Trabajo TCP - Calculadora Remota

Yessica Tatiana Gualteros Ayala - 1123530064
Leidi Tatiana Nieto Goyeneche - 1048823584

Luz Marina Santos Jaimes
Docente

Universidad de Pamplona
Facultad de Ingeniería y Arquitectura
Redes
Pamplona, Colombia
2025

Trabajo TCP - Calculadora Remota

Objetivo General: Reforzar los conocimientos del protocolo TCP mediante el desarrollo de una calculadora remota que utilice sockets tcp.

Descripción: Desarrollar una aplicación C/S empleando programación en sockets tcp con cualquier lenguaje de programación. Los procesos servidor y cliente deben estar en la misma red en máquinas diferentes.

Servidor: Escuchando permanentemente conexiones de clientes que solicitan realizar una operación matemática entre: +, -, * y /. El servidor recibe los operandos y operador y procede a realizar la operación, y a continuación devuelve el resultado al proceso cliente.

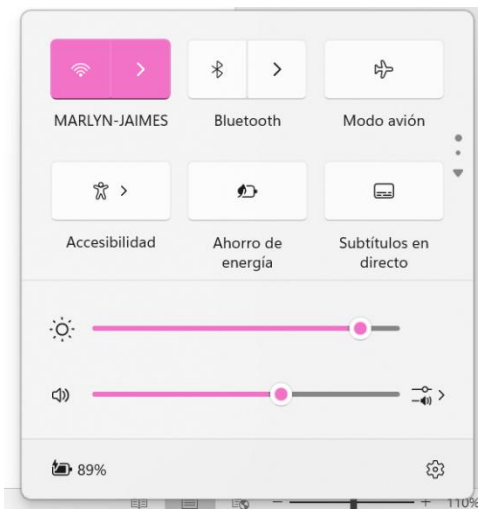
Cliente: El cliente solicita al usuario dos números (operandos) y la operación (+, -, * y /). A continuación, se conecta con el servidor para enviarle la información necesaria para que procese y devuelva el resultado. Una vez devuelto el resultado por parte del servidor se muestra en pantalla del cliente y se cierra la conexión con el servidor.

Entregable del trabajo:

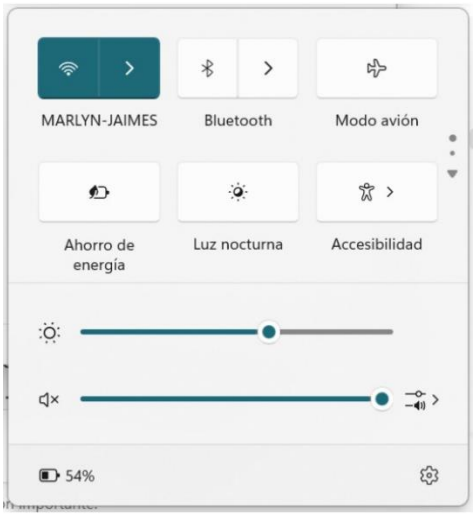
1. Código fuente completo, documentado y funcional (subido en GitHub).
2. Archivo readme.txt con explicación del funcionamiento de la calculadora remota.
3. Documento wireshark con captura de mensajes intercambiados entre cliente y servidor de la captura remota.
4. Enviar al correo sólo enlace del trabajo a GitHub con los integrantes del mismo.

Configuración de Máquinas

1. Conecta ambas computadoras a la misma red local (mismo Wi-Fi o LAN).



Cliente



Servidor

2. Verifica que tienen Python instalado.

```
Microsoft Windows [Versión 10.0.26100.6899]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>python --version
Python 3.11.9
```

Cliente

```
Microsoft Windows [Versión 10.0.26100.4349]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>python --version
Python 3.11.9
```

Servidor

3. En la PC Servidor, abre una consola y ejecuta:

```
Microsoft Windows [Versión 10.0.26100.4349]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\tatia>ipconfig

Configuración IP de Windows


Adaptador de Ethernet Ethernet 3:

   Sufijo DNS específico para la conexión. . :
   Vínculo: dirección IPv6 local. . . : fe80::f328:63a8:b6ee:ab59%4
   Dirección IPv4. . . . . . . . . . . . : 192.168.56.1
   Máscara de subred . . . . . . . . . . : 255.255.255.0
   Puerta de enlace predeterminada . . . . . :
```

4. Verifica que ambas computadoras pueden comunicarse**:**

   - Desde la PC cliente:

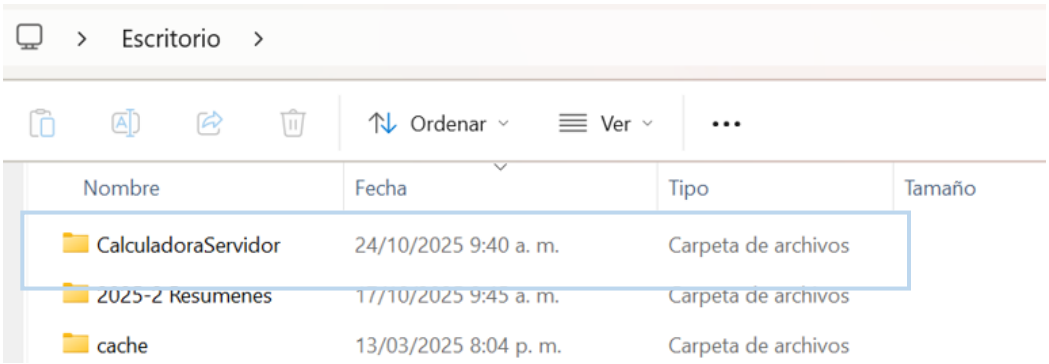
```
C:\>ping 192.168.1.8

Haciendo ping a 192.168.1.8 con 32 bytes de datos:
Respuesta desde 192.168.1.8: bytes=32 tiempo=41ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=605ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=96ms TTL=64
Respuesta desde 192.168.1.8: bytes=32 tiempo=233ms TTL=64

Estadísticas de ping para 192.168.1.8:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 41ms, Máximo = 605ms, Media = 243ms
```

   - Desde la PC servidor:

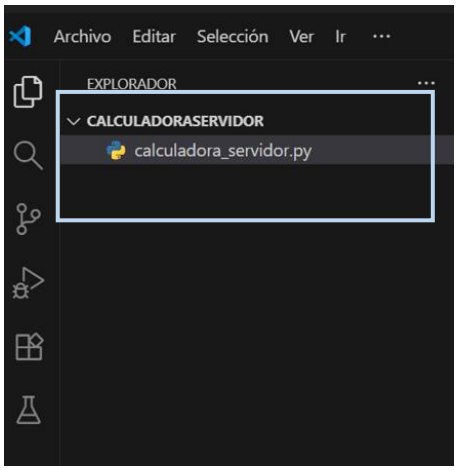
```
C:\Windows\System32>ping 192.168.1.10

Haciendo ping a 192.168.1.10 con 32 bytes de datos:
Respuesta desde 192.168.1.10: bytes=32 tiempo=166ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=234ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=3ms TTL=128
Respuesta desde 192.168.1.10: bytes=32 tiempo=7ms TTL=128

Estadísticas de ping para 192.168.1.10:
    Paquetes: enviados = 4, recibidos = 4, perdidos = 0
    (0% perdidos),
Tiempos aproximados de ida y vuelta en milisegundos:
    Mínimo = 3ms, Máximo = 234ms, Media = 102ms

C:\Windows\System32>
```

5. Preparar la pc servidor:

   - En el Escritorio, crea una carpeta llamada “CalculadoraServidor”:



- Dentro de esa carpeta, abre Visual Studio Code.



- Código:

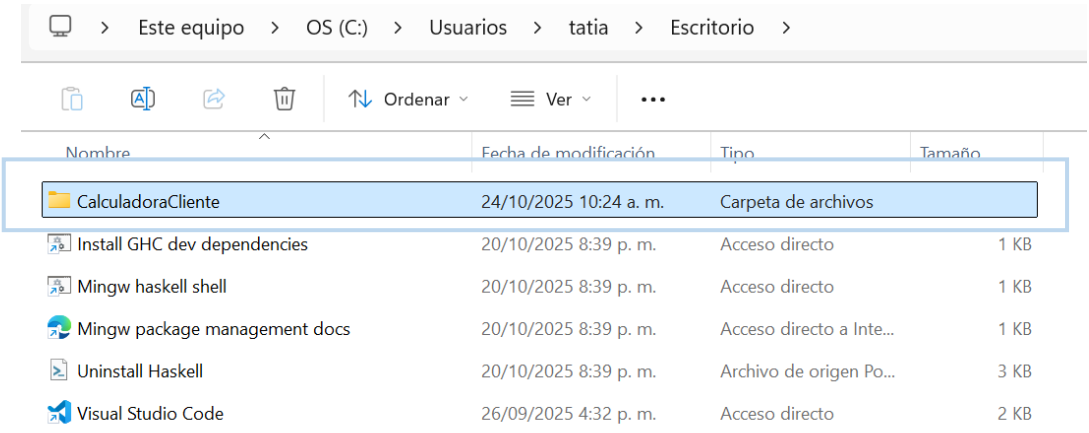
calculadora_servidor.py

```python
import socket # Importamos la librería necesaria para manejar sockets

# CONFIGURACIÓN DEL SERVIDOR
HOST = "0.0.0.0"  # Escucha en todas las interfaces de la red disponible.
PORT = 5000       # Puerto donde escuchará el servidor

# CREACIÓN DEL SOCKET TCP

# AF_INET -> Familia de direcciones IPv4
# SOCK_STREAM -> Tipo de socket TCP (orientado a conexión)
servidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
servidor.bind((HOST, PORT))# Asociamos el socket con la dirección IP y el puerto definidos
servidor.listen(1) # Ponemos el socket en modo escucha, máximo 1 cliente en cola
print(f"Servidor escuchando en {HOST}:{PORT}...")

# CICLO PRINCIPAL DEL SERVIDOR

# Este ciclo permite atender múltiples clientes uno tras otro
while True:
    # Esperamos a que un cliente se conecte
    conn, addr = servidor.accept()
    print(f"Conexión establecida con: {addr}")

    data = conn.recv(1024).decode() # Recibimos los datos enviados por el cliente (máximo 1024 bytes
    # Si no se recibe nada, se sale del bucle
    if not data:
        break
```

```
28
29      # Dividimos la cadena recibida (por ejemplo: "5 + 3")
30      partes = data.split()
31      num1 = float(partes[0])   # Primer número
32      operador = partes[1]      # Operador (+, -, *, /)
33      num2 = float(partes[2])   # Segundo número
34
35
36      # PROCESAMIENTO DE LA OPERACIÓN
37
38      if operador == '+':
39          resultado = num1 + num2
40      elif operador == '-':
41          resultado = num1 - num2
42      elif operador == '*':
43          resultado = num1 * num2
44      elif operador == '/':
45          # Evitar división por cero
46          resultado = num1 / num2 if num2 != 0 else "Error: división por cero"
47      else:
48          resultado = "Operador inválido"
49
50      # ENVÍO DEL RESULTADO AL CLIENTE
51
52      conn.send(str(resultado).encode())
53      print(f"Resultado enviado al cliente: {resultado}")
54      conn.close()  # Cerramos la conexión con ese cliente
```

6. Preparar la pc cliente:

   - En el Escritorio, crea una carpeta llamada “CalculadoraCliente”:



- Dentro de esa carpeta, abre Visual Studio Code.

