



UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

RETI DI CALCOLATORI

Francesco De Micco Matricola: 0124002556

Francesca Formisano Matricola: 0124002481

RELAZIONE - IRC CHAT

a.a 2023 - 2024

Indice

| | | |
|----------|--|-----------|
| 1 | Traccia | 5 |
| 2 | Descrizione e schema dell'architettura | 6 |
| 2.1 | Schema | 6 |
| 2.2 | Architettura | 6 |
| 2.2.1 | Componenti fondamentali | 6 |
| 2.2.2 | Flusso di comunicazione (Server -> Client) | 7 |
| 2.2.3 | Flusso di comunicazione (Client -> Server) | 7 |
| 3 | Dettagli implementativi del client/server | 8 |
| 3.1 | Client | 8 |
| 3.1.1 | Connessione al server | 8 |
| 3.1.2 | Gestione input utente | 8 |
| 3.1.3 | Autenticazione | 8 |
| 3.1.4 | Comunicazione con il Server | 8 |
| 3.1.5 | Utilizzo dei thread | 8 |
| 3.2 | Server | 9 |
| 3.2.1 | Ascolto connessioni in entrata | 9 |
| 3.2.2 | Gestione connessioni multiple | 9 |
| 3.2.3 | Comunicazione con il Client | 9 |
| 3.2.4 | Autenticazione degli Utenti | 9 |
| 3.2.5 | Esecuzione Comandi | 9 |
| 4 | Manuale utente ed esecuzione | 10 |
| 4.1 | Avvio del Server | 10 |
| 4.2 | Avvio del Client | 10 |
| 4.3 | Accesso | 11 |
| 4.4 | Comandi di base | 12 |
| 4.4.1 | Join Channel | 12 |
| 4.4.2 | Leave Channel | 12 |
| 4.4.3 | Send Private Message | 13 |
| 4.4.4 | Send Message | 13 |
| 4.4.5 | List Users | 14 |
| 4.4.6 | List Channels | 14 |
| 4.5 | Comandi admin | 15 |
| 4.5.1 | Kick User | 15 |
| 4.5.2 | Ban User | 15 |
| 4.5.3 | Unban User | 16 |
| 4.5.4 | Promote User | 16 |

1 Traccia

Simulare una chat multiutente basata su IRC. Utilizzare un approccio client/server.

Server:

- Permette agli utenti di connettersi
- Mostra agli utenti una serie di possibili canali attivi (identificati con #)
- Rappresenta un gruppo in cui tutti gli utenti connessi possono inviare messaggi visibili a tutti coloro che sono connessi in quel canale
- Permette all'utente di cambiare il canale su cui è connesso
- Gestisce la collisione tra nomi utenti uguali
- Permette a due utenti di parlare in privato

Client:

- Si connette ad un server specificando un nome utente, non è richiesta la password
- Può richiedere la lista dei canali inviando
Comando: `/list`
- Può connettersi ad un canale
Comando: `/join #channel_name`
- Può vedere gli utenti connessi
Comando: `/users`
- Può inviare messaggi
Comando: `/msg messaggio`
- Può inviare un messaggio privato ad un utente
Comando: `/privmsg nickname messaggio`
- Può cambiare il canale su cui è connesso in qualunque momento

Implementare l'utente amministratore che può:

- Espellere un utente dal canale
Comando: `/kick nickname`
- Bannare/sbannare un utente dal canale
Comando: `/ban nickname`
Comando: `/unban nickname`
- Promuovere un utente come moderatore
Comando: `/promote nickname`

