# Relazione tecnica

**Traccia 1** – Web Server + Sito Web Statico

Ilaria Piras

Matricola: 0001114122

## **Obiettivo**

Il codice realizza un semplice server HTTP in Python, capace di servire file statici (HTML, CSS, immagini, ecc.) su localhost:8080. Il server gestisce richieste GET, restituisce il file richiesto se esiste (con codice 200 e MIME type corretto) oppure una pagina di errore 404 se il file non esiste. Viene fatto il Logging di tutte le richieste.

**Struttura repository**

Il repository contiene un file Server.py, contenente l’implementazione del server, e una cartella www/.

Immagine che contiene testo, Carattere, schermata, linea

Il contenuto generato dall'IA potrebbe non essere corretto.Quest’ultima contiene i file html, l’index e altre 4 pagine, e una cartella images/ con tutte le immagini utilizzate. Per passare da una pagina all’altra, ognuna di essa contiene un menu come quello in figura.

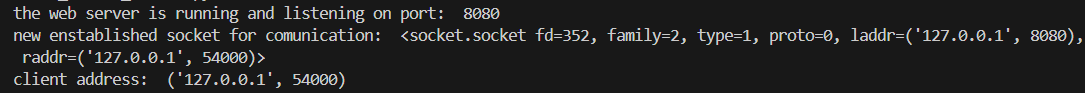
**Inizializzazione**

Per l’inizializzazione del server vengono utilizzati i seguenti comandi:

* **Socket:** viene creata una socket TCP, specificando SOCK\_STREAM, che fa uso dell’indirizzamento IPv4.
* **Bind:** richiamato sulla serverSocket, collega la socket all’indirizzo del server (in questo caso localhost:8080).
* **Listen:** la socket è in ascolto per richieste di connessione.

**Gestione delle richieste**

Il server entra in un ciclo infinito dove, tramite il metodo accept(), rimane bloccato fino a che non è stata accettata una richiesta di connessione di un client. Accept() restituisce quindi una nuova socket per la comunicazione e l'indirizzo del client, entrambi stampati su terminale.



Successivamente viene ricevuta la richiesta HTTP, tramite il metodo recv(), e vengono estratti il metodo e il percorso del file richiesto. Quest’ultimo viene quindi aperto in modalità binaria (rb) per supportare sia file di testo che immagini.

Se il percorso è /, viene servito index.html di default.

A questo punto si presentano due possibili situazioni da gestire:

1. File trovato
2. File non trovato

Nel primo caso il server risponde inviando, tramite il metodo send(), l’header “HTTP 200 OK”, per comunicare la buona riuscita della richiesta, e il contenuto del file in bytes. Nel secondo caso, invece, viene gestita un’eccezione IOError e il server risponde con l’header “HTTP 404 Not Found” e una semplice pagina HTML di errore.

Dopo ogni richiesta, la socket di comunicazione viene chiusa.

**Gestione dei MIME types**

Per determinare il MIME type si utilizza un dizionario, mime\_types, che associa l’estensione del file richiesto al rispettivo tipo per la corretta intestazione HTTP. Se l’estensione non è riconosciuta, viene usato come default “application/octet-stream”.

Immagine che contiene testo, schermata, Carattere

Il contenuto generato dall'IA potrebbe non essere corretto.

Nel caso in cui il server trovi il file, e riesce quindi a soddisfare la richiesta, esso invia in risposta l’header “Content-Type” con il MIME type corretto.

**Logging**

Ogni richiesta e ogni errore 404 vengono registrati tramite il modulo Logging, configurato per mostrare a terminale anche i log di livello info, con timestamp, indirizzo del client, metodo e file richiesto.



Immagine che contiene testo, schermata, Carattere, linea

Il contenuto generato dall'IA potrebbe non essere corretto.