DEPARTAMENTO:	DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN	**CARRERA:**	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN		
ASIGNATURA:	APLICACIONES DE SISTEMAS OPERATIVOS	**PERIODO LECTIVO:**	OCTUBRE 2021-MARZO 2022	**NIVEL:**	5to
DOCENTE:	ING. ANDREA LÓPEZ	**NRC:**	4647	**PRÁCTICA N°:**	2.1
TEMA DE LA PRÁCTICA:	ARQUITECTURA CLIENTE - SERVIDOR				

OBJETIVOS:

1. **Conocer la arquitectura cliente – servidor.**
2. **Realizar la practica entre máquina virtual y maquina anfitrión.**
3. **Establecer conexión para el paso de mensajes Servidor – Cliente mediante herramienta Putty.**
4. **Interpretar el paso de mensajes.**

MARCO TEÓRICO:

- **Arquitectura cliente-servidor**

La arquitectura cliente-servidor es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios. Llamadas servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor es quien da la respuesta.

Describe la configuración detallada de cada servidor e incluye: Hardware que se necesita para cada servidor. Sistema operativo que se necesita para cada servidor.

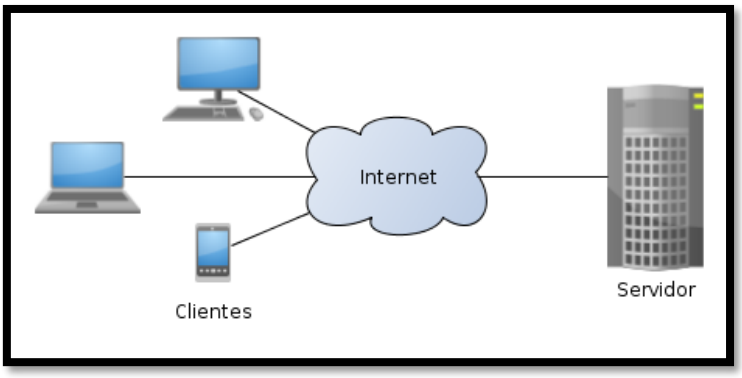


Ilustración 1. Arquitectura cliente-servidor

- **Lenguaje Python**

Pyhton es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo.

Es un lenguaje sencillo de leer y escribir debido a su alta similitud con el lenguaje humano. Además, se trata de un lenguaje multiplataforma de código abierto y, por lo tanto, gratuito, lo que permite desarrollar software sin límites.

- **Máquina virtual**

Es un software que simula un sistema de computación y puede ejecutar programas como si fuese una computadora real. Este software en un principio fue definido como “un duplicado eficiente y aislado de una máquina física”.

Tiene su propio disco duro, memoria, tarjeta gráfica y demás componentes de hardware, aunque todos ellos son virtuales.

- **Herramientas de gestión**

Son una serie de herramientas que sirven para mejorar el funcionamiento del sistema y proteger la información. Se dividen en tres herramientas que son:

1) De aplicación.
2) De configuración.
3) De optimización.

MATERIALES:

REACTIVOS: Dejar en blanco / No aplica	**INSUMOS:** Dejar en blanco / No aplica

EQUIPOS:

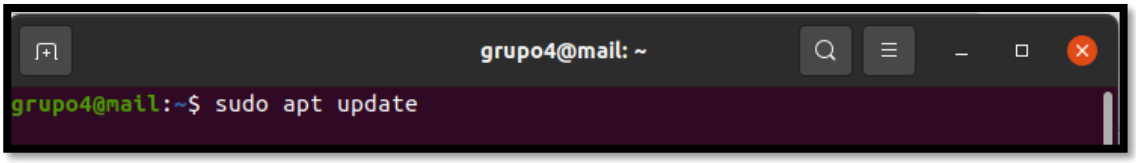
- PC

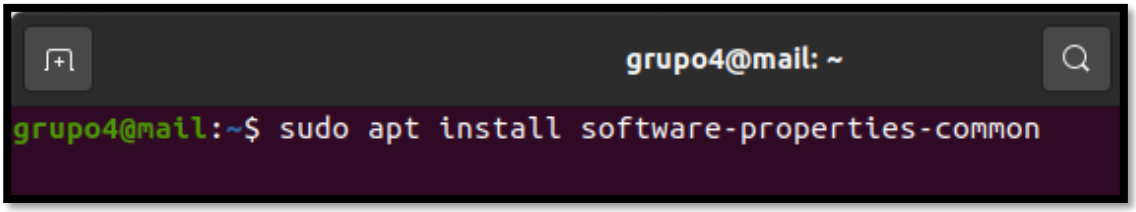
MUESTRA:
Dejar en blanco / No aplica

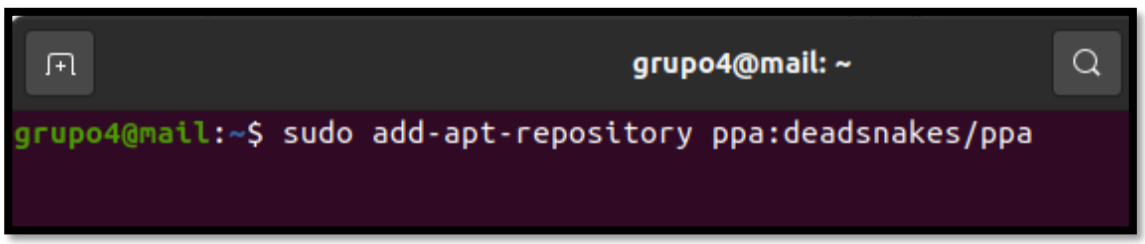
PROCEDIMIENTOS Y ACTIVIDADES REALIZADAS EN LA PRÁCTICA:

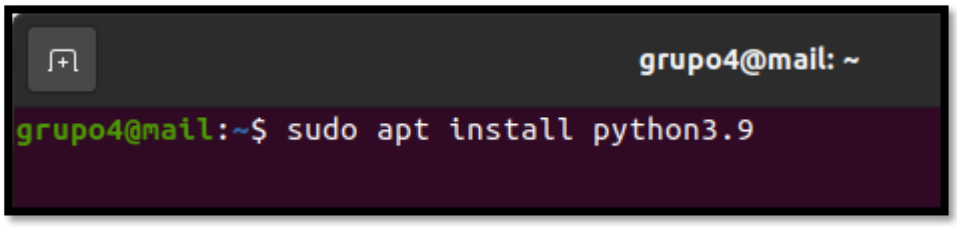
a) Implementar un modelo de paso de mensajes en una arquitectura cliente-servidor (se sugiere usar los siguientes lenguajes: C, C++, Python)

Instalación de Python en Ubuntu.

