Progetto per Programmazione di Reti: HTTP Server

Questo progetto è un semplice server http multithreaded che permette di servire ai client file statici presenti sul file system.

Questa repository contiene al suo interno il server (`server.py`) e la directory `static`, con al suo interno dei file di prova che possono essere aperti tramite un browser web.

Versioni di Python compatibili

Per eseguire il server, qualsiasi versione di Python 3.6 o successiva può essere utilizzata.
Durante lo sviluppo è stata utilizzata la versione di Python 3.11.

Come usare

Il server può essere lanciato in due modi, utilizzando i valori di default oppure specificando dei parametri attraverso la console

Eseguire il server utilizzando i valori di default

```
$ python3 server.py
```

Lanciando il server senza parametri, questo crea un server in ascolto su `localhost:1234` servendo file partendo dalla root `static`

Eseguire il server specificando dei parametri

Quello che segue è il messaggio di help, che elenca tutti i possibili parametri che possono essere utilizzati.

```
$ python3 server.py --help
Options available:
-h, --help: shows this help message
-a ADDRESS, --address=ADDRESS: specify the address used for this server. Default is localhost
-p PORT, --port=PORT: specify the port used for this server. Default is 1234
-r ROOT, --root=ROOT: specify where the root folder for serving files is located in the filesystem. Defaul
```

I parametri sono nel formato standard utilizzato dagli script bash e si possono utilizzare più parametri nello stesso comando (es. `python3 server.py -p 4004 --address=192.168.10.1`)

Output del server

```
$ python3 server.py
Server listening at ('127.0.0.1', 1234) (static)
Received Connection: ('127.0.0.1', 43356)
  Creating new thread with id: Thread-1 (serve_client)
  [Thread-1 (serve_client)]: GET /index.html
...
```

Una volta avviato il server, su stdout verra stampata una prima riga nel formato

```
Server listening at (<ADDRESS>, <PORT>) [<ROOT>]
```

dove verranno visualizzati i valori specificati dai parametri oppure quelli di default.
A seguire viene scritto il log di tutte le connessioni che hanno richiesto una risorsa dal server, con eventuale messaggio di errore se questa non è stata trovata nel filesystem.

Considerazioni aggiuntive

- Il server è stato scritto utilizzando esclusivamente librerie standard, in modo da poter eseguirlo in qualsiasi ambiente senza doversi

preoccupare di dipendenze esterne.

- Dato che il server utilizza la libreria standard `socket`, ho scelto di usare il paradigma procedurale in modo da poter scrivere del codice che sembri più a basso livello.
- Le funzioni sono state annotate con i tipi, in modo da rendere più facile la comprensione delle variabili