Reti di telecomunicazioni - LISI
Gugoasa Cosmin: 282322 Lo Casto Mattia: 283095 Savastre Cosmin Gabriele: 283110

# LISI IRC

# Applicativo Instant Messaging di tipo Client-Server

## Obiettivo

Realizzare una piattaforma di Instant Messaging (IM), composta da due o più UA e da un server centrale servizio di messaggistica base simile a Whatsapp con o senza possibilità di caching dei messaggi sul server in caso di mancato inoltro al UA remoto.

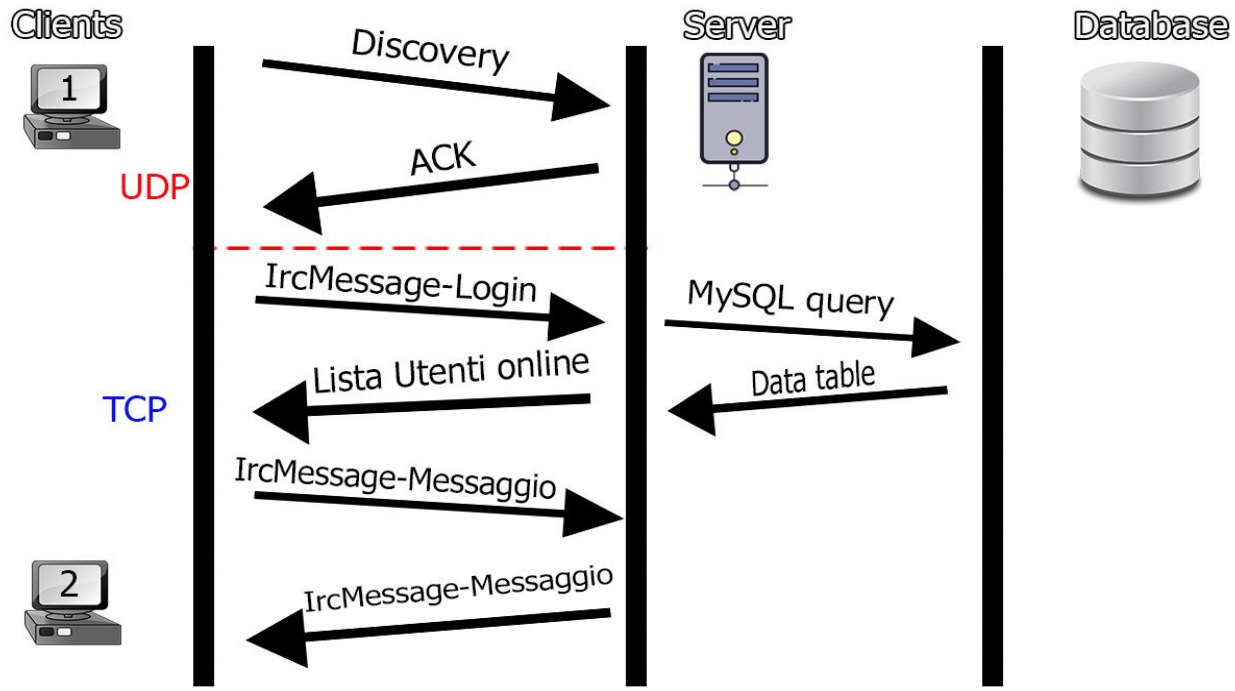
Implementazione:

- Protocollo di messaggistica ad-hoc oppure standard, SIP, XMPP, MQTT
- UA
- Server

## Descrizione

L'applicazione consiste in una piattaforma di Instant Messaging client-server, con registrazione ed autenticazione per ogni client, per poi poter inviare e ricevere messaggi da diversi utenti contemporaneamente.

## Architettura




### Client

Una volta avviato l'applicativo **Client** viene avviata una ricerca dei server disponibili usando pacchetti UDP trasmessi in broadcast sulla rete locale. Questi pacchetti verranno poi ricevuti da eventuali server presenti sulla stessa rete che risponderanno anche essi con un pacchetto UDP di acknowledgment (ACK) per rendere nota la propria presenza al client.

