|  |  |
| --- | --- |
|  | **Proyecto 2**  **Sistemas operativos**  **Universidad Nacional de Colombia**  **Tema Cliente/Servidor** |

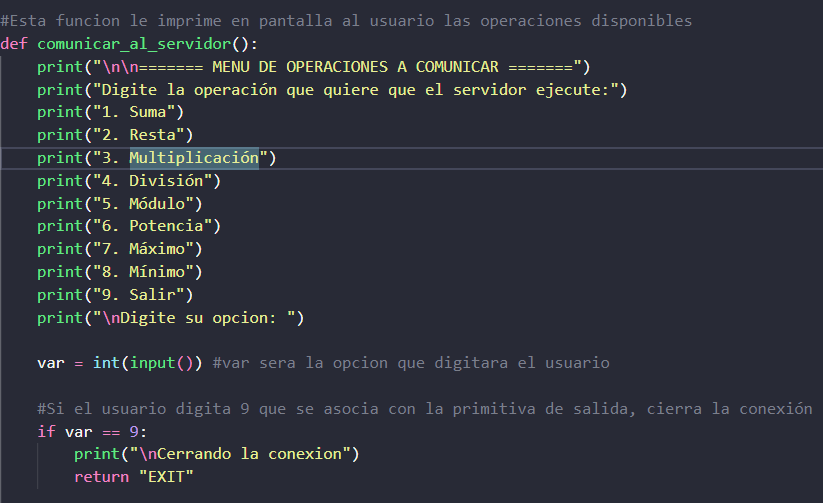
**Integrantes:**

* **Juan David Fajardo Betancourt**
* **Esteban Lopez Usma**

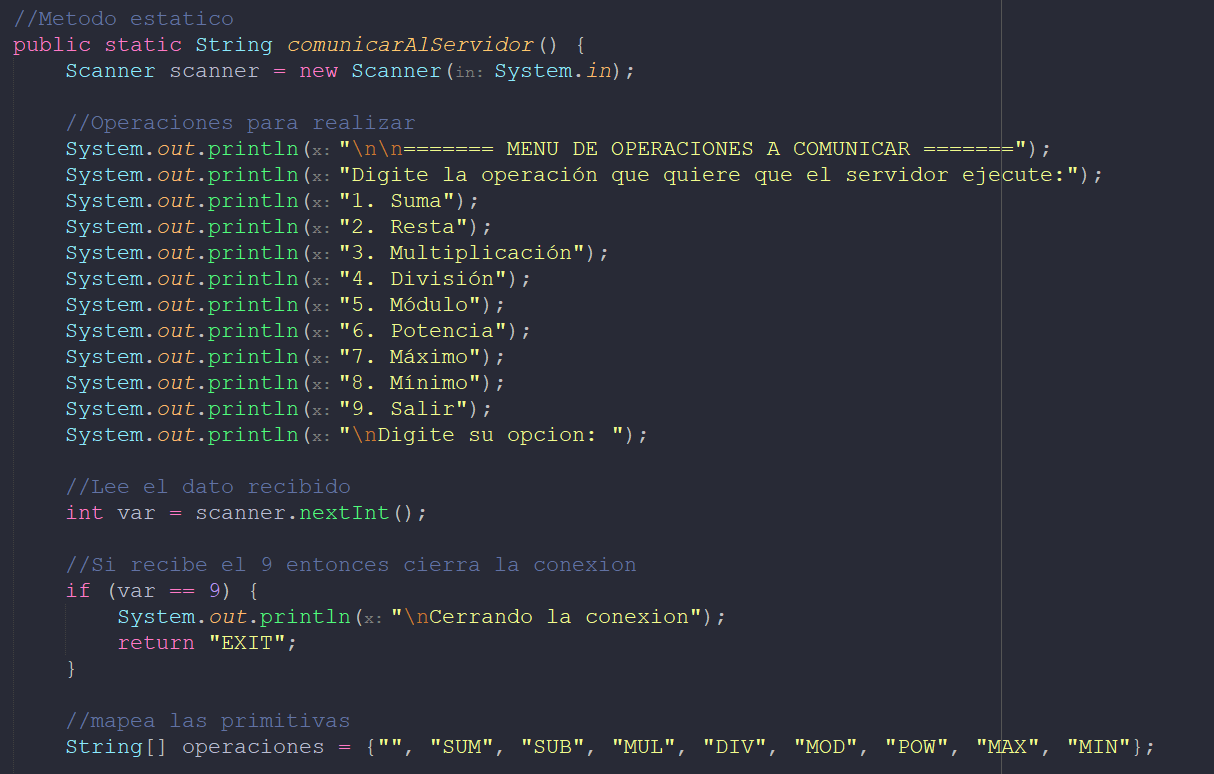
Dadas las aplicaciones TCPClient para Java y TCPServer para Java.

