

Relazione per  
“Programmazione di reti”  
Traccia 2 - Python Web Server

Valerio Di Zio - Matricola: 942637

24 luglio 2021

## 0.1 Introduzione

Il progetto consiste nella realizzazione di un web server in linguaggio Python per un'azienda ospedaliera che permetta all'utente di visualizzare, tramite browser, i servizi erogati e i dipartimenti disponibili.

## 0.2 Descrizione

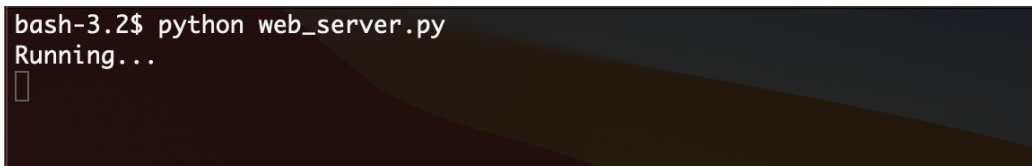
### 0.2.1 Requisiti minimi per l'avvio

- Python  $\geq$  2.7

### 0.2.2 Utilizzo del software

Per avviare il web server basterà eseguire: *"python web\_server.py"* nel terminale.

Il server resterà in ascolto sull'indirizzo 127.0.0.1:8080; se si vuole, invece, avviare il web server con una porta personalizzata basterà indicarla come argomento all'avvio: *"python web\_server.py 8081"*.



```
bash-3.2$ python web_server.py
Running...
█
```

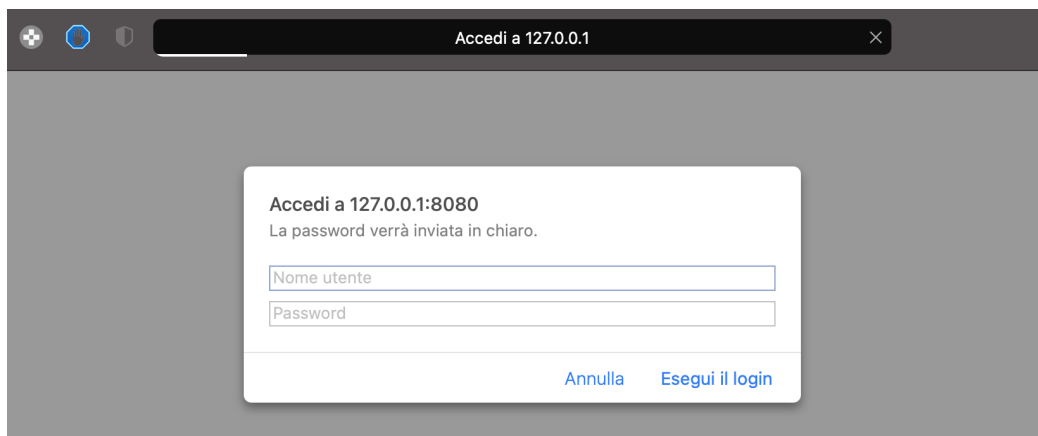
Il progetto contiene diversi file:

- **web\_server.py**: codice del web server.
- **index.html**: la pagina web principale.
- **style.css**: il foglio di stile utilizzato dalle varie pagine web.
- **pdf.pdf**: il documento pdf scaricabile dal web server.
- **dipartimenti/\*.html**: varie pagine che compongono il sito.
- **tests/close-ctrl+c-test.sh**: script per controllare le porte attualmente aperte.
- **tests/load-test.sh**: script per testare connessioni parallele.

Quando verrà effettuata una connessione al server, quest'ultimo risponderà soddisfacendo le richieste. Informazioni sulle richieste effettuate (ip, data, ora, tipo, etc...) saranno visualizzabili nella console.

