PROGETTO WORTH

Michele Velardita 578770

A.A. 2020/2021

Indice

1. Architettura generale

Lato Client

Il Client è costituito dal MainClient, dal DataClient e dai quattro Frame: LoginFrame, MiddleFrame, ProjectFrame e ChatFrame.

Nel MainClient viene effettuato l'accesso al registry attivato sul Server e viene creata un'istanza di DataClient. In particolare viene invocato il metodo remoto getUser() (riga 32) per caricare sul DataClient gli utenti registrati al servizio e il relativo stato (online/offline). Infine viene aperto un LoginFrame, ovvero la pagina di accesso.

Il DataClient contiene tutti I dati utili al funzionamento del client: la Socket per la comunicazione TCP con il Server (inizializzata nel metodo costruttore del LoginFrame), l'username del client (inizializzato al momento del login o registrazione avvenuto con successo) e un HashMap contenente tutti gli utenti registarti al servizio e il loro stato. Questa verrà poi aggiornata in modo asincrono tramite la NotifyEvent(utente, stato) presente nel MainClient.

Il LoginFrame viene creato nel MainClient passandogli il DataClient e lo stub.
Da qui è possibile compiere due operazioni: Login o Registrazione. La Registrazione efettuata con successo ha lo stesso effetto del Login, ovvero viene chiamato il metodo dispose() sul frame corrente e viene creato un MiddleFrame.

Dal MiddleFrame è possibile compiere quattro operazioni principali (corrispondenti ai quattro tasti presenti nel panel): Visualizzare la lista degli utenti registrati al servizio, visualizzare I progetti a cui l'utente appartiene, unirsi ad un progetto o crearne uno nuovo. Queste ultime due operazioni (se a buon fine) chiameranno la dispose() sul frame corrente e apriranno il ProjectFrame e il ChatFrame (settato a invisible fino a quando l'utente non premerà il tasto 'join chat' presente nel ProjectFrame). Insieme al ChatFrame viene inizializzato un Thread che si occupa di mandare I messaggi in multicast e riceverli per poterli poi stampare sul panel.
E' inoltre presente un tasto per poter effettuare il logout, il quale chiude la socket, il frame corrente ed istanzia un nuovo LoginFrame.

Nel ProjectFrame è implementata la maggior parte delle operazioni: tutte le operazioni relative alle card, la cancellazione del progetto, l'invito ad altri utenti e la 'join chat'. La classe prende come riferimento (oltre alla stub e al DataClient), la multicast socket per poterla chiudere nel caso in cui l'utente chiude la finestra (righe 333-350).

Lato Server

Il Server è formato dalla classe principale MainServer, dalle classi di supporto Card, Project, User e Addres, e dall'interfaccia ServerInterface.

Il MainServer ha una lista degli utenti con le relative informazioni, una lista dei progetti, una map<Socket, String> per memorizzare le socket dei client e I relativi dati da inviare, una lista di NotifyEventInterface per effettuare le callback, un Address in cui memorizza l'ultimo indirizzio usato per il multicast e una lista di Address contenente gli indirizzi dei progetti cancellati, quinid indirizzi riutilizzabili.

La classe Project contiene le quattro liste in cui possono essere inserite e card (TODO, INPROGRESS, TOBEREVISED e DONE), la lista di membri e l'indirizzo utilizzato per la chat di progetto.

La classe Card contiene nome e descrizione della card, la history e lo stato corrente.

La classe Address contiene semplicemente un numero di porta e un ip (rappresentato come un array di 4 dimensioni).

La classe User contiene I dati dell'utente, quindi username, password e stato (online/ offline)

2. Scelte di progetto

Il server è single-Threaded e utilizza un selector per il multiplexing dei canali e qundi per gestire la connessione simultanea con più client.
All'interno del metodo start() del MainServer vengono elobarate le richieste da parte dei client, le quali hanno tuttu lo stesso formato:

tipoRichiesta argomento1 argomento2 argomento3 …

I parametri vengono separati da una StringTokenizer (usando come separatore lo spazio). Alla ricezione della richiesta e dei parametri viene chiamata la funzione opportuna e memorizzata la risposta (ovvero il risultato della funzione) nell'HashMap<Socket, String> clients, per poter essere inviata alla socket corretta. Se la risposta memorizzata è un "Logout successfully" allora la socket del client che ha effettuato il logout viene rimossa dalla HashMap.

Indirizzi multicast e riuso

Viene assegnato a ciascun progetto un indirizzo di multicast a partire da 224.0.0.0 a 239.255.255.255. Se un progetto viene cancellato, il suo indirizzo sarà memorizzato in una lista (free_addr). Il campo addr contiene l'ultimo indirizzo utilizzato. Per assegnarne uno viene prima controllata la lista free_addr e nel caso sia vuota prendo l'indirizzo successivo a addr.

Serializzazione

Per la persistenza dei dati del server serializzo usando le librerie jackson (versione 2.11.2).
Il backup dei dati viene effettuato alla chiusura del server per mano di un thread (righe 545-555) che cattura il segnale di SIGINT e chiama il metodo savedata() e imposta tutti gli stati degli utenti in "offline".
Nel metodo savedata, prima di effettuare il backup, elimino le cartelle dei progetti per evitare inconsistenze, poichè un progetto precedentemente salvato in una sessione precedente potrebbe essere stato cancellato nella sessione corrente (righe 118-122).
I dati che vengono preservati sono:

- Lista degli utenti registrati
- Ultimo indirizzo ip e porta usati per gestire la chat di progetto
- Lista degli indirizzi inutilizzati (relativi ai progetti eliminati)
- Progetti e le relative card, in particolare:
  ogni progetto è una cartella contenente tre tipi di file .json:
  card, Address e Lista membri

All'apertura del server viene invocato il metodo restoredata() per ricostruire I dati del server.

3. Thread e concorrenza

Lato Server

Essendo il server gestito tramite Selector, non uso particolari metodi per sincronizzare le operazioni tranne che per I metodi remoti di callback (registerForCallback, unregisterForCallback e callbacks); per questi metodi utilizzo il costrutto synchronized per rendere atomica l'esecuzione del metodo.

Lato Client

Il Client è costituito da due Thread: Il MainClient con l’esecuzione sequenziale dei vari frame e il ChatFrame.
La Classe ChatFrame implementa l’interfaccia Runnable, e il suo metodo run si preoccupa solamente di ricevere I pacchetti ricevuti in multicast, leggerli e stamparne il contenuto nel panel. In questo modo è possibile chiamare concorrentemente il metodo sendmsg(String msg) e mandare messaggi sulla stessa Socket.

4. Compilazione

Eseguire prima lo script per compilare Client e Server, in seguito eseguire lo script per eseguire l’out del server e dopo il client.

~$ bash script.sh