1. Haga que la aplicación Servidora en vez de responder un mensaje, devuelva datos de 8 operaciones. Suma, Resta, Multiplicación y División y 4 que usted defina.

R/ **Python**

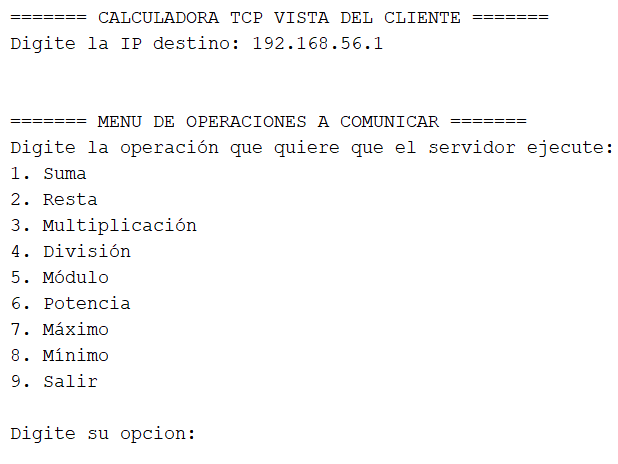
****

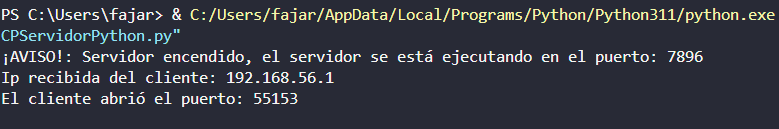
**Java**

****

* 1. La aplicación Cliente envíe como mensaje los números que desea sean calculados y la operación y reciba el resultado desde el servidor usando el paso de mensajes.

**Ejemplo con server Python y Cliente Java (Se hace lo mismo viceversa)**

****







* 1. Esta aplicación deberá recibir múltiples conexiones.

**Python**



**Java**

****

Estas líneas establecen una puerta para recibir múltiples conexiones en simultaneo.

* 1. Defina el protocolo que está usando y codifique los mensajes.

R/ El protocolo definido es TCP, (Transmission Control Protocol) es uno de los principales protocolos de la suite de protocolos de Internet (IP). TCP se encarga de establecer y mantener una conexión confiable entre dos dispositivos en una red para garantizar que los datos se transmitan de manera ordenada y sin errores. Algunas características clave de TCP incluyen:

1. Fiabilidad: TCP asegura que todos los paquetes de datos lleguen a su destino correctamente y en el orden adecuado. Si algún paquete se pierde o llega dañado, TCP se encarga de retransmitirlo.

2. Control de flujo: TCP regula la cantidad de datos que se envían en un momento dado, lo que ayuda a evitar la sobrecarga de la red.

3. Control de congestión: TCP ajusta la velocidad de transmisión de datos según la capacidad de la red para evitar la congestión.

4. Establecimiento de conexión: TCP utiliza un proceso conocido como "handshake de tres vías" para establecer una conexión antes de que los datos se transmitan. Esto asegura que ambos lados estén listos para comunicarse.

