UNIVERSITÀ DI PISA

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

WINSOME

Reti di Calcolatori e Laboratorio

Prof.ssa Federica Paganelli
Prof.ssa Laura Emilia Maria Ricci

Erica Pistolesi
518169 Corso A

Anno Accademico 2021/2022

Indice

Introduzione

WINSOME, acronimo di reWardINg SOcial MEdia, è una piattaforma social la cui caratteristica principale è quella di offrire una ricompensa agli utenti che pubblicano contenuti interessanti e ai loro curatori, ovvero chi vota/commenta tali contenuti.
Un utente all'interno di WINSOME, una volta che si è registrato e ha effettuato il login, può interagirvi tramite alcune funzionalità, in seguito saranno mostrate alcune delle più caratterizzanti.
Al momento della registrazione, gli utenti dovranno indicare una serie di *tag*, parole chiave che indicano i suoi interessi (minimo un tag e massimo cinque), tramite i quali potrà trovare altri utenti con interessi in comune con cui relazionarsi.
Gli utenti hanno la possibilità di gestire un *blog*, quindi pubblicare e rimuovere i post, seguire altri utenti, visualizzare il proprio *feed*, ovvero i vari blog degli utenti seguiti, valutare i contenuti degli altri utenti, effettuare il *rewin* di un post, cioè pubblicare sul proprio blog un post di un altro utente, e visualizzare il proprio portafoglio dove sono annotate le varie ricompense ottenute durante l'utilizzo di WINSOME.

Ambiente di sviluppo

Il progetto è stato sviluppato con i seguenti strumenti:

- IDE: Visual Studio Code versione 1.68.1
- OS: Ubuntu 20.04.4 LTS
- La versione di Java utilizzata è la numero 17, non sono state utilizzate funzioni introdotte da versioni successive a Java 8, tranne il metodo `readString`, disponibile da Java 11, durante il caricamento dello stato iniziale di WINSOME
- Per la serializzazione degli oggetti in file JSON è stata utilizzata la libreria GSON versione 2.8.9

Compilazione ed esecuzione

WINSOME può essere eseguito dai jar files, oppure compilando ed eseguendo separatamente i programmi client e server nel seguente modo:

```
#Supponendo di essere nella cartella del progetto

#Esecuzione dei jar
java -jar server.jar configServer.txt
java -jar client.jar configClient.txt

#Compilazione del server
javac -cp .:./libs/gson-2.8.9.jar server/*.java
#Esecuzione del server
java -cp .:./libs/gson-2.8.9.jar server.ServerMain configServer.txt

#Compilazione del client
javac client/ClientMain.java
#Esecuzione del client
java client.ClientMain configClient.txt
```

Architettura

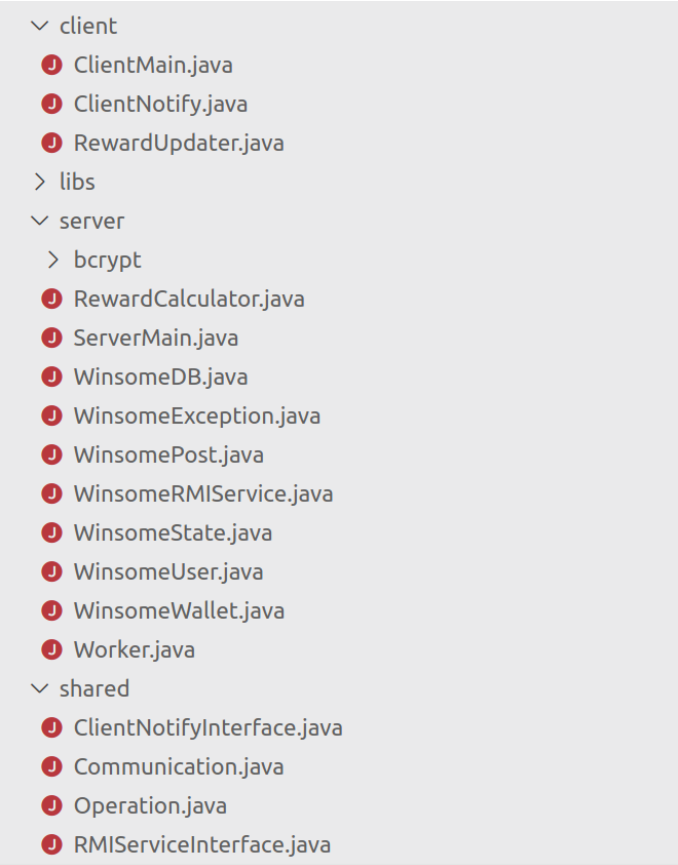
Il progetto è stato realizzato seguendo il paradigma client-server, la comunicazione segue diversi protocolli durante le varie fasi, in particolare si ha un protocollo richiesta/risposta su socket TCP, messaggi in multicast UDP e infine si fa anche uso della tecnologia RMI.

