Corso di Laurea triennale in
Ingegneria e Scienze Informatiche
Anno Accademico 2023-2024
Programmazione di Reti

Web Server Semplice

per il servizio di file statici e la gestione di richieste HTTP GET di base

Autore
Miriam Sonaglia, 0001071400

Obiettivi del progetto

Il progetto si propone di sviluppare un semplice web server HTTP multithread in Python che possa servire al client file statici, come HTML, CSS o immagini, e gestire una o più richieste simultanee HTTP GET di base.

Requisiti

Necessità di avere Python installato sul proprio sistema.

Guida all'utilizzo

Esecuzione del Server

Aprire il terminale o la riga di comando e spostare la directory corrente nella cartella in cui si trova il file del server con il comando:

```
cd /percorso/cartella
```

ed eseguire il comando:

```
python web_server.py
```

In alternativa è possibile aprire il file *web_server.py* in un IDE come Spyder ed eseguire il file (tasto F5).

Avvio del Server

Dopo l'avvio, nel terminale dovrebbe apparire la stringa:

```
Server in ascolto...
```

A questo punto il server sarà in ascolto su localhost:8080.

Accesso ai File Statici

Aprire un browser e accedere ai file presenti nella cartella del server, ad esempio:

http://localhost:8080/nomefile.estensione

Se il file dovesse trovarsi in una sottocartella, specificare il percorso completo:

http://localhost:8080/sottocartella/nomefile.estensione

Funzionamento del Codice

Importazione delle Librerie

Viene importato il necessario per la gestione delle connessioni di rete (socket), la gestione dei thread (threading) e l'interazione con il file system (os).

Funzione client_handle

Questa funzione gestisce le richieste dei client, legge la richiesta HTTP, determina il file richiesto, verifica se il file esiste e costruisce la risposta HTTP appropriata.

Creazione del Socket del Server

Viene creato un socket TCP/IP e configurato per ascoltare le connessioni in arrivo sulla porta 8080.

Ciclo di Ascolto e Gestione delle Connessioni

Un ciclo infinito accetta le connessioni in arrivo e crea un nuovo thread per ciascuna connessione, gestendo le richieste in modo concorrente.

Test del Server

Dopo aver avviato il server, aprire un browser e navigare verso:

http://localhost:8080/index.html

Esempio di output del server:

```
Connessione da: ('127.0.0.1', 56314)
Richiesta ricevuta: GET /index.html HTTP/1.1
Host: localhost:8080
```

Considerazioni Aggiuntive

Nel codice sono presenti annotazioni utili per comprendere il funzionamento del server e la gestione delle risposte e dei thread.