Nome: Alessio Barbanti  
Mat. 766318  
Traccia 2, Anno 2021

Istruzioni per l’avvio:  
Avviare Spyder ed eseguire il file Alessio\_Barbanti\_Traccia2\_Advanced.py.  
Al momento della stesura di questa guida non è necessario installare ulteriori librerie.

Premessa:  
Questo web server è studiato per funzionare in rete locale, una sua successiva implementazione potrebbe essere di usare ngrok, cercando l’ip del client nel campo “X-Forwarded-For” dell’header invece che nel campo client\_address.  
Successivamente si potrebbe implementare un hashing delle password e la connessione ad un database adatto (Es. MariaDB).

Il feedback per il successo o meno della registrazione e del login viene mandato sulla Console.

Analisi del codice lato server:

|  |  |
| --- | --- |
| **Riga** | **Descrizione** |
| **1-4** | Importazione delle librerie necessarie |
| **6-7** | Variabili da modificare in base alle impostazioni del proprio pc. |
| **10-38** | Codice HTML che tramite la funzione print\_page() (linea 118) verrà scritto nel file index.html dopo aver sostituito i placeholder con i rispettivi valori. |
| **45** | Crea un Handler di richieste personalizzato usando SimpleHTTPRequestHandler. |
| **49-53** | Vengono presi dal form i valori immessi e salvati in due variabili. |
| **56-64** | Dopo il post del form di registrazione viene aggiunto ?Registrati all’url, quindi viene controllato se lo username è già esistente, nel caso non lo sia viene aggiunto al database insieme alla sua password. |
| **67-78** | Dopo un’iniziale fase di login, se il login va a buon fine, viene aggiunto alla lista di ip abilitati l’indirizzo ip locale della macchina da cui ci si sta connettendo e tramite il codice 302 si viene reindirizzati alla pagina servizio.html contenente l’indice delle pagine dei servizi. |
| **82-90** | In questo metodo se si è già loggati si viene reindirizzati alla pagina con l’indice, invece nel caso non lo sia e si tenti di accedere a servizio.html, alla pagina di login. |
| **96-102** | Funzione che controlla se lo username e la password sono presenti all’interno del database, scorrendolo riga per riga e controllando se sono uguali a quelli immessi. |
| **106-113** | Simile alla funzione precedente, ma controlla solo se lo username è presente all’interno del database. |
| **115-118** | Funzione che scrive la pagina html con i dati inseriti dall’utente. |
| **121** | Inizializzo il server multithread. |
| **124-130** | Spegnimento del server e liberazione della risorsa socket. Spiegato nel dettaglio alla riga 134. |
| **133** | Il server viene settato come demonic, quando non avrà più thread aperti si chiuderà in automatico. |
| **134** | signal.signal(arg1, arg2) permette di gestire un signal, nel nostro caso arg1 è "signal.SIGINT" che corrisponde a alla pressione di ctrl+c, ovvero un'interruzione da tastiera, quando questo viene intercettato, lancia la funzione arg2, ovvero signal\_handler() (linea 139), che chiude il server liberando la risorsa socket (server\_close()) e procedendo con la chiusura del' interprete (sys.exit(0)). |
| **137-140** | Parte di codice puramente di DEBUG, ad ogni avvio del server cancella tutti gli account salvati nel file Account.txt |
| **144-145** | Loop infinito per il funzionamento del server |
| **146-147** | Cattura l'eccezione dell'interruzione da tastiera e usando *pass* ne previene la gestione |
| **150** | Lancio del main. |