Petrocchi Gianmarco 599116

Winsome: a reWardINg SOcial MEdia

Progetto corso laboratorio di reti"; 6/09/2022

Premessa

Il progetto consiste nell'implementazione del Social Media WINSOME, ispirato a STEEMIT e che prevede ricompense in una valuta chiamata WINCOIN.
WINSOME utilizza un'architettura client-server, nel quale il client manda le richieste attraverso la rete al server.
Il progetto è stato sviluppato tramite l'IDE Eclipse e testato in ambiene Windows.
Per la compilazione su Command Line delle due classi main (ClientMain e ServerMain) bisogna posizionarsi sulla cartella WINSOME e usare il seguente comando:

javac @paths.txt –d out –encoding utf8 –cp "lib/*"

Dove "paths.txt" è un file di testo dove sono presenti tutti i path relativi delle classi che compongono l'intero progetto. Nella cartella lib del progetto è presente la libreria **gson-2.8.9** usata sia lato client che server per serializzare e deserializzare oggetti.
Per l'esecuzione corretta del progetto prima bisogna eseguire il server con il comando:

java –cp "out;lib/*" server/ServerMain

e successivamente il client con il comando:

java –cp "out/lib*" client/ClientMain

Altrimenti è possibile eseguire i file .jar di client e server, sempre da Command Line, con i seguenti comandi:

java –jar jars/server.jar java –jar jars/client.jar

Struttura del progetto

Il progetto si articola in molti packages, ognuno con delle funzionalità:

client.

Contiene le classi e i metodi che vengono utilizzati **solo dal client:**

- *ClientMain*: classe la cui unica funzionalità è quella di simulare un utente che utilizza il Social Media, quindi manda richieste al server e riceverà le varie risposte o errori. Al momento dell'avvio, il client si connette via TCP al server seguendo le indicazioni precisate nel file di configurazione. Una volta connesso, legge da linea di comando attraverso un loop che si arresta solo quando il client digita il comando "logout" e questa richiesta viene eseguita correttamente. Le richieste che vengono mandate al server sono nella seguente forma: **command1:command2:arg1:arg2:….:argN.** Il numero di argomenti nella richiesta dipende da quale vogliamo eseguire, non tutte hanno la stessa lunghezza.(command2 serve per specificare comandi del tipo "show feed", quindi command2 = "feed"). Per sapere quali sono le possibili richieste e le sintassi di esse, andare alla sezione "**Lista**

comandi Winsome".

- *ClientStorageImpl*: classe che implementa il meccanismo di RMI callback per l'aggiornamento dei followers e dei following del client.
- *MulticastClient*: thread che riceve le notifiche di avvenuto calcolo delle ricompense tramite Multicast UDP. Viene settato e avviato successivamente dopo l'operazione di register o login, e viene interrotto solamente dopo l'operazione di logout.
- *PostWrapper*: classe che viene usata per poter serializzare oggetti json che vengono mandati dal server per rispondere alla richiesta "show feed".
- *PostWrapperShow*: classe che viene usata per poter serializzare oggetti json che vengono mandati dal server per rispondere alla richiesta "show post".
- *UserWrapper*: classe che viene usata per poter serializzare oggetti json che vengono mandati dal server per rispondere alla richiesta "list users".

Il package client contiene dei sub-package che corrispondono alle varie richieste che possono essere mandate al server.

configuration.

Il package contiene le classi che si occupano del parsin dei file di configurazione per client e server. Prima di tutto, si definisce il formato dei file di configurazione:

client_configuration.txt:
SERVERADDRESS=<a valid server address>
TCPPORT=<tcp_port>
UDPPORT=<udp_port>
RMIREGISTRYPORT=<rmi_registry_port>
RMIREGISTRYHOST=<rmi_registry_host>
REGISTRATIONSERVICENAME=<registration_service_name>
CALLBACKSERVICENAME=<callback_service_name>

server_configuration.txt:
SERVERADDRESS=<a valid server address>
TCPPORT=<tcp_port>
UDPPORT=<udp_port>
RMIREGISTRYPORT=<rmi_registry_port>
RMIREGISTRYHOST=<rmi_registry_host>
REGISTRATIONSERVICENAME=<registration_service_name>
CALLBACKSERVICENAME=<callback_service_name>
DELAYBACKUP=<milliseconds_delay_backup>
DELAYEARNINGSCALCULATION=<milliseconds_delay_earnings_calculation>
DELAYSHUTDOWNTHREADPOOL=<milliseconds_delay_shutdown_threadpool>
AUTHORPERCENTAGEERAN=<author_percentage_earn>
DIRECTORYFORRILE=<directory_to save_backup_files>

