

LATIHAN PERTEMUAN 7
PRAKTIKUM PEMOGRAMAN BERBASIS WEB
Untuk Memenuhi Praktikum Pemograman Berbasis Web



Oleh:

Nama : Siti Ghumaisa
NPM : 4522210131
Kelas : A
Semester : 4 (Genap)

Dosen :

ADI WAHYU PRIBADI ,S.SI.,M.KOM
S1-Teknik Informatika
Fakultas Teknik Universitas Pancasila
2023/2024

Link git-hub <https://github.com/Ghumaisa/PBW>

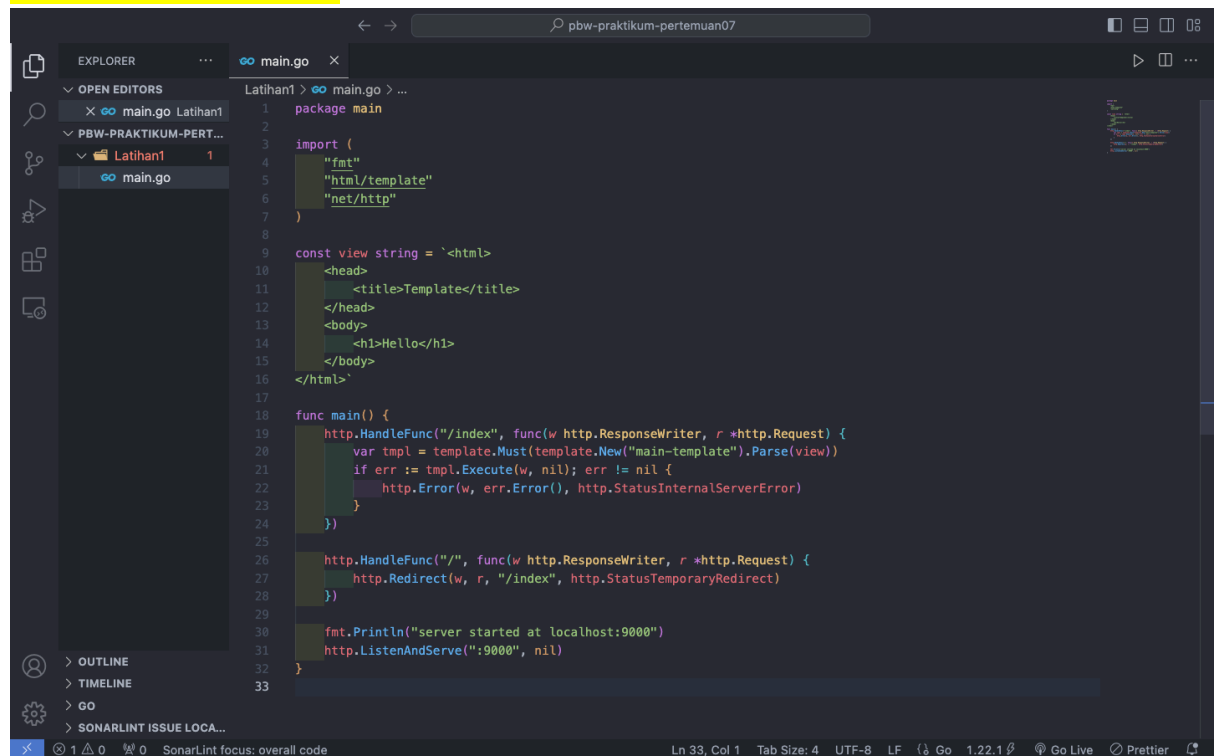
1. Jelaskan Html, parsing method

- **HTML** (HyperText Markup Language) adalah bahasa markah yang digunakan untuk membuat dan menyusun halaman web. HTML menggunakan serangkaian tag atau tanda yang ditempatkan di dalam tanda kurung sudut (<>) untuk menandai elemen-elemen dalam halaman web, seperti teks, gambar, dan tautan.
- **Parsing** adalah proses mengurai atau menganalisis suatu dokumen atau teks menjadi potongan-potongan yang lebih kecil atau lebih mudah dimengerti oleh komputer. Dalam konteks HTML, parsing adalah proses membaca dan menginterpretasikan kode HTML untuk membuat representasi struktur hierarkis dari halaman web tersebut.

