

AVANCE DEL PROYECTO

GRUPO 4

¿Cómo funciona?

En archivo servidor.py podemos observar la implementacion de distintas funcionalidades a las cual el cliente tiene acceso, como tal cuando corremos nuestro archivo podemos observar que nos muestra un manual donde tenemos acceso a:

1. Listar todos los procesos
2. Iniciar un proceso específico
3. Matar un proceso
4. Monitorear un proceso específico
5. Salir

Mediante el uso del módulo socket podemos enviar señales o comandos a otros dispositivos a través de la misma dirección IP, compartiendo el mismo puerto que lo podemos ver como una llave. En pocas palabras el cliente puede interactuar con el servidor desde un dispositivo aparte, podemos ver cómo al iniciar un proceso sencillo como la calculadora, este se abre en el dispositivo servidor. Así como también podemos detener el proceso y monitorearlo.

Código Tarea 1

```
import socket
import psutil
import subprocess
from prettytable import PrettyTable
import time

def monitor_proceso(pid):
    """
    Monitorea un proceso dado su PID.
    Retorna información sobre el proceso, incluyendo nombre, estado, uso de CPU y memoria.

    :param pid: Identificador del proceso a monitorear
    :return: Cadena con la información del proceso o mensaje de error si no se encuentra
    """
    try:
        p = psutil.Process(pid)
        info = f"Monitoring process {pid}:\n"
        info += f"Name: {p.name()}\n"
        info += f"Status: {p.status()}\n"
        info += f"CPU Usage: {p.cpu_percent(interval=1.0)}%\n"
        info += f"Memory Usage: {p.memory_percent()}\n"
    except:
```

```
def listar_procesos():
    """
    Lista los procesos activos en el sistema.
    Devuelve una tabla con los PID, nombres, uso de CPU y memoria de cada proceso.

    :return: Cadena con la tabla de procesos en formato de texto
    """
    tabla = PrettyTable()
    tabla.field_names = ["PID", "Nombre", "Uso de CPU (%)", "Uso de Memoria (%)"]

    for proceso in psutil.process_iter(attrs=['pid', 'name', 'cpu_percent',
'memory_percent']):
        info = proceso.info
        tabla.add_row([info['pid'], info['name'], info['cpu_percent'],
info['memory_percent']])

    return tabla.get_string()
```

```
def iniciar_proceso(nombre_proceso):
    """
    Inicia un nuevo proceso dado su nombre.

    :param nombre_proceso: Nombre del proceso a iniciar
    :return: Mensaje indicando si el proceso se inició correctamente o si hubo un error
    """
    try:
        subprocess.Popen(nombre_proceso, shell=True)
        return f"Proceso {nombre_proceso} iniciado correctamente"
    except Exception as e:
        return f"Error al iniciar el proceso: {e}"
```

```
def matar_proceso(pid):
    """
    Termina un proceso dado su PID.

    :param pid: Identificador del proceso a terminar
    :return: Mensaje indicando si el proceso se cerró correctamente o si hubo un error
    """
    try:
        p = psutil.Process(pid)
        p.terminate()
        return f"Proceso con PID {pid} terminado correctamente."
    except psutil.NoSuchProcess:
        return f"No se encontró ningún proceso con PID {pid}."
    except psutil.AccessDenied:
        return f"Acceso denegado al intentar matar el proceso con PID {pid}."
    except Exception as e:
        return f"Error al matar el proceso con PID {pid}: {e}"
```

```

def manejo_cliente(conex, addr):
    """
    Maneja la conexión con un cliente, procesando sus solicitudes.

    :param conex: Objeto de conexión del socket
    :param addr: Dirección del cliente conectado
    """
    with conex:
        print('Conectado por', addr)
        try:
            while True:
                data = conex.recv(1024)
                if not data:
                    break
                comando = data.decode('utf-8').split()
                if comando[0] == "listar":
                    respuesta = listar_procesos()
                elif comando[0] == "iniciar":
                    respuesta = iniciar_proceso(comando[1])
                elif comando[0] == "matar":
                    respuesta = matar_proceso(int(comando[1]))
                elif comando[0] == "monitorear":
                    respuesta = monitor_proceso(int(comando[1]))
                else:
                    respuesta = "Comando no reconocido"
                conex.sendall(respuesta.encode('utf-8'))
        except Exception as e:
            print(f"Error en manejo de cliente {addr}: {e}")

