

# UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO

## SISTEMAS OPERACIONAIS

### RELATÓRIO PROJETO: Comunicação cliente – servidor

**Discente:** Anderson Carlos da Silva Moraes

**Docente:** Reudismam Rolim de Sousa

**Enunciado:** simular uma comunicação entre dois processos, pelo menos um cliente e um processo servidor. O aluno pode utilizar a linguagem de sua preferência.

**Repositório:** [python-codes/Exercicios/ComunicacaoClienteServidor](https://github.com/python-codes/Exercicios/ComunicacaoClienteServidor) at main · [AndersonCSM/python-codes](https://github.com/AndersonCSM/python-codes)

#### **Descrição:**

O projeto simula a comunicação entre dois processos, sendo um cliente e um servidor. A comunicação entre os processos ocorrer de maneira sequencial, ou seja, somente um cliente consegue se comunicar com o servidor por vez.

Sendo usado a linguagem de programação Python3.13 para desenvolvimento do projeto.

O projeto está estruturado em três blocos principais:

- **Main:** função principal, responsável por garantir as chamadas das demais funções corretamente além de instruir ao usuário como o código deve ser usado:

```

if __name__ == "__main__": # tratamento para liberar permissões
    if len(sys.argv) < 2: # maneira intuitiva de mostrar ao usuário como utilizar o código
        print("Uso: python main.py [server|client]")
        print("Exemplo para iniciar o servidor: python main.py server")
        print("Exemplo para iniciar o cliente: python main.py client")
        sys.exit(1) # encerra o programa

    mode = sys.argv[1].lower() # recebe o argumento direito do terminal, faz o tratamento

    if mode == "server": # de acordo com o argumento informado, executa o servidor ou o cliente
        print("Iniciando o servidor...")
        run_server() # inicia o servidor
    elif mode == "client":
        print("Iniciando o cliente...")
        run_client() # inicia o cliente
    else: # se o usuário passar um argumento inválido
        print(f"Modo desconhecido: {sys.argv[1]}. Use 'server' ou 'client'.")

```

- **Run\_server:** Função para inicia a instância do servidor, como boa prática é esperado que ela seja executada antes de iniciar uma instância do cliente. Ela é responsável por criar uma conexão, receber as mensagens do usuário, e informar se a operação está funcionando corretamente.

```

5 # host como a porta padrão localhost, só aceita conexão da própria máquina
6 def run_server(host="127.0.0.1", port=65432):
7     """Inicia o servidor para escutar conexões de clientes.
8     Cria um objeto socket chamado s
9     1. Especifica o endereço IPv4 em socket.AF_INET
10    2. Especifica o tipo de protocolo de conexão como TCP em socket.SOCK_STREAM
11    """
12    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
13        s.bind((host, port)) # associa o host e port ao socket
14        s.listen() # usa o socket para ouvir conexões de entrada.
15        print(f"Servidor escutando em {host}:{port}") # exibe que o servidor está escutando
16
17        # O servidor pode aceitar múltiplas conexões sequencialmente (uma de cada vez)
18        while True: # loop para as múltiplas conexões possíveis do servidor
19            conn, addr = s.accept() # espera um cliente se conectar, atribuindo a conexão e endereço
20            with conn:
21                print(f"Conectado por {addr}") # endereço do cliente
22                while True: # loop para comunicação do cliente
23                    try: # tenta receber os dados do cliente
24                        data = conn.recv(1024) # Recebe até 1024 bytes
25                        if not data: # se não recebeu os dados é desconexão
26                            print(f"Cliente {addr} desconectou (sem dados).")
27                            break
28
29                        message = data.decode("utf-8") # decoder da mensagem
30                        print(f"Mensagem recebida de {addr}: {message}") # informa que recebeu uma mensagem
31
32                        if message.lower() == "sair": # se o cliente informou que quer encerrar a conexão
33                            print(f"Cliente {addr} solicitou desconexão.") # informa na tela
34                            conn.sendall("Servidor: Desconectando...".encode("utf-8")) # mensagem de desconexão
35                            break # sai do loop de comunicação
36
37                        # "Encaminha" a mensagem (neste caso, apenas processa e envia uma confirmação)
38                        response = f"Servidor recebeu: '{message}'"
39                        conn.sendall(response.encode("utf-8")) # mensagem de resposta ao cliente
40                    except ConnectionResetError: # raise erros possíveis durante o uso
41                        print(f"Conexão com {addr} foi resetada pelo cliente.")
42                        break
43                    except Exception as e: # raise erros possíveis durante o uso
44                        print(f"Erro ao comunicar com {addr}: {e}")
45                        break
46                print(f"Conexão com {addr} encerrada.")
47            # Após o cliente desconectar, o servidor volta a escutar por novas conexões.
48

