WEB SERVER

- ## INTRODUZIONE

L'obiettivo del progetto è quello di progettare un semplice server HTTP in Python (usando socket) e servire un sito web statico con HTML/CSS. I requisiti minimi del progetto sono i seguenti: il server deve rispondere su localhost:8080, deve servire almeno 3 pagine HTML statiche, deve gestire richieste GET e risposta con codice 200, deve gestire l'implementazione della risposta 404 per file non trovati.

- ## ARCHITETTURA DEL PROGETTO

Il progetto prevede una directory principale in cui si trovano il file server.py e una sotto-directory www. La sotto-directory www contiene i file html, css e una ulteriore sotto-directory img che contiene i file delle immagini utilizzate nel sito. Mentre il file server.py contiene il server scritto in python.

Reti/ ← cartella principale del progetto

├── server.py ← codice Python del web server

├── www/ ← cartella root dei file statici serviti dal server

│ ├── index.html ← homepage

│ ├── prodotti.html ← pagina dei prodotti

│ ├── contatti.html ← pagina contatti

│ ├── style.css ← file CSS principale

│ └── img/ ← cartella immagini

│ ├── home.jpg

│ ├── ciambelle.jpg

│ └── torta.jpg

- ## FUNZIONAMENTO DEL WEB SERVER

Il Web Server è stato realizzato in python utilizzando il modulo socket della libreria standard. Il server ascolta sulla porta 8080 dell'indirizzo locale (localhost) e gestisce le richieste HTTP inviate dai client. La logica principale è racchiusa in un ciclo while True, che permette di accettare connessioni in modo continuativo .Quando una richiesta viene ricevuta, il server legge il percorso del file richiesto e lo ricerca nella cartella www/. I file vengono serviti in modalità testo o binaria a seconda del loro tipo: file HTML e CSS vengono letti come testo in UTF-8, mentre immagini (JPEG, PNG, ICO) vengono gestite in modalità binaria. Il funzionamento del server è di tipo non persistente: apre il data socket quando viene fatta una richiesta e appena risolta, il server, chiude la connessione con il client. Quindi si crea una data socket per ogni richiesta di connessione da parte di un qualsiasi client, anche se si tratta del medesimo che effettua una richiesta diversa. Il server può rispondere alla richiesta del client in due modi: con codice 200 e l'invio della pagina richiesta; con codice 404 per indicare che la risorsa non è stata trovata

- ## GESTIONE AVANZATA / ESTENSIONI

-Gestione dei mime

tramite la funzione get_mime_type(), il server identifica automaticamente il tipo di file richiesto (HTML, CSS, JPEG, PNG, ICO, ecc.) e invia l'header HTTP corretto Content-Type. Questo permette al browser di interpretare correttamente sia i file testuali che le immagini.

-Logging delle richieste

ogni richiesta ricevuta dal server viene stampata a console, insieme all'esito (richiesta servita con successo o errore 404).

-Le animazioni

sono presenti leggere animazioni di hover su card e immagini, migliorando l'esperienza utente e rendendo il sito più moderno e gradevole.

- SITO WEB STATICO

Si è cercato di creare un sito web "vetrina" simulando le richieste di una pasticceria. È presente un file index HTML che contiene la home, un file contatti, all'interno del quale sono stati inseriti delle aree di testo per inserire i propri dati e inviare messaggi alla pasticceria(non funzionante) e una pagina prodotti.html che contiene le card dei prodotti con immagini e una breve descrizione.

- TEST E VERIFICA

Il server è stato testato attraverso chiamata a file HMTL, CSS e jpg. Per verificare il corretto invio del codice di errore sono state cercate pagine inesistenti da browser. Il MIME type è stato testato con chiamate a pagine html. Infine lo stile è stato testato posizionando il puntatore sopra le card nella sezione "I nostri prodotti"