```

```

def get_private_ipv4():
    """
    Obtiene la dirección IPv4 privada del servidor.

    :return: Dirección IP privada o mensaje de error si no se puede determinar
    """
    try:
        nombre_host = socket.gethostname()
        dir_IP = socket.gethostbyname(nombre_host)
        return dir_IP
    except socket.gaierror:
        return "Could not determine private IP address"

ip_privada = get_private_ipv4()
print(f"Server IP: {ip_privada}")
HOST = ip_privada
PORT = 65432

```

```

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()
    print(f"Server listening on {HOST}:{PORT}")
    while True:
        try:
            conex, addr = s.accept()
            manejo_cliente(conex, addr)
        except Exception as e:
            print(f"Error accepting connections: {e}")

if __name__ == "__main__":
    listar_procesos()
    mostrar_menu()

```

Código Tarea 2

```

import socket

def send_command(command):
    """
    Envía un comando al servidor a través de un socket.

    Args:
    command (str): El comando a enviar.
    """
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((HOST, PORT))
        s.sendall(command.encode('utf-8'))
        data = s.recv(4096)
        print(data.decode('utf-8'))

```

```

def mostrar_menu():
    """
    Muestra el menú de opciones y maneja la interacción del usuario.
    """
    salir = False
    while not salir:
        print("\n Gestión de Procesos")
        print("1. Listar procesos")
        print("2. Iniciar un proceso")
        print("3. Matar un proceso")
        print("4. Monitorear un proceso")
        print("5. Salir")

        opcion = input("Seleccione una opcion: ")

        if opcion == "5":
            print("Saliendo...")
            salir = True
        elif opcion == "1":
            send_command("listar") # Llama a la función para enviar el comando "listar"
        elif opcion == "2":
            nombre_proceso = input("Ingrese el nombre del proceso a iniciar: ")
            send_command(f"iniciar {nombre_proceso}") # Envía el comando "iniciar" con el
nombre del proceso
        elif opcion == "3":
            pid = input("Ingrese el PID del proceso a matar: ")
            send_command(f"matar {pid}") # Envía el comando "matar" con el PID del proceso
        elif opcion == "4":
            pid = input("Ingrese el PID del proceso a monitorear: ")
            send_command(f"monitorear {pid}") # Envía el comando "monitorear" con el PID
del proceso
        else:
            print("Opcion no valida, intente de nuevo")

```

```

if __name__ == "__main__":
    HOST = "10.74.80.170" # Asegurarse que este numero sea el mismo que el de server.py
    PORT = 65432
    mostrar_menu() # Llama a la función para mostrar el menú

```

Prueba

Computadora de Mafer

```
Símbolo del sistema - python servidor.py
11/02/2025 09:21 <DIR> ..
26/11/2024 14:14 <DIR> AvisaApi
11/03/2024 22:06 2.448 balenaEtcher.lnk
04/02/2025 11:22 1.090 Eclipse IDE for Java Developers - 2024-12.lnk
19/03/2024 20:02 <DIR> Fer-repositorio
11/02/2025 09:21 2.369 GitHub Desktop.lnk
20/01/2025 08:09 2.034 Greenfoot.lnk
26/12/2024 21:51 <DIR> librerias
28/12/2024 01:11 <DIR> Proyecto-Final-
06/02/2025 18:49 <DIR> So_final_project
04/02/2025 08:47 <DIR> Tarea 1 So
25/11/2024 19:05 <DIR> Technical-Documentation
18/11/2024 18:29 4.920.049.664 Windows10.iso
5 archivos 4.920.057.605 bytes
9 dirs 788.841.218.048 bytes libres

C:\Users\valen\Desktop>cd So_final_project
C:\Users\valen\Desktop\So_final_project>cd Final_project
C:\Users\valen\Desktop\So_final_project\Final_project>python servidor.py
Server IP: 192.168.56.1
Server listening on 192.168.56.1:65432
```

```
Administrador: Símbolo del sistema - python cliente.py

Gestión de Procesos
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opción: 1
```