5. Segmentación y reensamblaje: TCP divide los datos en segmentos antes de enviarlos y luego los vuelve a ensamblar en el destino.

Gracias a estas características, TCP es ampliamente utilizado en aplicaciones donde la fiabilidad es crucial, como la navegación web, el correo electrónico y la transferencia de archivos.

**Codificación**

La codificación de mensajes en el contexto del protocolo TCP se refiere a cómo los datos se estructuran y transmiten entre dispositivos en una red. Aquí tienes un resumen de cómo se codifican los mensajes en TCP:

1. Segmentación

- Los datos que se van a enviar se dividen en segmentos más pequeños, que se pueden gestionar de manera más eficiente por la red. Cada segmento incluye un encabezado (header) y una parte de los datos (payload).

2. Encabezado (Header)

- Cada segmento de TCP contiene un encabezado que incluye información crucial para la comunicación, como:

- Número de puerto de origen y destino: Identifica las aplicaciones que están enviando y recibiendo los datos.

- Número de secuencia (Sequence Number): Indica la posición del primer byte de datos en este segmento en el flujo de datos completo.

- Número de acuse de recibo (Acknowledgment Number): Confirma la recepción de datos previamente enviados.

- Flags de control: Bits que indican el estado de la conexión, como SYN (para iniciar una conexión), ACK (para reconocer la recepción de datos), FIN (para finalizar una conexión), entre otros.

- Ventana de recepción (Window Size): Indica cuánto espacio hay disponible en el buffer del receptor para recibir más datos.

- Checksum: Un valor que se usa para verificar la integridad del segmento. El receptor utiliza este valor para comprobar si el segmento ha llegado intacto.

- Opciones: Información adicional opcional que se puede incluir, como el tamaño máximo de los segmentos (MSS).

3. Codificación binaria

- Toda la información en el encabezado y los datos del segmento se codifican en formato binario. Esto significa que se representa mediante una serie de unos y ceros (bits), lo que es adecuado para la transmisión a través de medios físicos como cables, fibra óptica o señales inalámbricas.

4. Transmisión

- Una vez que el mensaje está segmentado y codificado en binario, se envía a través de la red. El protocolo TCP se encarga de gestionar la retransmisión en caso de que algún segmento se pierda o llegue con errores.

5. Reensamblaje y decodificación

- En el receptor, los segmentos se reensamblan en el orden correcto utilizando los números de secuencia. Luego, los datos se decodifican para su uso por la aplicación destino, asegurando que la información se reciba tal como se envió.

Este proceso de codificación y decodificación garantiza la integridad y el orden de los datos transmitidos entre dispositivos en una red usando TCP.