Il formato dei messaggi richiesta/risposta ha una struttura precisa che deve essere rispettata per permettere il corretto funzionamento degli scambi degli stessi ed è la seguente:
la richiesta è nella forma TipoOperazione;UtenteRichiedente;EventualiAttributi
dove il tipo di operazione è uno fra le operazioni consentite in WINSOME, l'utente richiedente rappresenta l'utente attualmente loggato con quel client, e infine gli eventuali attributi rappresentano i parametri necessari per alcune operazioni;

il server risponde con il codice 200 OK in caso di successo o con un messaggio che specifica il tipo di errore in caso contrario. Dopo la parte iniziale, terminata con una `newline`, sono presenti gli eventuali attributi. Il messaggio di risposta termina con `newline`.

Struttura del progetto

La figura sotto rappresenta la vista dei packages in cui è raggruppato il progetto.
All'interno del package client sono presenti le classi che implementano il client, quindi la classe ClientMain, che gestisce la connessione con il server, la configurazione dei parametri iniziali e il parsing delle richieste da linea di comando, la classe ClientNotify, che gestisce le notifiche ricevute tramite callback RMI, e la classe RewardUpdater che sta in ascolto del gruppo multicast per ricevere la notifica del calcolo delle ricompense.
All'interno del package shared sono presenti le interfacce ClientNotifyInterface e RMIServiceInterface che consentono la comunicazione con tecnologia RMI, inoltre vi sono le due classi Communication e Operation che supportano la comunicazione su socket TCP.



All'interno del package server sono presenti le classi che implementano il server, quindi le entità i thread, che saranno illustrate più dettagliatamente più avanti. Si noti

il package *bcrypt*: è stato importato per effettuare l'hashing con salt e la verifica delle password, rispettando licenza e copyright dell'autore.

Implementazione del server

Dettagli implementativi

Il server sfrutta il meccanismo del multiplexing dei canali per gestire la comunicazione con i client in modalità non bloccante. Il thread che se ne occupa è implementato tramite la classe Worker.

La scelta di delegare a un thread diverso dal main la gestione delle connessioni è stata la conseguenza di un progetto più elaborato in cui le varie richieste client venivano prese in carico da più thread worker per sfruttare al massimo l'efficienza del multithreading che poi non è stato realizzato. Altri piccoli dettagli simili si possono notare all'interno del codice, ad esempio la scelta di lasciare AtomicInteger come contatore dei post anche se soltanto un thread ne crea di nuovi e non si verificano corse critiche.
È stato quindi preferito l'utilizzo di un selettore che esamina uno o più canali e determina quali sono pronti per la lettura o la scrittura, ovvero quali client sono pronti a connettersi o ricevere o inviare dati, in alternativa all'approccio "un thread per connessione" che si verificherebbe con l'utilizzo di threadpool, per una migliore performance e per eventuali futuri miglioramenti all'architettura master-worker.

All'avvio del server, la classe ServerMain legge i parametri per la configurazione iniziale da un file che viene passato in ingresso al momento dell'esecuzione. I parametri presenti in tale file devono rispettare la precisa sintassi Parametro=valore, parametro con tutte lettere maiuscole, senza spazi vuoti e con ogni parametro su una nuova linea. I parametri da indicare sono:

- MULTICAST_ADDRESS → Indirizzo per il multicast
- MULTICAST_PORT → Porta per il multicast
- TCP_PORT → Porta per la socket TCP
- RMI_PORT → Porta per il servizio RMI
- RMI_NAME_SERVICE → Nome del servizio RMI
- REWARD_PERIOD → Periodo ogni quanto viene effettuato il calcolo delle ricompense
- PERC_AUTH → Percentuale di ricompensa che spetta all'autore del post
- AUTOSAVE_PERIOD → Periodo ogni quanto viene effettuato il salvataggio dello stato
- DATABASE → Nome del file dove salvare lo stato di WINSOME

Una volta stabilita la configurazione iniziale il ServerMain procede al ripristino dello stato di WINSOME se è presente un backup precedente, altrimenti inizializza le strutture, e crea il thread che si occupa dell'autosalvataggio periodico dello stato. In seguito viene istanziato l'oggetto remoto che fornisce il servizio di registrazione a WINSOME e al servizio di notifica per l'aggiornamento dei follower tramite callback con tecnologia RMI. Quindi viene creato il thread per il calcolo delle ricompense, viene aperta la connessione TCP e viene creato il thread per la gestione delle richieste client.
A questo punto il ServerMain avvia i vari thread (l'esecuzione di WINSOME adesso è a regime) e si sospende in attesa di un input da linea di comando, quando legge "quit" avvia la fase di terminazione. A questo punto sveglia i thread in caso fossero sospesi e setta loro un flag che indica la terminazione, effettua la `join` e quando ritorna termina a sua volta.