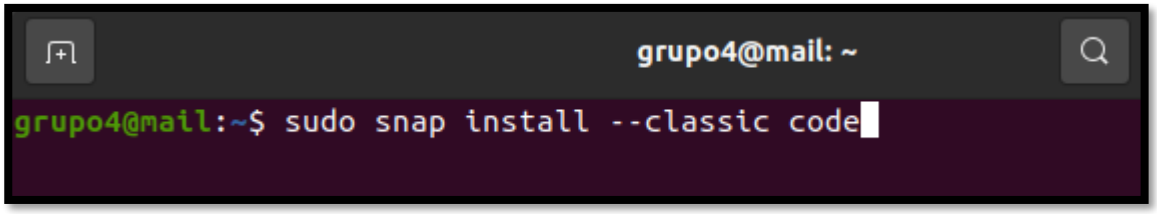




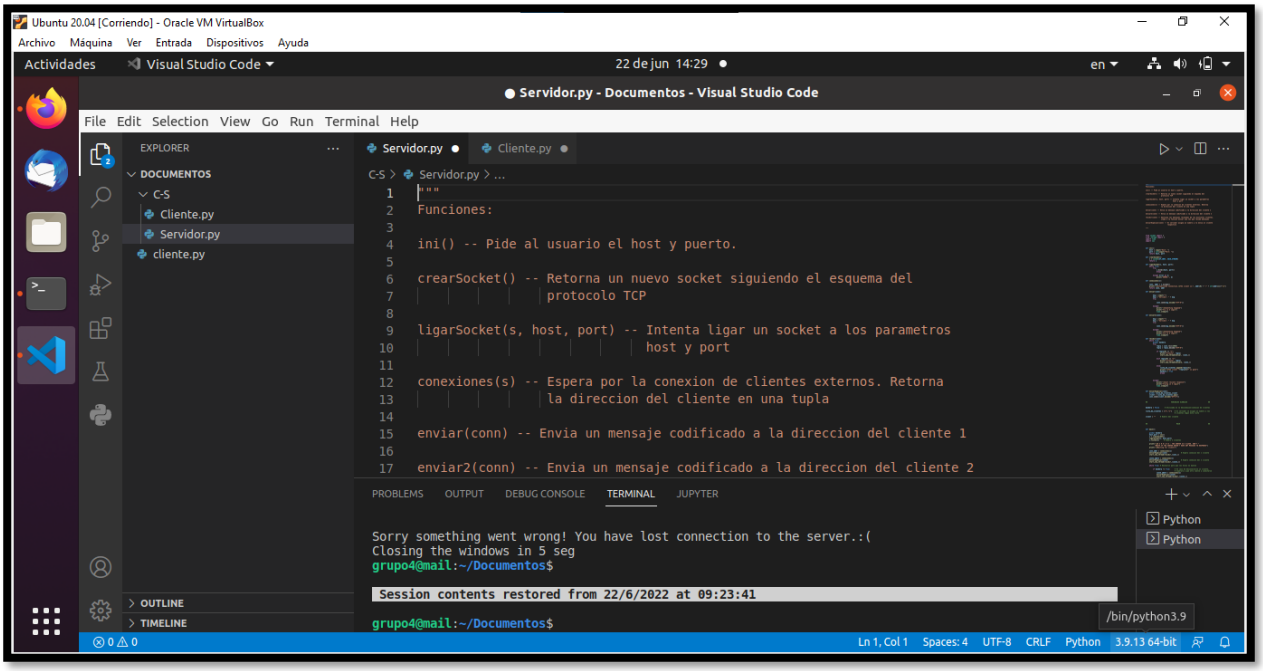




Instalación de Visual Studio Code en Ubuntu



Visual Studio Code Instalado en máquina virtual Ubuntu



Creación de código en Python para el Servidor y Cliente

SERVIDOR

```
from socket import *
from _thread import *
import time
import sys

#DEFINIR FUNCIONES

def ini():
    host = input("Host: ")
    port = int(input("Port: "))
    return host, port

def crearSocket():
    s = socket(AF_INET, SOCK_STREAM)
    return s

def ligarSocket(s, host, port):
    while True:
        try:
            s.bind((host, port))
            break

        except error as e:
            print("ERROR:", e)

def conexiones(s):

    conn, addr = s.accept()
    print("\nEstablished Connection.\nThe client is:", addr[0] + ":" + str(addr[1])+"\n")
    return conn, addr

def enviar(conn):

        msg = input("")
        msg = "Servidor: " + msg
        try:

            conn.send(msg.encode("UTF-8"))

        except:
            print("\nSomething happend")
            print("Try in 5 seg\n")
            time.sleep(5)
```

```
def enviar2(conn):

        msg = input("")
        msg = "Servidor: " + msg
        try:

            conn.send(msg.encode("UTF-8"))

        except:
            print("\nSomething happend")
            print("Try in 5 seg\n")
            time.sleep(5)

def recibir(conn):
    while True:
        global bandera
        try:
            reply = conn.recv(2048)
            reply = reply.decode("UTF-8")

            if reply[0] == "1":
                print("Cliente", reply)
                start_new_thread(enviar, (conn,))

            elif reply[0] == "2":
                print("Cliente", reply)
                start_new_thread(enviar2, (conn,))

            else:
                lista_de_clientes.append(reply[4])
                print("\nThe client "+reply[4]+" is gone")
                bandera = True
                break

        except:
            print("\nCant recieve response")
            print("Trying in 5 seg\n")
            time.sleep(5)


def enviarEspecial(conn):
    global lista_de_clientes,client
    client = lista_de_clientes.pop()
    conn.send(client.encode("UTF-8"))

bandera = False      # Utilizada en la desconexion/conexion de clientes
```

```
lista_de_clientes = ["2","1"]   # El servidor le asigna un numero a los
                                # clientes segun esta lista

client = ""     # Numero del cliente

#FUNCION PRINCIPAL

def main():

    global bandera #VARIABLE GLOBAL
    host,port = ini()
    s = crearSocket()
    ligarSocket(s, host,port)
    s.listen(2)     # Espero 2 clientes

    print("\nW A R N I N G : THE SERVER IS A SLAVE. DON'T "
          "WRITE IF THE SERVER DOESN'T HAVE ANY MESSAGE TO RESPONSE")
    print("\nWaiting for clients")

    conn,addr = conexiones(s)
    enviarEspecial(conn)               # Espero conexion del 1 cliente
    start_new_thread(recibir,(conn,))

    conn2,addr2 = conexiones(s)
    enviarEspecial(conn2)              # Espero conexion del 2 cliente
    start_new_thread(recibir,(conn2,))

    while True: # Necesario para que los hilos no mueran

        if bandera != True:     # En caso de desconectarse un cliente,
                                # esperara a que otro vuelve a conectarse
            conn3,addr3 = conexiones(s)
            enviarEspecial(conn3)
            start_new_thread(recibir,(conn3,))
            bandera = False

main()
```

CLIENTE

```
from socket import *
import time
from _thread import *

#DEFINIR FUNCIONES
```

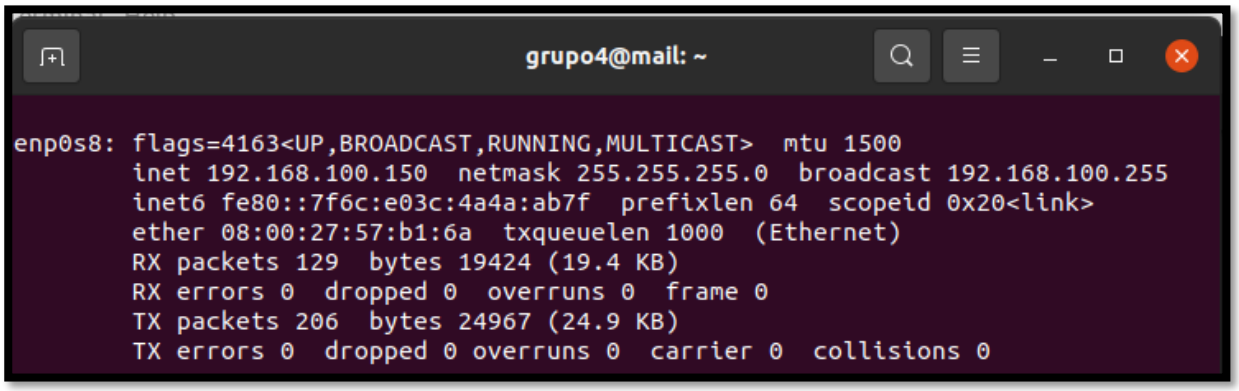
```
def ini():
    host = input("Server Address: ")
    port = int(input("Port: "))
    return host, port

def crearSocket():
    s = socket(AF_INET, SOCK_STREAM)
    return s

def conectarse (host, port, s):
    s.connect((host, port))

def intentoConexion(host, port, s):

        while True:
            print("\nTrying to connect to:", host + ":" + str(port))
            try:
                conectarse(host, port, s)
                break
            except:
                print("There is no Server at:", host + ":" + str(port))
                print("Trying again in 5 Seconds\n")
                time.sleep(5)

def enviar(s):

    while True:

        global exit

        try:
            msg = input("")
            msg = client +": " + msg
            if msg == client+": salir":
                exit = True
                msg = "The "+client+" Client is gone"
                s.send(msg.encode("UTF-8"))
                s.close
                break
            else:
                s.send(msg.encode("UTF-8"))
                start_new_thread(recibir,(s,))
        except:
            print("Something happend\n")
            print("Trying in 5 seg")
```

```
            time.sleep(5)

def recibir(s):
    while True:

        try:
          reply = s.recv(2048)
          print(reply.decode("UTF-8"))
          break
        except:
            print("Cant recieve response\n")
            print("Trying in 5 seg")
            time.sleep(5)

def recibirEspecial(s):
    global client  #VARIABLE GLOBAL
    client = s.recv(2048).decode("UTF-8")


exit=False      # Si el cliente envia salir, exit se pone en true y el
                # el programa termina
client = ""
def main():

    host, port = ini()
    s = crearSocket()
    intentoConexion(host,port,s)
    recibirEspecial(s)
    print("\nConnection To Server Established!\nThe server is:", host+":"+str(port)+"\n")
    print("Write your messages\n")
    start_new_thread(enviar,(s,))

    while exit!=True:   # Necesarios para que los hilos no mueran
        pass

    print("\nSorry something went wrong! You have lost connection to the server.:(")
    print("Closing the windows in 5 seg")
    time.sleep(10)

main()
```

