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

Lorenzo Marchegiani - Matricola 0001126380

27 maggio 2025

Introduzione

Il presente documento descrive la realizzazione di un server HTTP minimale in Python, capace di servire file statici da una cartella locale, con l'obiettivo di pubblicare un sito web statico. Il server gestisce richieste GET, restituisce risposte HTTP appropriate, logga le richieste su file, e gestisce i tipi MIME dei file serviti.

Struttura del progetto

Il progetto è composto dai seguenti file e cartelle principali:

- `server.py`: lo script Python che implementa il server HTTP minimale.
- `www/`: cartella contenente le pagine HTML, i file CSS e una sottocartella `images/` con le immagini del sito.
- `server.log`: file di log generato automaticamente alla prima esecuzione del server, dove vengono registrate le richieste ricevute.
- `relazione.pdf`: il documento PDF contenente la relazione tecnica del progetto.

Struttura del Server e Funzionalità

Il codice Python sviluppato implementa un server TCP che ascolta sulla porta 8080 dell'host locale. Di seguito vengono commentate le principali funzionalità implementate.

Gestione delle richieste GET

Il server analizza le richieste in arrivo e verifica che siano richieste di tipo GET. In tal caso, estrae il nome del file richiesto, lo cerca all'interno della cartella www, e in caso di successo lo restituisce con una risposta HTTP 200:

```
    if len(message.split())>0 and message.startswith('GET'):
    method = message.split()[0]
    filename = message.split()[1]
    ...
    if filename == "/":
    filename = "/index.html"
    filepath = "www" + filename
    ...
    with open(filepath, 'rb') as f:
    outputdata = f.read()
    header = (
    f"HTTP/1.1 200 OK\r\n"
    f"Content-Type: {mime_type}; charset=utf-8\r\n"
    "\r\n"
    )
    connectionSocket.send(header.encode())
    connectionSocket.send(outputdata)

```

Listing 1: Gestione richieste GET

Gestione della risposta 404

Nel caso in cui il file richiesto non sia presente, viene sollevata un'eccezione FileNotFoundError. Il server risponde con una pagina HTML semplificata e uno status HTTP 404:

```
    except FileNotFoundError:
    logging.warning(f"File Not Found: {filepath}")
    connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n".
   encode())
    connectionSocket.send("404 Not Found".encode())

```

Listing 2: Risposta 404 Not Found

Gestione dei MIME Types

Il modulo mimetypes viene utilizzato per determinare il tipo MIME corretto del file richiesto. Questo garantisce che il browser interpreti correttamente il contenuto:

```
    mime_type, _ = mimetypes.guess_type(filepath)
    if mime_type is None:
    mime_type = "application/octet-stream"

```

Listing 3: Gestione MIME type

Logging delle richieste

Il server registra ogni richiesta in un file server.log grazie al modulo logging. Vengono tracciate sia le connessioni in ingresso che le richieste HTTP:

```
    logging.basicConfig(
    filename='server.log',
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s'
    )
    ...
    logging.info(f"Connection from: {addr}")
    logging.info(f"Request: {method} {filename}")

```

Listing 4: Logging delle richieste

Descrizione del sito web statico

Il sito realizzato è dedicato al fumetto *Watchmen*. L'idea alla base è stata quella di creare una piattaforma di propaganda a favore dei vigilanti protagonisti del fumetto. Questi personaggi, all'interno della narrazione, sono fortemente criticati dall'opinione pubblica e dalla politica, che cerca di metterli al bando tramite il *Keene Act*.

Il sito si compone di tre pagine HTML, ciascuna contenente testo e immagini, con l'obiettivo di riabilitare l'immagine degli Watchmen e sensibilizzare il pubblico sul loro operato.

Responsive layout e animazioni

Nel file `styles.css` è presente il seguente codice che consente un layout responsive:

```
    <meta name="viewport" content="width=device-width,
   initial-scale=1.0">

```

Listing 5: Meta tag per il responsive design

```
    .hero img {
      width: 80%;
      max-width: 600px;
    }

```

Listing 6: Immagini adattabili

Questo consente alle immagini nella sezione `.hero` di adattarsi alla larghezza dello schermo, mantenendo una massima larghezza di 600px, rendendo il sito fruibile anche da dispositivi mobili.

Per rendere il sito più dinamico, è stata introdotta anche una semplice animazione CSS:

```
    animation: pulse 2s infinite;

```

Listing 7: Animazione pulsante

Questa animazione applica un effetto visivo ciclico che richiama l'attenzione dell'utente su specifici elementi del sito.

Conclusioni

Il server realizzato rappresenta un esempio minimale ma funzionante di server HTTP in Python. Permette di comprendere le basi della comunicazione client-server, la struttura di una richiesta e risposta HTTP, e la gestione dei contenuti statici.