Strutture dati utilizzate

Le strutture dati che compongono WINSOME sono raccolte nella classe WinsomeDB, che rappresenta infatti il cuore del progetto. Le principali sono tre `Map` che contengono rispettivamente le informazioni relative agli utenti, ai post e ai tag. Sarebbe stata sufficiente la sola `Map` degli utenti per raccogliere tutte le informazioni relative a WINSOME, la scelta di mantenere una ridondanza è motivata dalla maggior efficienza nel raggiungere, quindi votare, commentare, ecc..., i vari post, operazioni piuttosto comuni e quindi ipoteticamente frequenti durante l'esecuzione del programma, e nel reperire gli utenti con tag in comune, anche questa un'operazione presumibilmente frequente soprattutto durante i primi utilizzi di WINSOME da parte di un utente.

Thread principali

Worker. Il thread implementato con la classe Worker è il thread che si occupa di gestire le connessioni e le richieste client. All'avvio, si mette in ascolto su un `Selector` per effettuare il multiplexing dei canali sui quali i client invieranno le richieste di connessione o le richieste per soddisfare le operazioni degli utenti. In caso di errori su un canale, questo viene chiuso. Al momento di una nuova connessione accettata, il canale relativo viene reso non bloccante e registrato per la lettura. Quando un canale è pronto per la lettura è possibile doverne effettuare più d'una per poter leggere la richiesta completa, in questo caso i vari frammenti di messaggio vengono salvati in `attachment`, per poi essere ricomposti nelle iterazioni successive. Solo quando la lettura è completa, il canale viene registrato per la scrittura.

Una volta ricevuta la richiesta del client, il thread la risolve chiamando i metodi di WinsomeDB, se l'esito è positivo la risposta indicherà il successo con il codice 200 OK ed eventuali attributi saranno scritti a seguito, altrimenti la risposta conterrà il motivo del fallimento. La risposta viene quindi salvata in `attachment` al canale e poi inviata.

WinsomeState. Il thread che si occupa di salvare in modo persistente lo stato di WINSOME è implementato con la classe WinsomeState. Periodicamente recupera da WinsomeDB la `Map` degli utenti e la scrive su disco utilizzando la serializzazione JSON. L'unica

struttura che scrive è quella degli utenti, come già detto in precedenza, le strutture che raccolgono i post e i tag rappresentano una ridondanza che si è preferito non riportare anche su disco. La struttura degli utenti sarà sufficiente per ricreare l'intero stato corretto di WINSOME. Oltre al salvataggio periodico, il thread salva lo stato anche al momento della chiusura del server.

RewardCalculator. RewardCalculator è la classe che implementa il thread per il calcolo delle ricompense. Periodicamente richiede la lista degli utenti a WinsomeDB così da eseguire l'algoritmo di calcolo, per poi avvisare gli utenti iscritti al gruppo di multicast.

Ad ogni iterazione il thread recupera gli utenti di WINSOME, per calcolare la ricompensa per ognuno. Per ogni utente recupera i post da lui pubblicati, quindi per ogni post conta i nuovi voti e i nuovi commenti, salvandosi i curatori e la percentuale spettante in una struttura dedicata. Per distinguere i nuovi dai vecchi voti e commenti, nella classe WinsomePost, che rappresenta i post, sono presenti quattro strutture apposite, una `Map` per i nuovi voti e una per i vecchi, una per i nuovi commenti e una per i vecchi. Per vecchi si intende quei voti e commenti già considerati dopo l'ultima iterazione del thread. Una volta calcolata la ricompensa relativa a un post, il thread si occupa di trasferire i nuovi voti e commenti nelle strutture di quelli vecchi.

Al termine dell'iterazione il thread, con un pacchetto sulla socket UDP per il multicast, notifica tutti gli utenti che il calcolo delle ricompense è stato effettuato e si sospende fino alla prossima iterazione, o finché non ne è richiesta la terminazione.