I parametri possono essere scritti in qualsiasi ordine, non vengono ammessi argomenti non validi (i.e. non si permettono stringhe dove ci si aspetterebbe un numero) e nessuno è opzionale. La sintassi adottata è quella dei file properties:
Le classi all'interno del package sono:

- *ClientConfiguration*: fa il parsing del file di configurazione del client.
- *ServerConfiguration*: fa il parsing del file di configurazione del server.

exceptions.

Contiene eccezioni che vengono lanciate dal Social Media:

- *InvalidConfigurationExcpetion*: eccezione checked che viene lanciata durante il parsing, solo se il file di configurazione non rispetta la sintassi specificate sopra.
- *InvalidPortNumber*: eccezione checked che viene lanciata durante il parsing, solo se i parametri riguardanti le porte di comunicazione non sono validi (i.e. non sono un valore compreso tra 1024 e 65535)
- *InvalidAmountException*: eccezione checked che viene lanciata quando si prova a creare un oggetto **Transaction** con parametro **amount** minore o uguale a zero.
- *IllegalFileExcepetion*: eccezione checked che viene lanciata quando, al riavvio del server, alcuni file di backup non contengono dati, e di conseguenza non possono essere caricati.
- *ClientNotRegisteredException*: eccezione checked che viene lanciata quando un utente non risulta registrato.

RMI.

Contiene interfacce e classi che servono per poter implementare servizi RMI e RMI callback offerti da WINSOME:

- *ClientStorage*: interfaccia che contiene le firme dei metodi per l'aggiornamento dei follower e following di un utente, viene implementata dalla classe ClientStorageImpl nel package client.
- *RMICallback*: interfaccia che contiene le firme dei metodi usati per poter registrare e deregistrare un utente dal servizio di aggiornamento dei suoi follower e following.
- *RMICallbackImpl*: classe che implementa l'interfaccia RMICallback.
- *RMIRegistration*: interfaccia che contiene la firma del metodo per poter registrare un nuovo utente in WINSOME.
- *RMIRegistrationImpl*: classe che implementa l'interfaccia RMIRegistration.

utility.

Contiene classi utilizzate lato server per poter gestire gli utenti, post e i guadagni dei vari utenti:

- *User*: classe che contiene la definizione di un utente all'interno di WINSOME. I metodi messi a disposizione fanno uso del "synchronized" all'interno delle sezioni critiche in modo tale da permettere accessi in mutua esclusione (allo stesso oggetto di tipo User) a thread concorrenti.

• *Transactions*: classe che contiene la definizione di una transizione.
Post: classe che contiene la definizione di un post all'interno di WINSOME. I metodi messi a disposizione fanno uso del "synchronizes" all'interno delle sezioni critiche in modo tale da permettere accessi in mutua esclusione (allo stesso oggetto di tipo Post) a thread concorrente.
Comment: classe che contiene la definizione di commento riferito ad un post presente in WINSOME.

• *Vote*: classe che contiene la definizione di un voto riferito ad un post presente in WINSOME.

• *GainAndCurators*: classe che contiene il totale di WINCOIN da dover distribuire a tutti i curatori di un certo post.

• *TypeError*: classe che contiene tutti i possibili errori che possono occorrere durante le esecuzioni delle richieste che il server riceve dal client. Questi errori sono delle stringhe le quali vengono mandate al client e quest'ultimo visualizza il tipo di errore su STDIN.

