Relazione tecnica per il Progetto di programmazione di reti:

Realizzazione di un Web Server minimale in Python e pubblicazione di un sito statico

Nicolas Tazzieri

1 giugno 2025

Indice

Capitolo 1

Introduzione

1.1 Descrizione e requisiti

Il progetto ha come obiettivo la realizzazione di un web server in Python che risponde a richieste HTTP effettuate sulla porta 8080. Il server deve essere in grado di:

- Gestire richieste HTTP.
- Rispondere con 200 OK per le richieste valide (4 pagine HTML statiche) e 404 per quelle non valide.
- Gestire il MIME type per i diversi tipi di file da restituire (text/html, style/css, image/img, application/pdf).
- Effettuare il logging delle richieste.

Capitolo 2

Implementazione

Il server Web è implementato in Python e utilizza le librerie standard per la gestione delle socket, del logging e dei MIME types. Il server è in ascolto sulla porta 8080 e accetta connessioni dai client. La coda di backlog è settata ad 1. Quando un client invia una richiesta HTTP, il server la riceve, la analizza e restituisce il file richiesto o una pagina di errore 404 se il file non viene trovato. Di seguito sono analizzati aspetti significativi del codice e della struttura del progetto.

2.1 Organizzazione dei file

La struttura dei file del progetto è la seguente:

- **HTTPserver.py**: file principale che contiene il codice del server web.
- **utilities.py**: file che contiene le funzioni di utilità per la gestione del logging, del MIME type e della lettura della 404 page.
- **www/**: cartella che contiene i file statici serviti dal server (HTML, CSS, immagini, report).
- **log/server.log**: file di log.
- **page404.html**: pagina di errore 404 che viene restituita quando il file richiesto non viene trovato.

2.2 Librerie utilizzate

Per la realizzazione del progetto sono state utilizzate le seguenti librerie standard di Python:

- **socket**: utilizzata per la gestione della connessione con i client.
- **time**: utilizzata per la gestione dei timestamp del logging.
- **mimetypes**: utilizzata per il riconoscimento del MIME type del file richiesto dall'utente in fase di richiesta.

2.3 Analisi delle parti significative del codice

2.3.1 Gestione delle richieste HTTP

Il server è in ascolto sulla porta 8080 e accetta connessioni dai client. Quando un client invia una richiesta HTTP, il server la riceve e la analizza per determinare il file richiesto.

Se la richiesta è valida, ovvero non viene lanciata l'eccezione `IOError` in caso di file mancante, il server restituisce il contenuto del file richiesto insieme a un'intestazione *HTTP 200 OK*, con il MIME type corretto (Il cui funzionamento è discusso nella sezione 2.3.3). Se il file non viene trovato, il server restituisce un'intestazione *HTTP 404 Not Found* e recupera la pagina `page404.html` con la funzione `get_404_page()` definita in `utilities.py`.

2.3.2 Gestione del logging

Il server registra le richieste in un file di log `log/server.log`. Questa funzionalità è implementata nella funzione `print_and_log()` in `utilities.py`, che si occupa sia di scrivere le informazioni di log nel file ma anche di stamparle sul terminale. In particolare, il log contiene informazioni riguardanti:

- L'indirizzo IP del client che si è connesso e ha effettuato la richiesta.
- La socket.
- Il mime type del file richiesto.
- Il percorso del file richiesto.
- La risposta inviata (*200 OK* o *404 Not Found*).
- Chiusura della connessione.

Per ogni informazione stampata, viene aggiunto un timestamp per tenere traccia del momento in cui la richiesta è stata ricevuta. Questa funzionalità è implementata attraverso la funzione `local_time()` della libreria `time` che restituisce la data e l'ora corrente, e `strftime()` che formatta il timestamp in una stringa leggibile.

2.3.3 Gestione dei MIME types

Il server gestisce i MIME types per i diversi tipi di file da restituire. Quando un client richiede un file, il server determina il suo tipo attraverso la funzione `guess_type(path)` della libreria `mimetypes`, che analizza la sua estensione e ritorna il suo tipo MIME se riconosciuto. Il server poi imposta l'intestazione `Content-Type` di conseguenza. Ad esempio, per un file HTML, l'intestazione sarà `Content-Type: text/html`.
Se un file non ha un MIME type riconosciuto, il server restituisce il tipo generico `application/octet-stream`. Siccome il server deve aprire il file in modalità diverse prima di inviarli al client (rb → file binari, r → file di testo), prima di inviare la risposta, controlla se il tipo di file comincia con `text/`. In tal caso, il server codifica il contenuto del file in UTF-8 utilizzando la funzione `encode('utf-8')`, altrimenti lo invia senza codificarlo.

2.3.4 Organizzazione delle pagine servite

Il server è configurato per servire tre pagine HTML statiche e una pagina di errore 404. Queste pagine sono organizzate nella cartella `www/` e sono accessibili tramite le seguenti URL:

- `http://localhost:8080/www/index.html`: la pagina principale del server.
- `http://localhost:8080/www/halloffame.html`: una seconda pagina HTML.
- `http://localhost:8080/www/landscape.html`: una pagina che mostra un paesaggio (*HG: Sunrise on the Reaping* per gli appassionati).
- `http://localhost:8080/www/report.html`: pagina che contiene la relazione tecnica del progetto e un link per scaricarla.
- `http://localhost:8080/page404.html`: la pagina di errore 404 che viene restituita quando il file richiesto non viene trovato.

A tutte le pagine HTML viene applicato il file CSS `www/style.css` per la formattazione e la gestione delle animazioni. Questo viene referenziato all'interno delle pagine HTML.

Capitolo 3

Conclusioni

3.1 Riflessioni finali e commenti personali

La realizzazione di questo progetto mi ha sicuramente permesso di approfondire la conoscenza del protocollo HTTP e della gestione delle richieste in un server web. È stato interessante implementare la funzionalità di gestione dei MIME types, che grazie alle librerie di Python è stata semplificata notevolmente. Inoltre ciò mi ha permesso di semplificare la selezione della modalità di lettura dei file, senza fare ulteriori parsing. Ammetto che nella realizzazione delle pagine HTML mi sono sentito particolarmente ispirato, spendendo tempo un po' più di quanto fosse necessario. Però mi sono divertito a farlo e credo sia questo l'importante (oltre ad imparare ovviamente).