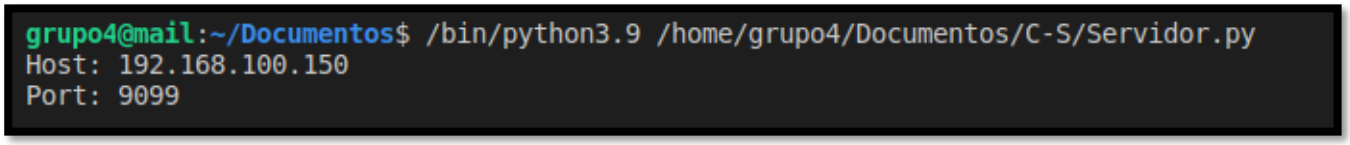
b) Se pueden usar las máquinas en donde Ubuntu sea el servidor y Windows 10 el cliente.

Si se puede usar

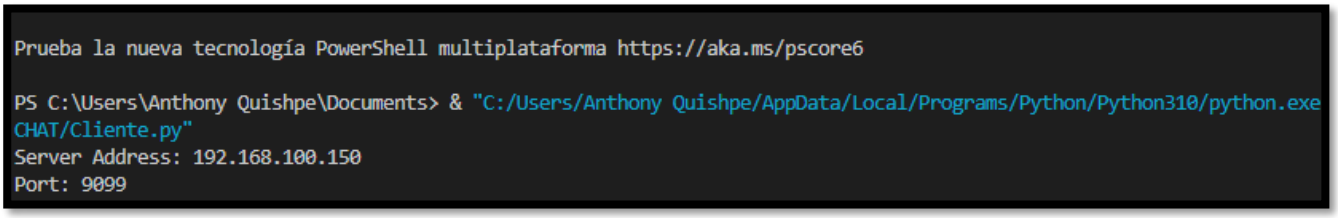
Verificamos la dirección IP de nuestro servidor a la cual se va conectar el cliente desde Windows 10



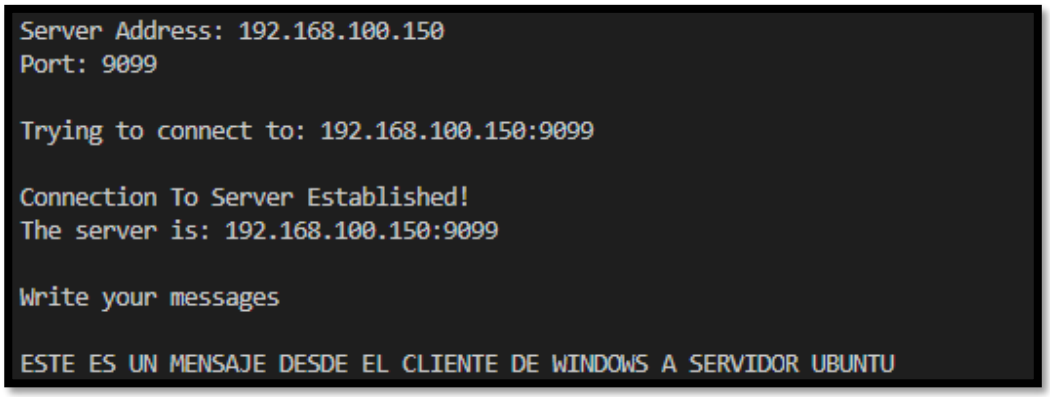
Ejecutamos el servidor con la dirección IP e ingresamos la dirección por el puerto que se va ingresar en este caso por el puerto 9099



Ejecutamos al cliente con la dirección IP del Servidor y el puerto por donde se va ingresar en este caso 9099



Desde el cliente escribimos
ESTE ES UN MENSAJE ES ENVIADO DESDE EL CLIENTE WINDOWS AL SERVIDOR UBUNTU



