Relazione sul Protocollo di Comunicazione Client-Server

Introduzione

Questa relazione descrive come funziona la comunicazione tra il client e il server realizzati per il progetto “Rifugi Alpini della Provincia Autonoma di Bolzano”.
Il programma è scritto in Java e permette di consultare da remoto i dati dei rifugi contenuti in un file CSV, usando una connessione TCP.

Descrizione del Progetto

Il progetto sviluppa un’applicazione client-server in Java, progettata per consentire la consultazione remota di un file CSV contenente informazioni sui rifugi alpini della provincia di Bolzano. Il server si occupa di leggere il file CSV e di organizzarlo in una struttura dati interna, gestendo le richieste provenienti dal client attraverso connessioni socket TCP. Il protocollo di comunicazione è semplice e consente al client di richiedere righe specifiche del file, con il server che risponde inviando i dati corrispondenti o messaggi di errore se le richieste non sono valide.

Struttura del Progetto

Il progetto si articola in più componenti principali, tutti localizzati all’interno di una struttura di cartelle organizzata. La parte server è costituita dal file Server.java, che implementa la logica del server, e dal file CSVReader.java, che si occupa della lettura e gestione dei dati provenienti dal file CSV. Per la parte client, c'è il file Client.java, che gestisce la comunicazione e l’interazione con il server. Il file CSV con i dati sui rifugi alpini è salvato sotto il nome Provincia-Autonoma-di-Bolzano-Elenco-dei-rifugi-alpini.csv e si trova nella cartella dei dati. Il progetto è accompagnato dal documento README.md, che descrive l’implementazione.

Funzionamento del Server

Il server è progettato per gestire le richieste del client in modo efficiente. All’avvio, il server legge il file CSV e lo carica in una struttura dati interna, come una lista di oggetti o una mappa, per permettere un rapido accesso ai dati. Una volta che il file è stato caricato, il server apre una connessione

socket in ascolto delle richieste provenienti dal client. Ogni richiesta del client viene gestita in modo indipendente tramite l'uso di thread, consentendo così al server di supportare la comunicazione con più client contemporaneamente. Il server risponde alle richieste inviando la riga del file CSV richiesta dal client, oppure restituisce un messaggio di errore nel caso in cui la richiesta non sia valida o la riga richiesta non esista.

Funzionamento del Client

Il client si connette al server utilizzando una connessione socket TCP. Una volta stabilita la connessione, il client invia richieste al server seguendo un protocollo definito, come ad esempio una richiesta per ottenere una riga specifica del file CSV. Ogni richiesta inviata dal client è formulata in modo che il server possa comprenderla facilmente e restituire una risposta adeguata. Quando il server invia la risposta, che può essere o i dati richiesti o un messaggio di errore, il client li riceve e li visualizza per l'utente. In questo modo, l'interazione con il server è semplice e mirata a permettere all'utente di ottenere informazioni specifiche sui rifugi alpini.

Protocollo di Comunicazione

Il protocollo di comunicazione tra il client e il server è strutturato in modo semplice ma preciso. Le richieste del client sono inviate come comandi di testo, come nel caso di MOSTRA_RIGA n, dove n rappresenta il numero della riga richiesta dal file CSV. Se la richiesta è valida, il server risponde con i dati relativi alla riga richiesta, come, ad esempio, "Rifugio XYZ, 2500m, Bolzano". Se la richiesta non è valida, ad esempio se il numero della riga è fuori dal range o il formato della richiesta è errato, il server invia un messaggio di errore come "Indice fuori intervallo." o “Comando non riconosciuto. Scrivi AIUTO per vedere i comandi disponibili.”. Questo protocollo è pensato per essere semplice, consentendo al client di comunicare in modo diretto e chiaro con il server.

Requisiti Tecnici

Il progetto è sviluppato in Java e fa uso di socket TCP per stabilire la comunicazione tra il client e il server. La gestione delle connessioni è implementata con threading, per consentire al server di rispondere simultaneamente a più client senza bloccare il servizio. Inoltre, il server è in grado di gestire situazioni di errore, restituendo messaggi di errore quando le

richieste non sono valide. La gestione degli errori è una componente fondamentale per garantire un'esperienza utente fluida e senza intoppi.

Comandi supportati dal protocollo

Il client invia comandi al server scrivendo testo.
I comandi che il client può inviare sono i seguenti:

- HELP, il server risponde con l'elenco dei comandi disponibili
- GET_ALL, il server risponde con l'elenco di tutti i rifugi contenuti nel file
- GET_ROW <n>, il server invia i dati del rifugio che si trova alla riga n
- GET_COMUNE <nome_comune>, il server invia i dati del rifugio/i rifugi al comune specificato
- GET_ALTITUDINE <min> <max>, il server invia i dati del rifugio/i rifugi all' altitudine indicata nel range
- QUIT, il client termina la connessione con il server

Risposte del server

Le risposte del server sono sempre in formato testo. In base al comando ricevuto, il server invia una delle seguenti risposte:

Esempi

HELP:
Comandi disponibili:

- HELP: mostra questo messaggio
- GET_ALL: elenca tutti i rifugi
- GET_ROW <n>: mostra il rifugio alla riga n
- GET_COMUNE <nome_comune>: elenca i rifugi nel comune specificato
- GET_ALTITUDINE <min> <max>: elenca i rifugi con altitudine nel range indicato
- QUIT: termina la connessione

GET_ROW 5:
Nome: Rifugio Firenze - Regensburgerhütte,
Località: Santa Cristina Valgardena,
Comune: Santa Cristina Valgardena,
Indirizzo: Val d'Anna,
CAP: 39047,
Altitudine: 2037,
Telefono: 0471794666,
Email: info@rifugio-firenze.com,
Posti letto: 65,

Provincia: Bolzano

GET_ALL
Il server risponde con un elenco testuale di tutti i rifugi, uno per riga.

Errori
Se il comando non è valido o l'indice fornito è sbagliato, il server invia messaggi come:
- Comando non riconosciuto.
- Indice fuori intervallo.
- Uso corretto: GET_ROW <numero>

Chiusura della connessione

Il comando QUIT viene usato dal client per chiudere in modo sicuro la connessione.
Anche il server può rilevare quando il client si disconnette e chiude la comunicazione in modo corretto.

Dettagli tecnici

- Porta usata dal server: 1300
- Formato dei messaggi: stringhe UTF-8
- Gestione di più client: ogni client è gestito su un thread diverso
- Lettura dati: il server carica i dati dei rifugi dal file CSV all'avvio

Conclusione

Il protocollo realizzato è semplice da usare e permette di accedere a dati reali in modo remoto e strutturato.
È un buon esempio per imparare come si crea una comunicazione tra client e server usando le socket in Java.