2. Praktek program

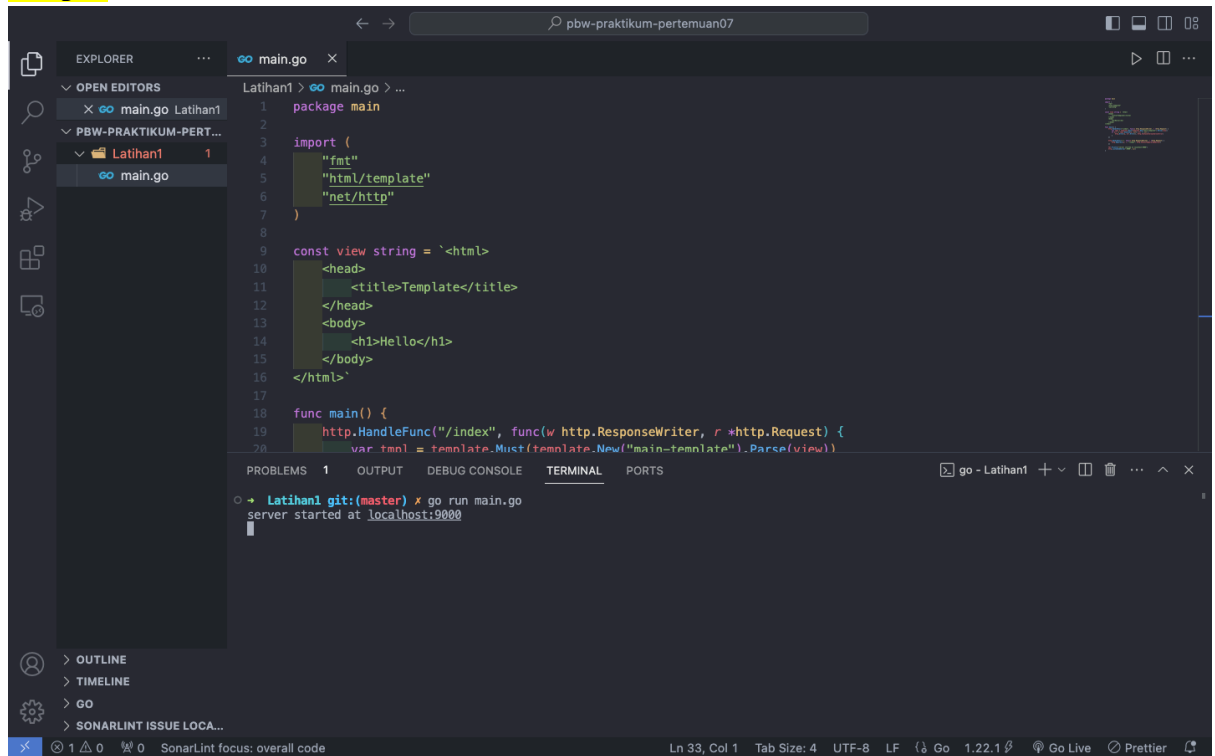
- Siapkan folder project baru beserta file main.go, isi dengan kode berikut

Screenshot Source Code



```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 const view string = `<html>
10     <head>
11         <title>Template</title>
12     </head>
13     <body>
14         <h1>Hello</h1>
15     </body>
16 </html>`
17
18 func main() {
19     http.HandleFunc("/index", func(w http.ResponseWriter, r *http.Request) {
20         var tmpl = template.Must(template.New("main-template").Parse(view))
21         if err := tmpl.Execute(w, nil); err != nil {
22             http.Error(w, err.Error(), http.StatusInternalServerError)
23         }
24     })
25
26     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
27         http.Redirect(w, r, "/index", http.StatusTemporaryRedirect)
28     })
29
30     fmt.Println("server started at localhost:9000")
31     http.ListenAndServe(":9000", nil)
32 }
33
```

