CHAT

Introduzione

La chat è una applicazione che tutti noi usiamo quotidianamente, essa permette a 2 o più utenti di scambiarsi messaggi tramite una comunicazione istantanea, nella quale verranno trasmessi messaggi di vario tipo passando dal semplice file di testo a file multimediali.
Per realizzare ciò è opportuno sviscerare sulle tecnologie che andremo ad utilizzare per la corretta realizzazione di questo progetto.

Tecnologie utilizzate

-Come primo punto la chat segue il modello client-server,nel quale un server si occupa di gestire tutte le connessioni e l'inoltro di tutti i messaggi ai vari client che gli si connettono richiedendo di accedere a tale servizio, mentre ogni client potrà connettersi al server e dopodichè potrà scegliere se e a chi inviare i messaggi.

-Per la comunicazione client-server dovremmo necessariamente scegliere il tipo di protocollo da adottare, nel nostro progetto utilizzeremo il tcp, tale protocollo appartiene al livello trasporto del modello tcp/ip.
Quest'ultimo in particolare permette di instaurare una connessione sicura, affidabile e full-duplex tra due processi, ciò significa che ogni host invierà una conferma dell'arrivo del pacchetto, confermando l'arrivo dei pacchetti all'altro host, durante l'inserimento della connessione avvera il three-way-handshake ovvero lo scambio di tre pacchetti tra client e server che permette appunto la scelta di un percorso affidabile.

-Inoltre, sempre al livello di trasporto del tcp/ip dobbiamo usufruire della tecnologia dei socket, questa tecnologia serve a smistare univocamente i pacchetti tra i vari processi, in modo da far comunicare più applicazioni, infatti Il socket è formato dal protocollo(tcp o udp), l'indirizzo ip e la porta logica.
inoltre esistono 2 tipi di socket, stream socket e datagram socket, si differenziano soltanto per il tipo di protocollo che andremo ad adottare.
Quindi se noi scegliamo il protocollo tcp allora utilizzeremo lo stream socket, invece se scegliamo l'udp utilizzeremo il datagram socket.

-I socket utilizzano le socket API, esse sono delle primitive che si occupano di gestire lo stato dei socket, che sono di fondamentale importanza come per esempio:

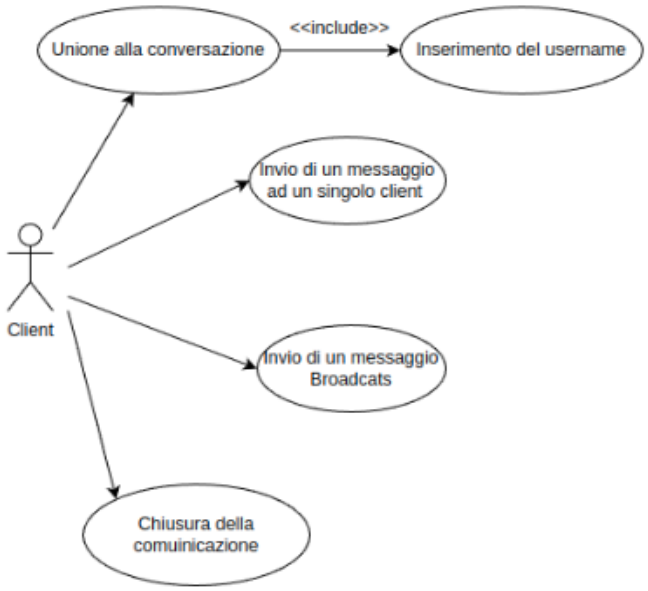
-accept() → istanza una connessione;
-close() → chiude una connessione;
-write() →trasmette dati da una connessione attiva;
-read() →riceve dati da una connessione attiva.

-Nel nostro server permetteremo di fare comunicare non un singolo client ma più client contemporaneamente. Infatti il nostro server sarà MultiThread, ovvero che quest'ultimo permette di eseguire più thread simultaneamente sia lato server, che lato client. Il server utilizzerà un thread per ogni client che gli si connette, mentre il client sfrutterà 2 thread, uno per la ricezione e uno per l'invio dei pacchetti.

-Per l'invio e la ricezione dei messaggi tra client-Server, abbiamo utilizzato JSON per trasformare delle strutture di dati in stream di byte e viceversa, tali processi si chiamano serializzazione e deserializzazione, nel nostro caso abbiamo scelto di utilizzare questo linguaggio perché è ritenuto più flessibile, e leggero rispetto al XML.

