

# Informe Callback

Para el desarrollo de este taller se tomó como base el ejercicio de números de fibonacci realizado en el curso. Las actualizaciones que se solicitaban agregar estaban divididas en dos partes.

1. Implementar un esquema sencillo para determinar el número de clientes que generan la excepción de timeout.
2. Modificar el server implementando las siguientes funcionalidades:
  - a. Agregar multihilos para que pueda responder a múltiples clientes concurrentemente
  - b. Registrar clientes para hacer callback
  - c. Listar clientes
  - d. Enviar mensajes
  - e. Enviar broadcast

Luego de implementar las funcionalidades se probaron y se realizó el siguiente experimento para determinar el número de clientes con el que el server daba timeout.

Para tener control sobre el experimento se creó un arreglo con las siguientes constantes:

```
private int FIBONACCI_TIMEOUT_ARRAY_SIZE = 100000;  
  
private int FIBONACCI_TIMEOUT_MAX_VALUE = 50;  
private int FIBONACCI_TIMEOUT_MIN_VALUE = 0;
```

Con esto se obtienen la cantidad de peticiones a realizar y el número a pedir.

Con el siguiente método se realiza el experimento:

```
public void executeFibonacciTimeout() {  
    int index = 0;  
    String hostname = "";  
    ArrayList<CompletableFuture<String>> calculatedNumbers = new ArrayList<>();  
  
    try {  
        int[] array = createFibonacciTimeOutArray();  
        while(index < array.length - 1){  
            int number = array[index];  
            senderPrx.printFibonacci(hostname, String.valueOf(number));  
            index++;  
        }  
    } catch (TimeoutException timeoutException) {  
        System.out.println("TIMEOUT");  
        System.out.println(index+1);  
    }  
}
```

Número Clientes	Solicitud de Fibonacci	Timeout
1	100000 solicitudes	No
2	100000 solicitudes	Si

```
^C[swarch@hgrid16 tarea3]$ java -jar client.jar
Who are you? Enter your current user:
jp
TIMEOUT
17
```

Hace time out luego de 17 solicitudes en 1 cliente con 2 clientes al tiempo.

```
:165580141
:102334155
:63245986
:39088169
:24157817
:14930352
:9227465
:5702887
:3524578
:2178309
:1346269
:832040
:514229
:317811
:196418
:121393
:75025
:46368
:28657
:17711
```

El cliente calcula los números de la petición y luego los envia todos.

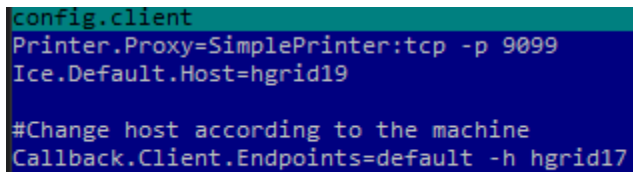
Número Clientes	Solicitud de Fibonacci	Timeout
1	100000 solicitudes	No
2	100000 solicitudes	No
3	100000 solicitudes	No

The image displays three terminal windows on a Kali Linux desktop. The top-left window, titled 'swarch@hgrid17:~/A00365918/tarea3', shows the execution of 'mc' and 'java -jar client.jar'. The top-right window, titled 'swarch@hgrid16:~/A00365918/tarea3', shows the execution of 'mc' and 'java -jar client.jar'. The bottom window, titled 'swarch@hgrid18:~/A00365918/tarea3', shows the execution of 'java -jar client.jar'. The desktop background is a landscape image with a body of water and a cloudy sky.

Al correr 3 clientes con 100000 solicitudes podemos ver que las respuestas del servidor tardan menor tiempo. La concurrencia de los multihilos es virtual ya que la máquina esta ocupando un hilo por cada petición que se le realiza. Si fuera real necesitaríamos otro nodo de procesamiento y asignarle la tarea para que se realicen en paralelo.

## Guía para correr el programa:

1. Realizar el deploy automático utilizando el comando:  
./deploy.sh
2. Correr el servidor ubicado en el xhgrid19/A00365918/tarea3  
Utilizando el comando java -jar server.jar
3. Correr los clientes ubicados en  
xhgrid17/A00365918/tarea3  
xhgrid18/A00365918/tarea3  
Antes de correrlos editar el archivo de configuración utilizando el comando “mc”



```
config.client
Printer.Proxy=SimplePrinter:tcp -p 9099
Ice.Default.Host=hgrid19

#Change host according to the machine
Callback.Client.Endpoints=default -h hgrid17
```

Cambiando el endpoint del cliente correspondiente. En caso de ejecutar desde el xhgrid17, el valor de Endpoints deberá ser hgrid17.

Luego correr utilizando el comando java -jar client.jar

4. Ingresar el nombre del usuario
5. Presionar 1 en el menu para registrarse
6. Usar libremente el programa