Output



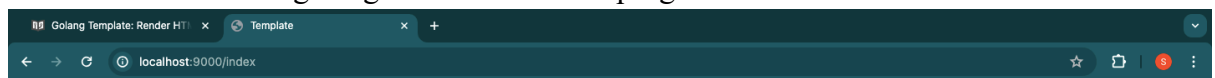
The screenshot shows the Visual Studio Code editor with a Go file named `main.go` open. The code defines a simple HTTP server using the `net/http` and `html/template` packages. The `main` function sets up a handler for the `/index` path that renders an HTML template. The terminal at the bottom shows the command `go run main.go` being executed, resulting in the message "server started at localhost:9000".

```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 const view string = `<html>
10     <head>
11         <title>Template</title>
12     </head>
13     <body>
14         <h1>Hello</h1>
15     </body>
16 </html>`
17
18 func main() {
19     http.HandleFunc("/index", func(w http.ResponseWriter, r *http.Request) {
20         var tmpl = template.Must(template.New("main-template").Parse(view))
21     })
22 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Latihan1 git:(master) X go run main.go
server started at localhost:9000

- Lakukan testing dengan melakukan run program



Hello

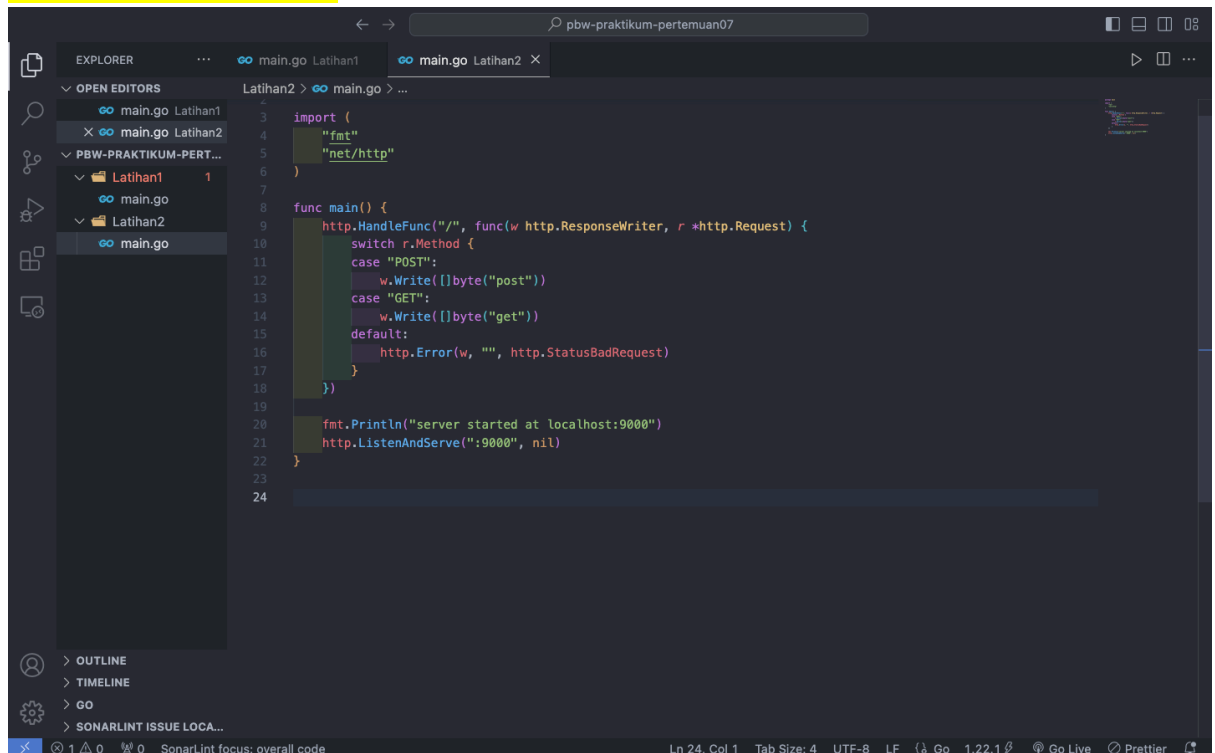
3. Apa itu postman

Postman adalah sebuah aplikasi yang digunakan oleh pengembang perangkat lunak untuk menguji, memodifikasi, dan mengelola API (Application Programming Interface). Dengan Postman, pengembang dapat membuat permintaan HTTP ke API, mengirimkan data seperti parameter dan header, dan melihat respons yang diterima dari server API. Salah satu fitur postman adalah mampu untuk menyimpan dan mengatur permintaan API dalam koleksi, yang memungkinkan pengguna untuk mengelola dan menyimpan permintaan API yang sering digunakan.

4. Praktek program

- Siapkan folder project baru beserta file main.go, isi dengan kode berikut

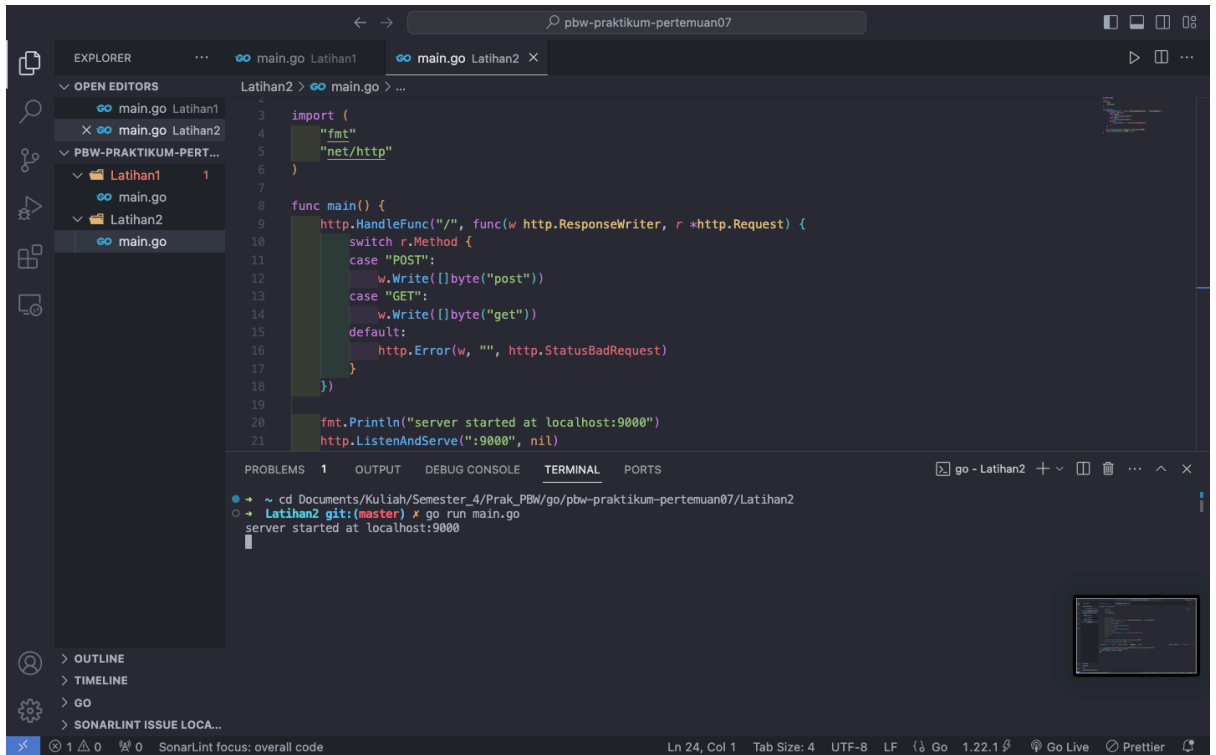
Screenshot Source Code



