# Unidad 8

Rodriguez Segura Fernando

## 1 Introducción

**Juice Tracker** es una aplicación de ejemplo diseñada por Google para enseñar cómo migrar una aplicación de Android construida con vistas tradicionales (XML) a Jetpack Compose, un toolkit moderno y declarativo para crear interfaces de usuario de forma más eficiente.

Esta documentación describe los temas cubiertos en el ejercicio, explica el código clave y detalla el proceso de migración.

## 2 Temas que se Aprenden

Este proyecto permite aprender los siguientes conceptos:

- **Migración parcial**: cómo introducir Jetpack Compose dentro de una app basada en vistas.
- **Uso de ComposeView**: integración de componentes composables dentro de layouts XML.
- **Declaración de UI con Composables**: cómo construir componentes de UI como tarjetas, listas y botones con funciones Composable.
- **Gestión de eventos y estado**: cómo pasar callbacks y manejar eventos como editar o eliminar un jugo.
- **Uso de RecyclerView junto con Compose**: demostrando interoperabilidad entre sistemas antiguos y nuevos.

## 3 Estructura del Proyecto

La aplicación permite a los usuarios agregar jugos con nombre, descripción y calificación, luego mostrarlos en una lista donde pueden ser editados o eliminados.

Estructura básica:

- `MainActivity`: inicia la aplicación.
- `Juice`: modelo de datos.
- `JuiceListAdapter` y `JuiceListViewHolder`: adaptadores para mostrar los datos.
- `JuiceItemComposable`: componente Composable que reemplaza al XML.

# 4 Configuración del Proyecto

Activar Jetpack Compose en el archivo `build.gradle.kts`:

Listing 1: Habilitar Jetpack Compose

```
android {
    buildFeatures {
        compose = true
    }
    composeOptions {
        kotlinCompilerExtensionVersion = "1.5.1"
    }
}
```

Agregar dependencias necesarias:

Listing 2: Dependencias de Compose

```
dependencies {
    implementation("androidx.activity:activity-compose:1.7.2")
    implementation("androidx.compose.material3:material3")
    implementation("androidx.compose.ui:ui-tooling")
}
```

# 5 Migración de Vista XML a Composable

## 5.1 Antes: Uso de Layout XML

El elemento de lista usaba un layout tradicional XML con TextViews y Buttons, inflado desde un `ViewHolder`.

## 5.2 Después: Uso de Composable

Reemplazamos la vista por un Composable que describe la UI:

Listing 3: Composable para el ítem de lista

```
@Composable
fun JuiceItemComposable(
    juice: Juice,
    onEdit: (Juice) -> Unit,
    onDelete: (Juice) -> Unit
) {
    Card(
        modifier = Modifier
            .padding(8.dp)
            .fillMaxWidth()
    ) {
        Column(modifier = Modifier.padding(16.dp)) {
```

```
            Text(text = juice.name, style = MaterialTheme.
               typography.titleLarge)
            Text(text = juice.description, style = MaterialTheme.
               typography.bodyMedium)
            Row(horizontalArrangement = Arrangement.SpaceBetween) {
                Button(onClick = { onEdit(juice) }) {
                    Text("Editar")
                }
                Button(onClick = { onDelete(juice) }) {
                    Text("Eliminar")
                }
            }
        }
    }
}
```

### 5.3 Integración con ComposeView

En lugar de inflar el XML en el ViewHolder, usamos `ComposeView`:

Listing 4: Uso de ComposeView en ViewHolder

```
class JuiceListViewHolder(
    private val composeView: ComposeView,
    private val onEdit: (Juice) -> Unit,
    private val onDelete: (Juice) -> Unit
) : RecyclerView.ViewHolder(composeView) {

    fun bind(juice: Juice) {
        composeView.setContent {
            JuiceItemComposable(juice, onEdit, onDelete)
        }
    }
}
```

## 6 Ventajas de Jetpack Compose

- Menos código y más legible.
- Código más declarativo.
- Más fácil de mantener.
- Compatible con la arquitectura moderna de estados y eventos.