server.
Contiene classi e metodi che vengono utilizzati **solo dal server:**

- *ServerMain*: al momento dell'avvio il server, in base ai parametri specificati nel file di configurazione, inizializza: il socket TCP che viene utilizzato per soddisfare la maggior parte delle richieste del client; il multicast socket UDP che viene utilizzato per poter inviare ad un gruppo multicast le notifiche di avvenuto calcolo delle ricompense; fa il binding dei servizi RMI (RMICallback e RMIRegistration); il threadpool per la gestione delle connesioni e richieste dei client. Successivamente ripristina utenti e post precedentemente caricati. Può capitare che al riavvio del server alcuni file siano vuoti, questo si verifica solo quando si avvia il server e non si fa alcuna operazione con i client, in questo, vengono utilizzate le strutture dati del costruttore Database per la gestione degli utenti e dei post.
- *TaskHandler*: un thread TaskHandler viene istanziato subito dopo aver accettato la connesione da parte di un client. E' la colonna portante dell'interno sistema, in quanto si occupa di ricevere la richiesta del client, fa il parsing di essa, verifica la validità, richiama il servizio apposito per poter eseguire la richiesta, manda al client il risultato oppure un errore nel caso in cui si sia verificato durante l'esecuzione della richiesta.
- *TaskBackup*: un thread TaskBackup viene istanziato per gestire il meccanismo di backup. Si occupa di chiamare i metodi forniti dal database e si rimette in idle chiamando una sleep per l'intervallo di tempo specificato nel file di configurazione; ricevuta una interrupt, termina.
- *TaskRewards*: Un thread TaskRewards viene istanziato per gestire il meccanismo di calcolo delle ricompense e il conseguente invio del messaggio di avvenuto calcolo via Multicast UDP. La periodicità viene implementata eseguendo una sleep per l'intervallo di tempo specificato nel file di configurazione del server.
  Thread Shutdownhook: thread che viene istanziato succesivamente ad una SIGTERM, SIGINT o ctrl+C da Command Line; server per gestire la corretta terminazione del server.

Il package server contiene dei sub-package, molti dei quali contengono metodi per poter eseguire i vari servizi offerti da WINSOME, usati dal thread TaskHandler. L'altro sub-package è **database,** il quale contiene:

Database: classe che serve per gestire tutti i dati riguardanti gli utenti, i post, e le transazioni che vengono calcolate per i vari utenti. Contiene molte strutture dati concorrenti, le quali vengono usate dai TaskHandler thread per poter eseguire le richieste dei client. Al primo avvio del server le collezioni del Database non contengono alcun dato, ai successivi riavvii, queste vengono caricate tramite i dati presenti nei file di backup. La parte di concorrenza del sistema è quindi presente nel database del server che viene gestita attraverso l'uso delle concurrent collections (principalmente ConcurrentHashMap), infatti ci sono molti metodi che vanno a modificare lo stato interno del Database. Infine, si vada a notare che modificare lo stato interno di un certo utente o di un certo post, non corrisponda a modificare lo stato del Database:

aggiungendo a questo la classe User e Post siano thread safe, concludiamo che la classe è thread safe.

- *Storage*: classe astratta che viene estesa dalla classe Database. Essa implementa due metodi diversi per il backup dei dati:

1. *backupCached:* implementa un meccanismo di caching scrivendo in append nel file specificato tutti e soli i dati dichiarati nuovi.
2. *backupNonCached:* sovrascrive l'intero file specificato andando a salvare tutti i dati.

Entrambi i metodi utilizzano Gson per convertire ogni elemento dello storage in un dato JSON valido.

Lista comandi WINSOME

- **register <username> <password> <tags**>: permette la registrazione di un nuovo utente in WINSOME i parametri specificati nella richiesta. Quando l'utente viene registrato con successo eseguirà in automatico la login, così che non sia necessario fare due richieste diverse. <tags> è una lista di tag separati da uno spazio. L'utente sceglie liberamente la lista, quindi il serve rnon gestisce una lista predefinita di tag.
- **login <username> <password>**: login di un utente già registrato per accedere al servizio. Se la login viene eseguita con successo: il client viene registrato al servizio di aggiornamento dei suoi follower e following mediante RMI callback, e si unisce al gruppo multicast per riceve le notifiche di avvenuto calcolo delle ricompense; vengono mandate, al server, due richieste di caricamento dei follower e following nel suo ClientStorage stub.

  logout: effetua il logout dell'utente dal servizio. Se questa richiesta viene eseguita con successo, il ciclo principale del client viene interrotto e quindi termina, ma prima di ciò: viene chiuso il socket TCP usato per la comunicazione con il server; viene interrotto il multicast client thread che lascerà il gruppo multicast e chiuderà il relativo socket; viene de-registrato il client dal servizio di RMI callback.
- **list users:** visualizza la lista di utenti registrati al servizio, solo quelli però che hanno almeno un tag in comune con l'utente che ne fa richiesta. Per ogni utente, che fa parte di questa lista, vengono visualizzati solamente lo username e la lista di tag.