calculadora_cliente.py

```python
import socket                       # Librería estándar 'socket' para crear sockets TCP/IP.
import tkinter as tk                # Importa tkinter para crear la interfaz gráfica (alias 'tk' para abreviar).
from tkinter import messagebox      # Importa el submódulo messagebox para mostrar diálogos (errores, avisos).

# -------------------------
# Configuración de red
# -------------------------
SERVER_IP = "192.168.1.8"          # IP del servidor.
PORT = 5000                        # Puerto TCP donde el servidor está escuchando (debe coincidir con el servidor).

# -------------------------
# Función que envía la operación y recibe el resultado
# -------------------------
def calcular():
    """
    6. Función llamada cuando el usuario hace clic en "Calcular".
       Reúne los datos de la interfaz, valida, abre conexión TCP, envía el mensaje,
       recibe el resultado y lo muestra en la etiqueta de resultado.
    """

    num1 = entrada_num1.get()       # Lee el texto actual del campo de entrada 'entrada_num1'.
    num2 = entrada_num2.get()       # Lee el texto actual del campo de entrada 'entrada_num2'.
    operador = operacion.get()      # Obtiene el operador seleccionado en el OptionMenu ('+', '-', '*', '/').

    #Validación básica: asegurar que los campos no estén vacíos.

    if not num1 or not num2:
        messagebox.showwarning("Datos incompletos", "Debe ingresar ambos números.")
        return                      # Sale de la función si faltan datos.

    # Validación de formato: intentar convertir a float para asegurarnos que sean números.

    try:
        float(num1)                 #Intento de conversión para validar; no guardamos el valor convertido aún.
        float(num2)
    except ValueError:

        # Si la conversión falla lanzamos un aviso al usuario y no continuamos.
        messagebox.showwarning("Valor inválido", "Los valores deben ser números (ej: 3.5 o 7).")
        return

    # Construimos el mensaje con el formato que espera el servidor: "num1 operador num2"
    mensaje = f"{num1} {operador} {num2}"

    # Intento de conexión y comunicación dentro de un bloque try/except para capturar errores de red.
    try:
        cliente = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Crea un socket IPv4 (AF_INET) orientado a conexión/TCP (SOCK_STREAM).

        cliente.connect((SERVER_IP, PORT))
        # Inicia la conexión TCP con el servidor usando la IP y el puerto configurados.
        #     Si el servidor no responde o la IP es incorrecta, esto lanzará una excepción.
```

```
        cliente.send(mensaje.encode())
        # Envía el mensaje al servidor codificado a bytes (UTF-8 por defecto).
        #     .encode() convierte la cadena a bytes para la transmisión.

        resultado = cliente.recv(1024).decode()
        # Recibe hasta 1024 bytes de respuesta del servidor y los decodifica a string.
        #     Asumimos que el servidor envía una respuesta corta (resultado de la operación).

        etiqueta_resultado.config(text=f"Resultado: {resultado}")
        # Actualiza la etiqueta en la interfaz con el resultado recibido.

        cliente.close()
        # Cierra el socket para liberar recursos y terminar la conexión TCP.
    except Exception as e:
        # Si ocurre cualquier excepción (conexión rechazada, timeout, etc.), mostramos un error.
        messagebox.showerror("Error de conexión", f"No se pudo conectar al servidor.\n\nDetalle: {e}")

# -------------------------
# Construcción de la interfaz gráfica
# -------------------------
ventana = tk.Tk()                   # Crea la ventana principal de la aplicación.
ventana.title("Calculadora Remota TCP")  # Título de la ventana.
ventana.geometry("320x280")         # Tamaño inicial de la ventana.
ventana.resizable(False, False)     # Evita que el usuario cambie el tamaño.
ventana.configure(bg="#F3E6F8")
```

```
# Etiqueta y campo para el primer número
tk.Label(ventana, text="Número 1:", bg="#F3E6F8", anchor="w").pack(padx=10, pady=(12,2), fill="x")
entrada_num1 = tk.Entry(ventana)    # Campo de entrada para el primer número.
entrada_num1.pack(padx=10, pady=(0,8), fill="x")

# Etiqueta y campo para el segundo número
tk.Label(ventana, text="Número 2:", bg="#F3E6F8", anchor="w").pack(padx=10, pady=(6,2), fill="x")
entrada_num2 = tk.Entry(ventana)    # Campo de entrada para el segundo número.
entrada_num2.pack(padx=10, pady=(0,8), fill="x")

# Etiqueta y menú para seleccionar la operación
tk.Label(ventana, text="Operación:", anchor="w").pack(padx=10, pady=(6,2), fill="x")
operacion = tk.StringVar(value="+") # Variable de control que guarda la opción seleccionada; valor inicial '+'.
menu_operacion = tk.OptionMenu(ventana, operacion, "+", "-", "*", "/")

# OptionMenu crea un desplegable con las cuatro operaciones y lo enlaza a 'operacion'.
menu_operacion.pack(padx=10, pady=(0,8), fill="x")

# Botón principal que ejecuta la función 'calcular' al hacer clic
boton_calcular = tk.Button(ventana, text="Calcular", command=calcular)
boton_calcular.pack(padx=10, pady=(6,10), fill="x")
```

```
# Etiqueta del resultado
etiqueta_resultado = tk.Label(
    ventana,
    text="Resultado:",
    bg="#F3E6F8",
    font=("Arial", 10, "bold"),
    fg="black"
)
etiqueta_resultado.pack(padx=10, pady=(10,6), fill="x")

# Inicia el loop principal de la interfaz; la aplicación queda a la espera de eventos (clics, entradas).
ventana.mainloop()
```

7. Ejecutar:

   - En la cmd del servidor:

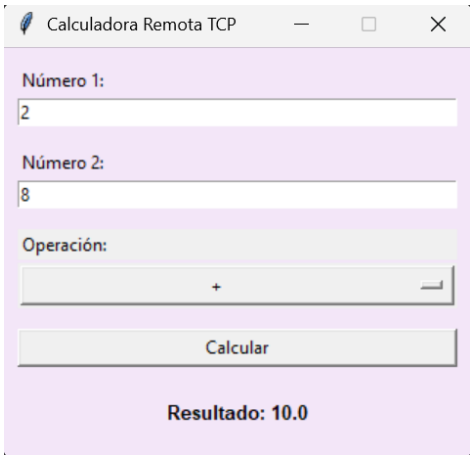
cd Escritorio\CalculadoraServidor
python calculadora_servidor.py

```
 Directorio de C:\Users\tatia\Desktop

24/10/2025  10:22 a. m.    <DIR>          .
20/10/2025  10:14 p. m.    <DIR>          ..
09/03/2025  06:44 p. m.    <DIR>          2024
22/06/2025  06:52 p. m.    <DIR>          2025
21/10/2025  10:42 a. m.    <DIR>          2025-2 Resumenes
13/03/2025  08:04 p. m.    <DIR>          cache
24/10/2025  09:45 a. m.    <DIR>          CalculadoraServidor
13/10/2025  03:47 p. m.    <DIR>          Chii
12/11/2024  06:14 a. m.             4.739 debug.log
25/09/2024  02:53 p. m.       109.732.304 draw.io-24.7.8-windows-installer.exe
06/10/2023  05:25 a. m.             1.107 Eclipse IDE for Java Developers - 2023-09.lnk
29/05/2025  06:16 p. m.       843.439.680 ideaIC-2025.1.1.1.exe
20/10/2025  08:36 p. m.               709 Install GHC dev dependencies.lnk
01/10/2024  09:15 p. m.    <DIR>          JUEGO 3 SEMESTRE IMAGENES
27/10/2023  06:11 p. m.             2.161 Liberar Espacio.lnk
21/02/2025  08:14 p. m.             2.361 Microsoft Edge.lnk
20/10/2025  08:36 p. m.               463 Mingw haskell shell.lnk
20/10/2025  08:36 p. m.               135 Mingw package management docs.url
15/09/2025  07:14 p. m.     1.182.435.288 OllamaSetup.exe
16/10/2024  02:02 p. m.       388.570.408 postgresql-17.0-1-windows-x64.exe
15/06/2025  02:55 p. m.    <DIR>          Tatiana Nieto
20/10/2025  08:36 p. m.             2.650 Uninstall Haskell.ps1
13/12/2024  01:55 p. m.             1.454 Visual Studio Code.lnk
              13 archivos  2.524.193.459 bytes
              10 dirs  195.582.320.640 bytes libres

C:\Users\tatia\Desktop>cd CalculadoraServidor

C:\Users\tatia\Desktop\CalculadoraServidor>python calculadora_servidor.py
Servidor escuchando en 0.0.0.0:5000...
```

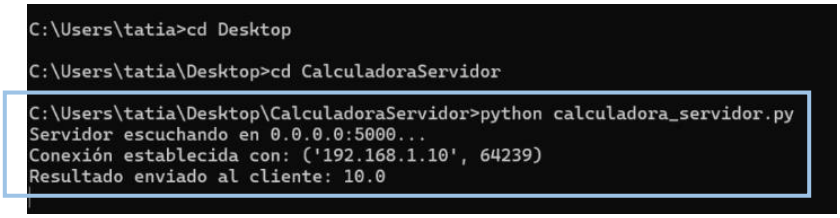
   - En la cmd del Cliente:

cd Escritorio\CalculadoraCliente
python calculadora_cliente.py

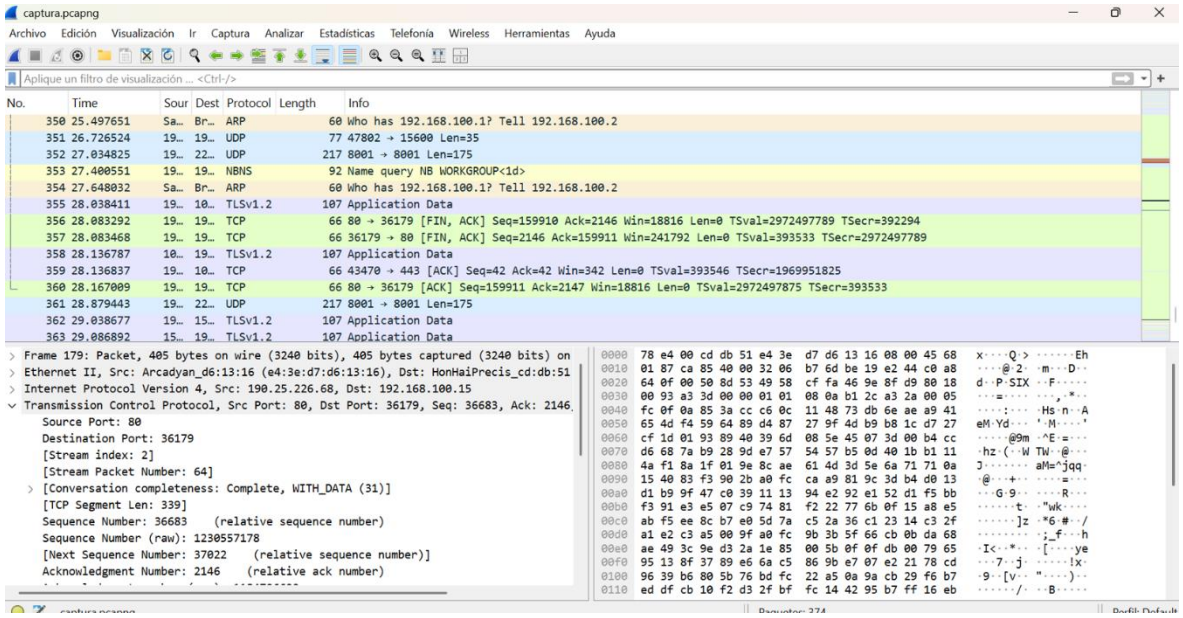
```
PS C:\Users\tatia> cd Escritorio
PS C:\Users\tatia\Escritorio> cd CalculadoraCliente
PS C:\Users\tatia\Escritorio\CalculadoraCliente> python calculadora_cliente.py
```



- CMD del pc Servidor

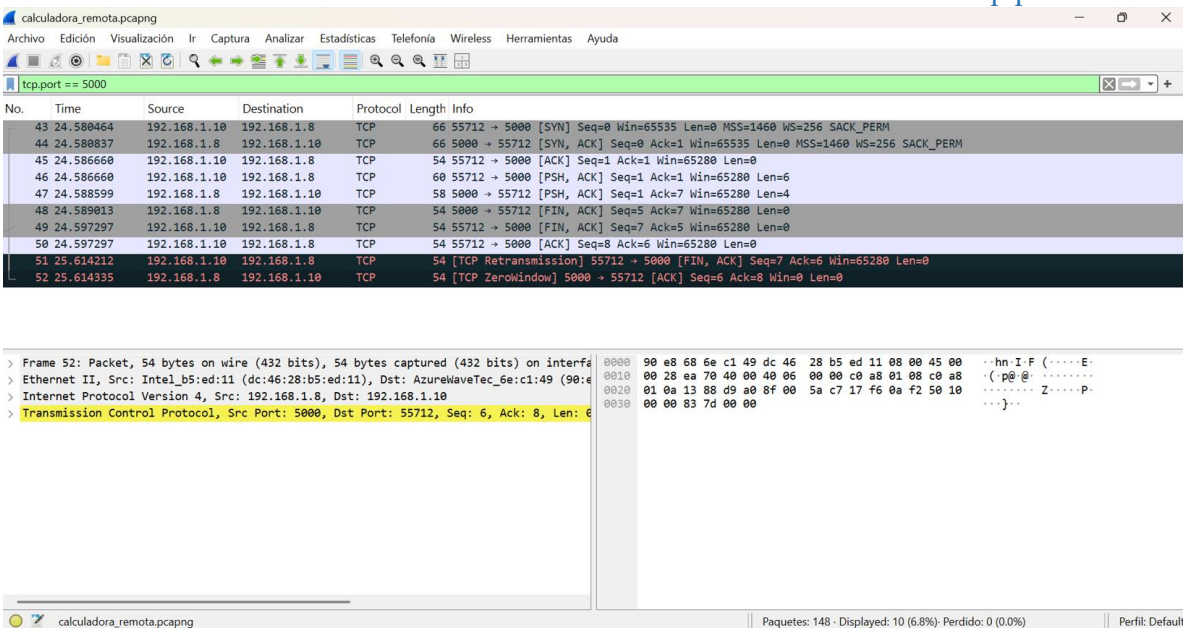


8. Captura En Wireshark



- Filtrar por el puerto 5000

Tcp port == 5000



9. Enlace del repositorio:

Link: https://github.com/Tatiana2412/CalculadoraTCP.git