Una volta ricevute le presenze dai server il client cercherà di avviare connessioni TCP mirate ai server in modo da controllare che siano ancora disponibili, se il tentativo di connessione TCP fallisce il client agirà in modo adeguato rimuovendo il server dalla lista di server disponibili.

L'utente dopo aver scelto il server con cui comunicare per accedere ai servizi di messaggistica dovrà eseguire un processo di registrazione oppure login nel caso avesse un account già registrato. Completato questo step all'utente verrà mostrato un form con la lista di utenti online collegati a quel server con cui poter avviare una chat e scambiare messaggi. Qualora l'utente riceva un messaggio da un utente online verrà avviata una nuova chat con quel determinato utente per la visualizzazione dei messaggi ricevuti.

## Server

All'avvio dell'applicativo **Server** verrà chiesto di inserire un nome da associare al server questo nome poi apparirà nella lista di server disponibili sugli applicativi client che vengono eseguiti sulla stessa rete locale.

Una volta scelto un nome, il server rimarrà in ascolto ed attiverà connessioni UDP e TCP che rispettivamente vengono usate per:

- UDP :
    - In ascolto di pacchetti contenenti il messaggio `DISCOVER_IRC_REQUEST` inviati dai client alla ricerca di server disponibili.
    - In risposta per l'invio di pacchetti acknowledgment `DISCOVER_IRC_REPLY:NOMESERVER`
- TCP:
    - Per confermare la propria presenza ai Client una volta inviato il messaggio di acknowledgment tramite UDP.
    - Per reindirizzare messaggi tra Client.
    - Per verificare la presenza dei Client cercando di avviare connessioni TCP e rimuoverli dalla propria lista di utenti online in caso la connessione fallisca.
    - Per inviare liste di utenti aggiornate ai vari Client che hanno eseguito l'accesso su quel particolare server.

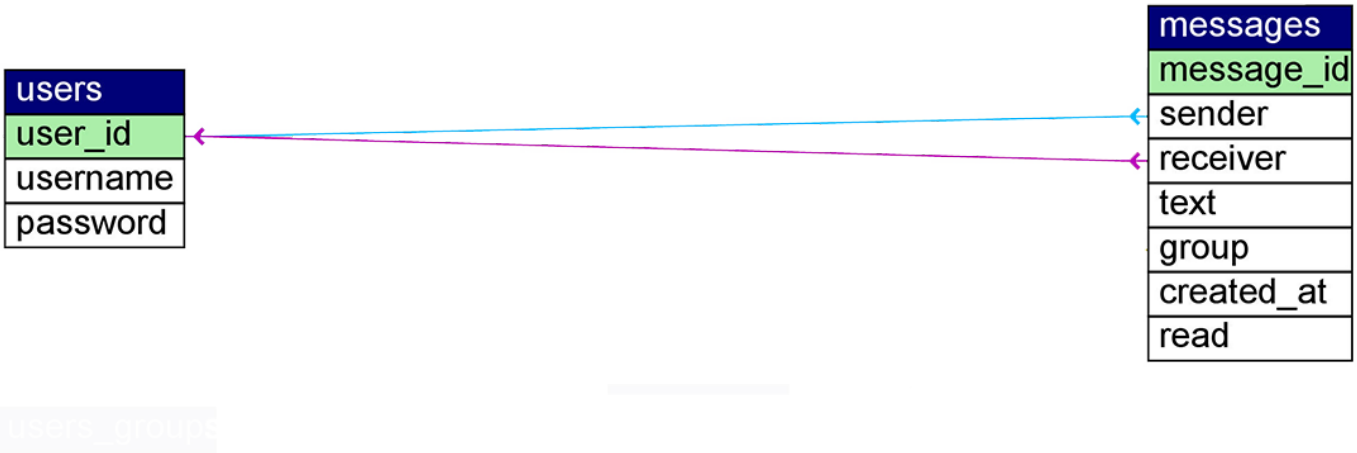
# Tecnologie utilizzate

Il progetto è stato sviluppato utilizzando il linguaggio di programmazione C# e le relative librerie presenti (LINQ, Threading, Etc…). Il progetto è stato creato utilizzando un'architettura client-server in cui i client dispongono di un'interfaccia grafica in modo da rendere l'uso dell'applicativo più intuitivo.