PID	Nombre	Uso de CPU (%)	Uso de Memoria (%)
0	System Idle Process	0.0	6.0110657709777935e-05
4	System	0.0	0.009106764643031355
172		0.0	0.369199659653456
220	Registry	0.0	0.4116077286677044
704	smss.exe	0.0	0.008655934710208021
992	svchost.exe	0.0	0.07264372984226662
1116	csrss.exe	0.0	0.03934242547104965
1308	wininit.exe	0.0	0.04799836018125768
1324	XboxPcTray.exe	0.0	0.1909114488862547
1328	csrss.exe	0.0	0.04661581505393279
1408	winlogon.exe	0.0	0.08583801920956288
1416	services.exe	0.0	0.11586329273559695
1460	LsaIso.exe	0.0	0.026298412748027848
1484	lsass.exe	0.0	0.20951569744743098
1612	svchost.exe	0.0	0.2421257292549855
1636	fontdrvhost.exe	0.0	0.022601607298876503
1640	fontdrvhost.exe	0.0	0.05142466767071502
1684	WUDFHost.exe	0.0	0.06519000828625417
1772	svchost.exe	0.0	0.12518044468061254
1840	svchost.exe	0.0	0.08866322012102245
1864	WUDFHost.exe	0.0	0.04673603636935234
1960	svchost.exe	0.0	0.0386812082362421
1968	svchost.exe	0.0	0.0781739103515662
1984	XboxPcAppFT.exe	0.0	0.3077064568163532
2012	svchost.exe	0.0	0.07667114390882175
2040	svchost.exe	0.0	0.0759498160163044
2052	svchost.exe	0.0	0.07129124004379662
2064	svchost.exe	0.0	0.06185386678336149
2096	svchost.exe	0.0	0.12091258798321831
2120	svchost.exe	0.0	0.06212436474305549
2136	svchost.exe	0.0	0.06266536066244349
2148	ApplicationFrameHost.exe	0.0	0.29508321869729986
2188	svchost.exe	0.0	0.05388920463681591
2304	svchost.exe	0.0	0.05749584409940259

CA. Administrador: Símbolo del sistema - python cliente.py

2096	svchost.exe	0.0	0.12091258798321831
2120	svchost.exe	0.0	0.06212436474305549
2136	svchost.exe	0.0	0.06266536066244349
2148	ApplicationFrameHost.exe	0.0	0.29508321869729986
2188	svchost.exe	0.0	0.05388920463681591
2304	svchost.exe	0.0	0.05749584409940259
2328	WmiPrvSE.exe	0.0	0.15021653361673504
2352	svchost.exe	0.0	0.10696691539454983
2568	svchost.exe	0.0	0.07096063142639285
2600	WUDFHost.exe	0.0	0.06170359013908704
2748	svchost.exe	0.0	0.0770919185127902
2756	svchost.exe	0.0	0.0768214205530962
2764	atiesrxx.exe	0.0	0.05965982777695459
2772	amdfendrsr.exe	0.0	0.05644390758948148
2780	svchost.exe	0.0	0.14519729369796858

