

## **Traccia 2: Python Web Server**

Realizzazione di un server che gestisca un sito  
per un'azienda ospedaliera

Matteini Mattia  
Paganelli Alberto

19 agosto 2021

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
1.1	Descrizione . . . . .	2
1.2	Requisiti . . . . .	3
<b>2</b>	<b>Design</b>	<b>4</b>
2.1	Client . . . . .	4
2.2	Server . . . . .	5
2.3	Threading . . . . .	5
<b>3</b>	<b>Guida</b>	<b>6</b>

# Capitolo 1

## Introduzione

### 1.1 Descrizione

La piattaforma web realizzata elenca i principali servizi offerti dal Policlinico di Milano. Riassume tutte le informazioni principali per usufruire di essi e le istruzioni necessarie per poter consultare esami, risultati o il proprio fascicolo sanitario.

Sarà presente una homepage del sito web contenente una piccola preview e descrizione dei servizi offerti oltre ad un bottone per poter scaricare un pdf che raccoglie le informazioni più importanti relative alla struttura.

Dalla homepage si possono visualizzare i servizi e si può navigare in pagine specifiche per ognuno di essi, all'interno di ogni pagina dedicata sarà presente anche un collegamento per tornare all'homepage.

Inoltre sarà possibile conoscere i volontari che si offrono all'aiuto dei cittadini nella sezione dedicata "Volontari". All'interno di quest'ultima si potrà scaricare un elenco dei volontari nei quali sarà presente il numero di telefono ed un nominativo.

Un'ulteriore sezione è quella dei contatti nella quale l'utente potrà trovare un contatto mail oltre a link Linkedin e Facebook del Policlinico di Milano.

## 1.2 Requisiti

1. IL SERVER DOVRÀ MANTENERE PIÙ CONNESSIONI IN CONTEMPORANEA PER GARANTIRE L'ACCESSO A PIÙ UTENTI
2. NELL'HOMEPAGE DOVRANNO ESSERE PRESENTI DEI COLLEGAMENTI AI SERVIZI OFFERTI
3. POSSIBILITÀ DI SCARICARE UN PDF RIASSUNTIVO NEL QUALE SONO ELENCATI I SERVIZI
4. INTERRUZIONE DELL'ESECUZIONE OPPORTUNAMENTE GESTITA
5. PAGINA HTML SINGOLA PER OGNI SERVIZIO

# Capitolo 2

## Design

### 2.1 Client

Per lo sviluppo del client sono state usate le seguenti librerie minimizzate:

1. BOOTSTRAP 4.5.2
2. JQUERY 3.5.1

Per quanto riguarda la navigazione tra le pagine HTML del sito è stato sviluppato modificando il `location.href` il quale effettua una richiesta GET alla risorsa desiderata.

Mentre il download dei pdf è stato fatto in maniera molto semplificata ed occorrerà sostituire il pdf in caso di aggiornamento dei servizi o dei volontari.

## 2.2 Server

Il server scritto in **Python** utilizza il package **socketserver** e in particolare la classe **ThreadingTCPServer** per gestire molteplici e semplici richieste alla volta. Per la costruzione dell'istanza non è stato inserito nessun indirizzo nello specifico così in automatico verrà riconosciuto l'accesso sia da localhost che da indirizzo ip del proprio computer nella rete al quale è connesso (esempio: 192.168.1.20). Per la gestione delle richieste utilizziamo la classe **SimpleHTTPRequestHandler**, appartenente al package **http**, che è più che necessario per il nostro applicativo.

Come operazioni preliminari rendiamo utilizzabile lo stesso indirizzo IP a più richieste in entrata (**allow\_reuse\_address**), e poi facciamo sì che quando il programma termina vengano terminati anche tutti i threads in sospenso generati dal server (**daemon\_threads**).

Successivamente definiamo un metodo che gestisca l'uscita dal programma e la chiusura del socket server, per questo utilizziamo il package **signal** che ci permette di intercettare i diversi segnali mandati al processo, in questo caso un SIGINT (2).

