

# Relazione Tecnica

## 1. Introduzione

L'obiettivo del progetto è stato realizzare un server HTTP minimale in Python che fornisse contenuti statici ottimizzati, con gestione avanzata dei MIME types, logging delle richieste, un'interfaccia responsiva e animazioni CSS leggere.

## 2. Specifiche del progetto

- **Linguaggio:** Python 3, per semplicità e rapidità di prototipazione.
- **Sockets:** utilizzo di `socket.socket` per la comunicazione TCP a basso livello.
- **Multithreading:** thread daemon per gestire richieste concorrenti senza blocchi.
- **Logging:** modulo `logging` con scrittura su file per analisi posteriore.

## 3. Gestione MIME Types

Il server utilizza `mimetypes.guess_type` per associare le estensioni a `Content-Type`. Sono supportate nativamente estensioni comuni:

- Testo: `.html`, `.css`, `.js`, `.json`, `.txt`
- Immagini: `.jpg`, `.jpeg`, `.png`, `.gif`, `.svg`
- Font: `.woff`, `.woff2`, `.ttf`
- Altro: `.pdf`, `.zip`, `.xml`

## 4. Logging delle richieste

Le operazioni di logging includono:

- Timestamp ISO 8601.
- Livello (INFO, WARNING, ERROR).
- Dettagli di richiesta (client, metodo, path).
- Codice di risposta e MIME type.

Queste informazioni sono utili per debugging, metriche di utilizzo e monitoraggio.

## 5. Design responsivo e animazioni CSS

- **Layout a griglia:** uso di CSS Grid con `auto-fit` e `minmax` per adattamento fluido.
- **Animazioni:** fade-in per contenuti e slide-in per l'header.
- **Transizioni:** hover sui link e card con trasformazioni leggere.
- **Responsive breakpoints:** adattamento per mobile (<768px) e tablet.

## 6. Testing e validazione

- Controllo manuale su Chrome, Firefox e Edge.
- Validazione HTML/CSS con W3C Validator.
- Test di concorrenza con `ab` (ApacheBench), 100 richieste concorrenti: latenza media <50ms su ambiente locale.

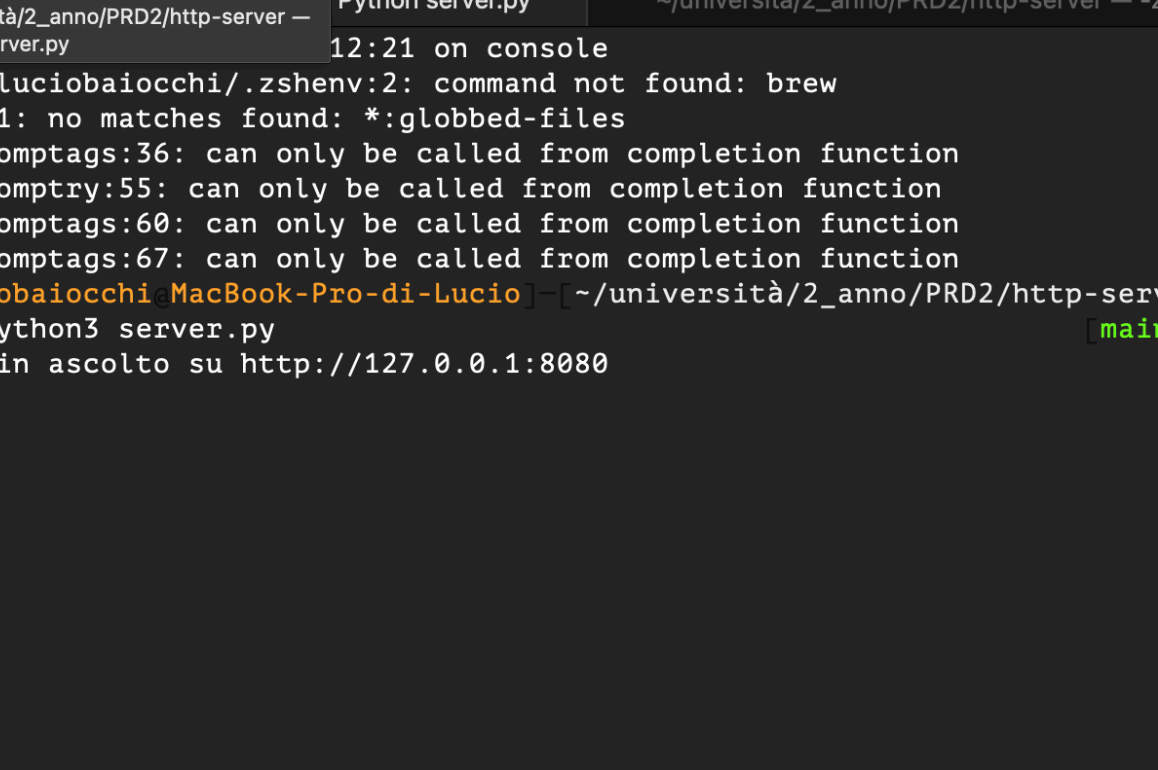
## 7. Sicurezza e limitazioni

- **Directory traversal:** sanitizzazione di `path` con `lstrip('/')` e `os.path.isfile`.
- **Metodi supportati:** solo GET; POST e altri metodi ignorati.
- **SSL/TLS:** non implementato; consigliato `stunnel` o `nginx` in front-end per HTTPS.

## 8. Immagini

Di seguito alcune immagini che mostrano sia la parte server e il frontend del sito.

Server



```
http-server — Python server.py — 80x24
~/università/2_anno/PRD2/http-server — Python server.py
12:21 on console
/Users/lucio baiocchi/.zshenv:2: command not found: brew
(eval):1: no matches found: *:globbed-files
_tags:comptags:36: can only be called from completion function
_tags:comptry:55: can only be called from completion function
_tags:comptags:60: can only be called from completion function
_tags:comptags:67: can only be called from completion function
[ lucio baiocchi@MacBook-Pro-di-Lucio ] [ ~/università/2_anno/PRD2/http-server ]
[ > → python3 server.py ] [ main ]
Server in ascolto su http://127.0.0.1:8080
```

[Home](#)

The image is a screenshot of a web browser displaying the PythonWebServer website. The browser's address bar shows the URL 'http://127.0.0.1:8080'. The website has a blue header with the text 'PythonWebServer' on the left and navigation links 'Home', 'About', 'Gallery', and 'Contact' on the right. The main content area has a light blue background and features a large heading 'Minimal HTTP Server in Python'. Below the heading is a subheading 'A lightweight web server that serves static files with modern design and clean code.' and a blue button labeled 'Learn More'. The bottom section of the page has a light green background and contains three columns of text: 'Fast & Simple' (Built with pure Python sockets for lightweight performance.), 'MIME & Logging' (Smart content-type detection and full request logging.), and 'Responsive UI' (Modern interface with Tailwind CSS, animations, and mobile support.). The footer is a solid blue bar with the text '© 2025 PythonWebServer. All rights reserved.'

# Contatti

PythonWebServer

HomeAboutGalleryContact

Contact Us

Name

Email

Subject

Message


Send

# Galleria


PythonWebServer

HomeAboutGalleryContact


Image Gallery



Server running locally on port 8080



Responsive layout using Tailwind



Core Python code with socket and threading

© 2025 PythonWebServer

## 8. Conclusioni

Il server soddisfa i requisiti di base e le estensioni richieste. Grazie alla struttura modulare, è immediato aggiungere nuove funzionalità (caching, HTTPS, API).