Gestión de Procesos

1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir

Seleccione una opcion: 2

Ingrese el nombre del proceso a iniciar: calc.exe


```
Administrador: Símbolo del sistema
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opción: 2
Ingrese el nombre del proceso a iniciar: calc.exe
Proceso calc.exe iniciado correctamente

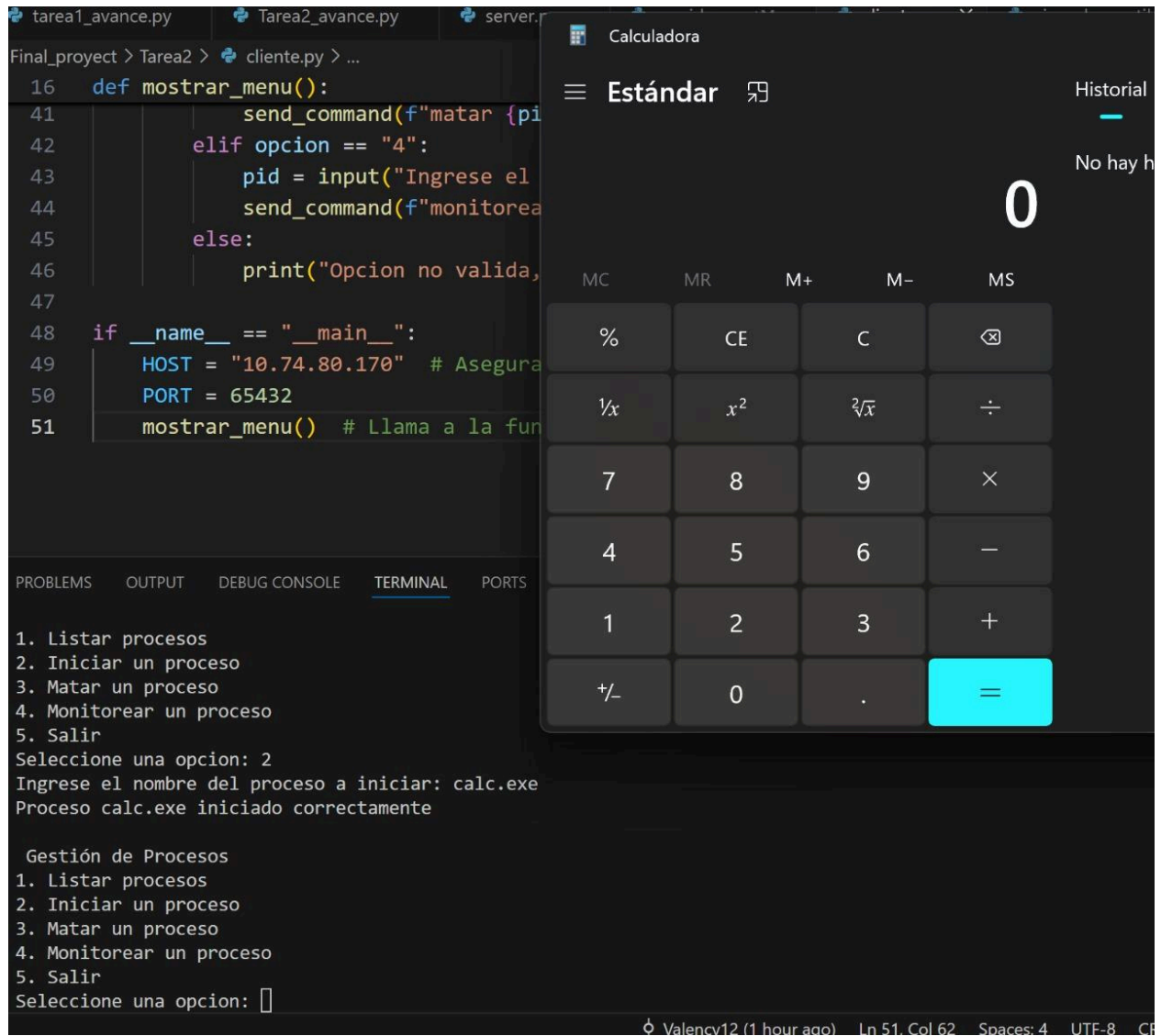
Gestión de Procesos
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opción: 3
Ingrese el PID del proceso a matar: 42420
Proceso con PID 42420 terminado correctamente.

Gestión de Procesos
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opción: 4
Ingrese el PID del proceso a monitorear: 37760
Monitoring process 37760:
Name: Code.exe
Status: running
CPU Usage: 0.0%
Memory Usage: 3.7964388643052995%

Gestión de Procesos
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opción: 5
Saliendo...

C:\Users\valen\Desktop\So_final_project\Final_project\Tarea2>
```