Respondemos desde el Servidor con el siguiente mensaje
MENSAJE RECIBIDO DESDE EL SERVIDOR UBUNTU

```
grupo4@mail:~/Documentos$ /bin/python3.9 /home/grupo4/Documentos/C-S/Servidor.py
Host: 192.168.100.150
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE ANY MESS
AGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:54740

Cliente 1: ESTE ES UN MENSAJE DESDE EL CLIENTE DE WINDOWS A SERVIDOR UBUNTU
MENSAJE RECIBIDO DESDE UBUNTU
```

Y verificamos que en el cliente Windows que el servidor Ubuntu respondió

```
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099

Write your messages

ESTE ES UN MENSAJE DESDE EL CLIENTE DE WINDOWS A SERVIDOR UBUNTU
Servidor: MENSAJE RECIBIDO DESDE UBUNTU
```

c) Utilizar una herramienta de gestión como Putty para acceder al servidor y ejecutar.

Ingresamos a la herramienta Putty, con la dirección IP de nuestro servidor, en este caso el servidor esta desde Ubuntu, por los cual se ingresa las credenciales de Ubuntu.

Una vez ingresado, nos dirigimos al directorio en donde esto código de servidor para ejecutar desde la consola de Putty.

```
grupo4@mail: ~/Documentos/C-S
login as: grupo4
grupo4@192.168.100.150's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.13.0-48-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

Se pueden aplicar 25 actualizaciones de forma inmediata.
22 de estas son actualizaciones de seguridad estándares.
Para ver estas actualizaciones adicionales ejecute: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Mon Jun 13 19:56:03 2022 from 192.168.56.1
grupo4@mail:~$ cd /home/grupo4/Documentos/C-S
```

Una vez ingresado al Servidor ingresamos la dirección IP de nuestro servidor y el puerto 9099.

```
grupo4@mail:~/Documentos/C-S$ python3 Servidor.py
Host: 192.168.100.150
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE AN
Y MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:54114
```

Iniciamos de la misma manera el cliente en Windows con la dirección IP del servidor y el mismo puerto para que exista conexión y vemos que se estableció la conexión.

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/Local/Programs/Python/Python310/pyth
on.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Cliente.py"
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099
```

Mandamos el primer mensaje desde el cliente al servidor.

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/Local/Programs/Python/Python310/pyth
on.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Cliente.py"
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099

Write your messages

HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTTY
```

Verificamos mediante un mensaje en el servidor el mensaje enviado desde el cliente.

```
grupo4@mail:~/Documentos/C-S$ python3 Servidor.py
Host: 192.168.100.150
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE AN
Y MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:54114

Cliente 1: HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTT
Y
HOLA Saludos desde el servidor al cliente WINDOWS
```

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppDat
on.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Cliente.py"
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099

Write your messages

HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTTY
Servidor: HOLA Saludos desde el servidor al cliente WINDOWS
```

Servidor abierto desde Putty para el paso de mensajes

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/L
on.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Cliente.py"
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099

Write your messages

HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTTY
Servidor: HOLA Saludos desde el servidor al cliente WINDOWS
```

```
grupo4@mail:~/Documentos/C-S$ python3 Servidor.py
Host: 192.168.100.150
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE AN
Y MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:54114

Cliente 1: HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTT
Y
HOLA Saludos desde el servidor al cliente WINDOWS
```

RESULTADOS OBTENIDOS:

EJECUCION DEL SERVIDOR WINDOWS A CLIENTE UBUTU.

Verificamos la dirección IP de nuestro servidor a la cual se va conectar el cliente desde Ubuntu.

Ejecutamos el servidor con la dirección IP e ingresamos la dirección por el puerto que se va ingresar en este caso por el puerto 9099

Ejecutamos al cliente con la dirección IP del Servidor y el puerto por donde se va ingresar en este caso 9099

Desde el cliente escribimos
HOLA ESTE MENSAJE ES ENVIADO DESDE EL CLIENTE UBUNTU

```
grupo4@mail:~/Documentos$ /bin/python3.9 /home/grupo4/Documentos/C-S/Cliente.py
Server Address: 192.168.100.143
Port: 9099

Trying to connect to: 192.168.100.143:9099

Connection To Server Established!
The server is: 192.168.100.143:9099

Write your messages

HOLA ESTE MENSAJE ES ENVIADO DESDE EL CLIENTE UBUNTU
```