1. Crear un Servidor o en C o en C++ o en Python para que el TCPClient de java se comunique con él.

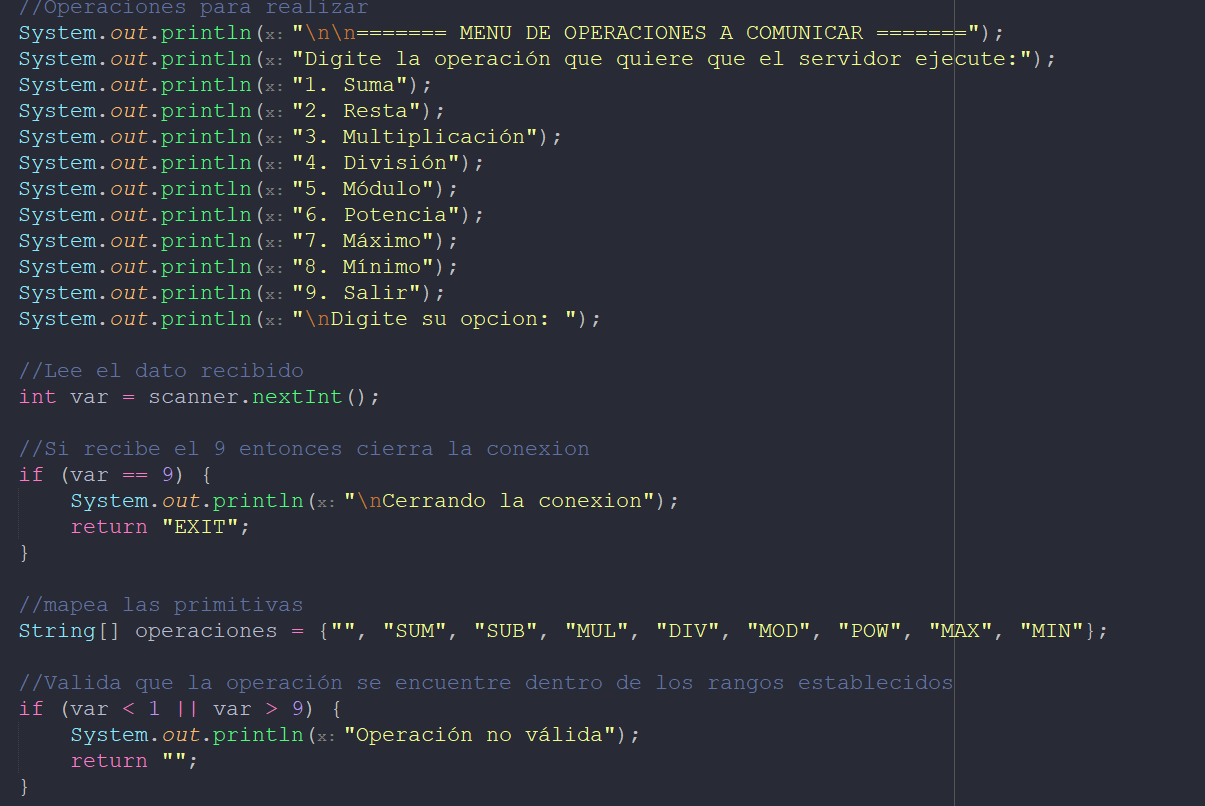
Adjuntos en la entrega de Moodle los señalados de arriba.

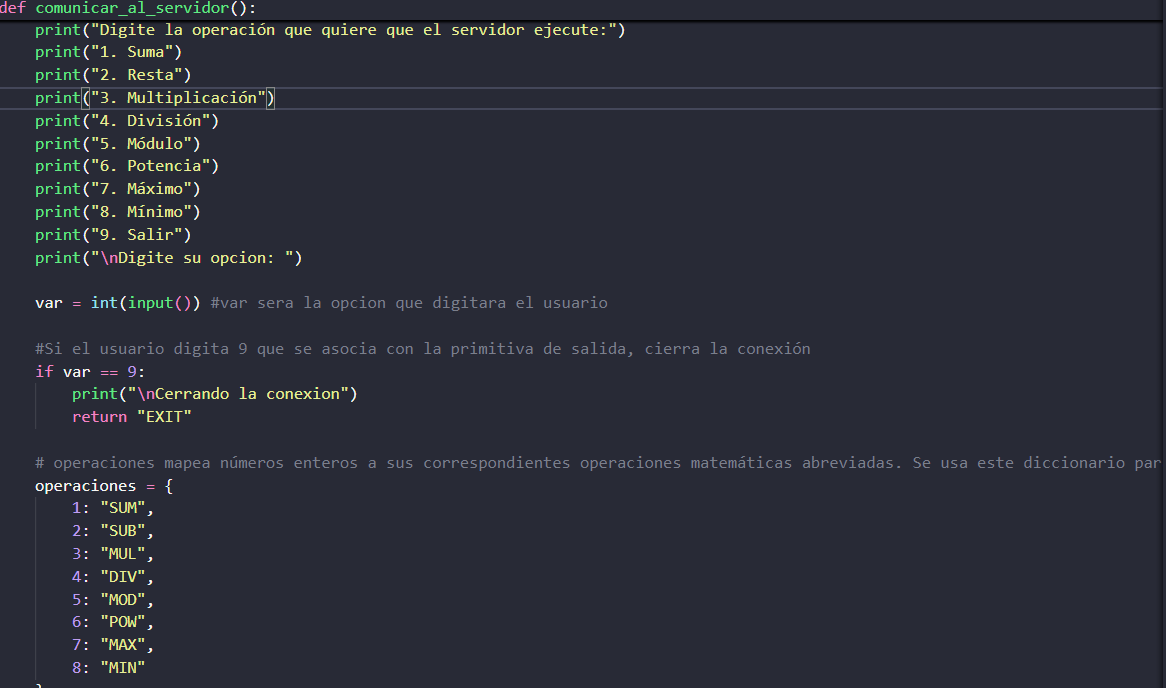
1. Realice una aplicación Cliente en C o en C++ o en Python que se comunique con el servidor TCPServer que está hecho en Java.

