



## **Socket e Hilos -1**

Alejandro Cantos y Gonzalo  
Durán



## /1. Fase 1

En esta fase partimos de un servidor iterativo básico. Añadimos un `Thread.sleep(15000)` justo después de aceptar un cliente:

Cuando el servidor ejecuta:

```
socket = server.accept();  
Thread.sleep(15000);
```

Durante esos 15 segundos, el hilo principal está bloqueado y no puede aceptar un segundo cliente.

Esto demuestra que un servidor iterativo es bloqueante:

solo puede atender a un cliente a la vez y no puede procesar nuevas conexiones hasta terminar la anterior.

### ¿Qué ocurre con el Cliente 2?

El Cliente 2 no puede conectarse al servidor. Se queda esperando indefinidamente y ni siquiera llega a enviar su mensaje.

### ¿Por qué no puede enviar su mensaje?

Porque el servidor solo tiene un hilo (el hilo principal) y está bloqueado durante 15 segundos después de aceptar al Cliente 1.

## /2. Fase 2

En esta fase implementamos un sistema de conversación interactiva:

- El cliente escribe mensajes en un bucle
- El servidor le responde
- Se finaliza cuando el cliente envía "FIN"

Esto permite mantener un diálogo continúa sin cerrar la conexión tras un único mensaje.



## /3. Fase 3 y 4

Ahora en la fase 3 y 4 convertimos el servidor en concurrente mediante hilos.

Estructura creada:

- **Servidor:** solo acepta clientes
- **GestorCliente:** un hilo por cada cliente
- **Cliente:** igual que fase 2

Cada cliente se atiende de forma independiente, permitiendo que varios usuarios hablen simultáneamente con el servidor.

```
"C:\Program Files\Java\jdk-23\bin\java.exe"  
Escribe mensaje: hola  
Servidor dice: Recibido: hola  
Escribe mensaje: que tal  
Servidor dice: Recibido: que tal  
Escribe mensaje: fin  
  
Process finished with exit code 0
```

Captura del cliente 1, interactuando con el servidor y poniendo 'fin'.

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent"  
Escribe mensaje: soy el segundo cliente  
Servidor dice: Recibido: soy el segundo cliente  
Escribe mensaje: voy a cerrar sin poner fin  
Servidor dice: Recibido: voy a cerrar sin poner fin  
Escribe mensaje:
```

Captura del cliente 2, interactuando con el servidor y cerrando sin poner 'fin'.



```
server x cliente x
Servidor iniciado...
Cliente conectado desde: /127.0.0.1
Cliente conectado desde: /127.0.0.1
[/127.0.0.1] dice: hola
[/127.0.0.1] dice: que tal
[/127.0.0.1] dice: fin
El cliente /127.0.0.1 ha enviado FIN.
[/127.0.0.1] dice: soy el segundo cliente
[/127.0.0.1] dice: voy a cerrar sin poner fin
Cliente /127.0.0.1 se ha desconectado inesperadamente.
```

Captura de los logs del servidor donde se ven los mensajes de ambos clientes y la salida inesperada del segundo.