**Progetto per il corso di Algoritmi Distribuiti**

Federico Bertoli Matricola: 157896

**Introduzione**

Per questo progetto ho deciso di implementare una semplice replica del social network Twitter, utilizzando Java RMI.

Immagine che contiene testo, schermata, Carattere, numero

Descrizione generata automaticamenteLa struttura del progetto è composta dai componenti principali Server e Client, metodi di utilità aggiuntivi come il file di costanti, i modelli dei dati scambiati tra le componenti, il DAO (Data Access Object) e l’interfaccia dei metodi implementati dal server e dal client.

Le funzionalità offerte sono le seguenti:

**Server**

* Registrazione al servizio (creazione handle come Twitter)
* Login utente
* Scrittura post
* Commento di un post
* Possibilità di aggiungere un like ai post
* Possibilità di seguire le persone (concetto di following) e dunque di essere seguiti (concetto di followers)
* Recupero dei soli post delle persone che si seguono
* Recupero di tutti i post
* Eliminazione di un proprio post

**Client**

* Implementazione di una UI apposita da terminale
* Scambio di messaggi diretti tra utenti

**Metodi implementati e descrizione**

Figura 1 Struttura del progetto

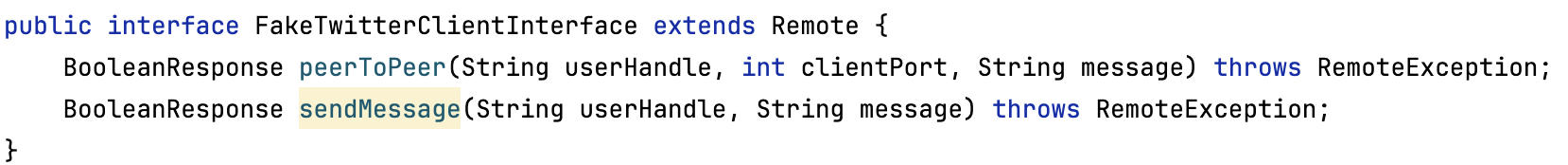
Il server implementa la seguente interfaccia:

**Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente**

Ad ogni operazione effettuata sugli utenti e sui post, ove necessario, le informazioni vengono salvate su file per garantirne la persistenza al riavvio.

Il client implementa invece questa interfaccia:



**Avvio:**

Appena avviato, il Server si configura ed effettua le operazioni di verifica e caricamento dei dati persistenti.

Il client mostra l’interfaccia iniziale, dove l’utente può registrarsi o, se già registrato, fare login:

Immagine che contiene testo, Carattere, ricevuta, bianco

Descrizione generata automaticamente

Figura 2 Avvio del programma

I metodi utilizzati in questa fase sono i seguenti:

**registerUser**(String userHandle):

Permette la registrazione dell’utente. È presente un controllo sull’esistenza dello stesso handle per evitare duplicati.

**login**(String userHandle):

Permette l’accesso ad un utente già registrato. Se l’utente non è presente, viene invitato ad effettuare prima la registrazione.

**Post Registrazione / Login:**

Il client visualizza il menu principale, dove viene offerta la possibilità di visualizzare i post degli utenti che si seguono o quelli globali. Inoltre, è presente la funzione di visualizzazione della lista utenti, per la messaggistica istantanea tra client.

Immagine che contiene testo, Carattere, ricevuta, schermata

Descrizione generata automaticamente

Figura 3 Menu principale

I metodi utilizzati sono:

**getLatestPosts():**

Recupera la lista di tutti i post presenti fino a quel momento, senza distinzione di utenti.

**getFollowedPosts**(String userHandle):

Recupera solamente i post degli utenti seguiti da quel particolare utente, se presenti.

**getClientsList**(String userHandle):

Recupera la lista degli utenti e le informazioni dei loro client, per permettere la comunicazione diretta utente – utente.