También adjuntos.

1. Entregue el los códigos y explique el protocolo que uso para realizar las operaciones.

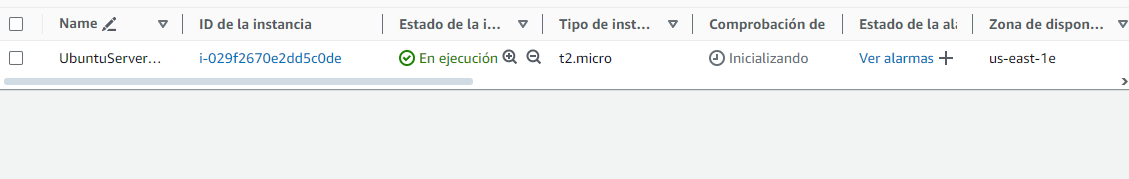
El protocolo que se uso fue TCP, y la forma en la que se adecuaron las opciones del protocolo fueron por medio de primitivas con denominaciones mayúscula y solo 3 letras, a excepción de la primitiva EXIT que utiliza cuatro para su denominación





En el servidor se encuentran las operaciones mediante switch case localiza por medio del numero de denominación dicha primitiva, para los códigos de los clientes se usaron diccionarios con un numero asociado que mapea las primitivas que el usuario puede realizar y por ende al código ubicara mas eficiente la operación mediante el numero que el cliente digito

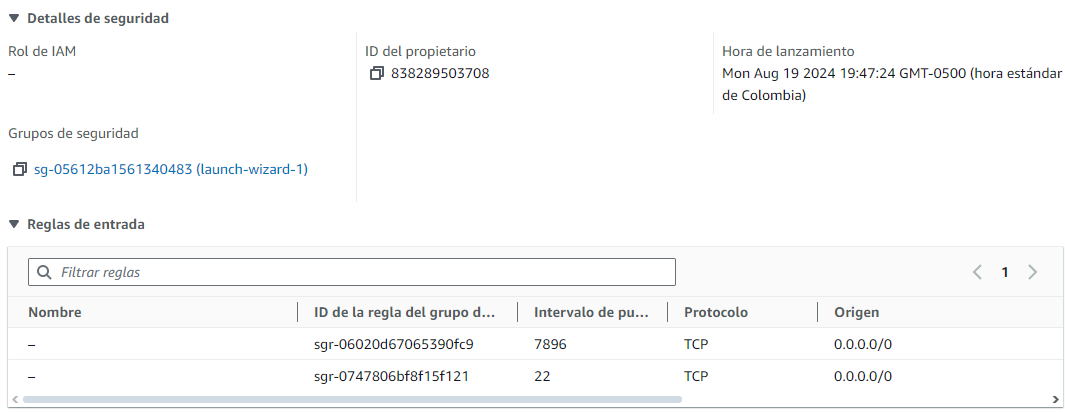
1. Los servidores deben ejecutarse en AWS.



