LAPORAN TUGAS BESAR

disusun untuk memenuhi salah satu tugas mata kuliah
CII2J4 Jaringan Komputer

Oleh :

Dafa Rafiansyah	(1301213290)
Gregoreus Marcelino	(1301213483)
Rifqah Amaliyah	(1301213241)

PROGRAM STUDI S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG

2022/2023

STATIC WEB SERVER BERBASIS TCP SOCKET PROGRAMMING

1. Topik yang dipilih dan sistem yang dibangun

Topik pada laporan kali ini adalah static webserver sederhana berbasis TCP socket programming. Adapun rincian sistem yang akan dibangun adalah sebagai berikut :

a. Implementasi pembuatan TCP socket dan mengaitkannya ke alamat dan port tertentu.
b. Program web server dapat menerima dan memparsing HTTP request yang dikirimkan oleh browser.
c. Web server dapat mencari dan mengambil file (dari file system) yang diminta oleh client.
d. Web server dapat membuat HTTP response message yang terdiri dari header dan konten file yang diminta.
e. Web server dapat mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client. Dapat berupa teks maupun gambar.
f. Jika file yang diminta oleh client tidak tersedia, web server dapat mengirimkan pesan “404 Not Found” dan dapat ditampilkan dengan benar di sisi client

2. Cara Kerja Program :

Webserver sederhana berbasis TCP socket programming akan bekerja untuk merespon permintaan HTTP dari klien, mencari dan mengirimkan file yang diminta, serta memberikan respons yang tepat jika file tidak ditemukan.

Berikut urutan kerja dari program yang dirancang :

1. Impor modul socket yang diperlukan.
2. Buat socket server menggunakan AF_INET dan SOCK_STREAM.
3. Tentukan port server yang akan digunakan.
4. Bind socket ke alamat dan port yang ditentukan.
5. Tunggu koneksi masuk dengan menggunakan metode listen().
6. Masuk ke dalam loop utama (while True) untuk menerima permintaan klien.
7. Saat koneksi diterima, terima pesan HTTP dari klien menggunakan metode recv().
8. Parsing pesan HTTP untuk mendapatkan nama file yang diminta.
9. Buka file yang diminta.
10. Kirim header HTTP ke klien yang menyatakan bahwa respons berhasil (HTTP/1.1 200 OK).
11. Kirim tipe konten (Content-Type) ke klien.
12. Kirim konten file yang diminta ke klien dengan menggunakan loop.
13. Tutup koneksi setelah konten file selesai dikirim.
14. Jika file tidak ditemukan (IOError), kirim respons kepada klien yang menyatakan "404 Not Found".
15. Tutup koneksi dengan klien.
16. Tutup socket server.
17. Selesaikan program dengan sys.exit() setelah data terkirim

3. Hasil Program

Dapat diakses pada tautan berikut :
https://drive.google.com/drive/folders/1CZjy5-YuQ3gvizEKVpJEBt9rVWzbVSo-?usp=sharing

4. Analisis Program

a. TCP Socket

```
serverSocket.bind(('',serverPort))
```

- Code
  serverSocket. bind((' ', serverPort)) berfungsi sebagai soket pertama atau pintu pertama yang berguna untuk menunggu klien me request

b. Parsing HTTP Request

```
try:
    message = connectionSocket.recv(1024).decode()
    filename = message.split()[1]
```

- Code **connectionSocket.recv(1024)** berguna untuk menerima pesan HTTP dari koneksi socket, yang kemudian pesan tersebut di **decode()** yang mana mengonversi pesan menjadi string.
- Code **split()[1]** berguna untuk memisahkan string, lalu indeks kedua (**[1]**) dari string tersebut disimpan ke variabel filename.

c. Mengambil file (dari file system) yang diminta oleh client

```
try:
    message = connectionSocket.recv(1024).decode()
    filename = message.split()[1]
    f = open(filename[1:], 'rb')
    outputdata = f.read()
```

- **open()** adalah sebuah fungsi bawaan Python yang digunakan untuk membuka file dan mengembalikan objek file. Pada code di atas, fungsi **open()** menerima argumen berupa string **filename[1:]** yang merupakan nama file yang akan dibuka. Notasi **[1:]** digunakan untuk mengambil karakter kedua hingga akhir string, karena karakter pertama (indeks 0) dianggap sebagai tanda **/** atau \ pada path file, tergantung pada sistem operasi yang digunakan.

- **'rb'** (read binary), berarti kita membaca file tersebut secara byte-by-byte, bukan sebagai teks biasa. Mode biner digunakan ketika kita ingin membaca atau menulis data yang tidak berbentuk teks, seperti gambar, audio, video, atau file biner lainnya.

- Setelah file berhasil dibuka, fungsi **read()** kemudian digunakan untuk membaca seluruh isi file dan menyimpannya ke dalam variabel outputdata. Dengan demikian, pada akhirnya variabel **outputdata** akan berisi string yang merepresentasikan isi file yang telah dibaca.

d. Menentukan tipe data yang dibaca

```
# Determine the content type based on the file extension
if filename.endswith('.html'):
    content_type = "text/html"
elif filename.endswith('.jpg') or filename.endswith('.jpeg'):
    content_type = "image/jpeg"
elif filename.endswith('.png'):
    content_type = "image/png"
else:
    content_type = "text/html"
```

Pengkondisian ini memungkinkan program untuk mengidentifikasi tipe data yang dbaca. Dapat berupa text, file berkekstensi jpeg, maupun png

e. HTTP response message yang terdiri dari header dan konten file yang diminta

```
#Send one HTTP header line into socket
connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())
connectionSocket.send("Content-Type: text/html\r\n\r\n".encode())


#Send the content of the requested file to the client
for i in range(0, len(outputdata)):
    connectionSocket.send(outputdata[i].encode())
connectionSocket.send("\r\n".encode())
connectionSocket.close()
```

- Code **connectionSocket.send("HTTP/1.1 200 OK\r\n".encode())** berfungsi mengirimkan HTTP response header line ke client. Header line ini menunjukkan bahwa request dari client telah diterima dan diproses oleh server dengan status **"200 OK"**.

- Baris **connectionSocket.send("Content-Type: text/html\r\n\r\n".encode())** akan memberitahu client bahwa tipe konten yang dikirimkan adalah **HTML (Content-Type: text/html)**.

- Baris **for i in range(0, len(outputdata)): connectionSocket.send(outputdata[i].encode())** berfungsi mengirimkan isi file yang telah dibaca sebelumnya ke client dalam bentuk byte. Dalam loop **for**, setiap karakter dalam isi file **(outputdata)** diambil satu per satu, diubah menjadi bentuk byte menggunakan fungsi **encode()**, dan dikirimkan ke client menggunakan method **send()** pada **connectionSocket**.

- **connectionSocket.send("\r\n".encode())** akan mengirimkan karakter newline terakhir ke client sebagai penanda akhir dari HTTP response.
- **connectionSocket.close()** kemudian menutup koneksi socket dengan client setelah seluruh isi file telah dikirimkan.

f. Web server mengirimkan response message yang sudah dibuat ke browser client

```
connectSocket.send(bytes('HTTP/1.1 200 OK\r\n\r\n', 'UTF-8'))
for i in range(0, len(outputdata)):
    connectSocket.send(outputdata[i].encode())
connectSocket.send("\r\n".encode())
connectSocket.close()
```

- **ConnectSocket.send** berarti mengirimkan pesan dalam bentuk byte. Pesan yang dikirimkan adalah pesan 200 OK yang berarti file sudah sesuai. Apabila file tidak sesuai atau tidak ditemukan maka pesan yang dikirim adalah 404 File not Found
- **connectSocket.close()**

yang berfungsi sebagai menutup socket koneksi. Oleh karena itu sekarang klien lain dapat menggunakan socket /server

g. Menampilkan pesan “404 Not Found”

```
except IOError:
#Send response message forfile not found
    connectionSocket.send("HTTP:/1.1 404 Not Found /r/n".encode())
    connectionSocket.send("Content-Type: text/html\r\n\r\n".encode())
    connectionSocket.send("<html><bbody><h1>404 Not Found</h1></body></html>\r\n".encode())
```

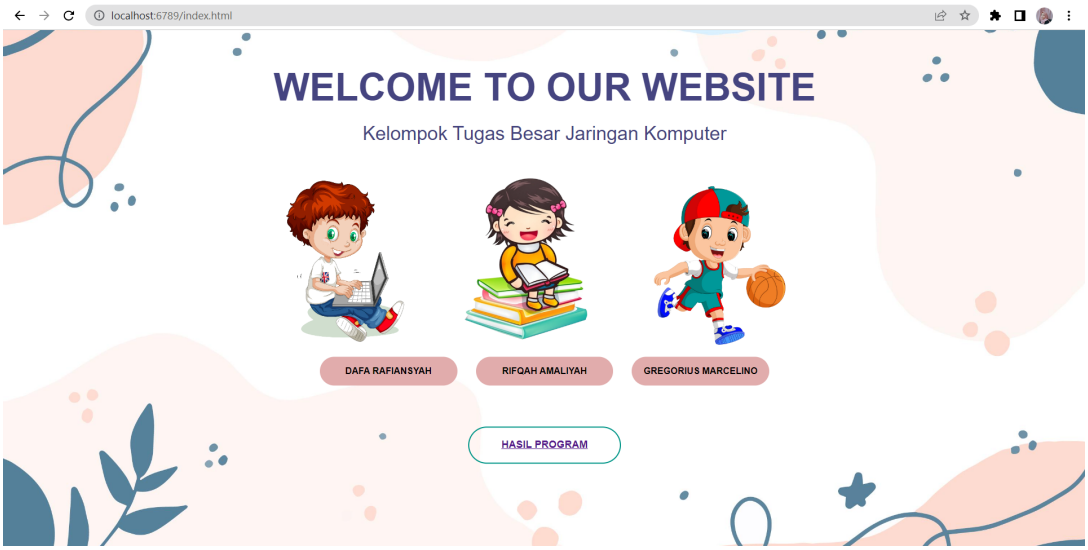
- Kode **connectionSocket.send("HTTP:/1.1 404 Not Found /r/n".encode())** ini berarti mengirimkan respon status line ke klien yang menunjukkan bahwa file yang diminta tidak ditemukan.
- **connectionSocket.send("Content-Type: text/html\r\n\r\n".encode())**, kode ini berarti mengirimkan respon ke klien yang menunjukkan bahwa tipe konten merupakan text/html.
- Kode **connectionSocket.send("<html><body><h1>404 Not Found</h1></body></html>\r\n".encode())** ini berarti mengirimkan pesan body HTTP response ke klien yang berisi bahwa pesan yang diminta tidak ditemukan (404 Not Found).

5. Uji Coba Web Server

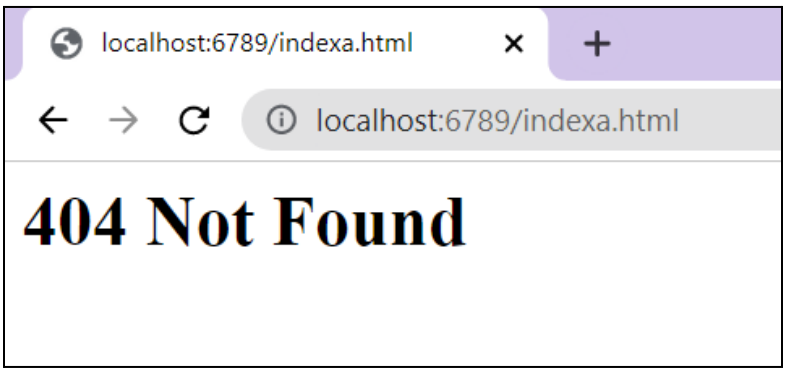
- Untuk menguji coba Web Server, kita perlu membuat file HTML. Disini kami mengujinya dengan menampilkan isi dari file html terebut. Berikut file html yang akan digunakan
  https://drive.google.com/drive/folders/1CZjy5-YuQ3gvizEKVpJEBt9rVWzbVSo-

- Kemudian buka terminal lalu jalankan web server yang telah dibuat

- Buka browser, kemudian tuliskan localhost:6789/index.html
  - 6789 merupakan serverPort yang sudah dibuat di web server
  - Index.html meeruakan file html yang akan diminta oleh client ke sisi server



- Jika file html yang diminta tidak ditemukan, maka server akan memberikan respon “404 Not Found” kepada client



6. Kesimpulan

Dengan menggunakan TCP Socket Programming, dapat dibangun program web server sederhana yang dapat menerima dan memproses HTTP request dari browser. Program tersebut dapat mengambil file yang diminta oleh client dari file system dan membuat HTTP response message yang terdiri dari header dan konten file yang diminta.

Web server juga dapat mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client. Selain itu, web server juga dapat mengirimkan pesan "404 Not Found" jika file yang diminta oleh client tidak tersedia dan dapat ditampilkan dengan benar di sisi client.

Implementasi praktikum ini memberikan pemahaman tentang cara kerja protokol HTTP dan dasar-dasar pembangunan program web server dengan menggunakan TCP Socket Programming.

7. Referensi

- Python Webserver Static : https://www.youtube.com/watch?v=uZJGklHPhVs
- Modul 8 pada Modul Praktikum Jarkom IF
- Chatgpt, keyword : how to load png/jpeg content type

8. Kontribusi Anggota

NO	NAMA	TUGAS
1	Dafa Rafiansyah	- Memparsing HTTP request - Menampilkan pesan “404 Not Found - Membuat analisis code yang dikerjakan - Membuat halaman html yang akan diujicoba sebagai request client ke server
2	Gregoreus Marcelino	- Membuat TCP socket dan mengaitkannya ke alamat dan port tertentu. - Mengirimkan response message yang sudah dibuat ke browser (client) dan dapat ditampilkan dengan benar di sisi client. - Membuat analisis code yang dikerjakan - Melengkapi comment pada setiap baris pada file program tcpserver
3	Rifqah Amaliyah	- Mencari dan mengambil file (dari file system) yang diminta oleh client. - Membuat HTTP response message yang terdiri dari header dan konten file yang diminta. - Membuat analisis code yang dikerjakan - Melakukan revisi laporan dan PPT Presentasi