**newPost**(String userHandle, String post):

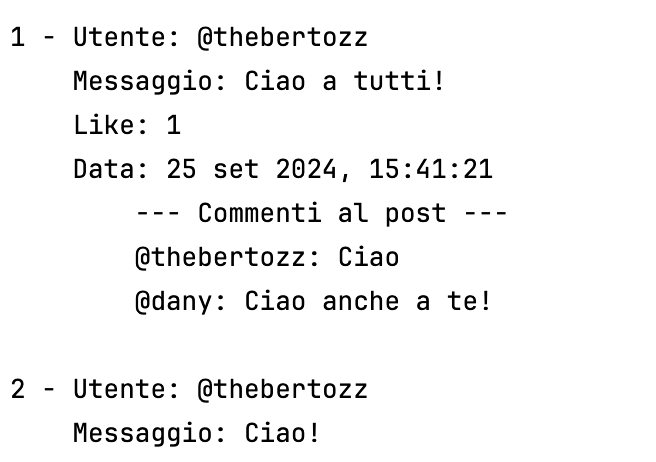
Crea un nuovo post, aggiungendolo alla lista di quelli già presenti.

Figura 4 Visualizzazione dei post degli utenti

**Opzioni :**

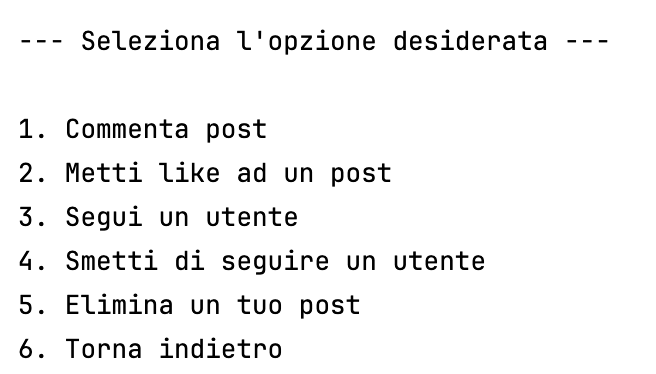
Selezionando di visualizzare la lista post, è quindi possibile inserire commenti, like, seguire un utente o eliminare un proprio post.

Figura 5 Opzioni per i post

**Metodi utilizzati:**

**commentPost**(String userHandle, String postUuid, String comment):

Aggiunge un commento al post selezionato; questo sarà visualizzato al di sotto del post. È possibile aggiungere un numero potenzialmente infinito di commenti.

**likePost**(String userHandle, String postUuid):

Aggiunge un like al contatore di ogni post, visibile nell’interfaccia utente.

**followUser**(String follower, String followed):

Permette di seguire un utente. È presente il controllo per evitare che un utente possa seguire sé stesso.

**unFollowUser**(String follower, String followed):

Permette la rimozione di un utente dalla propria lista di utenti seguiti.

**deletePost**(String userHandle, String postUuid):

Permette l’eliminazione di un post; questa è possibile solamente se la cancellazione è richiesta dall’utente che l’ha creato.

**Messaggistica istantanea:**

Per implementare questa funzionalità, i client fungono anche da server per permettere ad un altro client di contattarli e scambiare i messaggi; questo è implementato tramite un meccanismo per il quale il client all’avvio si configura su una determinata porta, fornita dal serve: di fatto il server funge da coordinatore dei client, permettendogli di ottenere porte con numerazione incrementale per evitare conflitti.

I metodi utilizzati sono:

**registerNewClient**(String host, String userHandle): | Server

Registra un nuovo client; questo metodo aggiunge al pool dei client presenti a runtime nel server il nuovo client, fornendogli come ritorno alla chiamata la porta che dovrà essere utilizzata.

**unRegisterClient**(String userHandle): | Server

Permette al client di chiedere al server di deregistrare la sua presenza online, evitando così che gli utenti possano inviargli messaggi diretti.

**peerToPeer**(String userHandle, int clientPort, String message): | Client

Metodo che configura il client per permettere la comunicazione, registrandosi per la comunicazione sulla porta fornita dal server

**sendMessage**(String userHandle, String message): | Client

Immagine che contiene testo, Carattere, bianco, ricevuta

Descrizione generata automaticamenteMetodo che permette all’altro client di ricevere il messaggio inviato.

Immagine che contiene testo, schermata, Carattere, bianco

Descrizione generata automaticamente

Figura 6 Conferma della registrazione del client sulla porta specificata dal server

Figura 7 Visualizzazione della lista degli utenti attivi

**Esempio di funzionamento**

**Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente**

**@thebertozz:**

**Immagine 14**

**@dany:Immagine che contiene testo, Carattere, schermata, bianco

Descrizione generata automaticamente**

**@dany:**

**Immagine 16**

**@thebertozz:**

**Diagramma UML del progetto**

Per descrivere meglio le relazioni tra le varie componenti, è stato generato un diagramma UML: