Corrutinas vs Callbacks

Julio García Salas - 22076

Enlace al vídeo: https://www.youtube.com/watch?v=iBUXycCivzU

1. Manejo de Errores:

Corrutinas:

- Pro: Permiten el uso de estructuras de control de errores tradicionales como try/catch, facilitando el manejo de errores.
- Con: El manejo de errores en flujos de corrutinas complejas puede requerir una planificación cuidadosa.

Callbacks:

- Pro: Permiten el manejo de errores mediante el paso de funciones de error como argumentos.
- Con: El manejo de errores puede volverse complicado en flujos asíncronos complejos y anidados.

2. Control de Flujo:

Corrutinas:

- Pro: Ofrecen un control de flujo sofisticado con operadores como launch, async, withContext, etc.
- Con: Requieren una comprensión de los diferentes constructores y contextos de corrutinas.

Callbacks:

- Pro: Son simples y directos para flujos de control básicos.
- Con: No ofrecen un control de flujo avanzado y pueden ser difíciles de gestionar en operaciones asíncronas complejas.

4. Escalabilidad:

Corrutinas:

- Pro: Son ligeras y permiten lanzar miles de ellas sin agotar los recursos del sistema.
- Con: La escalabilidad puede depender de la correcta utilización de contextos y despachadores.

Callbacks:

- Pro: Son adecuados para tareas asíncronas simples y no bloqueantes.
- Con: Pueden no ser tan escalables como las corrutinas para aplicaciones de alta concurrencia.

5. Interoperabilidad:

Corrutinas:

- Pro: Kotlin ofrece una buena interoperabilidad con otras bibliotecas y lenguajes.
- Con: La integración con algunas bibliotecas basadas en callbacks puede requerir wrappers o adaptadores.

Callbacks:

- Pro: Son un patrón comúnmente utilizado y ampliamente compatible con muchas bibliotecas y APIs.
- Con: Pueden no ofrecer las mismas capacidades de interoperabilidad y adaptabilidad que las corrutinas en algunos contextos.

Conclusión:

- Uso de Corrutinas: Es preferible cuando buscas un código más limpio, legible y escalable, y cuando trabajas con operaciones asíncronas o concurrentes complejas.
- Uso de Callbacks: Puede ser adecuado para tareas asíncronas más simples y cuando trabajas con bibliotecas o APIs que utilizan este patrón**.**