```
import (  
    "fmt"  
    "net/http"  
)  
  
func main() {  
    http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {  
        switch r.Method {  
            case "POST":  
                w.Write([]byte("post"))  
            case "GET":  
                w.Write([]byte("get"))  
            default:  
                http.Error(w, "", http.StatusBadRequest)  
        }  
    })  
  
    fmt.Println("server started at localhost:9000")  
    http.ListenAndServe(":9000", nil)  
}
```

Output

Lakukan go run dan buka postman

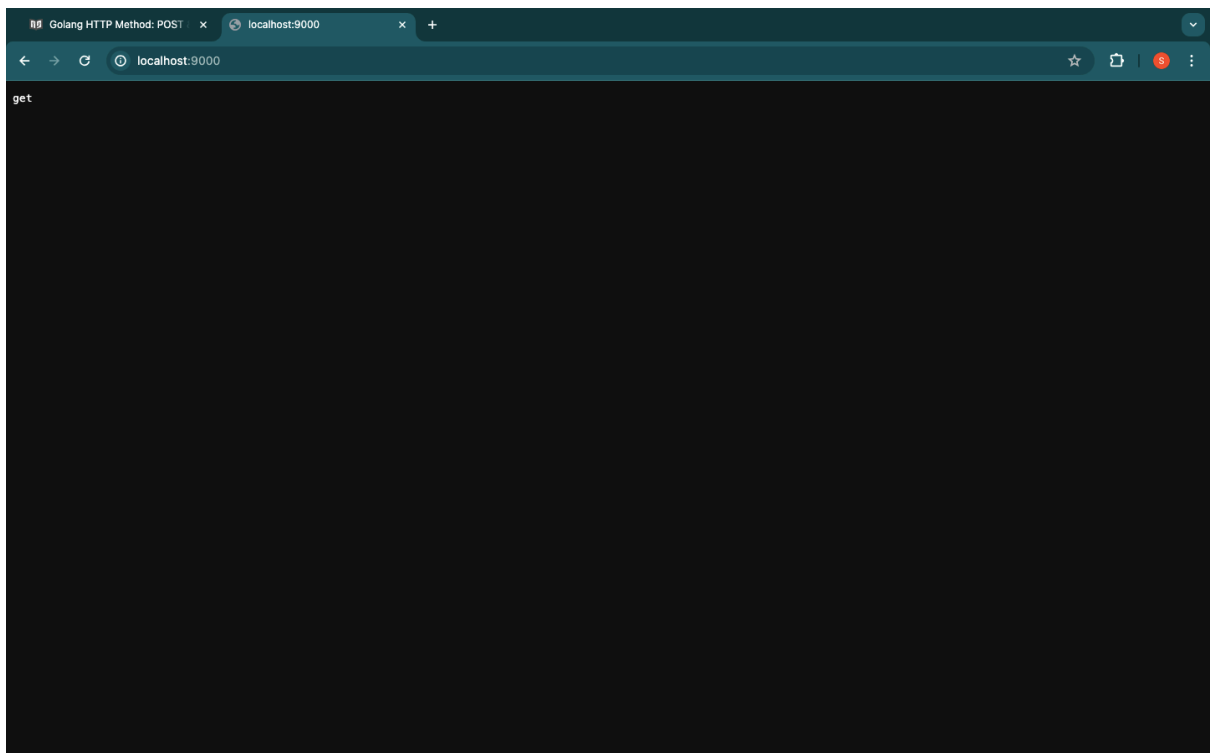


The screenshot shows the Visual Studio Code editor with a Go file named `main.go` open. The code defines a simple HTTP server