Relazione per il progetto di "Programmazione di Reti"

Arianna Pagano

A.A 2020/2021

0.1 Introduzione

A seguito della presa visione delle tracce dei progetti di fine corso 2020/2021, ho scelto di svolgere la *Traccia numero 2*.

0.2 Traccia del progetto

Si immagini di dover realizzare un *Web Server* in *Python* per una Azienda Ospedaliera.
I requisiti del Web Server sono i seguenti:

- Il web server deve consentire l'accesso a più utenti in contemporanea
- La pagina iniziale deve consentire di visualizzare la lista dei servizi erogati dall'azienda ospedaliera e per ogni servizio avere un link di riferimento ad una pagina dedicata
- L'interruzione da tastiera (o da console) dell'esecuzione del web server deve essere opportunamente gestita in modo da liberare la risorsa socket
- Nella pagina principale dovrà anche essere presente un link per il download di un file pdf da parte del browser

0.3 Specifiche generali progetto

Questo progetto ha l'obbiettivo di creare un *Web Server* in *Python* rendendo visibile in maniera chiara, precisa e senza alcun tipo di problema, tutti i servizi offerti da un'azienda ospedaliera, facilitando la visualizzazione grazie ad una barra menù intuitiva.

Nella pagina inziale infatti, troviamo un menù in cui grazie al solo *click*, è possibile essere portati ad una pagina del servizio scelto, con la presenza di una descrizione.
All'interno delle varie pagine disponibili per i servizi, è possibile anche notare foto, disegni, o più semplcemente immagini, presenti per migliorare la visione dell'intero sito. Tutto ciò è possibile grazie alla presenza della classe 'http.server.SimpleHTTPRequestHandler' che recupera i file dalla directory corrente e da quelle in essa contenute.

Oltre a ciò è possibile, grazie alla presenza dell'apposito tasto nel Menù posto a destra, di poter scaricare questo file pdf direttamente da lì, così da

poterlo visionare senza problemi.

In più, come specifica aggiuntiva, in basso a sinistra, è possibile notare la possibilità di poter aggiungere dei commenti in un modo molto semplice grazie al prorpio Nome, Cognome ed e-mail, così da dare un parere e poter richiedere servizi che al momento non sono ancora presenti all'interno dell'Azienda Ospedaliera. Dopo aver scritto il commento e messo i dati relativi, basterà cliccare il tasto *Invia* e si aprirà una nuova pagina in cui sarà possibile visualizzare il messaggio scritto.

0.4 Specifiche dettagliate del progetto

All'interno del Menù possiamo trovare molti pulsanti differenti, che possono essere riassunti in questo modo:

- Home, che permette di tornare alla schermata iniziale
- Prelievi per visite mediche
- Tamponi molecolari
- Riabilitazione
- Consulenza con medici specializzati
- Farmaci
- Download pdf, che permette di poter scaricare nel proprio computer questa Relazione

Tutte le richieste effettuate sul *Web Server* vengono salvate nel file "AllRequestsGET.txt" per mantenere uno storico delle azioni eseguite. Questo file viene azzerato ogni volta che il webserver viene riavviato.
Mentre tutti i commenti effettuati, saranno salvati nel file chiamato "AllPOST.txt" così da poter essere sempre visionati.

0.5 Dettagli implementativi

Lo sviluppo di questo progetto è stato implementato in modo tale da poter permettere il collegamento da più dispositivi.
Per far si che questo accada, come prima cosa, il server HTTP viene messo in ascolto sulla porta fornita o in alternativa in quella specificata come di

deafult (in questo caso la 8080).
Il server crea *thread*, o processi figli, incaricati di rispondere e che, quando avranno esaurito il loro compito, termineranno.
Per far si che si riesca a gestire più richieste, in Windows è presente la funzione ThreadingTCPServer e la relativa classe http.server.SimpleHTTPRequestHandler.

Successivamente per gestire al meglio il comportamento della connessione thread ereditata da ThreadingTCPServer, è necessario dichiarare esplicitamente come si desidera che i thread si comportino in caso di arresto improvviso. Infatti la classe ThreadingTCPServer definisce un attributo 'daemon threads', che indica se il server deve attendere la terminazione del thread.

Infine è presente la funzione 'signal.signal' che consente di definire handler personalizzati da eseguire quando si riceve un segnale. Infatti in questo caso, viene inserito un handler ben preciso (SIGINT) che interrompe l'esecuzione da tastiera quando arriva la sequenza (CTRL + C).

0.6 Librerie utilizzate

Le librerie utilizzate per lo sviluppo del progetto, sono state:

- sys, che serve per accettare informazioni inviate tramite la riga di comando signal
- signal, che rappresenta la riga di comando tramite la quale vengono inviate le informazioni e che consente di definire handlers personalizzati da eseguire quando si riceve un segnale
- http.server, che crea e ascolta sul socket HTTP, inviando le richieste a un handler
- socketserver
- cgi

0.7 Conclusioni

In caso si verifichino problemi con la cartella compressa da me inviata, la informo che è possibile visionare l'intero progetto al seguente link di GitHub:

`https://github.com/AriannaPagano/WebServer-Project.git`

0.8 Informazioni Personali

Nome e Cognome: *Arianna Pagano*
Mail istituzionale: *arianna.pagano@studio.unibo.it*
Numero matricola: *0000936491*