```

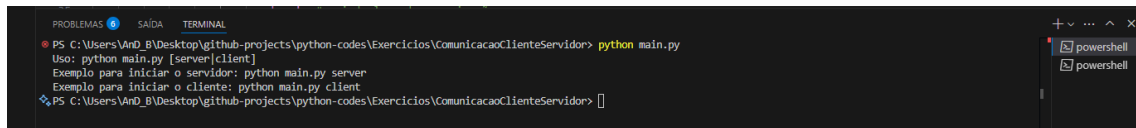
- **Run\_client:** Função para iniciar uma instância de cliente, como apresentado anteriormente, o projeto é uma comunicação sequencial, logo, só é possível ter

uma cliente se comunicando com o servidor. É dentro do código que o usuário pode inserir uma mensagem e enviá-la ao servidor, assim como, desconectar e também receber retornos do servidor.

```
49
50 def run_client(host="127.0.0.1", port=65432):
51     """Inicia o cliente para conectar ao servidor e enviar mensagens.
52     Cria um objeto socket chamado s
53     1. Especifica o endereço IPv4 em socket.AF_INET
54     2. Especifica o tipo de protocolo de conexão como TCP em socket.SOCK_STREAM"""
55
56     with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
57         try: # tenta estabelecer conexão com o servidor
58             s.connect((host, port)) # associa o host e port ao socket
59             print(f"Conectado ao servidor em {host}:{port}") # informa que conectou a um servidor
60         except ConnectionRefusedError: # raise erros possíveis durante o uso
61             print(f"Não foi possível conectar ao servidor em {host}:{port}. Verifique se o servidor está em execução.")
62             return
63         except Exception as e: # raise erros possíveis durante o uso
64             print(f"Erro ao conectar: {e}")
65             return
66
67     while True: # loop para comunicação entre cliente e servidor
68         message_to_send = input(
69             "Digite a mensagem para o servidor (ou 'sair' para desconectar): "
70         ) # mensagem para ser enviada
71         if not message_to_send.strip(): # Evita enviar mensagens vazias
72             print("Por favor, digite uma mensagem ou 'sair'.")
73             continue
74
75         try: # bloco para tentar enviar mensagem
76             s.sendall(message_to_send.encode("utf-8")) # envia para o servidor
77
78             if message_to_send.lower() == "sair": # solicitação de desconexão
79                 print("Enviando solicitação de desconexão para o servidor...")
80                 # Espera uma confirmação do servidor antes de fechar, se houver
81                 try: # bloco esperando resposta do servidor para desconexão
82                     response_data = s.recv(1024) # recebe mensagem do servidor
83                     if response_data: # se receber, imprima
84                         print(f"Resposta do servidor: {response_data.decode('utf-8')}")
85                 except Exception as e: # raise erros possíveis durante o uso
86                     print(f"Nenhuma resposta final do servidor ou erro: {e}")
87                 break
88
```

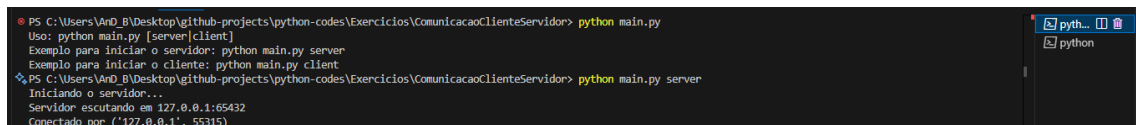
```
SE... 88
89
90     # Espera a resposta do servidor do envio da mensagem em # 74
91     response_data = s.recv(1024)
92     if not response_data: # Servidor fechou a conexão inesperadamente
93         print("Servidor fechou a conexão.")
94         break
95     print(f"Resposta do servidor: {response_data.decode('utf-8')}")
96
97     except ConnectionResetError: # raise erros possíveis durante o uso
98         print("A conexão foi resetada pelo servidor.")
99         break
100     except BrokenPipeError: # raise erros possíveis durante o uso
101         print(f"A conexão com o servidor foi perdida (Broken pipe).")
102         break
103     except Exception as e: # raise erros possíveis durante o uso
104         print(f"Erro ao enviar/receber dados: {e}")
105         break
106     print("Cliente desconectado.")
```

**Exemplo de comunicação:** vamos abrir dois terminais, um para executar o processo servidor e outro para o processo cliente, iremos executar o código em ambos para que seja mostrado as instruções corretamente de como iniciar cada processo.