Utilizzo di un protocollo ad hoc per i pacchetti da inviare trasportati tramite tcp.

Inoltre viene previsto anche l'utilizzo di un database di tipo MySQL hostato sul servizio gratuito Alwaysdata.com per la memorizzazione degli utenti registrati e dei messaggi inviati verso utenti offline su un dato server.

## Schema ER del database



# Protocollo ad hoc

E' stato realizzato un protocollo ad hoc orientato agli oggetti per la costruzione dei pacchetti inviati con il protocollo TCP.

## IrcMessage

E' stata creata una classe denominata **IrcMessage** con i seguenti attributi:

- Username del mittente.
- Username del destinatario.
- Testo del messaggio
- Tipo di richiesta contenuta nel messaggio.

Affinché si possa costruire un messaggio in base alle necessità che esso richiede (che sia l'invio di un testo o una richiesta di registrazione) è stato utilizzato l'overloading dei costruttori dell'oggetto IrcMessage che permettono all'applicativo di costruirlo in base alle proprie esigenze.

Affinché un oggetto possa essere inviato attraverso la rete locale è stato necessario implementare i seguenti metodi statici:

- **ObjToBytes()**: L'oggetto IrcMessage viene convertito in array di byte per poter essere inviato con TCP attraverso la rete verso un Client o Server.
- **BytesToObj()**: Converte l'array di byte ricevuto attraverso una comunicazione TCP in un oggetto per poter essere utilizzato dall'applicativo.

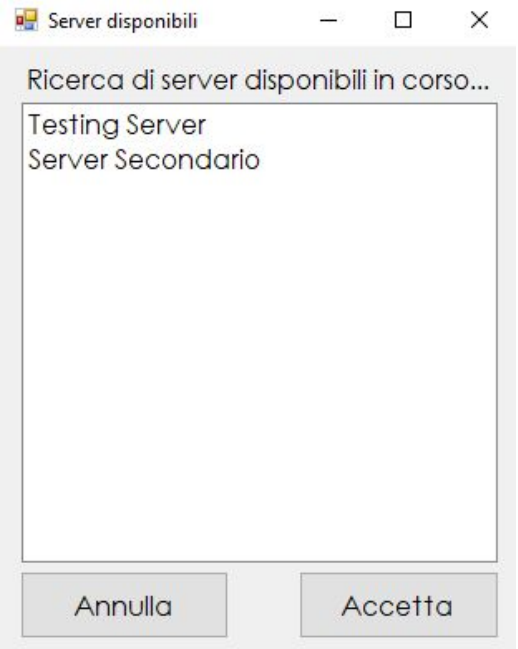
## IrcUser

Classe utilizzata per la creazione di oggetti che rappresentino gli utenti contenenti username e password in modo tale che dal lato client si abbia il proprio oggetto utente contenente i dati e dal lato server per creare una lista di utenti online e poterli differenziare tramite username per permettere l'aggiornamento della lista.

