Socket

&

Python

Sommario

Creazione codici 2

1. socket_server.py - Lato Server 2
2. socket_client.py - Lato Client 3
3. socket_registrazione.py - Funzioni di Registrazione 4

Creazione codici

1. socket_server.py- Lato Server

Questo file rappresenta il **server** e si occupa di gestire le richieste in arrivo dai client. Le funzioni principali del server sono le seguenti:

- **Creazione del socket**: Il server crea un socket TCP utilizzando socket.AF_INET (che indica l'uso di IPv4) e socket.SOCK_STREAM (che indica l'uso di TCP).
- **Binding (associazione)**: Il socket è associato a un indirizzo IP e a una porta specifica (0.0.0.0 per accettare connessioni da qualsiasi IP e la porta 44444 per identificare il servizio).
- **Ascolto delle connessioni**: Il server si pone in ascolto delle richieste di connessione in arrivo da parte dei client.
- **Accettazione della connessione**: Una volta ricevuta una richiesta di connessione, il server accetta la connessione e crea un socket dedicato per la comunicazione con il client.
- **Scambio di dati**: Dopo aver stabilito la connessione, il server invia un messaggio di conferma al client: "Connessione stabilita con successo!".
- **Chiusura della connessione**: Infine, sia il socket del client che quello del server vengono chiusi per completare la sessione.

```python
import socket

# definizione del server IP e PORTA
SRV_ADDR = "192.168.50.100"
SRV_PORT = 44444

# creazione del socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# collegamento all'indirizzo e porta specificato sopra
s.bind((SRV_ADDR, SRV_PORT))

# ascolto della porta
s.listen(1)
print("Server avviato! In attesa della connessione ... ")

# accettare la connessione
connection, address = s.accept()
print(f"Client connesso all'indirizzo: {address}")

# inviare un messaggio al client
welcome_message = f"Client connesso all'indirizzo: {address}"
connection.sendall(welcome_message.encode('utf-8'))

# continua il ciclo finché c'è connessione
while True:
    # scambio dati
    data = connection.recv(1024)

    # se non c'è connessione ferma
    if not data:
        break

    # stampa in utf-8 i dati della connessione
    print(data.decode('utf-8'))

# per chiudere la connessione
connection.close()
```

2. socket_client.py- Lato Client

Questo file rappresenta il **client** e ha il compito di stabilire una connessione con il server e ricevere dati. Le operazioni principali includono:

- **Creazione del socket**: Il client crea un socket TCP, similmente al server, per poter iniziare la connessione.
- **Connessione al server**: Il client si collega al server specificando l'indirizzo IP e la porta (in questo caso 0.0.0.0 e 44444, come indicato nel server).
- **Ricezione di dati**: Una volta che la connessione è stabilita, il client riceve il messaggio inviato dal server ("Connessione stabilita con successo!").
- **Chiusura del socket**: Al termine della comunicazione, il socket del client viene chiuso per interrompere la connessione.

```
import socket

# definizione dell'indirizzo e porta del server
SERVER_ADDR = "192.168.50.100"
SERVER_PORT = 44444

# creazione del socket
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# connessione al server
s.connect((SERVER_ADDR, SERVER_PORT))

# ricezione e stampa del messaggio di benvenuto
welcome_message = s.recv(1024)
print(welcome_message.decode('utf-8'))

# chiusura della connessione
s.close()


```

3. socket_registrazione.py- Funzioni di Registrazione

Questo file contiene logiche aggiuntive per la gestione della **registrazione delle connessioni** e delle interazioni tra client e server. In particolare, si occupa di:

- **Gestione degli utenti**: Il file implementa un sistema di registrazione che permette di registrare i client che si connettono al server, associando eventualmente l'IP o le credenziali dei client a una sessione attiva.
- **Memorizzazione delle connessioni**: Può contenere meccanismi per tracciare e memorizzare le connessioni stabilite con il server, conservando informazioni sulle interazioni e sugli indirizzi di rete dei client.
- **Log delle attività**: In aggiunta, potrebbe implementare un sistema di logging per tracciare eventi come connessioni stabilite, disconnessioni e possibili errori di rete.

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

server_socket.bind(("0.0.0.0", 44444))
server_socket.listen(1)

print("Connessione stabilita sulla porta: 44444 ")

client_socket, addr = server_socket.accept()
print (f"Connessione stabilita con {addr}")

client_socket.sendall(b"Connessione stabilita con successo!")

client_socket.close()
server_socket.close()


```