2020/21

Relazione progetto Programmazione di Reti

TRACCIA 2

LUCA TOMIDEI

MATRICOLA: 902747

INTRODUZIONE

L'obbiettivo era quello di realizzare un Web Server in Python per una azienda ospedaliera.

I requisiti del Web Server, espressi dalla traccia, 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
- Come requisito facoltativo si chiede di autenticare gli utenti nella fase iniziale della connessione.

DESCRIZIONE

Il sito si presenta con una Home Page minimale e molto intuitiva, nella quale troviamo una breve descrizione dell'azienda e le funzionalità del sito.

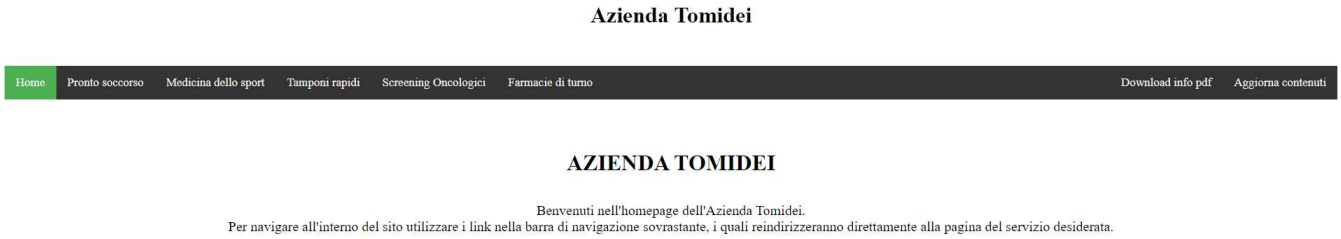


Figura 1. Home Page del sito.

Tutti i servizi, come mostrato nella figura 1, sono raggiungibili attraverso la barra di navigazione, la quale presenta anche un pulsante per scaricare una locandina esplicativa del sito.

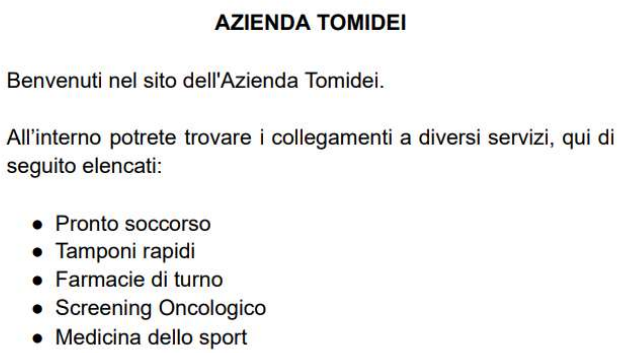


Figura 2. Screenshot della locandina scaricabile dal sito

DETTAGLI IMPLEMENTATIVI

Multithreading

```
38
39      # ThreadingTCPServer per gestire più richieste
40      server = socketserver.ThreadingTCPServer(('127.0.0.1',port), ServerHandler)
41
```

Per soddisfare il requisito di accesso da più utenti in contemporanea si è utilizzato il modulo socketserver, utilizzato come nell'immagine. 'port' rappresenta il valore di porta utilizzato, inserito opzionalmente come argomento.

Creazione pagine servizi

```
206     #metodo lanciato per la creazione delle pagine servizi
207     def create_page_servizio(title,file_html, end_page):
208         f = open(file_html,'w', encoding="utf-8")
209         try:
210             message = header_html + title + navigation_bar + end_page
211             message = message + footer_html
212         except:
213             pass
214         f.write(message)
215         f.close()
216
217     # creazione della pagina specifica del pronto soccorso
218     def create_page_pronto_soccorso():
219         create_page_servizio("<h1>Pronto soccorso</h1>"  , 'pronto_soccorso.html', end_page_pronto_soccorso )
220
```

Le pagine sono tra di loro simili, quindi, è stato deciso per un metodo comune per gestire la loro creazione. Questo metodo viene chiamato ogniqualvolta necessario passando i rispettivi argomenti di volta in volta.