Verificamos desde el servidor es decir Windows el mensaje

```
PROBLEMAS   SALIDA   CONSOLA DE DEPURACIÓN   TERMINAL   JUPYTER

Host: 192.168.100.143
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE ANY MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:59647

Cliente 1: HOLA ESTE MENSAJE ES ENVIADO DESDE EL CLIENTE UBUNTU
```

Respondemos desde el Servidor con el siguiente mensaje
MENSAJE RECIBIDO Y VERIFICADO DESDE EL SERVIDOR WINDOWS

```
Host: 192.168.100.143
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE ANY MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:59647

Cliente 1: HOLA ESTE MENSAJE ES ENVIADO DESDE EL CLIENTE UBUNTU
MENSAJE RECIBIDO Y VERIFICADO DESDE EL SERVIDOR WINDOWS
```

Verificamos desde el cliente Ubuntu el mensaje enviado desde el servidor Windows

```
grupo4@mail:~/Documentos$ /bin/python3.9 /home/grupo4/Documentos/C-S/Cliente.py
Server Address: 192.168.100.143
Port: 9099

Trying to connect to: 192.168.100.143:9099

Connection To Server Established!
The server is: 192.168.100.143:9099

Write your messages

HOLA ESTE MENSAJE ES ENVIADO DESDE EL CLIENTE UBUNTU
Servidor: MENSAJE RECIBIDO Y VERIFICADO DESDE EL SERVIDOR WINDOWS
```

Servidor abierto desde Putty para el paso de mensajes

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/L
on.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Cliente.py"
Server Address: 192.168.100.150
Port: 9099

Trying to connect to: 192.168.100.150:9099

Connection To Server Established!
The server is: 192.168.100.150:9099

Write your messages

HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTTY
Servidor: HOLA Saludos desde el servidor al cliente WINDOWS
```

```
grupo4@mail:~/Documentos/C-S$ python3 Servidor.py
Host: 192.168.100.150
Port: 9099

W A R N I N G : THE SERVER IS A SLAVE. DON'T WRITE IF THE SERVER DOESN'T HAVE AN
Y MESSAGE TO RESPONSE

Waiting for clients

Established Connection.
The client is: 192.168.100.143:54114

Cliente 1: HOLA SALUDOS DESDE EL CLIENTE WINDOWS A SERVIDOR UBUNTU MEDIANTE PUTT
Y
HOLA Saludos desde el servidor al cliente WINDOWS
```

CONCLUSIONES:

1. La red cliente servidor ha sido una red de comunicación en la cual el cliente está conectado con el servidor y se puede centralizar los diversos recursos y aplicaciones con que se cuenta cada vez que son solicitados.
2. Con este laboratorio se puede concluir que establecer una conexión para el paso de mensajes del servidor cliente, no es tan complicado, se ha logrado realizar las conexiones solicitadas sin inconvenientes y recibiendo los mensajes tanto al servidor, como al cliente.
3. Se realizaron pruebas de conectividad entre las máquinas virtuales y la máquina anfitrión, antes de realizar cualquier otra actividad, y todo fue exitoso.
4. Esta tecnología nos proporciona el acceso transparente a las aplicaciones, datos, servicios de cómputo o cualquier otro servicio del grupo de trabajo y/o, a través de la organización, en múltiples plataformas.

RECOMENDACIONES:

1. Es recomendable antes de iniciar la práctica, consultar en fuentes de información acerca de la arquitectura cliente – servidor, en máquinas virtuales con la implementación de un lenguaje de programación ya sea Python, C, C++, entre otros.
2. Es necesario conocer acerca de los sockets que se han implementado.
3. Para la programación de los códigos del cliente como del servidor es importante conocer las librerías que permiten crear la conexión para el paso de mensajes, mediante las direcciones IP y puerto por donde se va ingresar para la respectiva conexión.
4. Antes de iniciar la práctica, es recomendable revisar las configuraciones de red de la máquina virtual ya que, si se tiene conectado mediante, una dirección IP estática, el cliente no va responder a dicha dirección asignada, por lo que es recomendable tener en DHCP para así establecer conexión a la misma red.

ELABORADO POR:

F: .....

CONSTANTE ROBERSON
ESTUDIANTE

F: .....

QUISHPE ANTHONY
ESTUDIANTE

F: .....

TIPANTIZA NAYELI
ESTUDIANTE