2 Descrizione e schema dell'architettura

2.1 Schema

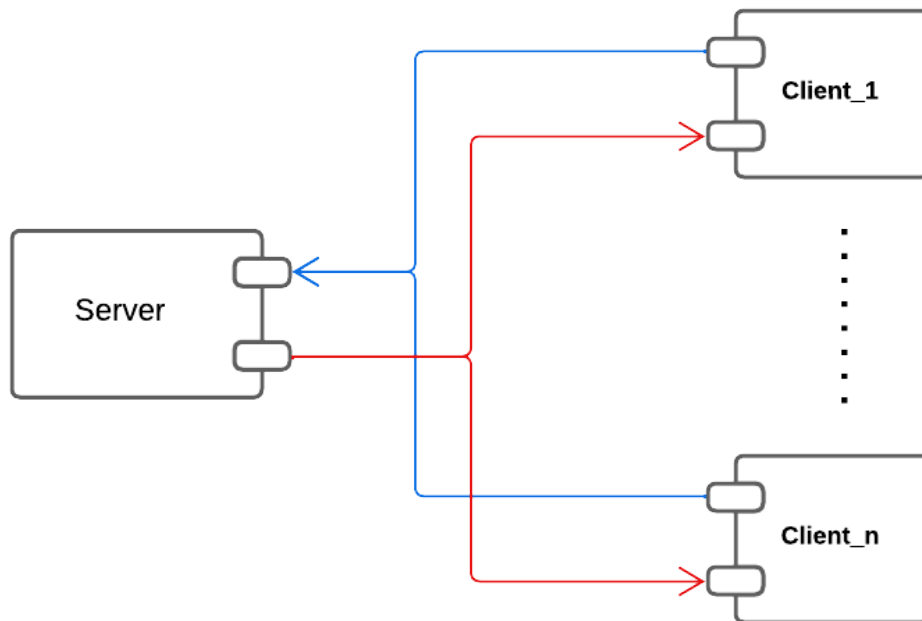


Figura 1: Schema dell'architettura

La Figura 1 mostra la comunicazione tra i vari client ed il server. Il client prevede un writer ed un reader per la lettura e scrittura su socket del server. Il server, al contempo, prevede un writer ed un reader per la lettura e scrittura su socket del client.

2.2 Architettura

E' stato optato per Java come linguaggio principale per il suo mix di portabilità, semplicità d'uso e solidità. Come richiesto, è stata scelta come architettura di comunicazione quella "client/server". L'implementazione di essa è stata resa possibile grazie all'utilizzo delle Socket.

2.2.1 Componenti fondamentali

Si procede di seguito ad elencare i principali componenti dell'applicativo in fase di sviluppo. Essi sono:

- **Client:** risulta essere l'utente che utilizza l'applicativo. E' stata pensata una sola interfaccia (da terminale) per entrambe le tipologie di utenti (Base e Moderatore); in seguito alla connessione verrà stabilito automaticamente il livello di privilegio. In quanto tale, può utilizzare l'applicativo inviando comandi al server e visualizzando il risultato.

- **Server:** accetta le connessioni in entrata e permette l'autenticazione degli utenti. Solo se il nome utente non è già stato utilizzato permette l'accesso ad esso; altrimenti richiede un'altro nome utente. Solamente dopo l'autenticazione, permette l'esecuzione dei comandi lato server. Riceve i comandi inviati dall'utente e li esegue; se necessario restituisce quanto richiesto dal comando.

2.2.2 Flusso di comunicazione (Server -> Client)

1. Apertura della socket del Server
2. Ascolto connessioni in entrata
3. Ascolto messaggi dai client ed esecuzione comandi (Thread)
4. Chiusura della connessione

Essendo che il server è stato predisposto per accogliere più client e gestirli in maniera concorrente, sono stati predisposti due flussi di esecuzione paralleli, di cui uno è un thread il quale gestisce l'ascolto dei messaggi dai client, per eseguire i comandi utente inviatigli. Il flusso principale di esecuzione gestisce invece le connessioni in arrivo. Più nello specifico, vi è un loop per accogliere le connessioni in entrata, una volta stabilita una connessione, viene fatto partire un thread per tale connessione che ascolterà i messaggi in arrivo dal client specifico.

2.2.3 Flusso di comunicazione (Client -> Server)

1. Autenticazione al server tramite nome univoco
2. Accesso al server
3. Invio comandi al server
4. Visualizzazione risultato

In questo caso, il client non richiede nessuna particolare azione, se non quella di effettuare l'accesso al server tramite nome univoco. Una volta effettuato l'accesso con successo, esso sarà in grado di eseguire i comandi sul server ed ottenere una risposta.

3 Dettagli implementativi del client/server

Di seguito si procede ad entrare più nello specifico per quanto riguarda l'aspetto implementativo. Anche in questo caso verrà descritto, nello specifico, ogni aspetto di ogni componente dell'applicativo.

3.1 Client

3.1.1 Connessione al server

Il client si connette al server IRC tramite il protocollo TCP/IP utilizzando la primitiva `Socket` che richiede il nome host o l'indirizzo IP del server e la porta IRC predefinita scelta (Nel caso specifico il server si trova in locale quindi l'indirizzo sarà `127.0.0.1`). Una volta stabilita la connessione, il client invia un messaggio di identificazione al server.

3.1.2 Gestione input utente

Il client gestisce gli input dell'utente, consentendo agli utenti di inserire comandi IRC come `/join`, `/msg`, ecc. I comandi vengono interpretati e inviati al server IRC per l'esecuzione. Attraverso la classe `BufferedReader` si accetta l'input dall'utente da terminale.

3.1.3 Autenticazione

L'autenticazione del client avviene attraverso l'invio di un nickname al server IRC durante la connessione. Il server verificherà che il nome inserito sia univoco; in caso opposto richiederà all'utente di inserirlo.

3.1.4 Comunicazione con il Server

Il client comunica con il server inviando e ricevendo messaggi utilizzando il protocollo IRC. Questi messaggi includono comandi per unirsi a un canale, inviare messaggi, cambiare il nickname, ecc. Viene predisposto un thread per l'ascolto dei messaggi provenienti dal server. Per inviare quanto digitato al server si utilizza la classe `PrintWriter`, per appunto scrivere sulla socket del Server. Analogamente, sempre con la classe `BufferedReader` si legge ciò che il server risponde al client.

3.1.5 Utilizzo dei thread

In questo caso, sono predisposti due thread:

- uno per l'ascolto continuo dei messaggi da parte del server
- uno per l'invio continuo dei messaggi scritti da terminale ed inviati al server

3.2 Server

3.2.1 Ascolto connessioni in entrata

Il server IRC ascolta le connessioni in entrata sulla porta TCP/IP specificata inizializzando la Socket; in questo caso, viene utilizzata la classe `ServerSocket` perchè appunto la socket è del server. Accetta quindi le connessioni dai client che desiderano collegarsi in un loop infinito.

3.2.2 Gestione connessioni multiple

Il server IRC è in grado di gestire connessioni multiple simultaneamente da parte di diversi client. Per rendere possibile ciò, nel momento in cui viene accettata la richiesta, viene startato un Thread per ogni client che sta cercando di connettersi in modo tale da non vincolare il flusso di esecuzione al server. In questo modo un flusso separato adibito al solo ascolto dei messaggi dei client viene predisposto per ognuno di essi.

3.2.3 Comunicazione con il Client

Il server comunica con i client in maniera analoga a come il client lo fa con il server, inviando e ricevendo messaggi IRC tramite il protocollo IRC. In particolare, Viene predisposto tramite la classe `BufferedReader` un oggetto per leggere dalla socket del client, e, analogamente a come avveniva con il client, tramite la classe `PrintWriter` un oggetto per scrivere sulla socket del server, quindi per restituire al client che ne fa richiesta il risultato di un comando.

3.2.4 Autenticazione degli Utenti

Il server IRC autentica gli utenti durante la connessione. Ciò include la verifica del nickname. Essendo che il nickname è l'unico modo per identificare e distinguere gli utenti, il server mantiene una lista di utenti, in particolare dei loro nomi. In questo modo, in fase di accesso, il server andrà a controllare se il server inserito non è stato già preso; affermativamente, verrà connessa la connessione al server, in caso contrario verrà negata e richiesto l'inserimento del nickname da parte dell'utente

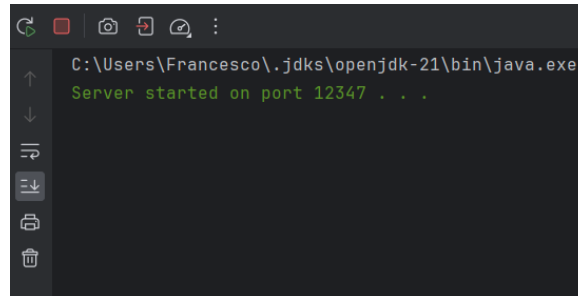
3.2.5 Esecuzione Comandi

Il server IRC esegue i comandi inviati dai client. Questi comandi possono includere operazioni di base, comuni a tutti gli utenti, o comandi relativi agli amministratori. Nel momento dell'invio di un comando al server, il server controllerà se l'utente che ha richiesto il comando avrà i privilegi necessari per eseguirlo. In caso affermativo eseguirà il comando altrimenti informerà l'utente che non può eseguirlo.

4 Manuale utente ed esecuzione

4.1 Avvio del Server

Per consentire ai client di connettersi al server, è ovviamente necessario far partire il Server. Se si esegue il codice da IntelliJ basta recarsi sul file `ChatServer.java` ed eseguirlo. L'output sarà il seguente:

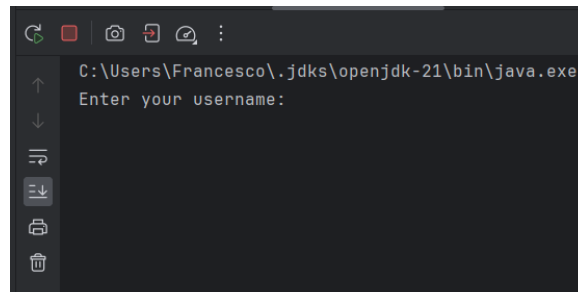
A screenshot of a terminal window with a dark background. The title bar shows standard window controls. The terminal text shows the command `C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe` and the output `Server started on port 12347 . . .` in green text. On the left side of the terminal, there is a vertical toolbar with icons for navigation and editing.

```
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe
Server started on port 12347 . . .
```

Figura 2: Avvio del Server

4.2 Avvio del Client

Per iniziare a messaggiare con altri utenti connessi al server, è ovviamente necessario far una (o più) istanza del client. Se si esegue il codice da IntelliJ basta recarsi sul file `ChatClient.java` ed eseguirlo. In primo luogo, verrà richiesto il nome utente. L'output sarà il seguente:

A screenshot of a terminal window with a dark background. The title bar shows standard window controls. The terminal text shows the command `C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe` and the output `Enter your username:`. On the left side of the terminal, there is a vertical toolbar with icons for navigation and editing.

```
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe
Enter your username:
```

Figura 3: Avvio del client

4.3 Accesso

Per eseguire l'accesso al server, sarà sufficiente fornire l'username, che deve essere univoco. Nel caso in cui il nome inserito risulti già scelto da un'altro utente, verrà mostrato un messaggio di errore che invita l'utente ad inserirne uno differente.

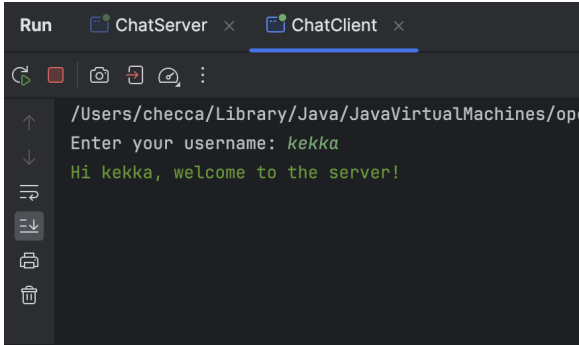


Figura 4: Inserimento nome utente valido

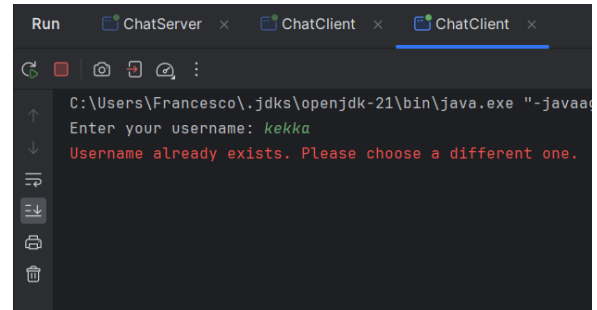


Figura 5: Inserimento nome utente già presente

Come si può notare, è stato simulato l'inserimento da un'altro client di un nome utente già scelto e presente nel server.

Inoltre è stato previsto un sistema di LOG del server che monitora connessioni in entrata ed uscita. Di seguito è quindi riportata una rappresentazione grafica.

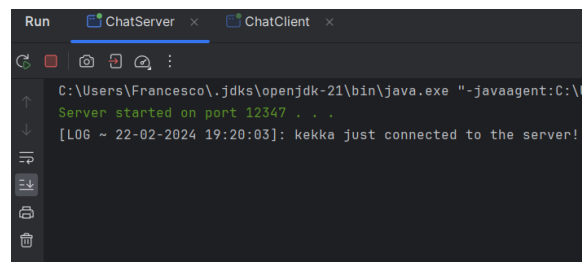
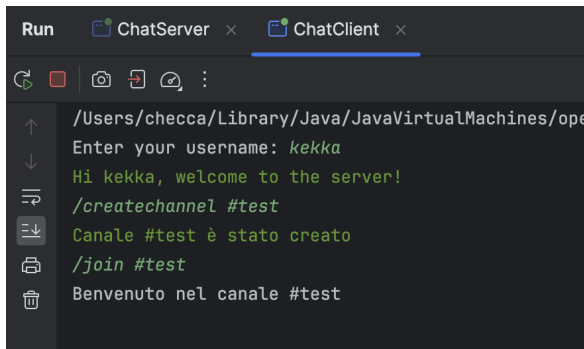


Figura 6: Visualizzazione log su Server

4.4 Comandi di base

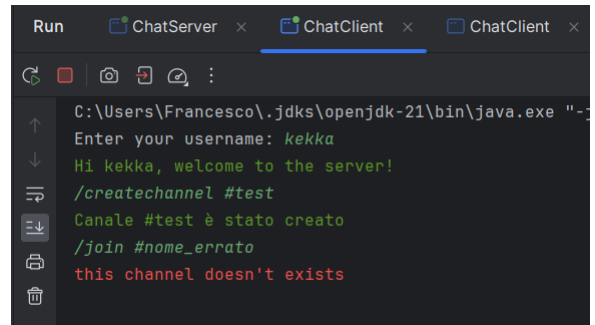
4.4.1 Join Channel

`/join <canale>` è un comando che consente di entrare in un canale esistente. Il parametro `<canale>` rappresenta il nome del canale a cui desideri unirti. Ad esempio, `/join test` ti unirà al canale denominato "test". Nel caso in cui il canale non esista verrà mostrato un'errore.



```
Run ChatServer x ChatClient x
/Users/checca/Library/Java/JavaVirtualMachines/ope
Enter your username: kekka
Hi kekka, welcome to the server!
/createchannel #test
Canale #test è stato creato
/join #test
Benvenuto nel canale #test
```

Figura 7: Creazione ed unione ad un canale

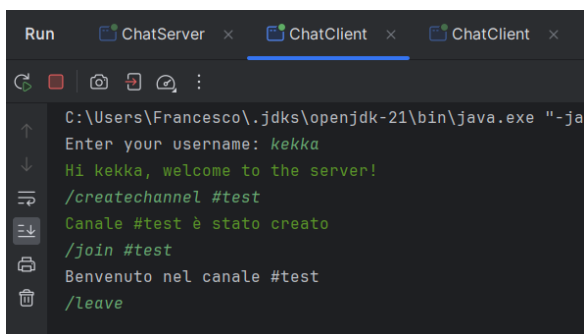


```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-j
Enter your username: kekka
Hi kekka, welcome to the server!
/createchannel #test
Canale #test è stato creato
/join #nome_errato
this channel doesn't exists
```

Figura 8: unione ad un canale inesistente

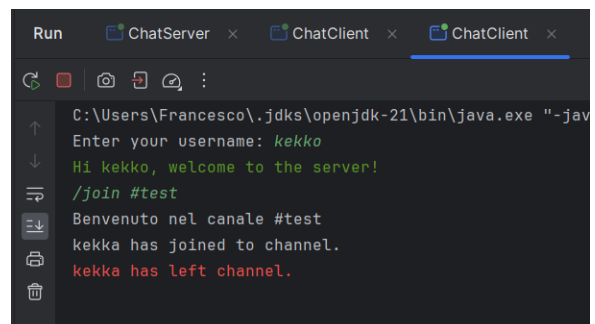
4.4.2 Leave Channel

`/leave` è un comando utilizzato per abbandonare il canale corrente. Non richiede parametri aggiuntivi. Se eseguito in un canale notificherà a tutti gli utenti l'uscita dal canale e quindi permetterà l'uscita all'utente che lo ha eseguito. Altrimenti verrà mostrato un errore in quanto questo comando è eseguibile solo se si è in un canale.



```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-ja
Enter your username: kekka
Hi kekka, welcome to the server!
/createchannel #test
Canale #test è stato creato
/join #test
Benvenuto nel canale #test
/leave
```

Figura 9: uscita da un canale

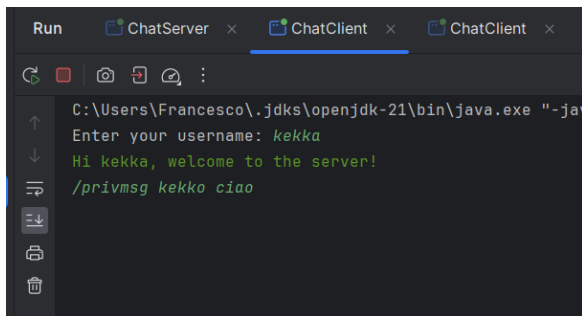


```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-jav
Enter your username: kekko
Hi kekko, welcome to the server!
/join #test
Benvenuto nel canale #test
kekka has joined to channel.
kekka has left channel.
```

Figura 10: messaggio per i partecipanti

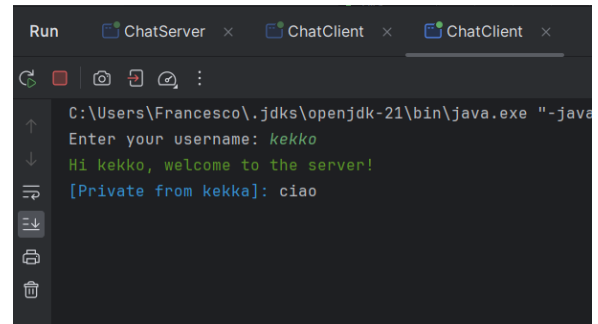
4.4.3 Send Private Message

`/privmsg <nickname> <messaggio>` è un comando che consente di inviare un messaggio in privato a un altro utente. Il parametro `<nickname>` rappresenta il nickname dell'utente a cui si vuole inviare il messaggio, mentre `<messaggio>` è il contenuto del messaggio che si desidera inviare. Nel caso in cui viene digitato il nome di un utente che non esiste (o che comunque non è connesso al server), verrà mostrato un'errore.



```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-java
Enter your username: kekka
Hi kekka, welcome to the server!
/privmsg kekko ciao
```

