Relazione tecnica

Programmazione di Reti – Progetto di laboratorio

Traccia 1

Nome: Matteo Grandini

E-mail: matteo.grandini@studio.unibo.it

Matricola: 0001115474

Funzionalità realizzate

Il codice python realizzato implementa un semplice server HTTP accessibile su localhost (127.0.0.1) alla porta 8080. Esso è in grado di rispondere alle richieste GET fornendo le risorse richieste, in particolare con codice 200 se la risorsa esiste e con codice 404 altrimenti. Vengono inoltre gestiti alcuni tipi MIME e viene effettuato il logging su terminale di tutte le richieste.

Inizializzazione server

Il server viene creato utilizzando il modulo **`socket`** di python, in particolare passando:

- `AF_INET:` Per utilizzare indirizzi IPv4
- `SOCK_STREAM:` Per utilizzare il protocollo TCP

Dopodiché viene chiamato il metodo **`bind()`** per collegare il socket all'indirizzo *127.0.0.1:8080* e il metodo **`listen()`** per iniziare l'ascolto delle richieste.

Viene utilizzato il costrutto **`with`** di python per garantire la chiusura del socket a fine esecuzione.

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen(1)
    print(f"Il server e' disponibile su {HOST}:{PORT}")
```

Gestione richieste

Una volta inizializzato il server, questo entra in un ciclo infinito utilizzando il metodo **`accept()`** per accettare le richieste di connessione entranti. L'oggetto riferito alla connessione viene poi passato alla funzione **`handle_request()`** (per maggiore leggibilità del codice).

```
while True:
    conn, addr = s.accept()
    log(f"Connessione da {addr[0]}:{addr[1]}")
    handle_request(conn)
```

A questo punto viene ricevuta la richiesta HTTP tramite il metodo **`recv(1024)`** dalla quale vengono estratti il metodo (GET) e il percorso alla risorsa richiesta.

```
request = conn.recv(1024).decode()
if not request:
    return

# Prendo il metodo e il percorso alla risorsa richiesta
method = request.split()[0]
path = request.split()[1]
```

Nota: Nel caso in cui il metodo richiesto non sia GET, viene mandata una risposta "*405 Method Not Allowed*", poiché il server realizzato non è in grado di gestire altri tipi di richieste.

Una volta ottenuto il percorso alla risorsa richiesta, questo viene concatenato al percorso della directory **`www/`** per ottenere il percorso reale sul file system. Il file viene letto in modalità binaria e inviato assieme all'header con codice "*200 OK*" tramite il metodo **`send()`**.

```
# Unisco il percorso della root del server con il percorso della risorsa richiesta
# (utilizzo l'os.path.join per evitare problemi con sistemi operativi diversi)
filepath = os.path.join(WEB_ROOT, path.lstrip("/"))
if os.path.isfile(filepath):
    with open(filepath, 'rb') as f:
        content = f.read()

    # Determino il mime type in base all'estensione del file richiesto
    ext = os.path.splitext(filepath)[1]
    mime_type = MIME_TYPES.get(ext, "application/octet-stream")

    # Invio l'header di risposta insieme al contenuto del file
    conn.send("HTTP/1.1 200 OK\r\n".encode())
    conn.send(f"Content-Length: {len(content)}\r\n".encode())
    conn.send(f"Content-Type: {mime_type}\r\n\r\n".encode())
    conn.send(content)
    log_request(method, path, 200)
```

Nel caso in cui il file non esista, viene invece mandata una risposta con codice "*404 Not Found*".

Infine, viene chiusa la connessione.

Gestione tipi MIME

Per identificare il tipo MIME delle risorse richieste, viene estratta l'estensione del file aperto utilizzando **`os.path.splitext(…)[1]`** e recuperando il corrispondente MIME type all'interno di un dizionario.

```
MIME_TYPES = {
    ".html": "text/html",
    ".css": "text/css",
    ".webp": "image/webp",
    ".jpg": "image/jpeg",
    ".jpeg": "image/jpeg",
    ".png": "image/png",
    ".gif": "image/gif"
}
```

Nota: Nel caso di tipi di file non presenti nel dizionario viene assegnato il tipo "*application/octet-stream*" come da standard.

Logging richieste

Il logging viene effettuato su terminale stampando stringhe formate da un timestamp e da un messaggio, in particolare:

- Provenienza della connessione
- Invio della risposta
    - Codice della risposta
    - Metodo utilizzato
    - Percorso alla risorsa richiesta

```
[2025-05-22 19:22:46.054448] Connessione da 127.0.0.1:61278
[2025-05-22 19:22:46.055432] 200 GET /index.html
```

```
[2025-05-22 19:23:07.882807] Connessione da 127.0.0.1:61290
[2025-05-22 19:23:07.883807] 404 GET /pippo/
```

Pagine HTML

Le pagine sono state realizzate utilizzando HTML e CSS. Da ogni pagina è possibile navigare alle altre attraverso dei collegamenti.