Gestione della concorrenza

Per evitare race conditions fra i vari thread che condividono le strutture di WinsomeDB, sono stati utilizzati diversi meccanismi per gestirne la concorrenza.
La struttura degli utenti è una `ConcurrentHashMap` per permettere a i client che registrano nuovi utenti tramite il servizio RMI di poterli aggiungere senza che si verifichino inconsistenze. Un altro punto del progetto in cui viene sfruttata la sincronizzazione della `ConcurrentHashMap` è quando il thread che calcola le ricompense risale agli utenti attualmente registrati a WINSOME.
Inoltre, è presente una `ReadWriteLock` per gestire la sincronizzazione delle strutture durante i vari accessi del thread che effettua il salvataggio periodico dello stato, ma non solo. Questa lock viene acquisita in modalità scrittura ogni volta che viene effettuata una modifica alle strutture di WinsomeDB, mentre viene acquisita in modalità lettura per il salvataggio dello stato e per l'acquisizione degli utenti da parte del thread per il calcolo delle ricompense. Viene acquisita in modalità scrittura anche mentre quest'ultimo effettua lo spostamento dei nuovi voti e commenti di un post nei vecchi e l'incremento del numero di iterazioni dell'algoritmo su quel post.
Infine, è stato fatto uso anche di blocchi e metodi `synchronized`, come nel caso della registrazione alle callback, durante l'esecuzione del thread che calcola le ricompense per evitare che nuovi voti o commenti vengano aggiunti mentre sono già stati contati gli altri, e per l'aggiunta di nuovi tag alla struttura dei tag al momento della registrazione di un nuovo utente. Per rappresentare questa struttura è stata usata una `HashMap` invece

che una `ConcurrentHashMap` perché l'operazione di aggiunta non è atomica e necessita comunque di due istruzioni, da qui la necessità di sincronizzare il blocco di codice.

Implementazione del client

Il client di WINSOME è un *thin* client, quindi delega la risoluzione delle richieste utente al server, tranne per la richiesta di visualizzare i propri follower. La classe che lo implementa è ClientMain. Al momento del login di un utente, infatti, il client si registra al servizio di callback tramite RMI per ricevere l'aggiornamento quando un altro utente inizia o smette di seguire quello attualmente loggato. In questo modo la struttura dei follower lato client rimane costantemente aggiornata.

Una volta avviato, legge i parametri iniziali dal file di configurazione, se sono corretti tenta di connettersi al server, altrimenti termina. Dopo essersi connesso, esegue un semplice parsing delle richieste da linea di comando finché non viene richiesta la terminazione con il comando "quit". Per poter utilizzare le funzionalità di WINSOME, gli utenti devono prima registrarsi al servizio e poi effettuare il login, a questo punto il client si iscrive ai servizi di notifica per l'aggiornamento delle ricompense e per i follower e ricevere la lista aggiornata degli stessi. Più utenti possono registrarsi tramite un solo client, ma solo uno per volta può loggarsi. Un utente può essere loggato su un solo client. Per visualizzare le operazioni consentite è possibile digitare il comando "help". Quando l'utente effettua il logout, il client si cancella dai servizi di notifica e un nuovo login è consentito.

Il ClientMain inoltre istanzia un oggetto ClientNotify e il thread RewardUpdater, che rappresentano appunto le classi che gestiscono l'arrivo delle notifiche, rispettivamente per l'aggiornamento riguardo i follower e quello per il calcolo delle ricompense.

```
register <username> <password> <tags>:   Effettua la registrazione dell'utente
login <username> <password>:             Effettua il login dell'utente
logout:                                  Effettua il logout dell'utente
list users:                              Restituisce gli utenti che hanno almeno un tag in comune
list followers:                          Restituisce la lista dei follower
list following:                          Restituisce la lista degli utenti seguiti
follow <username>:                       Permette di seguire un utente
unfollow <username>:                     Permette di smettere di seguire un utente
blog:                                    Visualizza i post di cui l'utente è autore
post <title> <content>:                  Crea un post
show feed:                               Visualizza il feed dell'utente
show post <id>:                          Visualizza il post
delete <idPost>:                         Elimina il post
rewin <idPost>:                          Effettua il rewin del post
rate <idPost> <vote>:                    Aggiunge un voto al post
comment <idPost> <comment>:              Aggiunge un commento al post
wallet:                                  Visualizza il portafoglio dell'utente
wallet btc:                              Visualizza il portafoglio dell'utente in bitcoin
help:                                    Visualizza questo messaggio
```

Vista delle possibili operazioni che un utente WINSOME può richiedere.