

INSTITUTO POLITÉCNICO NACIONAL



UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA Y TECNOLOGÍAS AVANZADAS

Unidad de Aprendizaje: Sistemas Distribuidos

Integrantes:

Mejía Hernández Karla Montserrat

Pedroza Velarde Luis Rodrigo

Grupo: 2TV7

Profesor: Miguel Félix Mata Rivera

PRÁCTICA 1: INTRODUCCIÓN AL MANEJO DE SOCKETS

Fecha de Entrega: 5 de marzo de 2023

Objetivos:

- Conocer cómo construir un Socket en Java, bajo la arquitectura CLIENTE-SERVIDOR y el enfoque de programación en red
- Comprender el concepto de arquitectura
- Comprender el concepto de transparencia en la comunicación
- Conocer el manejo básico de Github

I. COMUNICACIÓN CON SOCKETS LOCALMENTE

¿Qué necesitamos para que pueda comunicarse el programa Servidor (codificado en java) con un cliente (codificado en C) y viceversa?

Es necesario utilizar una arquitectura de comunicación en red, la forma que se utiliza más son los sockets. Un socket es un punto final de una conexión bidireccional entre 2 programas que se comunican a través de una red. Para implementarlo es necesario que se cree el servidor en Java que escuche las solicitudes entrantes, además de que se necesita un cliente en C que se conecte al servidor mediante un socket, esta acción también se realiza a la inversa para los casos de servidor en C y cliente en Java.

En Java las bibliotecas para el manejo de sockets se encuentran en el paquete `java.net`, mientras que en c la biblioteca está en el estándar de c, la forma de incluirlos en nuestros proyectos es la siguiente:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

Ilustración 1 Inclusión de librerías para manejo de sockets en C

```
import java.net.*;
import java.io.*;
```

Ilustración 2 Inclusión de paquetes para manejo de sockets en Java

II. COMUNICACIÓN CON SOCKETS REMOTAMENTE

¿Cómo se llama esta característica/funcionalidad en un sistema distribuido?

Que sea abierto

¿Qué es lo que permite que esta característica ocurra?

Como sabemos, que el sistema sea distribuido y se adapte al usuario desde los dispositivos compatibles al hacer una conexión, por lo que aquí también contamos con la flexibilidad y escalabilidad del sistema. Así que desde un socket podemos ver estas características en pequeña escala a diferencia de como se maneja un sistema distribuido

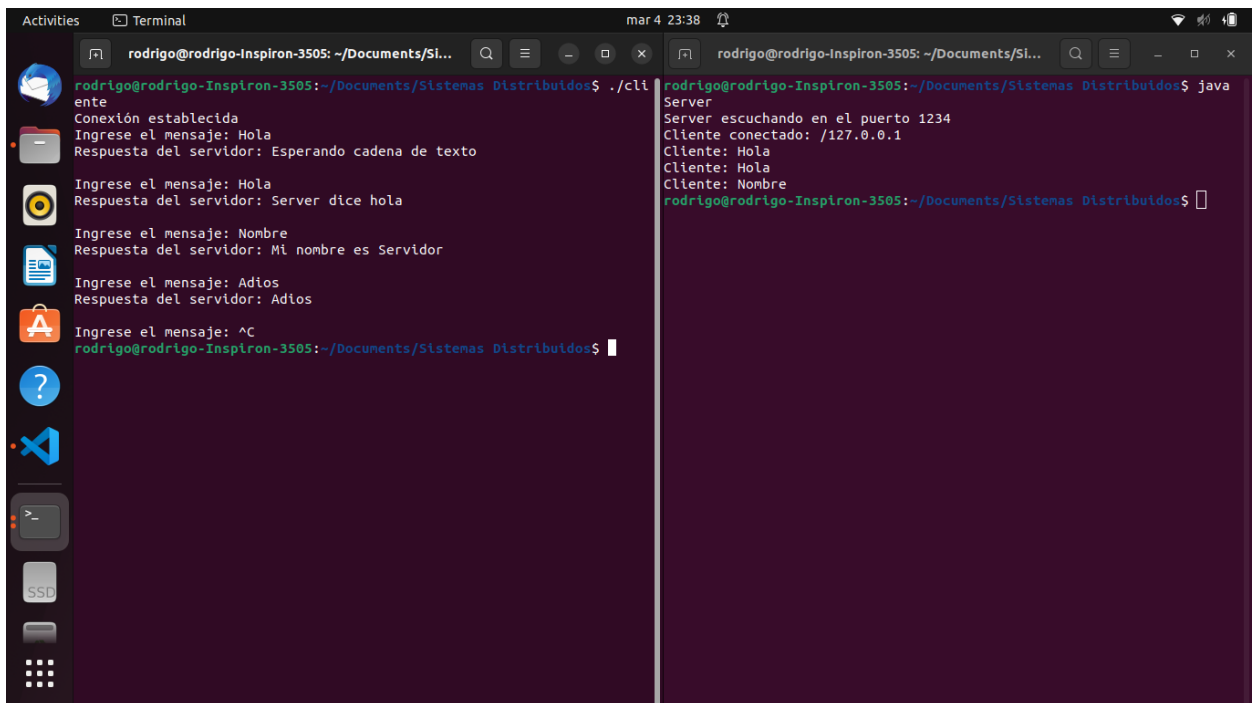
EJERCICIOS

Código en GitHub:

<https://github.com/rodri0319/Practica1>

Ejercicio 1.