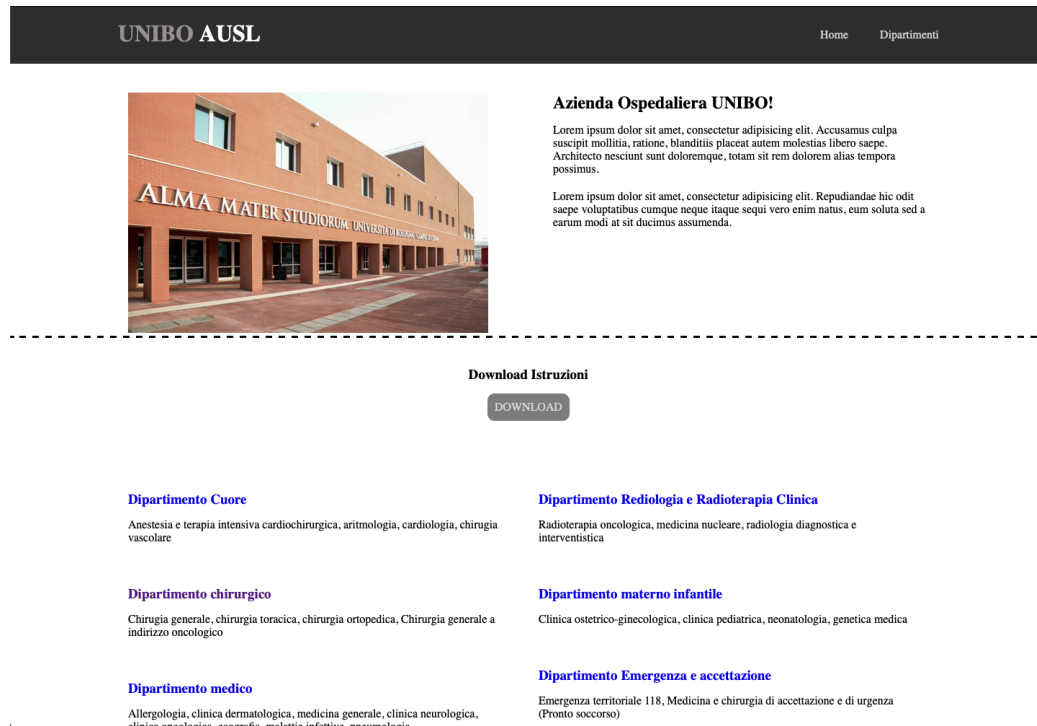
```
bash-3.2$ python web_server.py
Running...
127.0.0.1 - - [23/Jul/2021 15:57:02] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [23/Jul/2021 15:57:11] "GET /dipartimenti/cuore.html HTTP/1.1" 200 -
127.0.0.1 - - [23/Jul/2021 15:57:14] "GET /index.html HTTP/1.1" 200 -
127.0.0.1 - - [23/Jul/2021 15:57:18] "GET /dipartimenti/radiologia.html HTTP/1.1" 200 -
```

Spostandoci nel web browser che effettua la connessione, notiamo una finestra che ci chiede delle credenziali per accedere al server, le credenziali impostate di default sono USERNAME: admin, PASSWORD: programmazioneiredireti.



Se le credenziali inserite sono corrette, il server ci permetterà di visitare il sito web; in caso contrario, il server ci chiederà di reinserirle.

La prima pagina del sito ci permetterà di visionare una breve descrizione, un'elenco dei dipartimenti dell'azienda sanitaria e ci permetterà di scaricare un file pdf.



## 0.3 Implementazione

L'implementazione del web server è stata fatta utilizzando il linguaggio Python. Analizziamo nel dettaglio le varie funzionalità.

### 0.3.1 Accasso da più utenti simultaneamente

Grazie all'utilizzo del multithreading, il server riesce a gestire più richieste in contemporanea. L'implementazione è basata sulla libreria *"socketserver"*.

Il test per verificare questa funzionalità è eseguibile lanciando lo script:  
`./load-test.sh 100 127.0.0.1:8080`.

Si noti l'orario nelle risposte del server.

