Relazione Progetto

Traccia 2 - Python Web Server

Gianmaria Casamenti
0000925151
gianmaria.casamenti@studio.unibo.it

Introduzione

Il progetto consiste nel dover realizzare un Web Server in codice Python per un azienda ospedaliera.
L'applicativo deve presentare i seguenti requisiti funzionali:

- Connessione di più utenti contemporaneamente.
- La pagina iniziale deve far visualizzare i principali servizi dell'azienda e per ogni servizio avere un link di riferimento ad una pagina dedicata.
- L'interruzione da tastiera dell'esecuzione del Web Server e la giusta procedura per liberare la risorsa socket.
- Il Web Server deve presentare la possibilità di scaricare un file di tipo pdf.

Descrizione

Per la realizzazione del Web Server ho deciso di far uso di due linguaggi principali:

- Python : per la gestione delle risorse e per l'implementazione del programma, così da poter gestire in maniera migliore i threading per eseguire più richieste e l'aggiornamento delle pagine.
- HTML : per la creazione delle pagine con le relative disposizioni degli elementi all'interno di esse.

Dopo l'analisi dei requisiti fondamentali ho organizzato il progetto nel seguente modo:
All'esecuzione del progetto viene lanciata la pagina principale, che corrisponde a index.html, dove si può visualizzare oltre al titolo la barra di selezione dei servizi, su cui ci si può liberamente muovere per passare da un servizio ad un altro.

Figura 1: Homepage del Web Server

Sottostante alla barra ho optato per aggiungere una breve descrizione del servizio che si è scelto ed è sempre presente un link, interno o esterno al Web Server, che rimanda ad ulteriori informazioni.
Un esempio può essere il servizio “pronto soccorso” che una volta selezionato ti da l’opportunità di ricercare la sede più vicina a te.

Figura 2: Pagina Servizio ”pronto soccorso”

Infine ho aggiunto, all’interno della tabella 2 ulteriori scelte, nonostante non siano dei veri servizi erogati dall'azienda ospedaliera “Aggiorna” e “Accedi”
La prima dà la possibilità di fare un refresh del Web Server, ripartendo nuovamente dall’homepage, invece la seconda mostra un interfaccia di login che permette l’autenticazione del FSE (fascicolo sanitario elettronico).

Dettagli implementativi

Questa sezione riporta le spiegazioni tecniche riguardanti le soluzioni di alcuni problemi riscontrati in fase di sviluppo.

Gestione di più utenti:

Per la gestione di più richieste ho fatto affidamento alla programmazione multithreading, implementata all'interno del modulo "socketserver", utilizzando metodo utilizzato socketserver.ThreadingTCPServer(), che utilizza la generica classe socket utilizzando
il protocollo internet TCP, permettendo un continuo flusso di dati client-server.

Gestione del refresh:

Per aggiornare le informazioni delle pagine è stato necessario definire il metodo launch_thread_resfresh() che ogni 300s (cioè 5 min) utilizza i thread per fare un refresh dei servizi.

Gestione dell'uscita:

L'uscita dal web server è gestita sia da console che input da tastiera, proprio il secondo metodo è implementato all'interno della definizione signal_handler(signal, frame).
Di conseguenza si è presentata la necessità di fermare il thread refresh senza busy waiting e questo è stato possibile grazie al metodo waiting_refresh.set().

Librerie utilizzate

Durante lo svolgimento del programma ho ritenuto necessario l'utilizzo delle seguenti librerie:

- sys
- htt.server
- socketserver
- threading

import sys :

La libreria sys è stata utile perchè questo modulo fornisce l'accesso ad alcune variabili usate o mantenute dall'interprete, e a funzioni che interagiscono fortemente con l'interprete stesso come ad esempio sys.argv[..], sys.signal e sys.exit.

import http.server :

Questo modulo è stato utilizzato per facilitare la creazione del Web Server importando le classi HTTPServer e BaseHTTPRequestHandler.
Quest'ultimo include anche le funzionalità che, su Python 2, venivano implementate dal modulo SimpleHTTPServer.

import socketserver :

Ho ritenuto importare il modulo socketserver che semplifica il compito di scrivere server di rete, aggiungendo funzionalità di grande importanza come, ad esempio, il metodo socketserver.ThreadingTCPServer necessario per gestire più richieste contemporaneamente.

import threading :

Quest'ultimo modulo è stato utilizzato per programmare il multithreading, inoltre mi permette di eseguire parti del programma come sottoprocessi indipendenti.