RELAZIONE

Progetto Programmazione di Reti, Traccia 2

Lorenzo Leoni
916119
lorenzo.leoni5@studio.unibo.it
a.a. 2020/21

Indice

1. Introduzione - traccia progetto

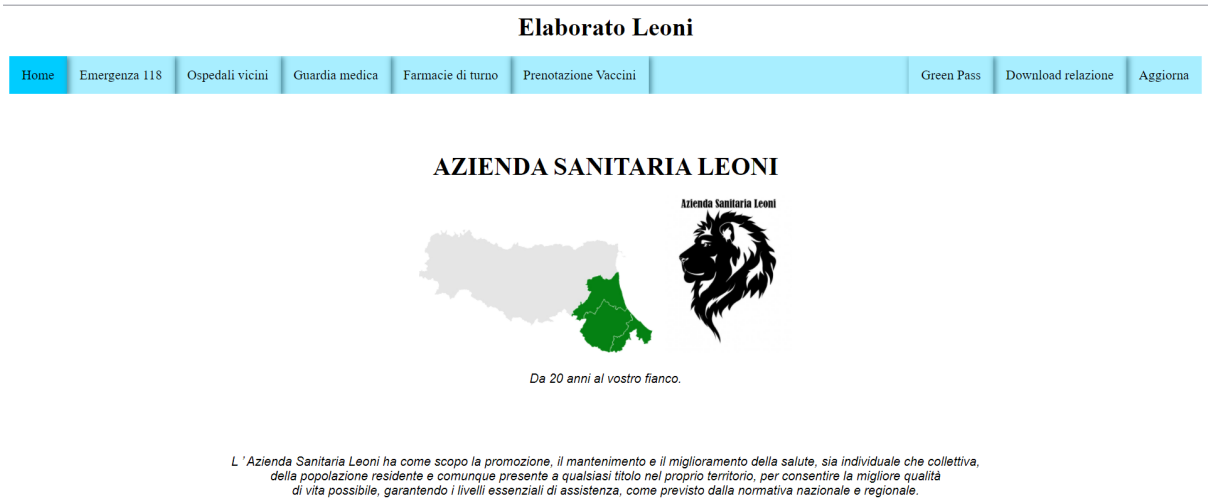
Il progetto consiste nella realizzazione di un Web Server in Python per una azienda ospedaliera. I requisiti da rispettare (definiti dalla traccia) sono i seguenti:

1. Il web server deve consentire l'accesso a più utenti in contemporanea.
2. 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.
3. L'interruzione da tastiera (o da console) dell'esecuzione del web server deve essere opportunamente gestita in modo da liberare la risorsa socket.
4. Nella pagina principale dovrà anche essere presente un link per il download di un file pdf da parte del browser.
5. Come requisito facoltativo si chiede di autenticare gli utenti nella fase iniziale della connessione

2. Descrizione

La homepage presenta una barra di navigazione orizzontale, dove vi sono indicati tutti i servizi offerti dall'Azienda Sanitaria Leoni, premendo su uno di essi si verrà indirizzati alla pagina ad esso dedicata.
I servizi proposti sono:

- Emergenze 118 e Pronto Soccorso
- Ospedali nelle vicinanze
- Guardia Medica
- Farmacia di turno
- Prenotazione vaccini
- GreenPass



Ogni pagina dei servizi mantiene la lista dei servizi e inoltre presenta una breve descrizione e un link che porta a un sito esterno, nel quale si possono visionare le informazioni richieste. Nella barra di navigazione sono presenti altri due pulsanti, il primo per il download della relazione, il secondo per il refresh della pagina

3. Dettagli implementativi

Per implementare quanto richiesto nella traccia dell'elaborato ho deciso di lavorare nel seguente modo.

3.1. Gestione di più utenti

Per la gestione di più utenti contemporaneamente ho sfruttato quanto offerto dal modulo socketSever. Port rappresenta il valore della porta che viene passato dalla shell di spyder.

```
# ThreadingTCPServer per gestire più richieste
server = socketserver.ThreadingTCPServer(('127.0.0.1',port),
ServerHandler)
```

3.2. Creazione pagine dei servizi

Per implementare questo requisito ho deciso di utilizzare il seguente metodo:

```
#metodo per la creazione di una generica pagina
def create_page_servizio(title,file_html, end_page):
    f = open(file_html,'w', encoding="utf-8")
    try:
        message = header_html + title + navigation_bar + end_page
        message = message + footer_html
    except:
        pass
    f.write(message)
    f.close()
```

In quanto header, navigation bar e footer sono comuni a tutte mi basterà creare l'end_page in html e passarla al metodo per creare una nuova pagina di un nuovo servizio.

3.3. Interruzione da tastiera

Premendo la combinazione ctrl+c è possibile far terminare l'esecuzione dei thread e quindi dell'intero programma, potendo così uscire.
Ho quindi definito la seguente funzione, che verrà poi richiamata all'interno del main:

```
# funzione per uscire con comando Ctrl+c
def signal_handler(signal, frame):
    print( 'Exiting http server (Ctrl+C pressed)')
    try:
      if(server):
        server.server_close()
    finally:
      # fermo il thread del refresh senza busy waiting
      waiting_refresh.set()
      sys.exit(0)
```

3.4. Download pdf

Il download del pdf permette di scaricare la relazione ed è stata implementata direttamenta all'interno della navigation bar:

```
<a href="http://127.0.0.1:{port}/Relazione_Leoni.pdf"
download="Relazione_leoni.pdf" style="float: right">Download
relazione</a>
```

3.5. Autenticazione

L'autenticazione è stata gestita attraverso nome utente e password, direttamente da linea di comando, se una delle due credenziali il programma chiude il server, per evitare così errori al prossimo avvio, ed infine esce.

```
 #controllo sulle credenziali di accesso
    username = input("Inserire username: ")
    pw = input("Inserire password: ")
    #se usr o pw sono diverse, chiudo il server e esco
    if (username != 'lorenzo' or pw != 'leoni') :
        print("Errore durante l'autenticazione dell'utente, riprovare")
        server.server_close()
        sys.exit(0)
    print("Autenticazione avvenuta con successo.\n\n")
```

4. Librerie utilizzate

Le librerie utilizzate per la realizzazione del Web Server sono le seguenti:

```
import http.server
import sys,signal
import socketserver
import threading
```

- **http.server:** Questo modulo definisce le classi per implementare i server HTTP (server web).
- **sys:** Questo modulo fornisce l'accesso ad alcune variabili usate o mantenute dall'interprete, e a funzioni che interagiscono fortemente con l'interprete stesso.
- **signal:** Questo modulo fornisce funzioni, variabili e meccanismi atti ad usare gestori di segnali in Python.
- **socketserver:** Il modulo Socket Server è una infrastruttura per la creazione di server di rete. Definisce le classi per la gestione di richieste di rete sincrone
- **threading:** Questo modulo realizza un'interfaccia ad alto livello per la gestione dei thread sulla base del modulo a basso livello thread. È stato utilizzato per realizzare un thread che aggiornare periodicamente i contenuti delle pagine e inoltre per gestire l'attesa in modo da evitare così il busy waiting.