that listens on `localhost:9000`. It handles `POST` and `GET` requests by writing `post` and `get` to the response body, respectively. The `main` function calls `http.ListenAndServe` to start the server.

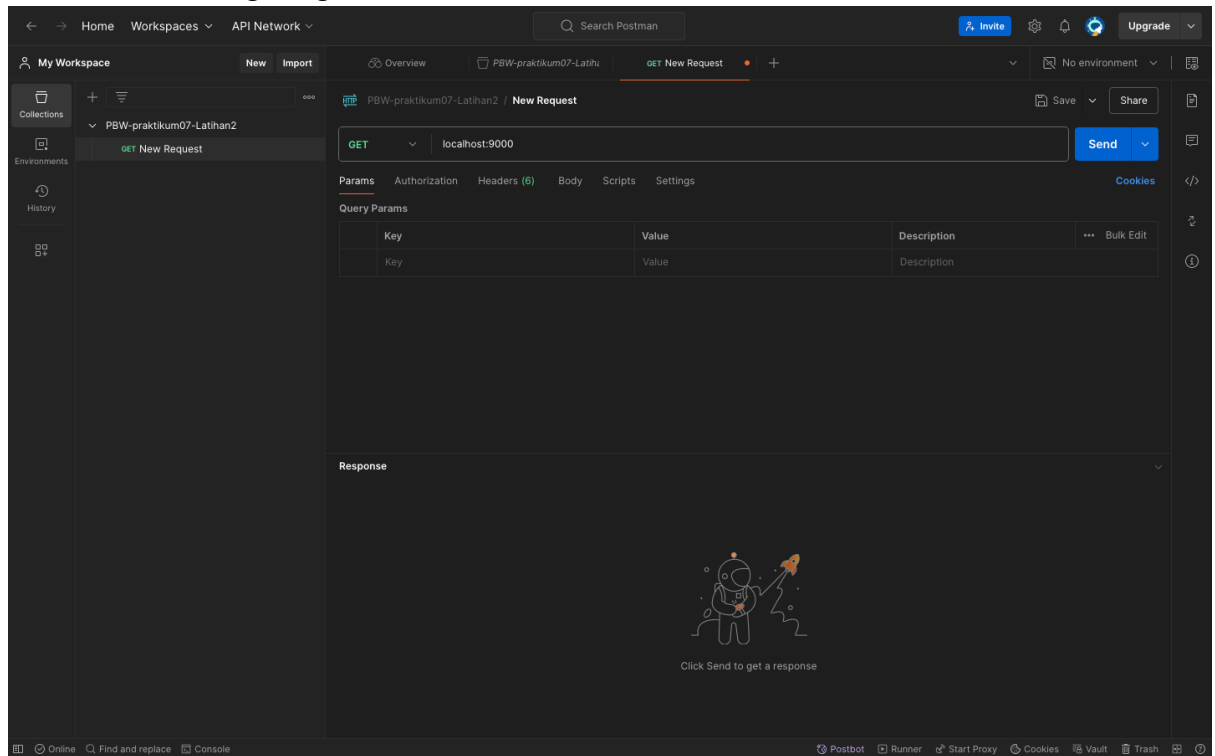
```
1 import (  
2     "fmt"  
3     "net/http"  
4 )  
5  
6  
7  
8 func main() {  
9     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {  
10         switch r.Method {  
11             case "POST":  
12                 w.Write([]byte("post"))  
13             case "GET":  
14                 w.Write([]byte("get"))  
15             default:  
16                 http.Error(w, "", http.StatusBadRequest)  
17             }  
18         }  
19     })  