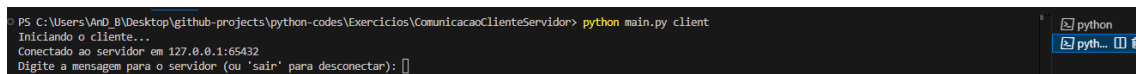
```
PROBLEMAS SAÍDA TERMINAL
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor> python main.py
Uso: python main.py [server|client]
Exemplo para iniciar o servidor: python main.py server
Exemplo para iniciar o cliente: python main.py client
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor>
```

Iniciando o processo servidor:



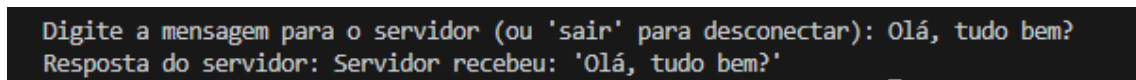
```
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor> python main.py
Uso: python main.py [server|client]
Exemplo para iniciar o servidor: python main.py server
Exemplo para iniciar o cliente: python main.py client
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor> python main.py server
Iniciando o servidor...
Servidor escutando em 127.0.0.1:55315
Conectado por ('127.0.0.1', 55315)
```

Iniciando o processo cliente:



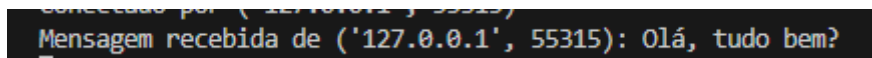
```
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor> python main.py client
Iniciando o cliente...
Conectado ao servidor em 127.0.0.1:55315
Digite a mensagem para o servidor (ou 'sair' para desconectar):
```

Enviando mensagem por cliente:



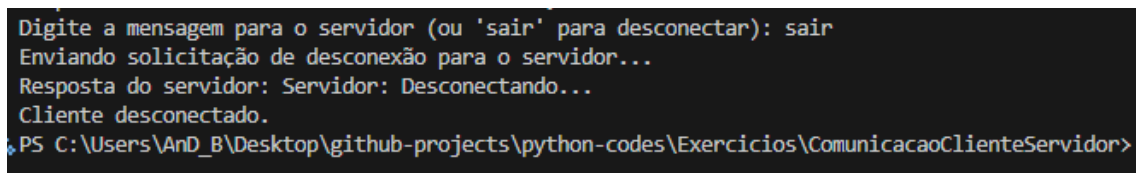
```
Digite a mensagem para o servidor (ou 'sair' para desconectar): Olá, tudo bem?
Resposta do servidor: Servidor recebeu: 'Olá, tudo bem?'
```

Mensagem recebida do lado servidor:



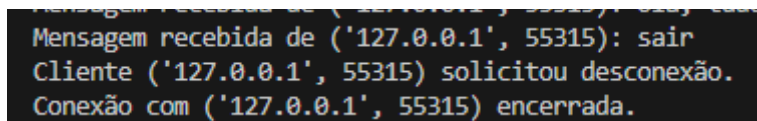
```
Conectado por ('127.0.0.1', 55315)
Mensagem recebida de ('127.0.0.1', 55315): Olá, tudo bem?
```

Cliente desconectado do servidor:



```
Digite a mensagem para o servidor (ou 'sair' para desconectar): sair
Enviando solicitação de desconexão para o servidor...
Resposta do servidor: Servidor: Desconectando...
Cliente desconectado.
PS C:\Users\AnD_B\Desktop\github-projects\python-codes\Exercicios\ComunicacaoClienteServidor>
```

Servidor tratando desconexão:



```
Mensagem recebida de ('127.0.0.1', 55315): sair
Mensagem recebida de ('127.0.0.1', 55315): sair
Cliente ('127.0.0.1', 55315) solicitou desconexão.
Conexão com ('127.0.0.1', 55315) encerrada.
```