

Elaborato per il corso di
”Programmazione di Reti”

Rebosio Alessandro

10 maggio 2025

Indice

1	Introduzione	2
2	Architettura del progetto	3
2.1	Descrizione dell'architettura	3
3	Implementazione del web server	4
3.1	Struttura generale	4
3.2	Gestione delle richieste	4
3.3	Gestione MIME types e logging	5
3.4	Sicurezza base	5
4	Sito web statico	6
4.1	Pagine HTML	6
4.2	Estendibilità	6
5	Test e risultati	7
6	Conclusioni	8
7	Appendice	9

Capitolo 1

Introduzione

L'obiettivo di questo progetto è la realizzazione di un **server HTTP** minimale sviluppato in **Python**, capace di gestire richieste di tipo GET sulla porta 8080 di localhost e di servire contenuti statici in formato HTML e CSS.

Il progetto intende fornire una comprensione pratica del protocollo HTTP, dell'utilizzo dei **socket** per la comunicazione di rete e del funzionamento di base di un **web server**.

Attraverso la realizzazione di un server costruito manualmente, è stato possibile esplorare in modo diretto e applicato i concetti fondamentali delle reti e della programmazione di basso livello.

Funzionalità principali

- Risposta con codice di stato 200 per risorse esistenti;
- Gestione dell'errore 404 per risorse non trovate;
- Riconoscimento e gestione dei MIME types per i file serviti;
- Logging delle richieste ricevute;
- Pubblicazione di un sito web statico composto da tre pagine HTML responsive.

Capitolo 2

Architettura del progetto

L'architettura del progetto si basa su una suddivisione tra la logica applicativa del server e i contenuti statici da servire al client.

2.1 Descrizione dell'architettura

- **src/server.py**: implementa il web server HTTP, ascolta sulla porta 8080 e serve file statici presenti nella cartella **www/**.
- **www/**: contiene il sito web statico, costituito da tre pagine HTML e un file CSS condiviso. Le pagine includono layout responsive e contenuti base.

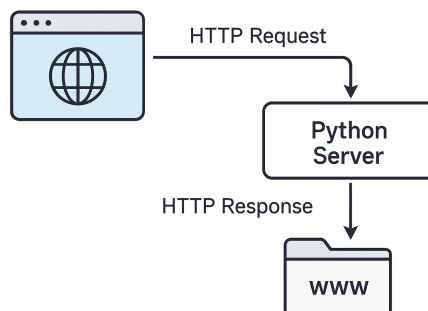


Figura 2.1: Schema dell'architettura client-server

In Figura 2.1 viene rappresentata l'interazione tra browser, server e file system. Quando un client invia una richiesta HTTP, il server interpreta la richiesta, individua il file richiesto nella directory **www/** e restituisce il contenuto.

Capitolo 3

Implementazione del web server

L'implementazione del server HTTP è interamente in `Python` utilizzando il modulo `socket` per la comunicazione e il modulo `threading` per la gestione concorrente delle connessioni.

3.1 Struttura generale

- **HTTPServer**: classe usata per stabilire una o più connessioni attraverso la creazione di un `socket`, che viene poi associato alla porta 8080 per accettare le connessioni in ingresso.
- **HTTPRequestHandler**: classe per il parsing delle richieste con il successivo invio di risposte.
- **MyServer**: sottoclasse per semplificare la logica, nella quale viene implementato il metodo `do_GET` per servire i file richiesti dai client.

3.2 Gestione delle richieste

Quando un client tenta la connessione e viene accettata, il server crea un nuovo `thread` per gestirla, permettendo le richieste in modo concorrente.

La richiesta viene letta, vengono estratti il `metodo`, `path` e la versione HTTP dalla prima riga.

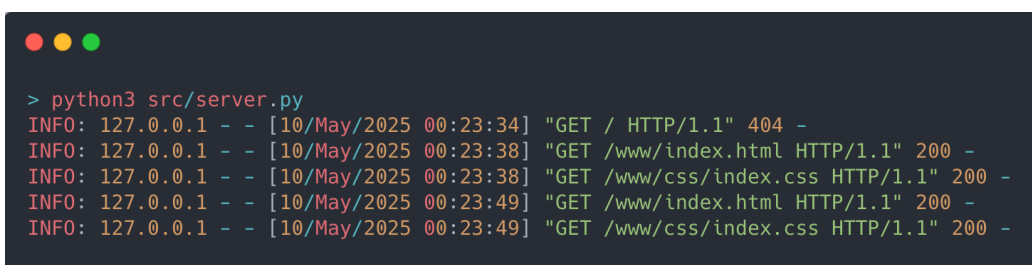
Successivamente se la risorsa richiesta esiste nella directory `www/`, viene caricata e restituita la risposta con il codice (200, OK). Altrimenti viene restituita la risposta di errore con il codice (404, Not Found).

3.3 Gestione MIME types e logging

Il server usa la libreria `mimetypes` per determinare il tipo di contenuto da servire in base all'estensione del file.

Ogni richiesta viene loggata Figura 3.1 in console con il formato: indirizzo IP del client, data e ora, metodo, risorsa richiesta e codice di stato restituito.

Esempi di logging:



```
> python3 src/server.py
INFO: 127.0.0.1 - - [10/May/2025 00:23:34] "GET / HTTP/1.1" 404 -
INFO: 127.0.0.1 - - [10/May/2025 00:23:38] "GET /www/index.html HTTP/1.1" 200 -
INFO: 127.0.0.1 - - [10/May/2025 00:23:38] "GET /www/css/index.css HTTP/1.1" 200 -
INFO: 127.0.0.1 - - [10/May/2025 00:23:49] "GET /www/index.html HTTP/1.1" 200 -
INFO: 127.0.0.1 - - [10/May/2025 00:23:49] "GET /www/css/index.css HTTP/1.1" 200 -
```

Figura 3.1: Log delle richieste in console

3.4 Sicurezza base

Per evitare accessi non autorizzati a file esterni alla cartella `www/`, il path richiesto viene sanificato rimuovendo sequenze potenzialmente pericolose come `../`.

Sicurezza del path:



```
self.path.lstrip("/").replace("../", "").replace("../\\", "")
```

Figura 3.2: Codice per path sicura

Capitolo 4

Sito web statico

Il sito web servito dal web server, al momento, è composto da tre pagine HTML statiche, con layout responsive e stile definito tramite file CSS dedicati. L'obiettivo è fornire un'interfaccia semplice per dimostrare il corretto caricamento dei contenuti da parte del server.

4.1 Pagine HTML

- **index.html** - Homepage che presenta una navigazione verso le altre sezioni del sito.
- **login.html** — Simula una pagina di login: nel caso venga inviata una richiesta POST, il server restituirà un errore 501 (metodo non implementato).
- **contact.html** — Pagina di contatto, con pulsanti per tornare alla home o alla pagina di login.

*Tutte le pagine fanno uso del framework **Bootstrap 5** per garantire un design responsive e compatibile con dispositivi di varie dimensioni.*

4.2 Estendibilità

Il sistema è progettato per essere facilmente estendibile: per aggiungere una nuova pagina è sufficiente **creare** un file HTML nella cartella **www/** e assicurarsi che il **collegamento** nel sito punti al percorso corretto.

Non è necessario modificare il codice del web server: qualsiasi file esistente sarà automaticamente servito, a condizione che venga effettuata una richiesta HTTP valida.

Capitolo 5

Test e risultati

Capitolo 6

Conclusioni

Capitolo 7

Appendice