# 7 Conclusión

La migración parcial a Jetpack Compose permite modernizar aplicaciones sin reescribir todo desde cero. El proyecto **Juice Tracker** muestra cómo integrar Compose de manera segura, elegante y progresiva dentro de apps existentes, facilitando una transición suave al nuevo paradigma declarativo.

# 8 Introducción

El proyecto **Juice Tracker App** es una aplicación educativa desarrollada en Kotlin utilizando Jetpack Compose. Su propósito principal es enseñar a los desarrolladores cómo construir una aplicación que gestione una lista de jugos, permitiendo agregar, editar y eliminar entradas, utilizando componentes modernos de arquitectura como ViewModel y Room.

# 9 Funcionalidad Principal

La aplicación permite:

- Visualizar una lista de jugos registrados.
- Agregar nuevos jugos con detalles como nombre y descripción.
- Editar la información de jugos existentes.
- Eliminar jugos de la lista.
- Persistir los datos utilizando Room.
- Observar y actualizar el estado de la interfaz de usuario utilizando Jetpack Compose y StateFlow.

# 10 Tecnologías Utilizadas

- **Jetpack Compose**: Toolkit moderno para construir interfaces de usuario en Android.
- **Room**: Biblioteca de persistencia para almacenar datos localmente.
- **Kotlin Coroutines y StateFlow**: Para manejar operaciones asincrónicas y observar cambios en los datos.
- **ViewModel**: Para gestionar el estado y la lógica de la interfaz de usuario.

# 11 Estructura del Proyecto

- `MainActivity.kt`: Actividad principal que configura la interfaz de usuario y observa el estado de los datos.
- `Juice.kt`: Modelo de datos que representa un jugo con su información relevante.

- `JuiceDao.kt`: Interfaz que define las operaciones de acceso a la base de datos.
- `JuiceDatabase.kt`: Clase que configura la base de datos utilizando Room.
- `JuiceViewModel.kt`: ViewModel que gestiona la lógica de negocio y el estado de la interfaz de usuario.
- `ui/`: Carpeta que contiene los componentes de la interfaz de usuario construidos con Jetpack Compose.

# 12 Ejemplo de Código

## 12.1 Definición de la Entidad Juice

Listing 5: Definición de la Entidad Juice

```
@Entity(tableName = "juice_table")
data class Juice(
    @PrimaryKey(autoGenerate = true) val id: Int = 0,
    val name: String,
    val description: String
)
```

## 12.2 Interfaz DAO para Acceso a Datos

Listing 6: Interfaz DAO para Acceso a Datos

```
@Dao
interface JuiceDao {
    @Query("SELECT * FROM juice_table")
    fun getAllJuices(): Flow<List<Juice>>

    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insert(juice: Juice)

    @Update
    suspend fun update(juice: Juice)

    @Delete
    suspend fun delete(juice: Juice)
}
```

## 12.3 Observación de Datos en la Interfaz de Usuario

Listing 7: Observación de Datos en la Interfaz de Usuario

```
val juiceList by juiceViewModel.juiceList.collectAsState()
```

```
LazyColumn {
    items(juiceList) { juice ->
        JuiceItem(juice = juice)
    }
}
```

# 13 Conclusión

El proyecto Juice Tracker App es una excelente referencia para aprender a construir aplicaciones Android modernas utilizando Kotlin, Jetpack Compose y componentes de arquitectura como ViewModel y Room. Además, demuestra cómo manejar operaciones de base de datos y observar cambios en los datos para mantener una interfaz de usuario reactiva y actualizada.

# 14 Recursos

- Repositorio GitHub: `https://github.com/google-developer-training/basic-android-kotlin-co`
- Curso Android Basics with Compose: `https://developer.android.com/courses/android-basics-co` `unit-7`
- Documentación de Room: `https://developer.android.com/training/data-storage/room`