Infine abbiamo un **while True** dentro al quale chiamiamo il metodo **server\_forever** che dà vita a un nuovo thread il quale gestirà le richieste che arrivano e bloccherà il thread principale, inoltre lasciamo il **poll\_interval** di default che è di 0.5 secondi, quindi ogni mezzo secondo verrà controllato se viene ricevuto una richiesta di shutdown.

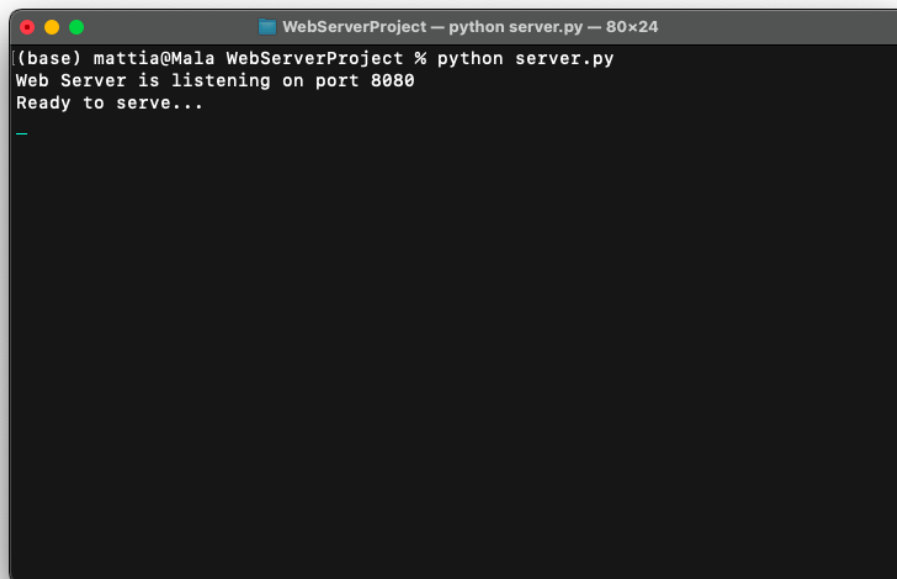
## 2.3 Threading

Per quanto riguarda la gestione di più richieste nello stesso momento, utilizzando un **ThreadingTCPServer** abbiamo un Main Thread avviato dal metodo **serve\_forever**, il quale crea un altro thread per ogni richiesta entrante in modo da poterle gestire tutte contemporaneamente.