- **list following:** visualizza la lista degli utenti di cui è follower. Per ogni utente in questa lista viene visualizzato solo lo username.
  - **follow <username>:** l'utente chiede di seguire l'utente che ha come username quello specificato nel comando. Da quel momento in poi può ricevere tutti i post pubblicati dal nuovo utente che segue. L'utente che vogliamo seguire deve essere registrato in Winsome e non possiamo seguire noi stessi.
- **unfollow <username>:** l'utente chiede di non seguire più l'utente che ha come username quello specificato nel comando. Da quel momento in poi non può più ricevere tutti i post pubblicati dall'utente che ha smesso di seguire. L'utente che vogliamo smettere di seguire deve essere registrato in WINSOME e non possiamo unfolloware noi stessi.
- **blog:** visualizza la lista dei post di cui l'utente è autore, quindi i post presenti nel blog dell'utente che ne fa richiesta. Per ogni post presente in questa lista, viene visualizzato id del post, autore e titolo.
- **post:** operazione per pubblicare un nuovo post in WINSOME. Dopo aver digitato post su STDIN, viene validata la richiesta da parte del server, se essa è valida allora l'utente inserirà da STDIN titolo e contenuto del post. In questo modo, le stringhe che rappresentano titolo e contenuto, possono contenere qualsiasi tipo di carattere perché verranno codificate e decodificate in Base64. Il titolo ha lunghezza massima di 20 caratteri e il contenuto ha lunghezza massima di 500 caratteri. Se l'operazione va a buon fine, il post è creato e disponibile per i follower dell'autore del post. Il sistema assegna un identificatore univoco a ciascun post creato (idPost).
- **show feed:** visualizza la lista di post presenti nel feed dell'utente che ne fa richiesta. Per ogni post presente in questa lista, viene visualizzato id, autore e titolo.
- **show post <id post>:** il server restituisce titolo, contenuto, numero di voti positivi, numero di voti negativi e commenti del post. Se l'utente è autore del post può cancellare il post con tutto il suo contenuto associato (commenti e voti). Se l'utente ha il post nel proprio feed può esprimere un voto e/o inserire un commento.
- **delete <id post>:** operazione per cancellare un post. La richiesta viene accettata e eseguita solo se l'utente che ne fa richiesta è l'autore del post. Il server cancella il post con tutto il suo contenuto associato (commenti, voti e rewin). Non vengono calcolate ricompense "parziali", ovvero se un contenuto recente (post, voto o commento) non era stato conteggiato nel calcolo delle ricompense perché ancora il periodo non era scaduto, non viene considerato nel calcolo delle ricompense.
- **rewin <id post>:** operazione per effettuare il rewin di un post, equivale al "condividi" su facebook o altri social, ovvero pubblicare nel proprio blog un post presente nel proprio feed. Non è possibile condividere post presenti nel proprio blog.
- **rate <id post> <vote>:** operazione per assegnare un voto positivo o negativo ad un post. Se l'utente ha il post nel proprio feed, allora la richiesta viene accettata e eseguita, se il post è presente nel blog dell'utente, allora viene rifiutata. Vote è un integer che può assumere solo due valori +1 (equivale al "mi piace") e –1 (equivale al "non mi piace"), **non sono accettati altri numeri**.

- **comment <id post>:** operazione per aggiungere un commento ad un post. Dopo aver digitato il comando citato su STDIN, viene validata la richiesta da parte del server, se essa è valida allora l'utente inserirà da STDIN il contenuto del commento. In questo modo, la stringa che rappresenta il contenuto, può contenere qualsiasi tipo di carattere perché verrà codificato e decodificato in Base64. Il contenuto del commento non ha una lunghezza massima. Il commento viene accettato solo se il post specificato è presente nel feed dell'utente, non se è presente nel blog.

- **wallet:** operazione per recuperare il valore del proprio portafoglio e la storia delle transizioni. Per ogni transizione viene visualizzato l'incremento (amount) e il timestamp (quando è stata fatta).

- **wallet btc:** operazione per recuperare il valore del proprio portafoglio convertito in bitcoin, Per poter fare questa conversione, il server comunica con il sito RANDOM.ORG per poter ottenere un tasso di cambio casuale.

- **help:** comando non presente nella specifica del progetto. Si limita solamente a fare vedere la lista dei possibili comandi da poter usare per interagire con WINSOME. Ad ogni comando è associata anche una brevissima spiegazione di cosa fa.