Computadora de Esteban



The screenshot shows a code editor with a Python script and a Windows calculator application. The code editor has tabs for 'tarea1_avance.py', 'Tarea2_avance.py', and 'server.py'. The active file is 'cliente.py' in the 'Final_proyect > Tarea2 >' directory. The code defines a 'mostrar_menu()' function and a main block that sets 'HOST' to '10.74.80.170' and 'PORT' to '65432', then calls 'mostrar_menu()'. The terminal output shows a menu with five options: 1. Listar procesos, 2. Iniciar un proceso, 3. Matar un proceso, 4. Monitorear un proceso, and 5. Salir. Option 2 is selected, and the user is prompted to enter the process name 'calc.exe', which is successfully started. The calculator is open in 'Estándar' mode, showing '0' on the display.

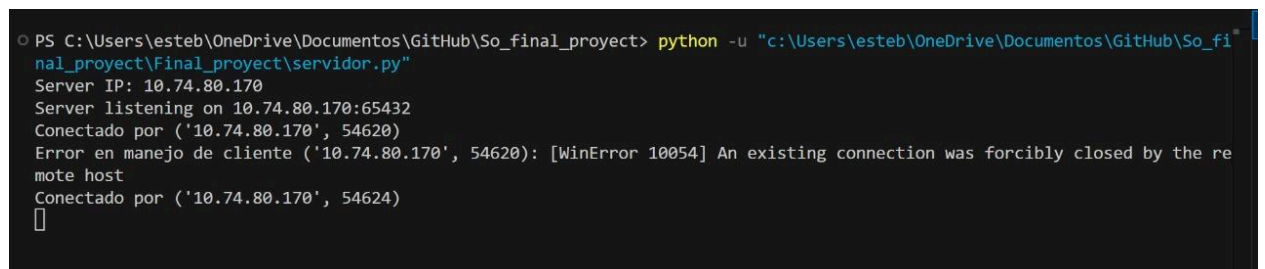
```
Final_proyect > Tarea2 > cliente.py > ...
16 def mostrar_menu():
41     send_command(f"matar {pi
42     elif opcion == "4":
43         pid = input("Ingrese el
44         send_command(f"monitorea
45     else:
46         print("Opcion no valida,
47
48 if __name__ == "__main__":
49     HOST = "10.74.80.170" # Asegura
50     PORT = 65432
51     mostrar_menu() # Llama a la fun

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opcion: 2
Ingrese el nombre del proceso a iniciar: calc.exe
Proceso calc.exe iniciado correctamente

Gestión de Procesos
1. Listar procesos
2. Iniciar un proceso
3. Matar un proceso
4. Monitorear un proceso
5. Salir
Seleccione una opcion: []
```

Calculadora
Estándar
Historial
No hay h
0
MC MR M+ M- MS
% CE C \times
 $\frac{1}{x}$ x^2 $\sqrt[n]{x}$ \div
7 8 9 \times
4 5 6 $-$
1 2 3 $+$
 \pm/\square 0 . $=$



The screenshot shows a terminal window with a PowerShell prompt. The user runs a command to execute a Python script. The output shows the server IP, the server listening on port 65432, a successful connection from '10.74.80.170' on port 54620, an error message '[WinError 10054] An existing connection was forcibly closed by the remote host', and another successful connection from '10.74.80.170' on port 54624.

```
PS C:\Users\esteb\OneDrive\Documentos\GitHub\So_final_proyect> python -u "c:\Users\esteb\OneDrive\Documentos\GitHub\So_fi
nal_proyect\Final_proyect\servidor.py"
Server IP: 10.74.80.170
Server listening on 10.74.80.170:65432
Conectado por ('10.74.80.170', 54620)
Error en manejo de cliente ('10.74.80.170', 54620): [WinError 10054] An existing connection was forcibly closed by the re
mote host
Conectado por ('10.74.80.170', 54624)
[]
```