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.
La 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.

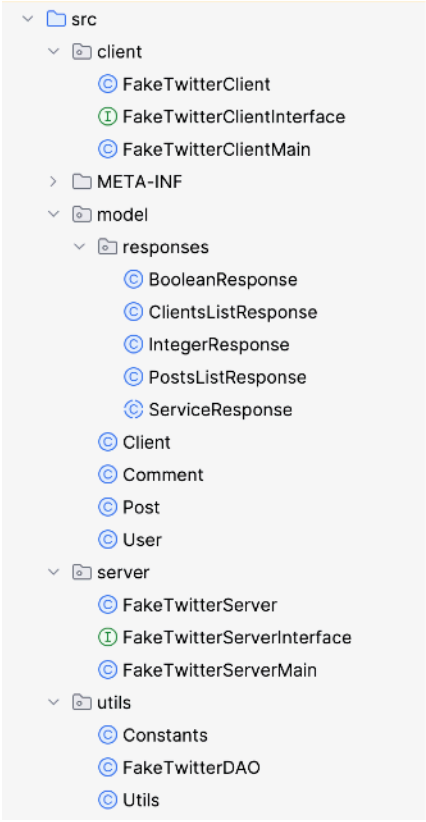

Figura 1 Struttura del progetto

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

Il server implementa la seguente interfaccia:

```
public interface FakeTwitterServerInterface extends Remote {
    BooleanResponse registerUser(String userHandle) throws RemoteException;
    BooleanResponse login(String userHandle) throws RemoteException;
    BooleanResponse newPost(String userHandle, String post) throws RemoteException;
    BooleanResponse deletePost(String userHandle, String postUuid) throws RemoteException;
    BooleanResponse likePost(String userHandle, String postUuid) throws RemoteException;
    BooleanResponse followUser(String follower, String followed) throws RemoteException;
    BooleanResponse unFollowUser(String follower, String followed) throws RemoteException;
    PostsListResponse getLatestPosts() throws RemoteException;
    PostsListResponse getFollowedPosts(String userHandle) throws RemoteException;
    BooleanResponse commentPost(String userHandle, String postUuid, String comment) throws RemoteException;
    ClientsListResponse getClientsList(String userHandle) throws RemoteException;
    IntegerResponse registerNewClient(String host, String userHandle) throws RemoteException;
    BooleanResponse unRegisterClient(String userHandle) throws RemoteException;
}
```

Ad ogni operazione effettuata sugli utenti e sui post, ove necessario, le informazioni vengono salvate su file per garantirne la **persistenza** al riavvio. Questo è possibile attraverso la classe FakeTwitterDAO.

Il client implementa invece questa interfaccia:

```
public interface FakeTwitterClientInterface extends Remote {
    BooleanResponse peerToPeer(String userHandle, int clientPort, String message) throws RemoteException;
    BooleanResponse sendMessage(String userHandle, String message) throws RemoteException;
}
```

Avvio del programma:

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

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

```
--- Benvenuti in Fake Twitter! --- Registrati o effettua il login

1. Registrati
2. Effettua il login
```

Figura 2 Avvio del programma client

I metodi utilizzati in questa fase sono i seguenti:

registerUser(String userHandle): | Server

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

login(String userHandle): | Server

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.

```
--- Cosa vuoi fare oggi, @thebertozz? ---

1. Nuovo post
2. Mostra la lista di tutti i post
3. Mostra la lista dei post di chi segui
4. Mostra la lista degli utenti del servizio
5. Esci
```

Figura 3 Menu principale

I metodi utilizzati sono:

newPost(String userHandle, String post): | Server

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

getLatestPosts(): | Server

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

getFollowedPosts(String userHandle): | Server

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

getClientsList(String userHandle): | Server

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

```
1 - Utente: @thebertozz
    Messaggio: Ciao a tutti!
    Like: 1
    Data: 25 set 2024, 15:41:21
        --- Commenti al post ---
        @thebertozz: Ciao
        @dany: Ciao anche a te!

2 - Utente: @thebertozz
    Messaggio: Ciao!
```

Figura 4 Visualizzazione dei post degli utenti, con commenti e like

Opzioni :

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

```
--- Seleziona l'opzione desiderata ---

1. Commenta post
2. Metti like ad un post
3. Segui un utente
4. Smetti di seguire un utente
5. Elimina un tuo post
6. Torna indietro
```

Figura 5 Opzioni per i post

Metodi utilizzati:

commentPost(String userHandle, String postUuid, String comment): | Server

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): | Server

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

followUser(String follower, String followed): | Server

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

unFollowUser(String follower, String followed): | Server

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

deletePost(String userHandle, String postUuid): | Server

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 server: di fatto quest'ultimo 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

Metodo che permette all'altro client di ricevere il messaggio inviato.

```
Welcome back, thebertozz!

Registrazione per la messaggistica completata. Porta: 1100
```

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

```
--- Lista degli utenti registrati ---

1 - @dany

--- Seleziona l'opzione desiderata ---

1. Chatta con un utente
2. Torna indietro
```

Figura 7 Visualizzazione della lista degli utenti attivi

Esempio di funzionamento

@thebertozz:
```
--- Messaggio (Q per uscire) ---
Ciao! Come va oggi?

Messaggio inviato!
```

@dany:
```
Messaggio ricevuto da @thebertozz: Ciao! Come va oggi?
```

@dany:
```
--- Messaggio (Q per uscire) ---
Tutto bene, grazie! Qui c'è il sole

Messaggio inviato!
```

@thebertozz:
```
Messaggio ricevuto da @dany: Tutto bene, grazie! Qui c'è il sole
```

Diagramma UML del progetto

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