==========

# TODO LIST PROGETTO FT_IRC

==========

---

## 1. IMPLEMENTAZIONE COMANDO PING / PONG (Priorità: Alta)

---

CONCETTO: I client IRC reali (HexChat, irssi, etc.) inviano periodicamente un comando "PING" al server per assicurarsi che la connessione sia ancora attiva. Se il server non risponde con "PONG" entro un certo tempo, il client si disconnette per "Timeout".

COSA MANCA: Manca la gestione del comando PING nella commandMap di HandleBuffer.

DA FARE:

1. Aggiungere "PING" alla commandMap in HandleBuffer::parseMessage.
2. Implementare HandleBuffer::handlePing.
    - Logica: Deve rispondere immediatamente al client.
    - Formato risposta: "PONG :<serverName>\r\n" (o col parametro inviato dal client).

---

## 2. IMPLEMENTAZIONE COMANDO QUIT (Priorità: Alta)

---

CONCETTO: Quando un utente chiude il client o digita "/quit", invia il comando QUIT. Attualmente il tuo server gestisce la disconnessione solo se la recv() ritorna 0 (chiusura socket brutale) o errore. Il comando QUIT permette una disconnessione pulita con un messaggio di addio opzionale.

COSA MANCA: Manca il gestore per il comando QUIT.

DA FARE:

1. Aggiungere "QUIT" alla commandMap.
2. Implementare HandleBuffer::handleQuit.
    - Logica: a. Iterare su tutti i canali in cui si trova l'utente. b. Inviare un messaggio di broadcast agli altri utenti in quei canali (es: ":Nick!User@Host QUIT :Bye Bye"). c. Rimuovere l'utente da tutti i

canali (vedi punto 4). d. Chiudere il socket e rimuovere il Client dalla mappa del Server (o settare un flag per farlo fare al main loop).

---

## 3. IMPLEMENTAZIONE COMANDO PART (Priorità: Alta)

---

CONCETTO: Permette a un utente di lasciare un canale specifico senza disconnettersi dal server (comando "/part #canale"). Attualmente un utente può solo entrare (JOIN) ma non uscire.

COSA MANCA: Manca la gestione del comando PART in HandleBuffer e Channel.

DA FARE:

1. Aggiungere "PART" alla commandMap.
2. Implementare Channel::handlePart (simile a JOIN/KICK ma inverso).
    - Logica: a. Trovare il canale richiesto. b. Verificare se l'utente è dentro. c. Inviare messaggio di PART a tutti nel canale (es: ":Nick!User@Host PART #canale"). d. Rimuovere l'utente dalla lista `clients` del canale. e. (Importante) Se il canale rimane vuoto, va distrutto (vedi punto 4).

---

## 4. GESTIONE MEMORIA CANALI (CLEANUP) (Priorità: Media/Alta)

---

CONCETTO: Attualmente, quando un canale viene creato con JOIN, rimane in memoria per sempre (fino allo shutdown del server). Se tutti gli utenti escono, il canale rimane "vuoto" ma allocato. Questo è un memory leak logico.

COSA MANCA: Un meccanismo che controlli se un canale è vuoto dopo un PART, un KICK o un QUIT.

DA FARE:

1. Modificare la logica ovunque si rimuova un client da un canale (PART, QUIT, KICK).
2. Dopo aver rimosso il client: if (channel->getClientsMap().empty()) { delete channel; server.getChannelsMap().erase(channelName); }

NOTA: Fare attenzione a non accedere al puntatore channel dopo averlo cancellato!

---

## 5. CORREZIONI MINORI E ROBUSTEZZA (Priorità: Bassa)

---

A. Validazione input numerici:

- In Channel::handleMode per il flag 'l' (limit), usi std::atoi.
- DA FARE: Assicurarsi che gestisca numeri negativi o caratteri non validi (es. se atoi ritorna 0 o negativo, ignorare o dare errore).

B. Gestione nomi canali Case-Insensitive (Opzionale ma consigliato):

- IRC standard tratta #CHANNEL e #channel come lo stesso canale.
- Attualmente la tua std::map usa stringhe case-sensitive.
- DA FARE: O converti tutto in lowercase prima di cercare nella mappa, o lasci così se il subject non lo vieta esplicitamente (spesso in 42 è accettato).

---

## 6. CORREZIONE HOSTNAME NEI REPLY (Priorità: Media)

---

CONCETTO: Il server attualmente invia `@localhost` come host in tutti i messaggi di notifica (JOIN, MODE, KICK, TOPIC). Questo non permette ai client di identificare correttamente la provenienza del messaggio.

COSA MANCA: L'utilizzo dinamico dell'hostname del client che ha generato l'evento.

DA FARE: Sostituire `"@localhost"` con `"@" + client.getHostname()` nelle seguenti funzioni:

1. Channel.cpp:
   - sendJoinChange
   - sendTopicChange
   - sendModeChange
   - sendKickChange
2. HandleBuffer.cpp:
   - handleTopic (nella risposta RPL_TOPICWHOTIME)