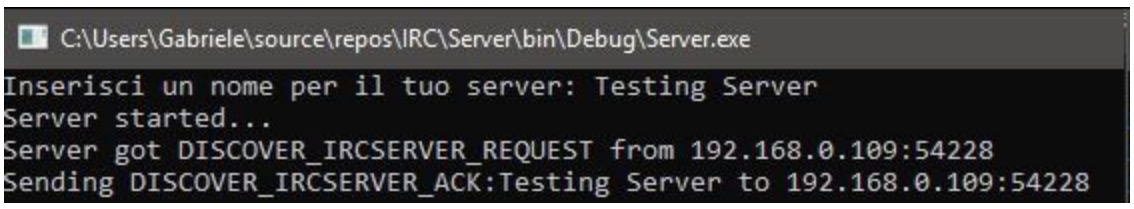
# Funzionalità

### Server Discovery

Questa funzione dal lato client prevede un form in cui viene mostrata la lista di server disponibili che hanno risposto ai pacchetti UDP inviati dal client in broadcast. Successivamente alla scelta del server si verrà reindirizzati alla schermata di login.




Lato server questa funzione è chiamata `discoveryListener` con porta di ascolto statica affidata ad un thread in parallelo che stia sempre in ascolto di pacchetti UDP, con formato standard per il riconoscimento, inviati dai client in modo tale che il server riferisca la propria disponibilità rispondendo con un ACK.

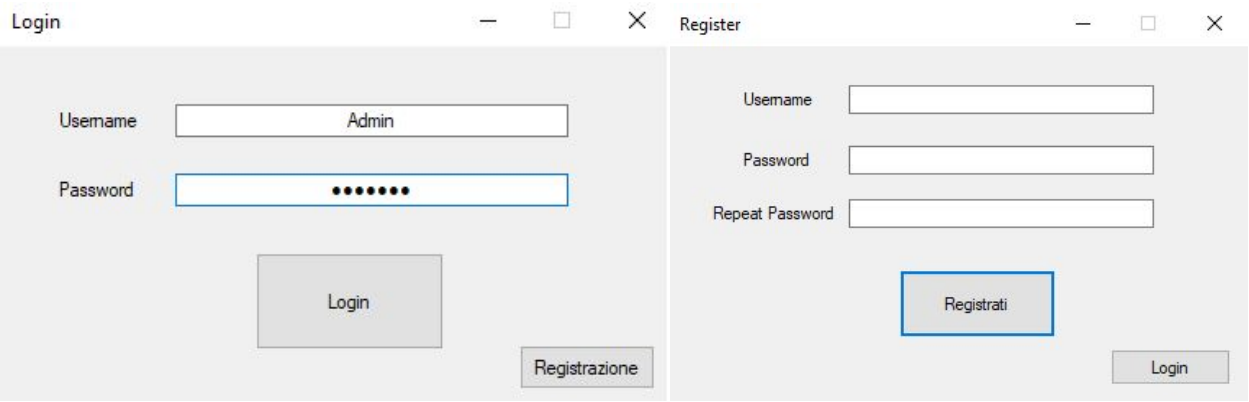


## Login e Register

Queste due funzionalità permettono all'utente di effettuare il login o la registrazione per identificarsi ed accedere ai servizi tramite protocollo ad hoc trasportato con TCP.

Dal lato client verrà mostrato un client grafico per il login con funzionalità di switch che permette di passare alla registrazione, in cui si inseriscono username e password e si invia la richiesta al server.

Da lato server invece viene ricevuto il pacchetto, viene poi identificata l'azione richiesta dal client ed effettuate le operazioni di login o registrazione tramite le funzioni implementate nel `DBManager`. Infine il server invia un messaggio al client per confermare il successo dell'operazione e quindi essere reindirizzati alla home oppure il fallimento.




```
LOGIN_USER_REQUEST Received
Inizio processo di login per Admin:segreto
Login avvenuto con successo per Admin
```

## Ping Users

Questa funzionalità, implementata con un thread sul server, invia pacchetti di ping TCP agli utenti online in modo tale da poter verificare lo stato degli utenti ed andare ad aggiornare la lista degli utenti online.

```
09:14:49:Invio ping TCP a Admin all'indirizzo 192.168.1.5
09:14:49:Connessione TCP avvenuta con successo per Admin
```