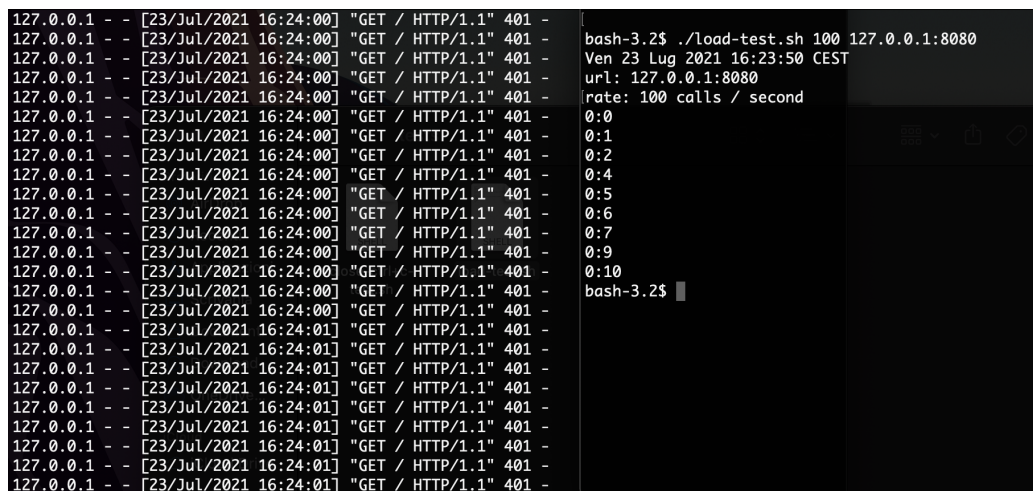
The image shows a terminal window with two columns of output. The left column contains a series of log entries from a server, each starting with '127.0.0.1 - -' followed by a timestamp in brackets, an HTTP method and path in quotes, and a status code. The timestamps range from 16:24:00 to 16:24:01. The right column shows the output of a shell script execution. It starts with 'bash-3.2\$ ./load-test.sh 100 127.0.0.1:8080', followed by the date and time 'Ven 23 Lug 2021 16:23:50 CEST', the URL 'url: 127.0.0.1:8080', and the rate 'rate: 100 calls / second'. Below this, there is a vertical list of numbers from 0:0 to 0:10, and finally 'bash-3.2\$'.

Figura 1: A sinistra le risposte del server, a destra l'esecuzione dello script

### 0.3.2 Interruzione da tastiera opportunamente gestita in modo da liberare la risorsa socket

L'interruzione del server è stata opportunamente gestita intercettando la combinazione di tasti CTRL+C da tastiera. Alla pressione simultanea dei due tasti il server, si chiuderà automaticamente rilasciando le risorse precedentemente occupate. L'implementazione è basata sulla libreria *"signal"*, specificando come parametro **signal.SIGINT**.

### 0.3.3 Download di un file pdf dal web server

Sul sito web è presente un pulsante, al centro della pagina, che permetterà di scaricare un documento pdf di dimensione 20,7 MB: per forzare il download del file è stato aggiunto l'header Content-Disposition alla risposta.

### Compressione dati con GZIP

Per diminuire la quantità di dati trasmessi (a discapito del carico del server) è stata aggiunta una funzionalità che permette di comprimere il documento pdf direttamente da server.

La compressione è effettuata utilizzando l'header Content-Encoding: gzip, sfruttando la libreria *"zlib"*.

Per disattivare la compressione basta impostare la costante **USE\_GZIP\_COMPRESSION** al valore **False**.

Dall'immagine possiamo notare come la Content-Length diminuisca utilizzando la compressione GZIP.

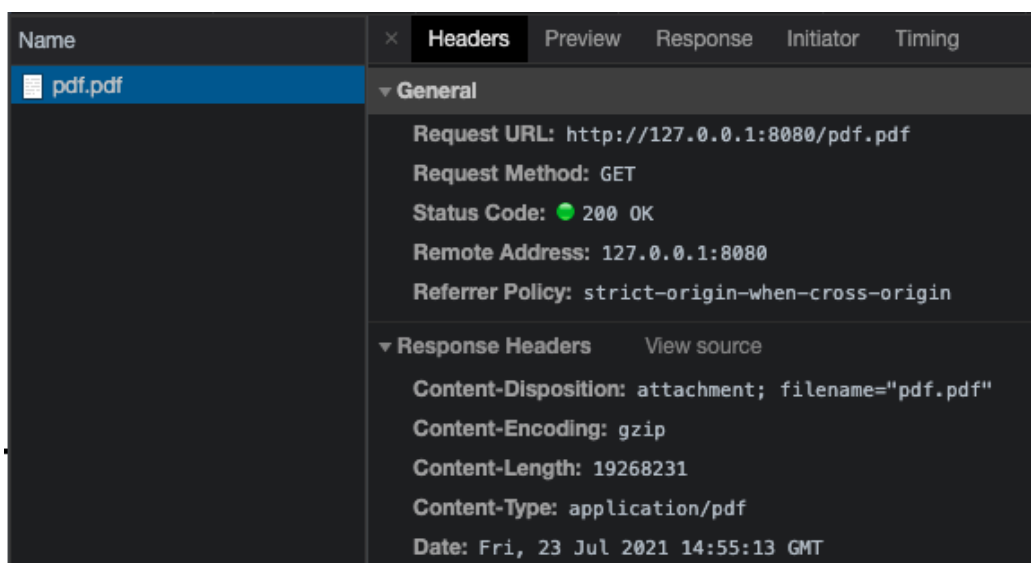


Figura 2: **Utilizzando** la compressione GZIP la Content-Length è: 19268231 Byte