Figura 11: Invio messaggio in privato

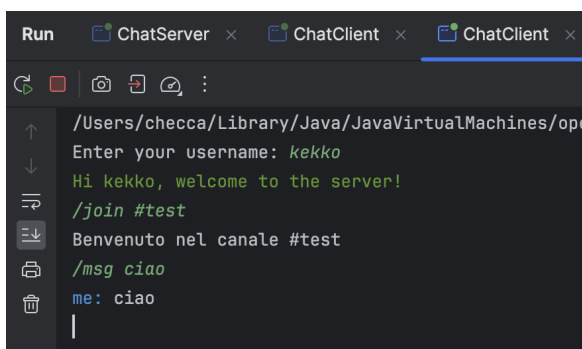


```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-java
Enter your username: kekko
Hi kekko, welcome to the server!
[Private from kekka]: ciao
```

Figura 12: ricezione messaggio in privato

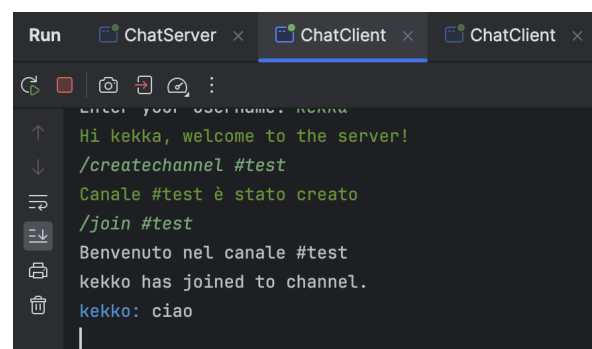
4.4.4 Send Message

`/msg <messaggio>` è un comando che consente di inviare un messaggio nel canale corrente. Il parametro `<messaggio>` rappresenta il contenuto del messaggio che desideri inviare. Il messaggio sarà visibile a tutti gli utenti presenti nel canale in cui viene eseguito. Se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale.



```
Run ChatServer x ChatClient x ChatClient x
/Users/checca/Library/Java/JavaVirtualMachines/open
Enter your username: kekko
Hi kekko, welcome to the server!
/join #test
Benvenuto nel canale #test
/msg ciao
me: ciao
```

Figura 13: invio messaggio in un canale



```
Run ChatServer x ChatClient x ChatClient x
Enter your username: kekko
Hi kekka, welcome to the server!
/createchannel #test
Canale #test è stato creato
/join #test
Benvenuto nel canale #test
kekko has joined to channel.
kekko: ciao
```

Figura 14: ricezione messaggio inviato in un canale

4.4.5 List Users

/users è un comando utilizzato per visualizzare un elenco degli utenti attualmente connessi al canale. Se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale.

```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-jav
Enter your username: kekko
Hi kekko, welcome to the server!
/join #test
Benvenuto nel canale #test
/users
[kekka, kekko]
```

Figura 15: esecuzione comando in un canale

```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-jav
Enter your username: kekko
Hi kekko, welcome to the server!
/users
[you're not in a channel !]
```

Figura 16: esecuzione comando NON in un canale

4.4.6 List Channels

/list è un comando utilizzato per visualizzare un elenco dei canali attivi sul server. Può essere eseguito in qualunque momento.

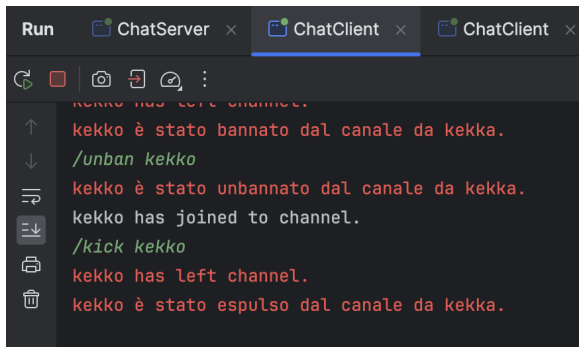
```
Run ChatServer x ChatClient x ChatClient x
C:\Users\Francesco\.jdk\openjdk-21\bin\java.exe "-ja
Enter your username: kekka
Hi kekka, welcome to the server!
/list
[#test]
```

