Descrizione

Il progetto consiste in una semplice chat tra un server e vari clients. Il server viene aperto prima dei clients e permette lo scambio dei messaggi. Ai vari utenti viene associata una gui per permettere un migliore scambio dei messaggi. Il sistema sfrutta il TCP e la programmazione concorrente tramite threads basandosi sul linguaggio python.

Server

`server.py` contiene lo script per far partire il server che gestisce la comunicazione tra client della chat.

Una volta lanciato lo script viene associata a Cntrl + c una funzione per terminare il server, che invia a tutti i client un messaggio di chiusura. Dopo di che viene creato un socket TCP e viene poi associato all'HOST (0.0.0.0) e poi alla porta, che viene presa in input oppure le è assegnato un valore di default.

```
HOST = ''

port_input = input("Inserisci la porta del server (Default %i): " % DEFAULT_PORT)
if port_input:
    PORT = int(port_input)
else:
    PORT = DEFAULT_PORT

BUFSIZ = DEFAULT_BUFF
ADDR = (HOST, PORT)

SERVER = socket(AF_INET, SOCK_STREAM)
SERVER.bind(ADDR)
```

Subito dopo il server si mette in attesa dei client e crea un thread `Thread(target=new_client_accept)` per l'accettazione dei client.

new_client_accept

la funzione `new_client_accept` gestisce l'accettazione di nuove connessioni. A ogni client viene inviato un messaggio in cui si chiede di inserire il nome e avvia un thread per la gestione di ognuno dei client.

```
client, client_address = SERVER.accept()
print("%s:%s si è collegato." % client_address)
client.send(bytes("CIAO! Digita il tuo Nome seguito dal tasto Invio!", "utf8"))
Thread(target=client_handler, args=(client,)).start()
```

client_handler

la funzione `client_handler` verifica che il client connesso abbia effetivamente inserito un nome e invia al nuovo client un messaggio di benvenuto e infine invia un messaggio a eventuali client gia' connessi segnalando che si e' aggiunto un nuovo client.

```
message = socket_client.recv(BUFSIZ)
if message != bytes("{quit}", "utf8"):
    name = message.decode("utf8")
    benvenuto = 'Benvenuto %s! Se vuoi lasciare la Chat, scrivi {quit} per uscire.' % name
    socket_client.send(bytes(benvenuto, "utf8"))
    msg = "%s si è unito all chat!" % name
    broadcast(bytes(msg, "utf8"))
        ...
else:
    #la prima cosa mandata dal client e' un {quit}
    socket_client.close()
    print("%s:%s si è scollegato." % addresses[socket_client])
```

Da questo momento in avanti entra in un ciclo in cui invia agli utenti collegati i messaggi del nuovo utente e segnala quando abbandona la chat.

```
msg = socket_client.recv(BUFSIZ)
if msg != bytes("{quit}", "utf8"):
    broadcast(msg, name + ": ")
else:
    socket_client.close()
    broadcast(bytes("%s ha abbandonato la Chat." % name, "utf8"))
    break
```

Client

Lo script Client.py inizia chimando la funzione `start_chat()` che fornisce all'utente un'interfaccia per inserire l'ip e la porta del server.

Poi vengono definite le caratteristiche della finestra della chat effettiva e viene associata una funzione `close(signum=None, frame=None)` a Cntrl + c.

Connessione al Server

Dopo aver ottenuto l'indirizzo IP e la porta dal prompt dell'utente, il programma tenta di connettersi al server con una socket TCP, se la connessione non riesce entro il `DEFAULT_TIMEOUT` la connessione fallisce e il programma si conclude con exit(1)

```
def connect_with_timeout():
    global connection_successful
    connection_successful = False
    try:
        client_socket.connect(ADDR)
        connection_successful = True
    except Exception as e:
        print("Errore durante la connessione:", e)

client_socket = socket(AF_INET, SOCK_STREAM)
connection_successful = False
connection_thread = Thread(target=connect_with_timeout)
connection_thread.start()
connection_thread.join(timeout=DEFAULT_TIMEOUT)
if not connection_successful:
    print("Connessione non riuscita")
    os._exit(1)

receive_thread = Thread(target=receive)
receive_thread.start()
```

receive e send

La funzione `receive()` e' associata a un thread che resta sempre in ascolto dei messaggi e li aggiunge a schermo, si occupa anche di chiudere la connessione nel caso in cui il server lo richieda.

```
while True:
    try:
        message = client_socket.recv(BUFSIZ)
        if message == bytes("{quit}", "utf8"):
            print("La connessione con il server e' stata interrotta")
            os._exit(1)
        msg=message.decode("utf8")
        msg_list.insert(tkt.END, msg)
    except OSError or RuntimeError:
        break
```

La funzione `send()` invia i messaggi al server quando l'utente preme il pulsante di invio o preme il tasto "Invio" sulla tastiera.

```
msg = my_msg.get()
my_msg.set("")
client_socket.send(bytes(msg, "utf8"))
if msg == "{quit}":
    client_socket.close()
    frame.quit()
```

Funzionamento e considerazioni

Requisito

Avere installato python, nel caso dei client e` anche neccesario che ci sia un server.

Avvio

Per utilizzare la chat e' sempre necessario avviare prima il server e dopo si possono avviare fino a `MAX_CLIENTS` client.

Considerazioni

Al client non e` permesso lasciare vuoto l'indirizzo ip e viene lasciato un timeout di 2 secondi per collegarsi al server nel caso in cui la connessione non riesca ad andare a buon fine.