## DBManager

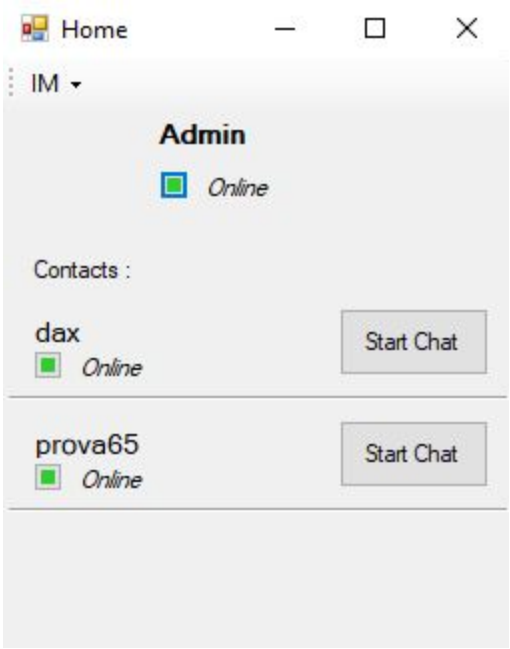
Classe statica implementata per connettersi al database ed effettuare operazioni sul database. Contiene funzioni di connessione al database, chiusura della connessione, insert, update, delete, select ed inoltre prima di effettuare operazioni sul db verifica che la tabella richiesta e la colonna della tabella siano presenti nel database.

Le query da inviare al DB sono implementate in modo dinamico in modo che in base ai valori che il server passa alle funzioni del DB Manager si crei la corretta query SQL.

Le operazioni che modificano il DB sono state implementate utilizzando le transazioni in modo tale da poter eseguire un'operazione di rollback per tutte le azioni di tipo DML.

## Home

Lato client prevede un form che visualizza la lista sempre aggiornata di utenti online fornita dal server e ne permette l'inizio di una conversazione da cui si aprirà il form "Chat".

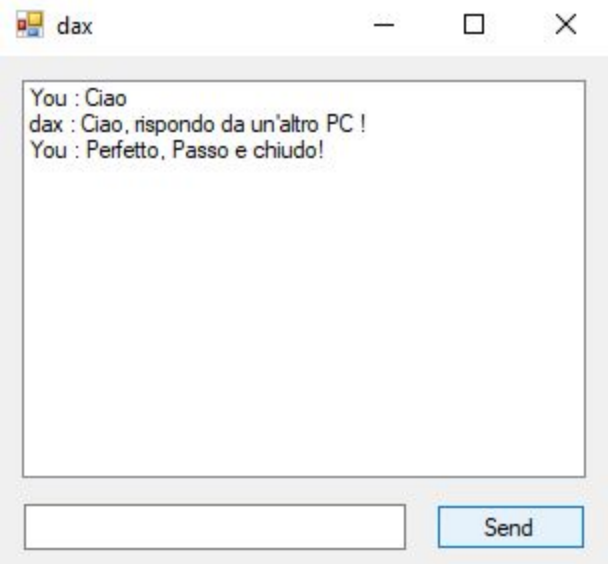



Inoltre per non bloccare l'utilizzo del form "Chat" si è implementato un thread in ascolto dei messaggi che vengono inviati all'utente e se non è ancora aperta una chat con quell'utente, che ha inviato il messaggio, viene aperta automaticamente, altrimenti aggiorna la chat già aperta inserendo graficamente il messaggio.

Lato server prevede la ricezione dei pacchetti inviati dai client, ne identifica il tipo di richiesta (in questo caso messaggio) e lo indirizza al destinatario.

### Chat

Viene visualizzata qualora l'utente inizi una conversazione oppure riceva un messaggio da un utente online con cui non abbia già instaurato una conversazione. Questo form si limita alla visualizzazione dei messaggi ricevuti dall'utente, ricevuti tramite il thread in ascolto presente in "home", e permette all'utente di inviare messaggi al destinatario della chat.




# Sommario