Università di Pisa

Dipartimento di Informatica
Corso di Laurea Triennale in Informatica

CROSS: an exChange oRder bOokS Service

Laboratorio III

Professore:
Prof. Laura Ricci

Autore:
Birindelli Leonardo

Anno Accademico 2024/2025

1 Spiegazione scelte adottate a livello di codice

1.1 Client

Nella realizzazione dell'interfaccia del client ho pensato di creare due distinte interfacce interattive : quella iniziale per il client che non ha ancora provveduto ad accedere al proprio profilo in cui sono presenti operazioni tra cui la registrazinoe, la richiesta di accesso nel servizio, il cambio delle credenziali (ovviamente se il profilo è presente nella base di dati lato server) e per concludere la possibilità di chiudere il client;

La seconda interfaccia è quella che si presenta all'utente solo dopo aver effettuato l'autenticazione di un profilo registrato in cui si da la possibilità all'utente di effettuare operazioni di trading permettendo l'inserimento di ordini di acquisto e vendita (tra cui gli ordini a mercato, ordini limite e ordini di stop), la cancellazione di uno specifico ordine non ancora evaso e la possibilità di richiedere l'elenco dei dati storici degli ordini effettuati in un dato mese e anno. Inoltre l'utente ha la possibilità di cambiare il profilo (tornarndo così alla schermata iniziale) e di chiudere il client.
La decisione di separare le due interfacce è basata sul fatto di mostrare una esplicita separazione tra gli utenti autenticati e quelli non garantendo anche una maggiore sicurezza nel caso in cui un utente non autorizzato provi ad accedere al servizio. In entrambe le implementazione ho deciso di creare un interfaccia di interazione con l'utente basata sull'inserimento di numeri corrispondenti alle operazioni che l'utente vuole effettuare in modo tale da garantire una interfaccia basilare e guidata permettendo coì di evitare controlli particolari sui tipi di input che l'utente può inserire e rendendola più semplice possibile sia lato utente che lato dell'elaborazione delle operazioni. Perlopiù il client gestisce lo spegnimento dell'applicativo permettendo così che l'utente autenticato venga disconnesso dal servizio in caso di un interruzione utente da linea di comando (crtl+c) e chiudendo opportunamente il socket di comunicazione con il server. Inoltre l'applicativo client è munito di un sistema di gestione degli errori che permette di gestire eventuali errori di comunicazione con il server e di input errati da parte dell'utente: sia gli errori che le risposte con successo inviate dal server vengono stampate vengono formattate a terminale per notificare lo stato dell'operazione effettuata. Per concludere, al momento dell'istanziazione di connessione con il server, il client utilizza un thread indipendente (gestito da un Executor perché permette di essere più resiliente nel caso in cui vengono lanciate eccezzioni nel thread e in più perché risulta essere più leggibile e organizzato permettendo anche di essere modulare ad eventuali future aggiunte) che si occupa di attendere e mostrare a terminale le notifiche di evasione degli ordini dell'utente autenticato .

1.2 Server

2 Schema thread attivati

2.1 Client

I thread attivati dal client sono due:

- il primo thread è il thread principale che si occupa di gestire l'interfaccia utente, di inviare e stampare a schermo le risposte del server;
- il secondo thread è un thread indipendente gestito all'interno di una thread pool contente un solo thread che si occupa di ricevere le notifiche di evasione degli ordini dell'utente autenticato.

2.2 Server

3 Strutture Dati

3.1 Client

Lato client non sono state utilizzate strutture dati particolari in quanto la comunicazione con il server avviene tramite l'invio di stringhe contenenti le operazioni da effettuare e i dati necessari per effettuare tali operazioni. L'uniche informazioni che faccio memorizzare al client sono quelle legate alla

sessione utente tra cui il nome dell'utente attualmente autenticato "`usernameLoggedIn`", il tempo massimo di sessione utente imposta nella configurazione del server "`maxLoginTime`" (passato al momento dell'autenticazione dell'utente con il server) e il timestamp di stato della sessione utente memorizzata lato server "`userSessionTimestamp`" (la quale viene continuamente aggiornata ogni volta che il client interagisce con il server).

4 Primitive di Sincronizzazione

4.1 Client

Lato client ho usato solo una primitiva di sincronizzazione che è l'utilizzo delle implicit lock tramite la parola chiave `synchronized` in Java nel punto in cui lo shutDownHook intercetta il segnale di chiusura del client e si occupa di disconnettere l'utente e chiudere il socket di comunicazione con il server. Questo è stato fatto per garantire che il client possa chiudere correttamente la connessione con il server e che non ci siano problemi di concorrenza tra il thread che gestisce lo shutDownHook e il thread principale del nel momento in cui il client viene chiuso.

- **Mutex (synchronized)**: Utilizzato per proteggere l'accesso concorrente all'`OrderBook`.
- **Condition Variables**: Gestiscono il coordinamento tra thread per notifiche.
- **Semaphore**: Limita il numero di connessioni concorrenti al server.