

INFORME DE PRÁCTICA DE LABORATORIO

DEPARTAMENTO:	CIENCIAS DE LA COMPUTACIÓN	CARRERA:	INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN		
ASIGNATURA:	APLICACIONES DE SISTEMAS OPERATIVOS	PERIODO LECTIVO:	MAYO 2023- SEP 2023	NIVEL:	5to
DOCENTE:	ING. ANDREA LÓPEZ	NRC:	10035	PRÁCTICA N°:	2.1
TEMA DE LA PRÁCTICA:	ARQUITECTURA CLIENTE - SERVIDOR				

OBJETIVOS:

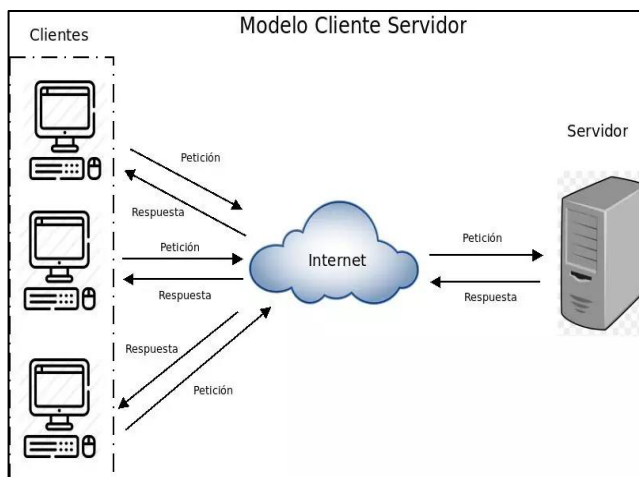
1. Implementar un modelo de paso de mensajes en un arquitectura cliente-servidor (se sugiere usar los siguientes lenguajes: C, C++, Python)
2. Se pueden usar las máquinas en donde Ubuntu sea el servidor y Windows 10 el cliente. (opcional)
3. Utilizar una herramienta de gestión como Putty para acceder al servidor y ejecutar.

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. Llamados 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.



• Lenguaje Python

Python 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”.

INFORME DE PRÁCTICA DE LABORATORIO

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.

De optimización.

MATERIALES:

EQUIPOS:

- PC

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.

```
grupo4@mail: ~  
grupo4@mail:~$ sudo apt update
```

```
grupo4@mail: ~  
grupo4@mail:~$ sudo apt install software-properties-common
```

```
grupo4@mail: ~  
grupo4@mail:~$ sudo apt install python3.9
```

Instalación de Visual Studio Code en Ubuntu

```
grupo4@mail: ~  
grupo4@mail:~$ python3.9 --version  
Python 3.9.13  
grupo4@mail:~$
```

Visual Studio Code Instalado en máquina virtual Ubuntu

```
grupo4@mail: ~  
grupo4@mail:~$ sudo snap install --classic code
```

Creación de código en Python para el Servidor y Cliente



ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

INFORME DE PRÁCTICA DE LABORATORIO

```
1 # Servidor.py
2 """
3 Funciones:
4 1. init() -- Pide al usuario el host y puerto.
5 2. crearSocket() -- Retorna un nuevo socket siguiendo el esquema del
6    protocolo TCP
7 3. ligarSocket(host, port) -- Intenta ligar un socket a los parametros
8    host y port
9 4. conexiones() -- Espera por la conexi3n de clientes externos. Retorna
10    la direcci3n del cliente en una tupla
11 5. enviar(conn) -- Envia un mensaje codificado a la direcci3n del cliente 1
12 6. enviar2(conn) -- Envia un mensaje codificado a la direcci3n del cliente 2
13 """
14
15 import socket
16
17 def init():
18     host = input("Ingrese el host: ")
19     port = input("Ingrese el puerto: ")
20
21 def crearSocket():
22     s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
23
24 def ligarSocket(host, port):
25     s.bind((host, port))
26
27 def conexiones():
28     s.listen(5)
29     while True:
30         conn, addr = s.accept()
31         print('Conexi3n de %s' % addr)
32         enviar(conn)
33         enviar2(conn)
34
35 def enviar(conn):
36     msg = input("Mensaje para el cliente 1: ")
37     conn.send(msg.encode())
38
39 def enviar2(conn):
40     msg = input("Mensaje para el cliente 2: ")
41     conn.send(msg.encode())
42
43 if __name__ == '__main__':
44     init()
45     crearSocket()
46     ligarSocket(host, port)
47     conexiones()
48     enviar(conn)
49     enviar2(conn)
```

Seridor

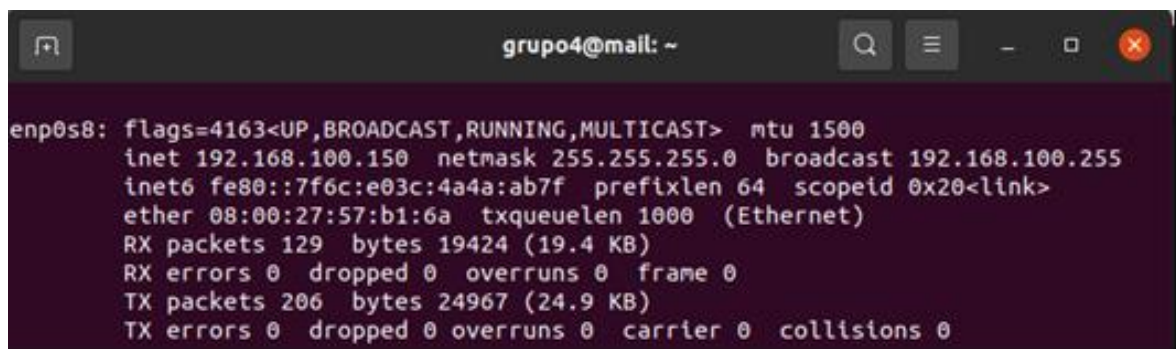
INFORME DE PRÁCTICA DE LABORATORIO

```
1 import socket
2
3 def run_server():
4     # Crear un socket TCP/IP
5     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7     # Asignar una dirección y un puerto al socket
8     server_address = ('localhost', 5000)
9     sock.bind(server_address)
10
11     # Escuchar conexiones entrantes
12     sock.listen(1)
13
14     print('El servidor está en funcionamiento y esperando conexiones...')
15
16     while True:
17         # Esperar una conexión entrante
18         connection, client_address = sock.accept()
19
20         try:
21             print('Conexión establecida desde', client_address)
22
23             # Recibir datos del cliente
24             data = connection.recv(1024)
25             print('Mensaje recibido:', data.decode())
26
27             # Ingresar el mensaje a enviar al cliente
28             message = input("Ingrese el mensaje a enviar al cliente: ")
29
30             # Enviar una respuesta al cliente
31             connection.sendall(message.encode())
32
33         finally:
34             # Cerrar la conexión
35             connection.close()
36
37 if __name__ == '__main__':
38     run_server()
39
```

Client

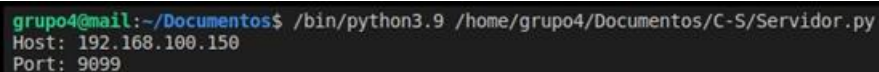
INFORME DE PRÁCTICA DE LABORATORIO

```
1  import socket
2
3  def run_client():
4      # Crear un socket TCP/IP
5      sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
6
7      # Conectar al servidor y puerto especificados
8      server_address = ('localhost', 5000)
9      sock.connect(server_address)
10
11     try:
12         # Ingresar el mensaje a enviar
13         message = input("Ingrese el mensaje a enviar: ")
14
15         # Enviar datos al servidor
16         sock.sendall(message.encode())
17
18         # Recibir respuesta del servidor
19         data = sock.recv(1024)
20         print('Respuesta del servidor:', data.decode())
21
22     finally:
23         # Cerrar la conexión
24         sock.close()
25
26 if __name__ == '__main__':
27     run_client()
28
```



```
grupo4@mail: ~
enp0s8: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.100.150  netmask 255.255.255.0  broadcast 192.168.100.255
        inet6 fe80::7f6c:e03c:4a4a:ab7f  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:57:b1:6a  txqueuelen 1000  (Ethernet)
        RX packets 129  bytes 19424 (19.4 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 206  bytes 24967 (24.9 KB)
        TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```

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



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



INFORME DE PRÁCTICA DE LABORATORIO

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

```
Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6  
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/Local/Programs/Python/Python310/python.exe  
CHAT/Cliente.py"  
Server Address: 192.168.100.150  
Port: 9099
```

Desde el cliente escribimos

ESTE ES UN MENSAJE ES ENVIADO DESDE EL CLIENTE WINDOWS AL SERVIDOR UBUNTU

```
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
```

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  
os que en el cliente Windows que el servidor Ubuntu respondió  
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

INFORME DE PRÁCTICA DE LABORATORIO

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

b) 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 lo cual se ingresa las credenciales de Ubuntu.

Una vez ingresado, nos dirigimos al directorio en donde está el 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

WARNING: 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: 192.168.100.143:54114
```

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.



INFORME DE PRÁCTICA DE LABORATORIO

```
PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/Local/Programs/Python/Python310/python.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.

```
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 PUTTY
Y
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/Local/Programs/Python/Python310/python.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
```

RESULTADOS OBTENIDOS:

```
C:\> Símbolo del sistema

Adaptador de LAN inalámbrica Wi-Fi:

    Sufijo DNS específico para la conexión. . . :
    Vínculo: dirección IPv6 local. . . : fe80::16f:d6c4:a7c5:f9cc%17
    Dirección IPv4. . . . . : 192.168.100.143
    Máscara de subred. . . . . : 255.255.255.0
    Puerta de enlace predeterminada. . . . : fe80::1%17
    192.168.100.1

PS C:\Users\Anthony Quishpe\Documents> & "C:/Users/Anthony Quishpe/AppData/Local/Programs/Python/Python310/python.exe" "c:/Users/Anthony Quishpe/Documents/CHAT/Servidor.py"
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
```




INFORME DE PRÁCTICA DE LABORATORIO

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

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
█
```

```
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

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

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  JUPYTER

Host: 192.168.100.143
Port: 9099

WARNING : 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
█
```

```
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
█
```

```
PS C:\Users\Anthony Qulshpa\Documents> & "C:\Users\Anthony Qulshpa\AppData\Local\Microsoft\Windows\Terminal\Profiles\Ubuntu\bin\python3.9" /home/grupo4/Documentos/C-S/Server.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

WARNING : 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
```

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

INFORME DE PRÁCTICA DE LABORATORIO

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:

CEVALLOS JUAN

ESTUDIANTE



F:

SANGOQUIZA DAVID

ESTUDIANTE



F:

DR. PANCHITO

ESTUDIANTE



F:

PACHACAMA FREDDY

ESTUDIANTE