# Capitolo 3

## Guida

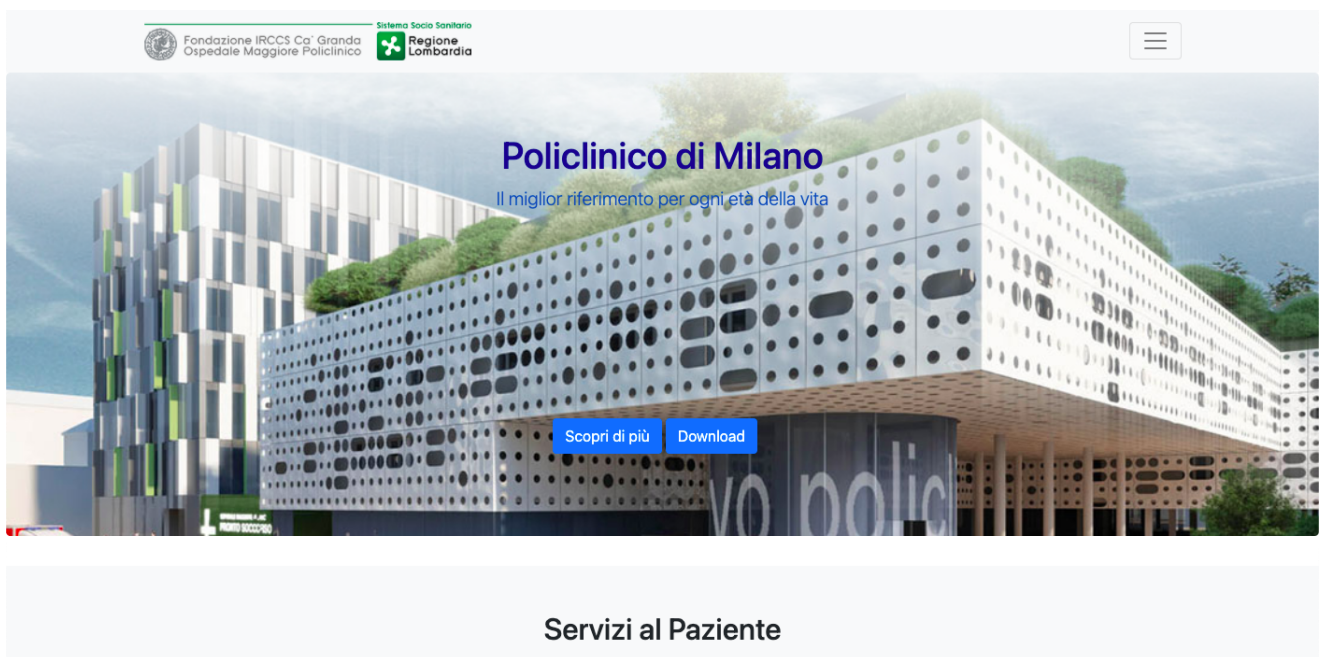
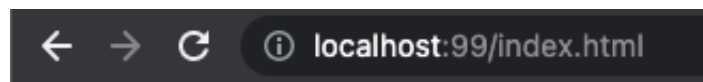
Per avviare il server basterà dare il seguente comando da terminale "`python server.py PORT`" dove `PORT` è il numero della porta che si vuole utilizzare per l'ascolto, se non si inserisce nessun parametro di default verrà utilizzata la porta 8080.

A screenshot of a terminal window with a dark background. The title bar at the top reads "WebServerProject — python server.py — 80x24". The terminal content shows a prompt "(base) mattia@Mala WebServerProject %" followed by the command "python server.py". Below the command, the output is "Web Server is listening on port 8080" and "Ready to serve...". A single hyphen "-" is visible on the line following the output.

```
(base) mattia@Mala WebServerProject % python server.py
Web Server is listening on port 8080
Ready to serve...
-
```

Per visualizzare la pagina principale basterà avviare un browser e digitare il seguente link "`localhost:PORT/index.html`", sostituendo PORT col numero di porta che si è data al server.

N.B. si può anche digitare l'indirizzo IP della macchina sulla quale sta girando il server al posto di utilizzare "`localhost`".





Si può notare la gestione delle richieste effettuate dal browser verso il server, il quale risponde con i percorsi delle risorse necessitate dal browser per far visualizzare la pagina web (file, immagini, ecc...).

```
WebServerProject — python server.py — 104x24
(base) mattia@Mala WebServerProject % python server.py
Web Server is listening on port 8080
Ready to serve...
127.0.0.1 -- [19/Aug/2021 14:38:26] "GET / HTTP/1.1" 304 -
127.0.0.1 -- [19/Aug/2021 14:38:27] "GET /img/logo/logo.png HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:30] "GET / HTTP/1.1" 304 -
127.0.0.1 -- [19/Aug/2021 14:38:30] "GET /img/logo/logo.png HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET / HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /libraries/jquery-3.5.1.min.js HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /libraries/bootstrap-4.5.2/css/bootstrap.min.css HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /libraries/bootstrap-4.5.2/js/bootstrap.min.js HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /js/article.js HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/logo/logo.png HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/services/esami-laboratorio.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/services/referti.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/services/cartella-clinica.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/services/ricovero.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:36] "GET /img/services/volontariato.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:37] "GET /img/NuovoPoliclinico.jpeg HTTP/1.1" 200 -
127.0.0.1 -- [19/Aug/2021 14:38:37] "GET /img/logo/logo.png HTTP/1.1" 200 -
```