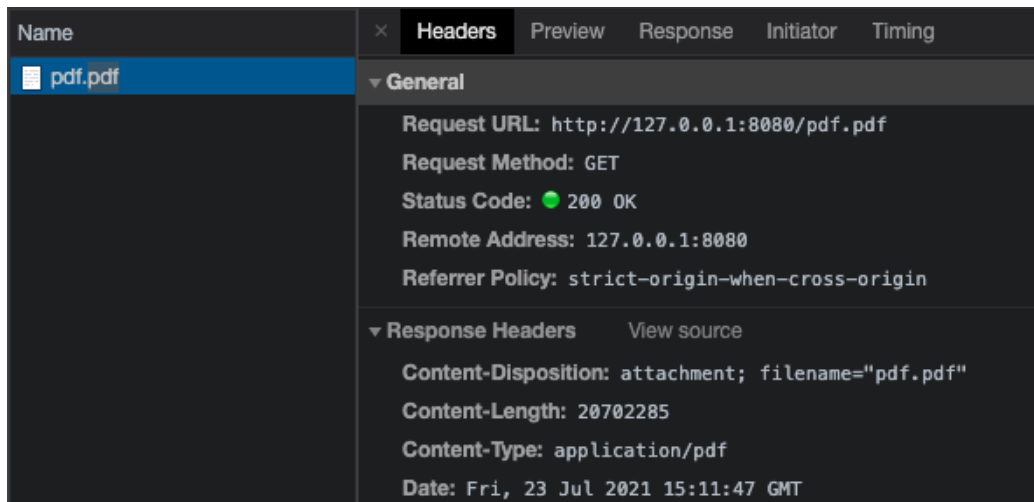


Figura 3: **Non Utilizzando** la compressione GZIP la Content-Length è: 20702285 Byte

### 0.3.4 Autenticare gli utenti nella fase iniziale della connessione

Quando ci colleghiamo al server, è richiesta un'autenticazione HTTP Basic attraverso un "Nome Utente" e una "Password". Questa funzionalità è stata implementata aggiungendo l'header **Authorization**.

#### Utilizzo dell'encoding Base 64 per la verifica delle credenziali

Per la verifica delle credenziali è necessario:

1. leggere in contenuto dell'header Authorization e prendere la parte dopo il prefisso "Basic".
2. convertire in Base 64 (libreria: *base64*) la stringa nomeUtente:password (memorizzate nel codice).
3. confrontare le due stringhe.

Nota: l'autenticazione HTTP Basic, senza l'utilizzo del protocollo HTTPS, è insicura.

```
Accept-Encoding: gzip, deflate, br
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7
Authorization: Basic YWRtaW46cHJvZ3JhbW1hem1vbmVkaXJldGk=
Cache-Control: max-age=0
Connection: keep-alive
Host: 127.0.0.1:8080
If-Modified-Since: Fri, 23 Jul 2021 09:20:19 GMT
```

Figura 4: Esempio del contenuto dell'header Authorization

## 0.4 Librerie utilizzate

- sys
- signal
- http.server
- socketserver
- threading
- os
- base64
- zlib

## 0.5 Repo Github

<https://github.com/v4l3rio/pythonWebServer-programmazioneDiReti>

## 0.6 Extra

Il web server fornisce, come funzionalità aggiuntiva, la compressione GZip permettendo un risparmio della quantità di dati scambiati.

Sono stati forniti degli script che permettono di aiutare l'utente nel capire se le risorse non sono state liberate e di verificare il supporto a connessioni multiple.



## 0.7 Conclusioni

Durante la realizzazione del progetto sono stati affrontati molti argomenti. Tra questi, alcuni erano stati precedentemente trattati a lezione e applicarli nella realizzazione del progetto mi ha permesso di capirli/approfondirli a fondo.

Altre invece, in particolare le “funzionalità extra” hanno richiesto uno studio più approfondito/nel dettaglio, anche di argomenti non trattati. Questo mi ha dato modo di ampliare le mie conoscenze e capire, ancora di più, gli argomenti trattati a lezione.