Università di Bologna
Corso di Laurea in Ingegneria Informatica

Relazione Tecnica
Mini-Webserver "Autoscuola Marte"

Matteo Onofri

Data: 19 maggio 2025

Contents

1 Introduzione **2**

2 Requisiti e specifiche **2**
2.1 Requisiti minimi soddisfatti 2
2.2 Estensioni opzionali implementate 2

3 Architettura del sistema **2**

4 Il mini-webserver (server.py) **3**

5 Logging **3**

6 Sicurezza e limiti noti **3**

7 Front-end statico **3**

8 Avvio del server **3**

9 Utilizzo **4**

10 Conclusioni **4**

1 Introduzione

Per il progetto di Programmazione di Reti è stato realizzato un server HTTP minimale in Python in grado di pubblicare un sito web statico denominato "Autoscuola Marte". L'obiettivo era rispettare i requisiti previsti dalla Traccia 1 - Web Server + Sito Web Statico - e implementare alcune estensioni facoltative per rendere l'applicazione più completa.

2 Requisiti e specifiche

2.1 Requisiti minimi soddisfatti

- Il server è in ascolto su localhost porta 8080.
- Sono presenti e correttamente servite tre pagine HTML statiche: index.html, lezioni.html, esami.html.
- Il server gestisce richieste GET restituendo 200 OK quando il file esiste.
- Il server restituisce 404 Not Found con pagina personalizzata quando il file richiesto non è presente.

2.2 Estensioni opzionali implementate

- Rilevamento automatico del MIME type dei file tramite la libreria standard mimetypes.
- Logging di tutte le richieste HTTP su file access.log.
- Layout responsive e piccole animazioni.
- Protezione da directory traversal.
- Intestazioni HTTP complete (Date, Content-Type, Content-Length, Server).

3 Architettura del sistema

Struttura delle directory:

- server.py
- access.log
- www/
  - index.html
  - lezioni.html
  - esami.html
  - 404.html
  - style.css
  - img/logo.png

4 Il mini-webserver (server.py)

Il server utilizza il modulo socket in modalità TCP. Il ciclo principale crea il socket, abilita SOREUSEADDR, rimane in ascolto sulla porta 8080 e per ogni connessione invoca la funzione di servizio del client. La funzione:

- legge la prima linea della richiesta HTTP (metodo, percorso, versione);
- registra la richiesta nel log con timestamp e indirizzo IP;
- rifiuta metodi diversi da GET restituendo 405 Method Not Allowed;
- previene percorsi contenenti ".." rifiutandoli con 400 Bad Request;
- mappa la radice "/" in "/index.html";
- verifica l'esistenza del file richiesto, determina il MIME type, costruisce l'intestazione e invia il contenuto al client;
- in assenza del file richiesto restituisce 404 e la pagina di errore.

5 Logging

Ogni richiesta è salvata nel file access.log con orario locale, indirizzo IP, metodo e risorsa.

6 Sicurezza e limiti noti

- Il server gestisce una richiesta per volta (single-thread).
- Non supporta HTTPS.

7 Front-end statico

index.html: pagina di benvenuto con sezione introduttiva e collegamenti alle altre pagine.
lezioni.html: calendario delle lezioni teoriche e dettagli sulle prove pratiche.
esami.html: riepilogo delle date d'esame teorico e pratico 2025.
404.html: pagina di errore personalizzata con collegamento per tornare alla home.
style.css: definisce schema di colore, tipografia, layout flessibile e animazioni al passaggio del mouse sui pulsanti.

8 Avvio del server

```
python3 server.py
```

9 Utilizzo

- Il server serve i file statici presenti nella cartella `www`.
- La Homepage è `index.html`.
- Se un file non viene trovato, viene restituita la pagina `404.html` che reindirizza alla homepage.
- I log degli accessi vengono salvati in `access.log`.

10 Conclusioni

Il progetto soddisfa tutti i requisiti minimi e integra diverse funzionalità opzionali, dimostrando come la sola libreria standard di Python consenta di realizzare un piccolo server HTTP sicuro ed estendibile per la pubblicazione di contenuti statici.