Interruzione da tastiera

```
259     # definiamo una funzione per permetterci di uscire dal processo tramite Ctrl-C
260     def signal_handler(signal, frame):
261         print( 'Exiting http server (Ctrl+C pressed)')
262         try:
263           if(server):
264             server.server_close()
265         finally:
266           # fermo il thread del refresh senza busy waiting
267           waiting_refresh.set()
268           sys.exit(0)
269
```

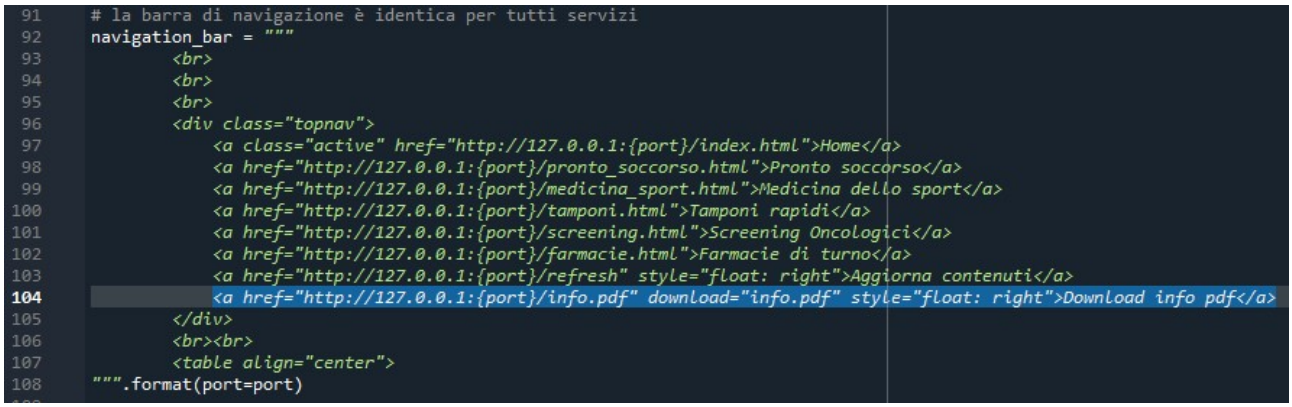
La funzione riportata qui sopra permette l'uscita dall'applicativo attraverso la combinazione di tasti CTRL+C (questo metodo è richiamato all'interno del main, dove viene selezionato appunto questa combinazione passata come argomento).

Autenticazione

```
270     # metodo che viene chiamato al "lancio" del server
271     def main():
272         usr = input("username: ") #richiesto l'username da tastiera
273         psw = input("password: ") #richiesta la password
274         if(usr != 'luca' or psw != 'tomidei'):
275             print("Errore.")
276             server.server_close() #Per evitare errori al prossimo avvio
277             sys.exit(0)
```

Per gestire l'autenticazione al web server si è semplicemente optato per un if condizionale e la richiesta da tastiera di username e password. In caso di errata digitazione il server viene chiuso.

Download Info.pdf



```
# la barra di navigazione è identica per tutti servizi
navigation_bar = """
        <br>
        <br>
        <br>
        <div class="topnav">
            <a class="active" href="http://127.0.0.1:{port}/index.html">Home</a>
            <a href="http://127.0.0.1:{port}/pronto_soccorso.html">Pronto soccorso</a>
            <a href="http://127.0.0.1:{port}/medicina_sport.html">Medicina dello sport</a>
            <a href="http://127.0.0.1:{port}/tamponi.html">Tamponi rapidi</a>
            <a href="http://127.0.0.1:{port}/screening.html">Screening Oncologici</a>
            <a href="http://127.0.0.1:{port}/farmacie.html">Farmacie di turno</a>
            <a href="http://127.0.0.1:{port}/refresh" style="float: right">Aggiorna contenuti</a>
            <a href="http://127.0.0.1:{port}/info.pdf" download="info.pdf" style="float: right">Download info pdf</a>
        </div>
        <br><br>
        <table align="center">
""".format(port=port)
```

Questa richiesta è stata gestita direttamente in HTML, all'interno della definizione della barra di navigazione (come riportato dalla riga evidenziata in figura) e permette di scaricare un file "info.pdf" posizionato necessariamente all'interno della directory.

LIBRERIE UTILIZZATE

Sys: Questo modulo fornisce l'accesso ad alcune variabili usate o mantenute dall'interprete, e a funzioni che interagiscono fortemente con l'interprete stesso. È sempre disponibile.

signal: Questo modulo fornisce meccanismi atti ad usare gestori di segnali in Python.

http.server: Questo modulo definisce le classi per implementare i server HTTP (server web).

socketserver: Il modulo socketserver semplifica il compito di scrivere server di rete.

threading: Questo modulo realizza un'interfaccia ad alto livello per i thread sulla base del modulo a basso livello thread.