Código Ejecutándose.



```
rodri@rodri-Inspiron-3505: ~/Documents/Sistemas Distribuidos$ ./cli
ente
Conexión establecida
Ingrese el mensaje: Hola
Respuesta del servidor: Esperando cadena de texto

Ingrese el mensaje: Hola
Respuesta del servidor: Server dice hola

Ingrese el mensaje: Nombre
Respuesta del servidor: Mi nombre es Servidor

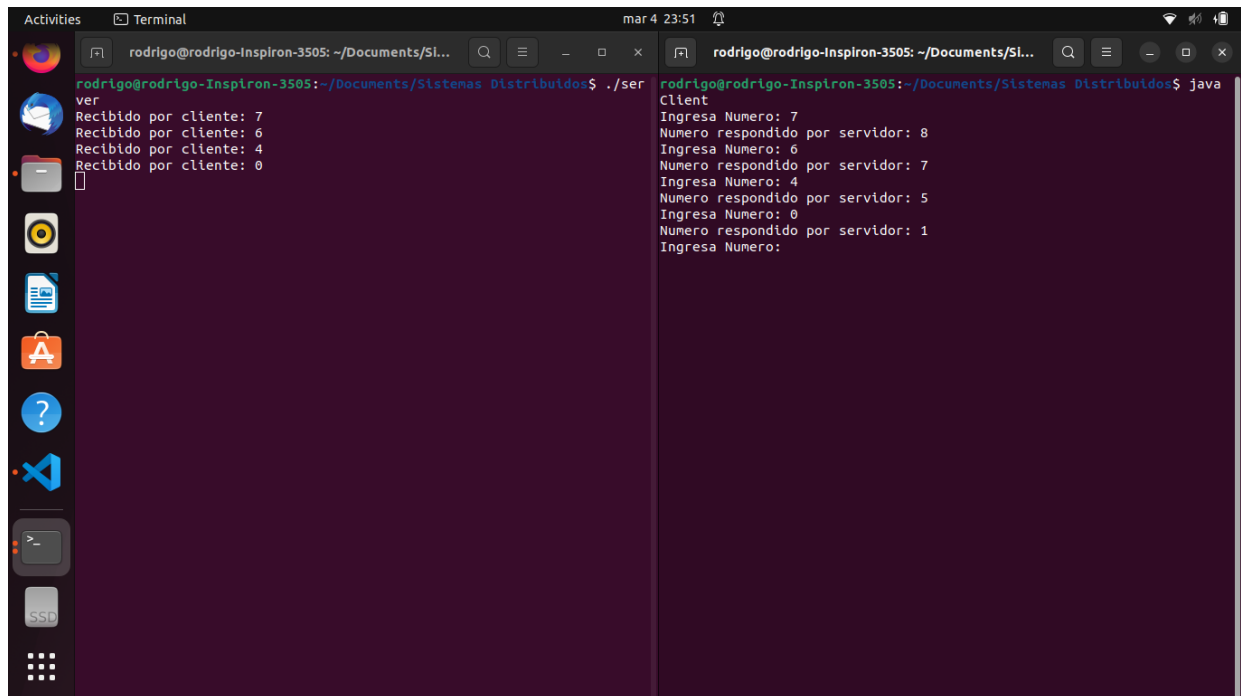
Ingrese el mensaje: Adios
Respuesta del servidor: Adios

Ingrese el mensaje: ^C
rodri@rodri-Inspiron-3505: ~/Documents/Sistemas Distribuidos$
```

```
rodri@rodri-Inspiron-3505: ~/Documents/Sistemas Distribuidos$ java
Server
Server escuchando en el puerto 1234
Cliente conectado: /127.0.0.1
Cliente: Hola
Cliente: Hola
Cliente: Nombre
rodri@rodri-Inspiron-3505: ~/Documents/Sistemas Distribuidos$
```

Ejercicio 2

Codigo Ejecutandose.



```
rodrico@rodrico-Inspiron-3505: ~/Documents/Sistemas Distribuidos$ ./server
ver
Recibido por cliente: 7
Recibido por cliente: 6
Recibido por cliente: 4
Recibido por cliente: 0

rodrico@rodrico-Inspiron-3505:~/Documents/Sistemas Distribuidos$ java Client
Ingreso Numero: 7
Numero respondido por servidor: 8
Ingreso Numero: 6
Numero respondido por servidor: 7
Ingreso Numero: 4
Numero respondido por servidor: 5
Ingreso Numero: 0
Numero respondido por servidor: 1
Ingreso Numero:
```