Relazione Progetto

Traccia 2 - Python Web Server

Michele Nardini

Matricola: 914548

Elisa Simoni

Matricola: 922134

1 Introduzione

Il progetto consiste nel realizzare un programma con il linguaggio Python, in particolare un Web Server per una azienda ospedaliera, con una pagina di login.

2 Descrizione

Il progetto presenta nella schermata principale una pagina di login in cui verrà chiesto all'utente d'inserire Username e Password. Tutti gli Username e le Password sono contenuti in un file credential.txt. In caso di mancata presenza delle informazioni il web server rindirizzerà a una pagina di registrazione che scriverà le informazioni utente nel file credential.txt. Una volta effettuato il login l'utente verrà mandato alla pagina di home dove sono elencati i servizi ospedalieri assieme una piccola descrizione. In caso di mancata compilazione di uno dei campi username/password delle pagine Login e Registrazione verrà dato un Error Response e si dovrà tornare indietro. L'utente cliccando sugli elementi della lista verrà rimandato alle pagine web dei suddetti servizi dell'ospedale di Forlì. In fondo alla lista è presente un link per il download di un pdf (lista_servizi.pdf).

3 Dettagli Implementativi

1. Multithreading del Server
2. Creazione e reindirizzamento pagine servizi
3. Download PDF
4. Pagina di login e registrazione
5. Chiusura con CTRL-C

1. Come specificato nella traccia, il web server doveva consentire l'accesso a più utenti in contemporanea. Per ovviare a questo problema siamo ricorsi all'uso della libreria socketserver.

```
# ThreadingTCPServer per consentire l'accesso a più utenti in contemporanea
server = socketserver.ThreadingTCPServer(('127.0.0.1',port),requestHandler)
```

2. Abbiamo definito una funzione generale che permettesse ai servizi di essere creati con la loro relativa pagina html.

```
#creo una funzione per creare le pagine dei vari servizi
def service_page(name, url):
    resp = requests.get(url)
    f = open(name + ".html",'w', encoding="utf-8")
    f.write(resp.text)
    f.close()
```

3. Per scaricare il file PDF abbiamo creato un file pdf nella cartella del progetto e l'abbiamo richiamato con una linea di codice html.

   ** <h1> Download lista servizi</h1>
**

4. Per la pagina di login abbiamo sfruttato il metodo DoPost per effettuare una richiesta di login/registrazione credenziali in un file di testo denominato credential.txt. Tale file all'inizio presenta una riga di descrizione del file e ogni volta che viene effettuata una registrazione delle credenziali viene salvato su tale file username e password dell'utente. Appena viene effettuata una richiesta di POST vengono letti gli input della pagina e scritti sul file credential.txt, se ci troviamo nella pagina di registrazione. Per verificare la presenza delle credenziali nel file credential.txt si procede con una lettura del file con controllo delle righe a due a due (username \n + password \n).

```
def do_POST(self):
    try:
        # Salvo i vari dati inseriti
        form = cgi.FieldStorage(
        fp=self.rfile,
        headers=self.headers,
        environ={'REQUEST_METHOD':'POST'})
        global x

        # Con getvalue prendo i dati inseriti dall'utente
        if x == 0:
            username = form.getvalue('username')
            password = form.getvalue('password')
            f=open("credential.txt","r")
            lines = f.readlines()
            print(self)
            print(username+"\n"+password)
            i=1
            while i < len(lines):
                print("sono entrato")
                if lines[i].strip() == username and lines[i+1].strip() == password:
                    print("okkk")
                    self.path='home.html'
                    return http.server.SimpleHTTPRequestHandler.do_GET(self)
                i+=2
            print("sono uscito")
            print(self.path)
            self.path='registrer.html'
            x=1
            return http.server.SimpleHTTPRequestHandler.do_GET(self)
```

5. Per far sì che il web server venga interrotto abbiamo sfruttato la libreria signal.

```
def signal_handler(signal, frame):
    print( 'Exiting http server (Ctrl+C pressed)')
    try:
      if(server):
        server.server_close()
    finally:
      sys.exit(0)
```

4 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.

signal

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

http.server

Questo modulo definisce le classi per l'implementazione di server HTTP (server Web).

socketserver

Il modulo SocketServer è una infrastruttura per la creazione di server di rete. Definisce le classi per la gestione di richieste di rete sincrone

requests

pip install requests

Requests è una libreria che ci permette di effettuare una richiesta http/https in modo semplice ed immediato basata su urllib3.

cgi

Uno script CGI viene richiamato da un server HTTP, in genere per elaborare l'input dell'utente inviato tramite un HTML.