Se lanzó una instancia desde AWS, usando Ubuntu como maquina virtual base de cloud

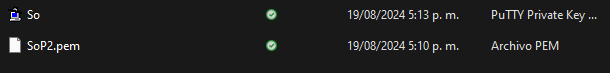


Información de la instancia lanzada, OJO, la ipv4 publica de la maquina es dinámica por lo que al inicar la instancia nuevamente cambia.

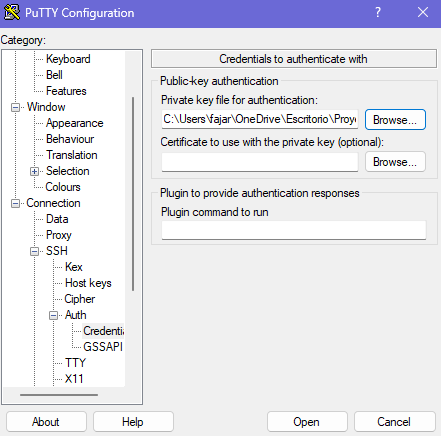


Acá se visualiza las reglas de tcp para la conexión de la maquina

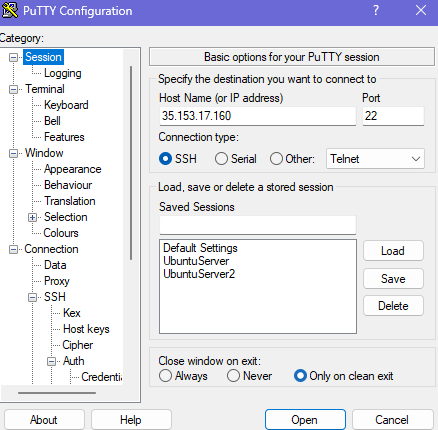
Para conectarse usando SSH desde el pc cliente a la maquina servidora, es necesario usar putty usando keys codificadas en estos archivos



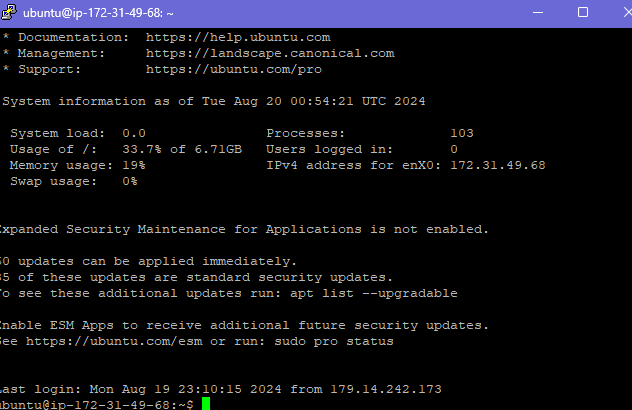
El archivo pem lo genera el AWS y el so solito es usando puttygen



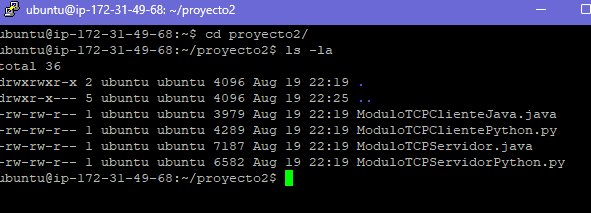
El archivo de llave es para hacer la autenticación inmediata



Conexión con putty usando ssh en puerto 22



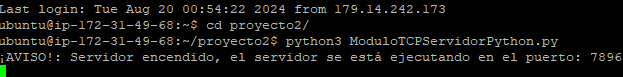
Maquina virtual de aws

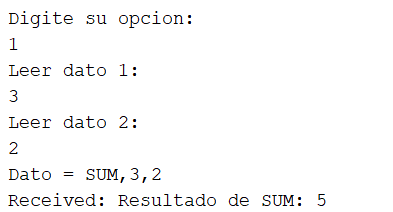


Creación de una carpeta por el cual se cuenta con los servidores de java y Python



Servidor python





Servidor Java