CASI D'USO

Client **Server**



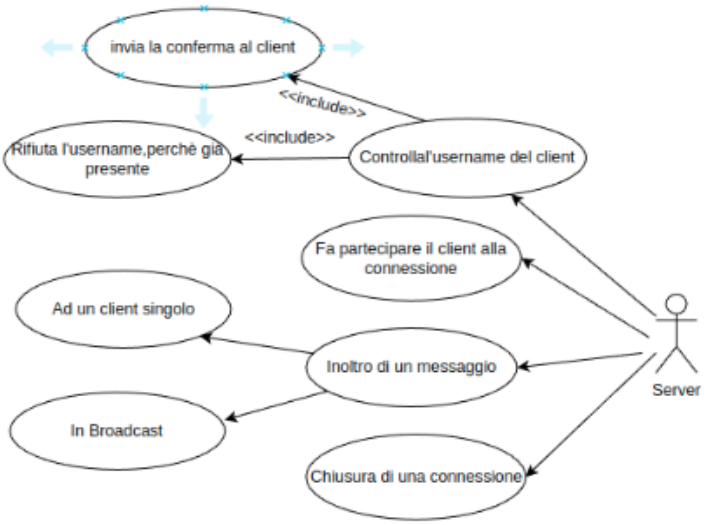
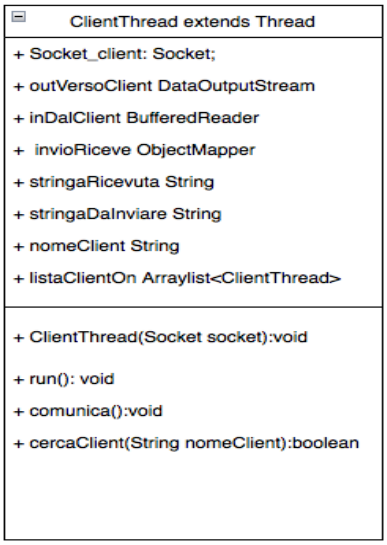
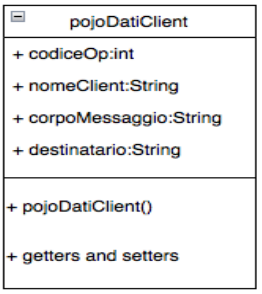
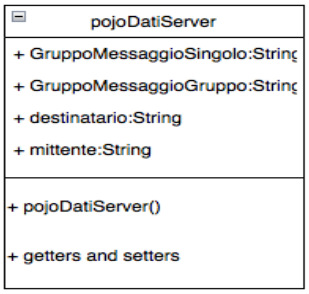


Diagramma delle classi

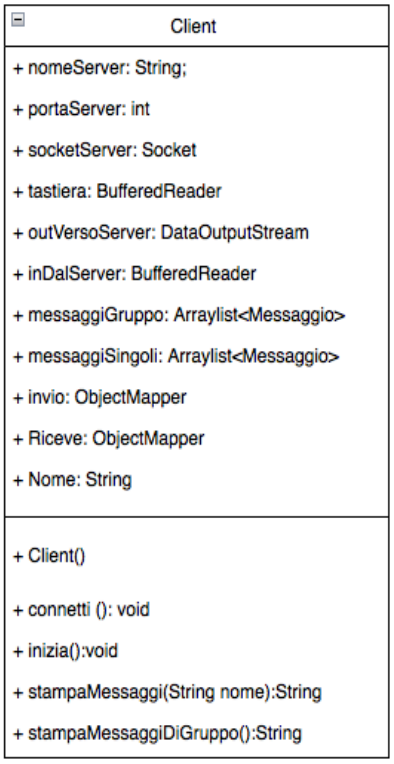
SERVER

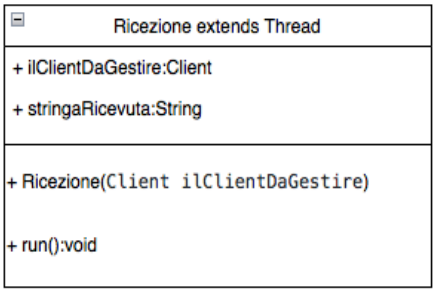


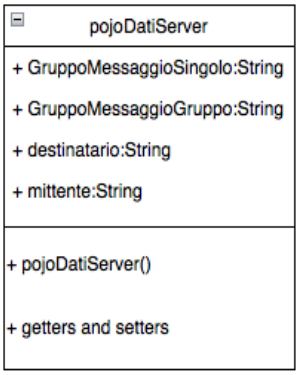


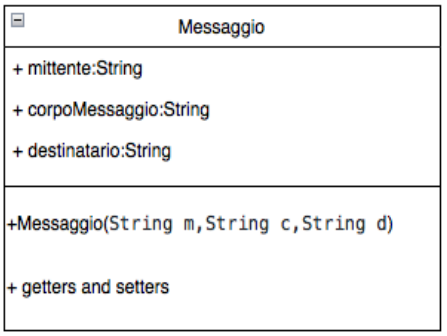


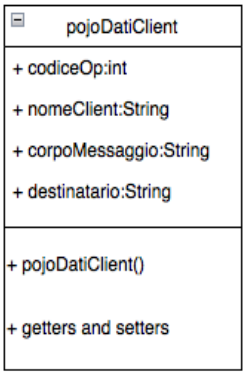
CLIENT











Messaggi scambiati tra client-server