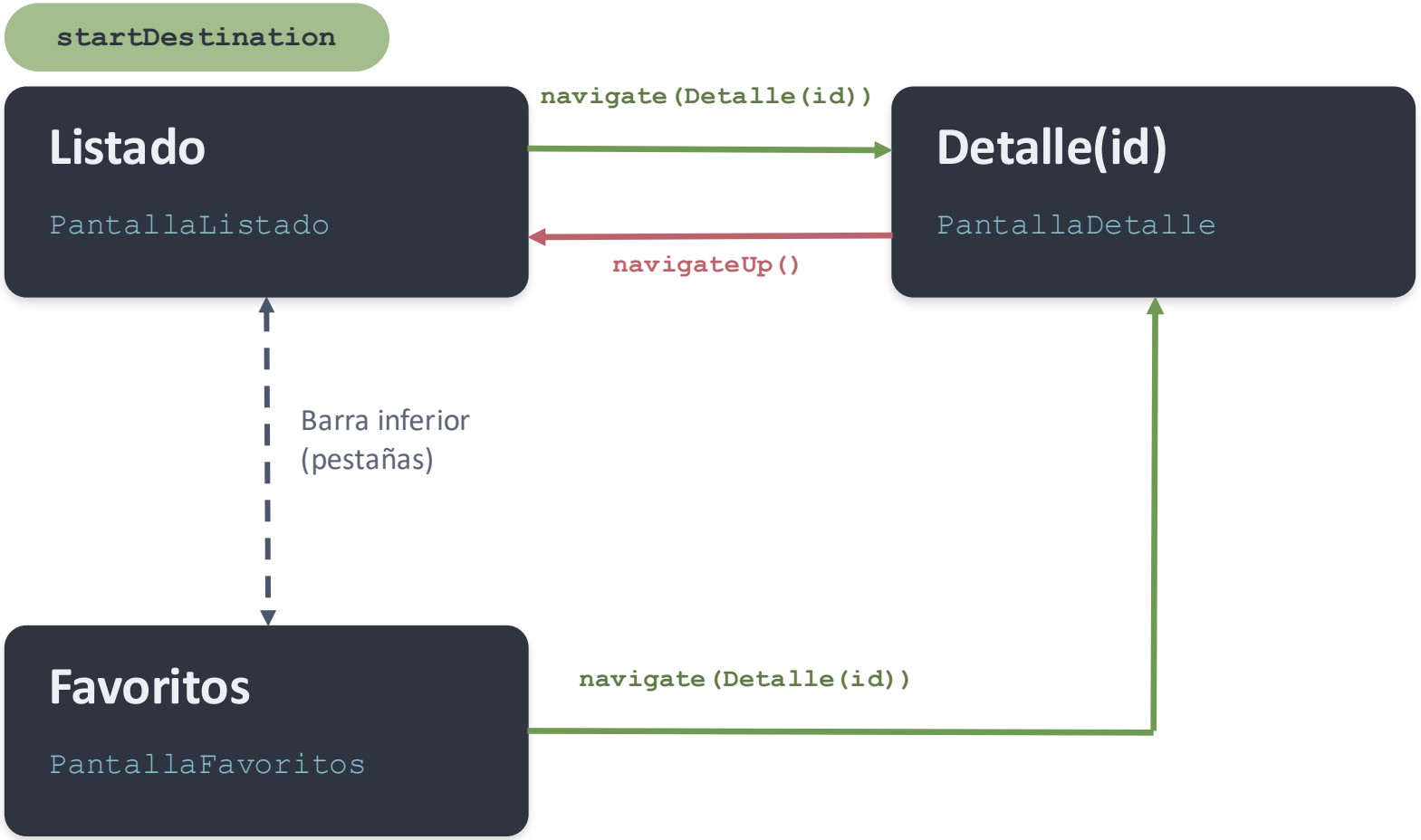
```
@Composable
fun PantallaListado(
    onNavegaADetalle: (Int) -> Unit
) {
    // ...
    onNavegaADetalle(libro.id)
}
```

Sin NavController no hay preview ni test sencillo

Preview y test con una lambda vacía: { }

El grafo de navegación de AppDummy

AppDummy



navegacion/Rutas.kt

Inicio	data object
Listado	data object
Detalle(id: Int)	data class
Favoritos	data object

