📘 Actividades de Verificación – Unidad 4: Componentes de la arquitectura (Android Basics with Compose)

# 🎯 Objetivo general

Comprobar que:  
- Has realizado el codelab y visto los videos de la Unidad 4.  
- Comprendes los conceptos de ciclo de vida, ViewModel, estado y pruebas unitarias.  
- Puedes aplicar lo aprendido en una pequeña app propia.

# 1. Cuestionario de repaso (10 min)

Instrucciones: Responde de manera individual. Algunas preguntas son de selección múltiple, otras de respuesta breve.

* **1. ¿Qué sucede con el ciclo de vida de la Activity al rotar el dispositivo?**  
   a) Se mantiene igual, no cambia.  
   b) Se destruye y se vuelve a crear.  
   c) Solo se reinicia el método onResume().
* **2. Verdadero o falso: Un ViewModel sobrevive a los cambios de configuración (como la rotación de pantalla).**

Verdadero.

* **3. Explica con tus propias palabras: ¿Por qué no es recomendable guardar el estado directamente en la Activity?**

Como la Activity se destruye y se vuelve a crear en cambios de configuración (como la rotación), por lo que si se guarda ahí, se perderán los datos, pero si usamos ViewModel podemos mantener la información.

* **4. ¿Qué tipo de lógica se recomienda ubicar dentro de un ViewModel?** a) Lógica de UI.  
   b) Lógica de negocio y manejo de estado.  
   c) Recursos de diseño (colores, estilos).
* **5. En el codelab de pruebas unitarias, ¿qué parte del ViewModel se pone a prueba y por qué?**

Se pone a prueba la verificación de respuestas correctas o incorrectas, el puntaje y el fin del juego. Esto porque la lógica es independiente de la interfaz y puede evaluarse con pruebas unitarias para asegurar que el comportamiento interno sea correcto.

# 2. Ejercicio de completar código (15 min)

Instrucciones: Completa los huecos en el siguiente código.

class CounterViewModel : ViewModel() {  
 private val \_count = MutableStateFlow(0)  
 val count: StateFlow<Int> = \_count.**asStateFlow()** // completar  
  
 fun increment() {  
 \_count.**update** { old ->  
 old + 1  
 }  
 }  
}

@Composable  
fun CounterScreen(viewModel: CounterViewModel = viewModel()) {  
 val value by viewModel.count.collectAsState()  
  
 Column {  
 Text(text = "Valor: $value")  
 Button(onClick = { viewModel.**increment()**}) { // completar  
 Text("Incrementar")  
 }  
 }  
}

Pregunta extra: ¿Qué pasará si giras el dispositivo? ¿Por qué?

# 3. Reto práctico: extensión del codelab (2 horas para 1 de octubre)

Enunciado:  
Crea una pequeña aplicación Compose que tenga:  
- Un contador controlado por un ViewModel.  
- Un botón para reiniciar el contador a 0.  
- Prueba unitaria que verifique el correcto funcionamiento de la función reset().  
- Evidencia de que al rotar el dispositivo el valor no se pierde.

Entrega:  
- Código fuente en un repositorio o ZIP.  
- Captura de pantalla antes y después de la rotación.  
- Resultado de las pruebas unitarias (pantallazo o log).

----------------------------------------------------------------------------------------------------------------------

**- El código fuente se añadió a GitHub como un archivo .ZIP**

**- Captura de pantalla antes de la rotación**

****

**- Captura de pantalla después de la rotación**

**A screenshot of a computer

AI-generated content may be incorrect.**

**- Resultado de las pruebas unitarias**

**A screen shot of a computer

AI-generated content may be incorrect.**