



Los 10 errores al manejar conurrencia y threads

(Y cómo solucionarlos)



PABLO DEL ÁLAMO

Introducción



Manejar hilos y concurrencia puede ser complejo. Pequeños errores pueden provocar:



- Bloqueos del sistema.
- Pérdida de datos.
- Comportamientos impredecibles.

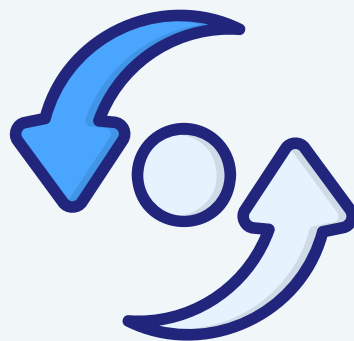


PABLO DEL ÁLAMO



Error 1: Olvidar sincronizar el acceso a recursos compartidos

-  Problema: Varios hilos acceden al mismo recurso sin control, causando inconsistencias.
-  Solución: Usa mecanismos como synchronized o Lock.





Error 2: Deadlocks por bloqueo mutuo

✗ Problema: Dos hilos esperando eternamente que el otro libere un recurso.

✓ Solución:

- Define un orden en la adquisición de recursos.
- Usa tryLock con timeout para evitar bloqueos infinitos.

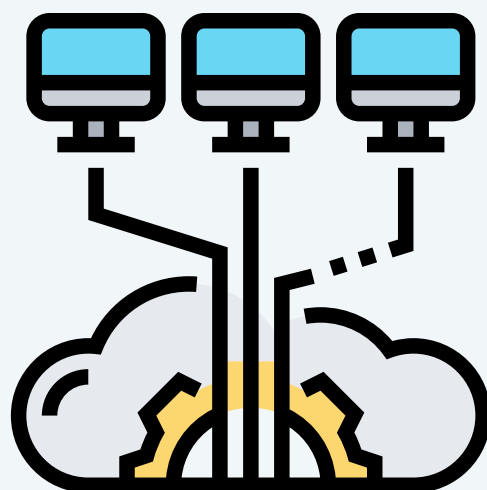




Error 3: No usar estructuras concurrentes

✗ Problema: Usar colecciones no seguras como ArrayList en entornos multihilo.

✓ Solución: Usa alternativas como ConcurrentHashMap o CopyOnWriteArrayList.



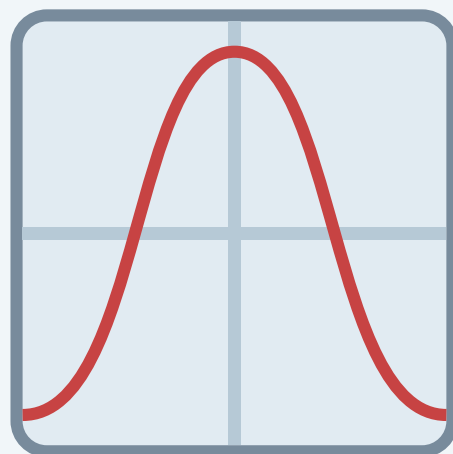
PABLO DEL ÁLAMO



Error 4: Abusar de threads individuales

✗ Problema: Crear demasiados hilos individuales puede sobrecargar el sistema.

✓ Solución: Usa un Thread Pool con un tamaño fijo para optimizar recursos.



PABLO DEL ÁLAMO



Error 5: No manejar excepciones dentro de hilos

✗ Problema: Una excepción no capturada puede detener un hilo sin avisar.

✓ Solución:

- Usa bloques try-catch dentro del código ejecutado por el hilo.
- Implementa un `Thread.UncaughtExceptionHandler`.





Error 6: No planificar correctamente el shutdown de threads

✗ Problema: Los threads no cerrados correctamente, siguen ejecutándose en segundo plano.

✓ Solución:

- Usa `ExecutorService.shutdown()`.
- Maneja interrupciones adecuadamente:



PABLO DEL ÁLAMO



Error 7: Condiciones de carrera (Race Conditions)

✗ Problema: Dos hilos acceden y modifican un dato al mismo tiempo.

✓ Solución: Usa mecanismos de sincronización o variables atómicas como AtomicInteger.



PABLO DEL ÁLAMO



Error 8: Confundir concurrencia con paralelismo

✗ Problema: Diseñar el sistema pensando que concurrencia siempre significa más velocidad.

✓ Solución: Entiende la diferencia:

- Concurrencia: Manejar múltiples tareas al mismo tiempo.
- Paralelismo: Ejecutar tareas simultáneamente en múltiples núcleos.

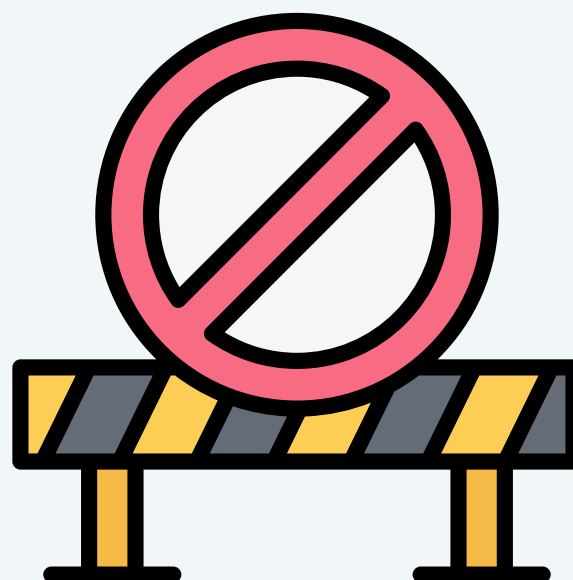




Error 9: Bloqueos innecesarios

✗ Problema: Bloquear recursos más tiempo del necesario, ralentizando el sistema.

✓ Solución: Reduce el tiempo de bloqueo al mínimo y usa bloques críticos cortos.



PABLO DEL ÁLAMO



Error 10: No probar en entornos concurrentes

✗ Problema: Bugs relacionados con hilos pueden no aparecer en pruebas simples.

✓ Solución: Usa herramientas como JUnit Stress Test o simulaciones con alta carga.



PABLO DEL ÁLAMO

Conclusión



Manejar concurrencia y threads es un desafío que requiere cuidado y planificación.

Evitar errores como condiciones de carrera, deadlocks y uso indebido de recursos compartidos no solo mejora el rendimiento, sino también la estabilidad de tus aplicaciones.

Implementar buenas prácticas como sincronización adecuada, uso de estructuras concurrentes y planificación de shutdowns, asegura que tus sistemas sean más eficientes y confiables.

La clave está en combinar el conocimiento teórico, con pruebas rigurosas en escenarios reales.



PABLO DEL ÁLAMO



¿Te ha resultado útil?



- Comparte esta guía con tu equipo o amigos desarrolladores.
- Guárdala para tenerla siempre a mano.
- ¡Dale un like o comenta si tienes preguntas!