Destino inicial del NavHost: **Listado**

AppNavigation: el grafo en código

NavHost

AppNavigation.kt (simplificado, sin Scaffold)

```kotlin
@Composable
fun AppNavigation(
    navController: NavHostController = rememberNavController()
) {
    NavHost(navController = navController, startDestination = Listado) {
        composable<Listado> {
            PantallaListado(
                onNavegaADetalle = { id -> navController.navigate(Detalle(id)) }
            )
        }
        composable<Detalle> {
            PantallaDetalle(onVolver = { navController.navigateUp() })
        }
        composable<Favoritos> { /* igual que Listado */ }
    }
}
```

1. **`startDestination`**
   La ruta con la que arranca el grafo

2. **`composable<T>`**
   Un destino por cada ruta

3. **Lambdas**
   Traducen los avisos de la pantalla en navegación

4. **`MainActivity`**
   Solo llama a AppNavigation() dentro del tema

04

Pila y ViewModel

Barra inferior, gestión de la pila y ámbito de los ViewModel



BottomNavigationBar: Catálogo y Favoritos

Bottom bar



1

```
popUpTo(graph.findStartDestination().id) { saveState = true }
```

No apila pestañas: vuelve a la raíz y guarda el estado de la que deja.

2

```
launchSingleTop = true
```

Pulsar la pestaña activa no crea otra copia del destino.

3

```
restoreState = true
```

Al volver a una pestaña recupera su estado (scroll, datos…).

`ItemNavegacion<T : Any>` · `currentBackStackEntryAsState()`

Cada ítem guarda etiqueta, ruta, icono e icono seleccionado. La entrada actual de la pila decide qué pestaña se marca y oculta la barra fuera de las pantallas principales.

navigateUp() frente a popBackStack()

Gestión de la pila

navigateUp()

- Sube un nivel en la jerarquía
- Pensado para la flecha ← de la TopAppBar
- No deja la pila vacía

popBackStack()

- Saca la cima de la pila
- Puede volver hasta una ruta concreta
- Si saca la última entrada: pantalla vacía

Opción en navigate(ruta) { … }	Qué hace
`popUpTo(Listado) { inclusive = true }`	Desapila hasta Listado (incluido si inclusive) antes de navegar
`launchSingleTop = true`	No duplica el destino si ya está en la cima
`saveState / restoreState`	Guarda y recupera el estado de los destinos desapilados

ViewModel: por destino o compartido

ViewModel

Por destino

Entrada de la pila · Detalle(7)

DetalleViewModel

Entrar en el destino: se crea

Salir (se desapila): se destruye

Dentro de composable<T>, viewModel() toma ese destino como dueño.

navigation<GrafoPedido>

PasoSeleccion → PasoConfirmacion

PedidoViewModel · uno para los dos pasos

```
val entradaPadre = remember(entry) {
    navController.getBackStackEntry<GrafoPedido>()
}
val vm: PedidoViewModel = viewModel(
    viewModelStoreOwner = entradaPadre,
    factory = PedidoViewModel.Factory
)
```

05

Practicar

Desarrollo práctico guiado: AppDummy con navegación completa



Desarrollo práctico guiado: AppDummy

Práctica

1 **Libro**

Modelo de datos del catálogo

2 **Rutas.kt**

Inicio, Listado, Detalle(id), Favoritos

3 **DetalleUiState**

Cargando · Exito(libro) · NoEncontrado

4 **DetalleViewModel**

Lee el id con toRoute y carga el libro

5 **PantallaDetalle**

Pinta cada estado y expone onVolver

6 **AppNavigation**

NavHost con Favoritos y barra inferior

Tu tarea: implementar PantallaFavoritos

Recibe onNavegaADetalle y abre Detalle(id), igual que el Listado. Cierra la navegación del Bloque 2.

Ideas clave



Una sola Activity

Un NavHost cambia el composable visible.



Rutas @Serializable

data object o data class: el compilador las comprueba.



toRoute<T>()

Lee los argumentos en el NavHost o en el ViewModel.



Lambdas, no NavController

La pantalla avisa; AppNavigation decide.



Bottom bar sin duplicados

popUpTo + launchSingleTop + restoreState.



ViewModel con ámbito

Por destino o compartido en un grafo anidado.

Referencias

Fuentes

Navegación con Compose
developer.android.com/develop/ui/compose/navigation

Seguridad de tipos en Navigation (type safety)
developer.android.com/guide/navigation/design/type-safety

Barra de navegación (Material 3 en Compose)
developer.android.com/develop/ui/compose/components/navigation-bar

Versiones de Navigation
developer.android.com/jetpack/androidx/releases/navigation

kotlinx.serialization
github.com/Kotlin/kotlinx.serialization

Tema 3. Navigation Compose (apuntes)
documentation.javiercarrasco.es/pmdm/B2/B2-T3_Navigation_Compose