Figura 17: comando per mostrare i canali attivi

4.5 Comandi admin

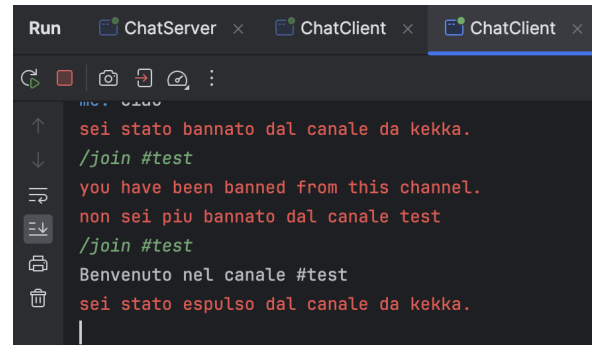
4.5.1 Kick User

`/kick <nickname>` è un comando utilizzato dagli amministratori per espellere un utente dal canale. Il parametro `<nickname>` rappresenta il nickname dell'utente che si desidera espellere. Se il nome dell'utente inserito non esiste (o se comunque non è presente in quel canale) verrà generato un'errore. Inoltre, se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale.



```
Run ChatServer x ChatClient x ChatClient x
↑
↓
kekko has left channel.
kekko è stato bannato dal canale da kekka.
/unban kekko
kekko è stato unbannato dal canale da kekka.
kekko has joined to channel.
/kick kekko
kekko has left channel.
kekko è stato espulso dal canale da kekka.
```

Figura 18: espulsione di un'utente

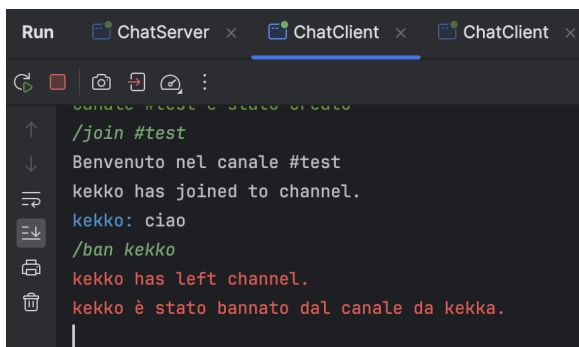


```
Run ChatServer x ChatClient x ChatClient x
↑
↓
sei stato bannato dal canale da kekka.
/join #test
you have been banned from this channel.
non sei piu bannato dal canale test
/join #test
Benvenuto nel canale #test
sei stato espulso dal canale da kekka.
```

Figura 19: messaggio mostrato all'utente espulso

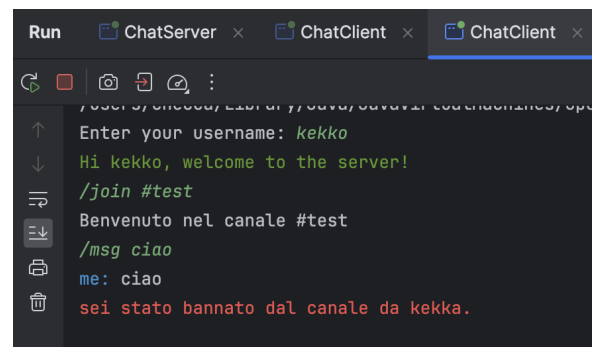
4.5.2 Ban User

`/ban` è un comando utilizzato dagli amministratori per bannare un utente dal canale. Se il nome dell'utente inserito non esiste (o se comunque non è presente in quel canale) verrà generato un'errore. Inoltre, se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale. Anche nel caso in cui l'utente da bannare risulta già bannato, verrà mostrato un'errore.



```
Run ChatServer x ChatClient x ChatClient x
↑
↓
Benvenuto nel canale #test
kekko has joined to channel.
kekko: ciao
/ban kekko
kekko has left channel.
kekko è stato bannato dal canale da kekka.
```

Figura 20: ban di un utente

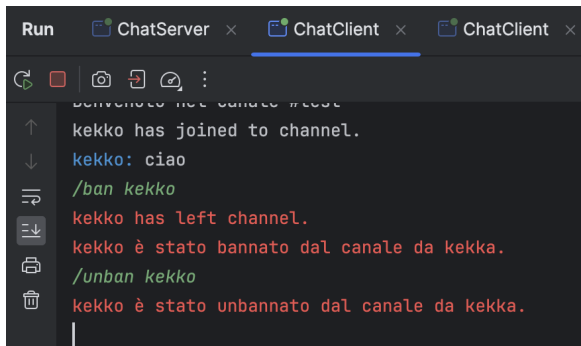


```
Run ChatServer x ChatClient x ChatClient x
↑
↓
Enter your username: kekko
Hi kekko, welcome to the server!
/join #test
Benvenuto nel canale #test
/msg ciao
me: ciao
sei stato bannato dal canale da kekka.
```

