Relazione Progetto

Traccia 2 – Web Server

Damiano Morandi

Matricola 0 000 921 149

Giugno 2021

1 – Introduzione e Requisiti

Il progetto punta a realizzare un semplice web server http in python per un'azienda ospedaliera.
In particolare, il server dovrà:

- Consentire l'accesso contemporaneo di più utenti
- La pagina iniziale dovrà elencare i servizi ospedalieri offerti e i file scaricabili dal sito
- Ogni servizio elencato dovrà avere una pagina dedicata accessibile dalla pagina iniziale
- Liberare correttamente le risorse di sistema (socket) quando viene terminato da riga di comando.

Questo progetto inoltre implementa:

- Autenticazione HTTP di base
- Configurazione tramite un singolo file

2 – Descrizione

Per semplificarne la progettazione e consentire maggiore flessibilità, il server è suddiviso in due moduli python:

- Main
  Nel file `main.py`, si occupa di inizializzare il server e di gestire le richieste GET dei client
- Resources
  Nel file `resources.py`, consente l'accesso alle risorse del sistema per la quale il server è configurato

L'applicativo è configurabile all'avvio, permettendo la facile aggiunta di nuove pagine di servizi o file scaricabili semplicemente editando il file `config.json` all'interno della cartella del progetto, seguendo la struttura dell'oggetto json ivi presente.
È possibile specificare un percorso alternativo per il file di configurazione aggiungendo l'opzione `-c` alla riga di comando del server seguita dal percorso del file scelto.

Il server richiede una versione di python 3.7 o superiore, ed è possibile avviarlo con `python3 main.py`, o con `./main.py` nelle piattaforme UNIX o UNIX-like

```
$ python3 main.py
Starting server at port 8081
```

Una volta avviato l'applicativo, il server rimarrà in attesa su tutte le interfacce di rete della macchina, e stamperà a video la porta di ricezione (di default 8081, ma è possibile cambiarla specificando l'opzione `-p` e il numero della porta alla riga di comando).

```
$ python3 main.py -p 3636
Starting server at port 3636
```

Per connettersi, aprire un browser web a scelta e inserire l'IP della macchina sui cui il server è in esecuzione, seguito dalla porta.
Il server richiederà al browser l'autenticazione di base HTTP, e pertanto questi aprirà un form richiedendo di inserire le credenziali.
(Per consentirne la prova, sono preconfigurate "Admin" "Admin" e "user" "password")

Una volta autenticati con successo, si verrà presentati con l'indice e pagina principale del sito, dalla quale sarà possibile prendere visione dei servizi disponibili o scaricare i file elencati sotto di essi.

Le richieste http GET dell'indice e delle altre pagine e file verranno stampate sulla console con il relativo indirizzo IP e risorsa richiesta.

```
$ python3 main.py
Starting server at port 8081
127.0.0.1 - - [19/Jun/2021 10:48:28] "GET / HTTP/1.1" 401 -
127.0.0.1 - - [19/Jun/2021 10:48:35] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2021 10:48:35] "GET /favicon.ico HTTP/1.1" 404 -
127.0.0.1 - - [19/Jun/2021 10:48:39] "GET /4fd7f200238f43eb5403919238e19fb74af02
c1c HTTP/1.1" 200 -
127.0.0.1 - - [19/Jun/2021 10:48:44] "GET /0a37ace3e960a26e37b35b0732c02d473a55f
c68 HTTP/1.1" 200 -
```

Per terminare il server è sufficiente interromperlo con un interrupt da tastiera (`Ctrl+C` sulla maggior parte dei sistemi).

È possibile ottenere ulteriori informazioni sulle opzioni supportate dalla riga di comando aggiungendo l'opzione `-h` all'avvio.

3 – Implementazione

Il web server utilizza solamente le librerie incluse con l'implementazione di riferimento dell'interprete python (CPython). In particolare:

- `argparse`
- `base64`
- `hashlib`
- `http.server`
- `io`
- `os`
- `pathlib`
- `sys`
- `json`

L'applicativo può supportare più connessioni contemporanee tramite l'utilizzo di thread: il thread principale rimane in attesa di una connessione in entrata. Una volta stabilita il programma crea un thread apposito per gestire le richieste della connessione creata, il quale verrà terminato alla chiusura della connessione.

Per evitare che la connessione stalli quando viene richiesto il download di un file di dimensioni considerevoli, il modulo `resources` fornisce accesso alle pagine web e ai file via un'interfaccia comune basata sugli stream, in modo da caricare e inviare progressivamente anche i contenuti più pesanti senza dover attendere che il thread in carico della connessione li carichi completamente in memoria prima di iniziare ad inviarli.

Tramite l'utilizzo di un `try-catch-finally`, il server chiude correttamente il socket utilizzato anche in caso di errore. Lo stesso costrutto viene utilizzato per gestire la terminazione con interrupt da tastiera.

Per prevenire l'accesso da parte di utenti malevoli a file non desiderati, il web server nasconde il percorso delle risorse (pagine html e file) utilizzando un hash derivato dal nome per indicizzarle.

Per garantire una maggiore riservatezza e sicurezza, le password degli utenti sono memorizzate sotto forma di sha2-256.