Figura 21: messaggio utente bannato

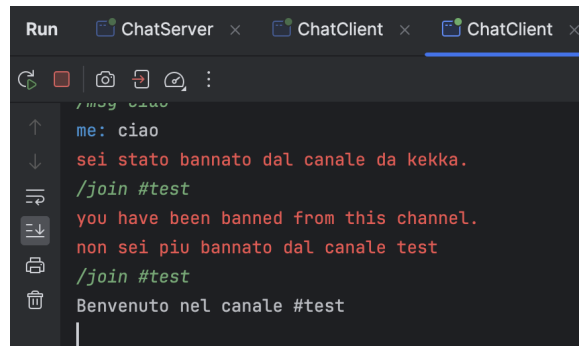
4.5.3 Unban User

/unban è un comando utilizzato dagli amministratori per rimuovere un ban precedentemente impostato su un utente nel canale. Se il nome dell'utente inserito non esiste (o se comunque non è presente in quel canale) verrà generato un'errore. Inoltre, se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale. Anche nel caso in cui l'utente dalla quale si vuole revocare il ban risulta non precedentemente bannato, verrà mostrato un'errore.



```
Run ChatServer x ChatClient x ChatClient x
↑
↓
kekko has joined to channel.
kekko: ciao
/ban kekko
kekko has left channel.
kekko è stato bannato dal canale da kekka.
/unban kekko
kekko è stato unbannato dal canale da kekka.
```

Figura 22: revoca ban utente



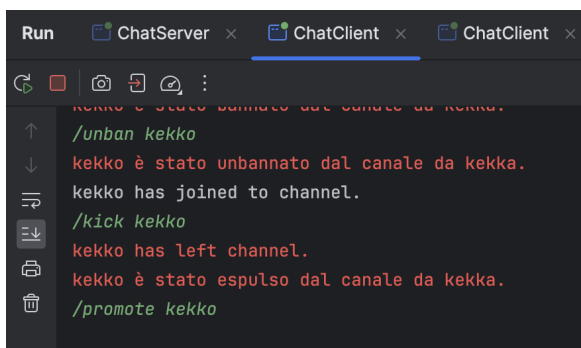
```
Run ChatServer x ChatClient x ChatClient x
↑
↓
me: ciao
sei stato bannato dal canale da kekka.
/join #test
you have been banned from this channel.
non sei piu bannato dal canale test
/join #test
Benvenuto nel canale #test
```

Figura 23: messaggio revoca ban utente

Importante notare come, nella Figura ??, l'utente precedentemente bannato, abbia cercato di accedere al canale a cui era appena stato bannato. Gli viene mostrato un'errore poichè non gli è consentito l'accesso in quanto bannato. Solamente dopo la revoca del ban all'utrente gli sarà concesso nuovamente l'accesso a quel canale.

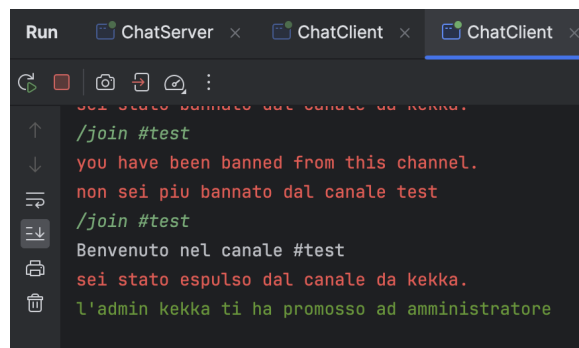
4.5.4 Promote User

/promote <nickname> è un comando utilizzato dagli amministratori per promuovere un utente a un ruolo con privilegi aggiuntivi nel canale. Se il nome dell'utente inserito non esiste (o se comunque non è presente in quel canale) verrà generato un'errore. Inoltre, se eseguito quando non si è in un canale verrà mostrato un'errore in quanto questo comando è eseguibile solo se si è in un canale.



```
Run ChatServer x ChatClient x ChatClient x
↑
↓
kekko è stato unbannato dal canale da kekka.
kekko has joined to channel.
/kick kekko
kekko has left channel.
kekko è stato espulso dal canale da kekka.
/promote kekko
```

Figura 24: promozione di un utente



```
Run ChatServer x ChatClient x ChatClient x
↑
↓
sei stato bannato dal canale da kekka.
/join #test
you have been banned from this channel.
non sei piu bannato dal canale test
/join #test
Benvenuto nel canale #test
sei stato espulso dal canale da kekka.
l'admin kekka ti ha promosso ad amministratore
```

Figura 25: messaggio di avviso all'utente appena promosso

N.B.: Solamente per brevità, non sosno stati mostrati tutte le possibile combinazioni di errori.

5 GitHub

Il codice sorgente di questo progetto è disponibile su GitHub al seguente indirizzo:

<https://github.com/r3aprz/net9project>