20     fmt.Println("server started at localhost:9000")  
21     http.ListenAndServe(":9000", nil)  
22 }
```

The terminal output shows the command `go run main.go` being executed, and the server starting on `localhost:9000`.

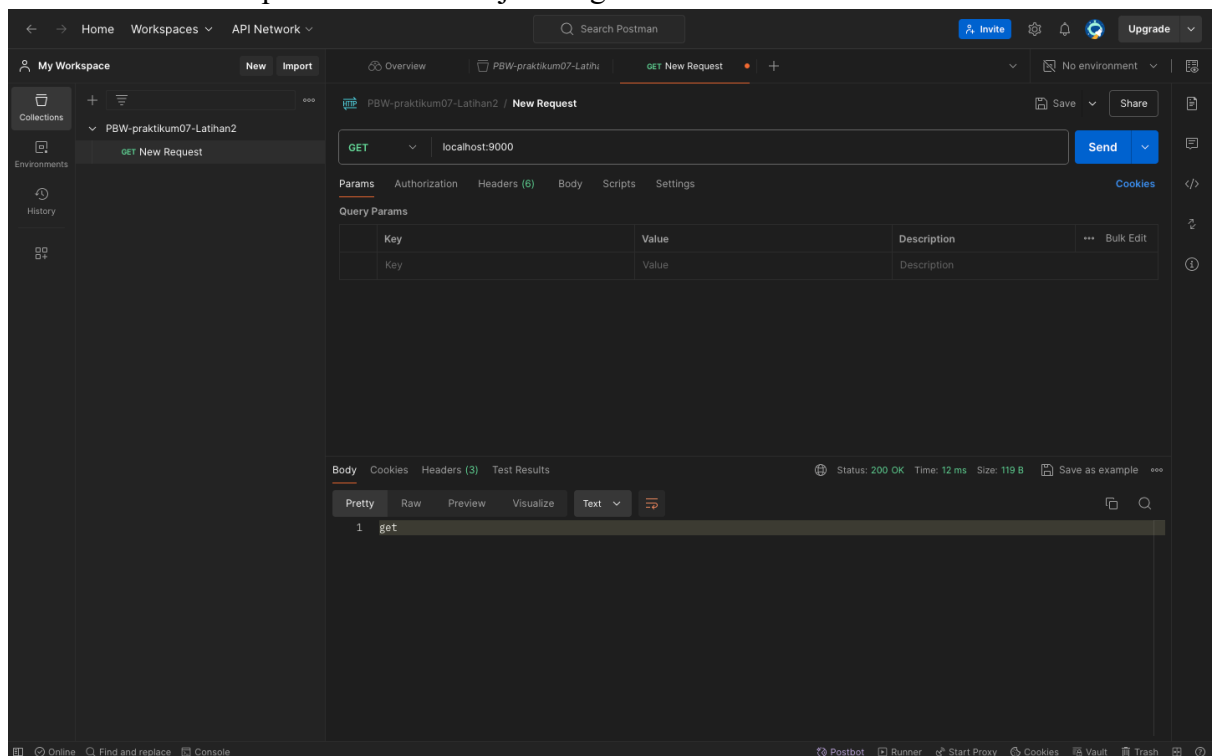
```
~ cd Documents/Kuliah/Semester_4/Prak_PBW/go/pbw-praktikum-pertemuan07/Latihan2  
~ Latihan2 git:(master) x go run main.go  
server started at localhost:9000
```



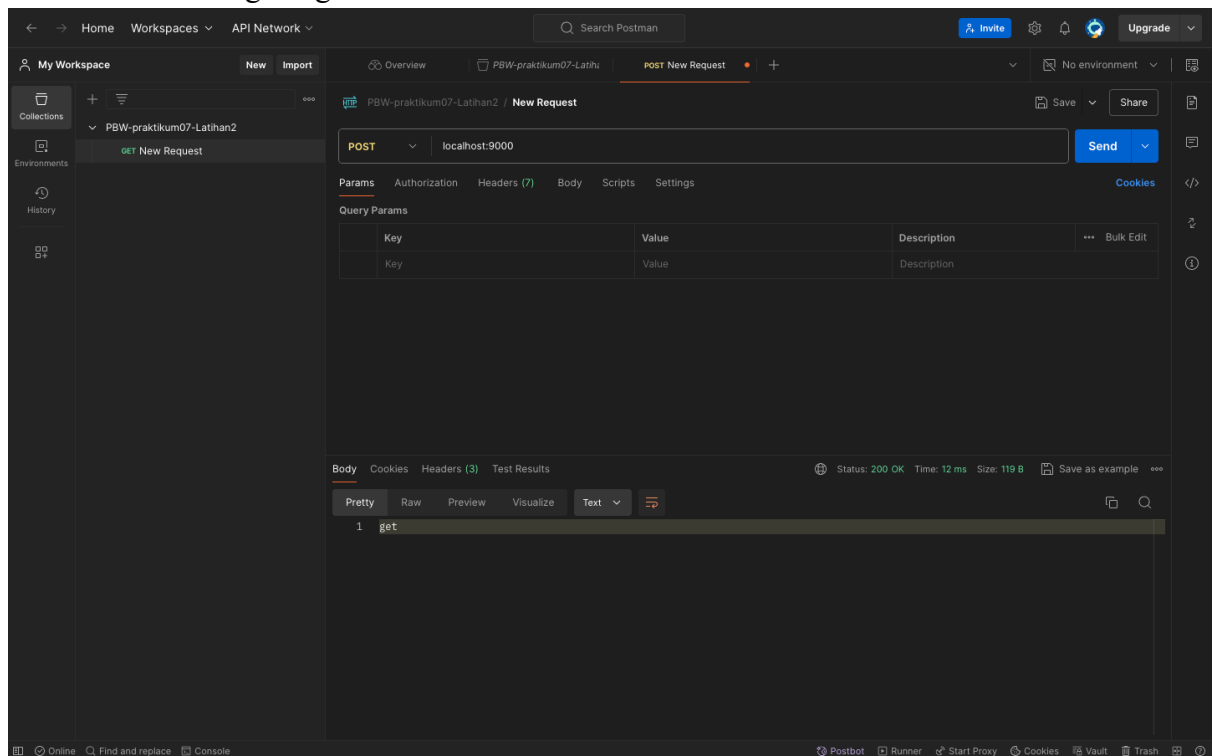
- Lakukan testing dengan Method GET



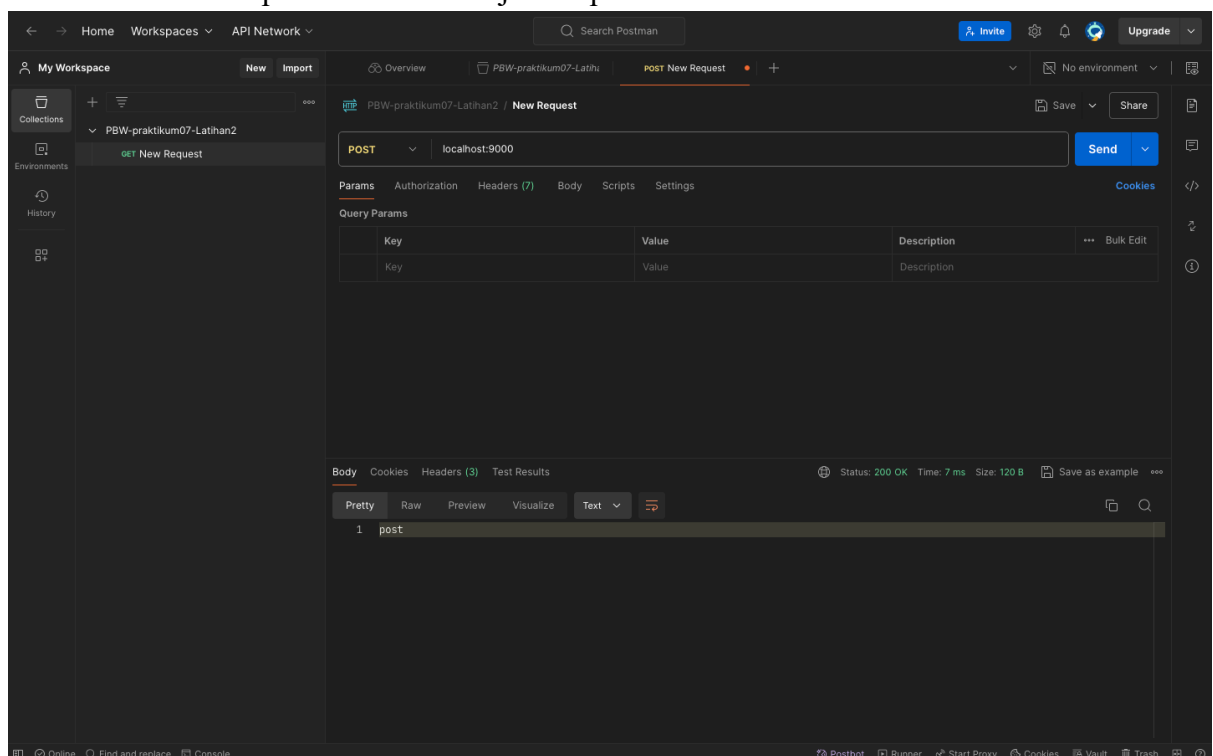
- Jika berhasil response akan menunjukan 'get'



- Lakukan testing dengan Method POST.



- Jika berhasil response akan menunjukan 'post'



5. Setelah melakukan testing dengan method GET dan POST. Namun ketika menggunakan selain dua method diatas akan menghasilkan message 400 Bad Request. Tolong jelaskan!

- Pada postman, jika pesan "400 Bad Request" muncul ketika metode selain GET dan POST digunakan karena server atau aplikasi yang diuji hanya mendukung kedua metode tersebut. Ini menandakan bahwa ada ketidakcocokan antara permintaan yang dikirim dengan yang dapat diproses oleh server. Saran yang bisa diberikan adalah memastikan bahwa metode yang digunakan sesuai dengan yang didukung oleh server, dan jika perlu menggunakan metode lain, pastikan server telah dikonfigurasi dengan benar atau cek kembali dokumentasi server untuk informasi lebih lanjut.

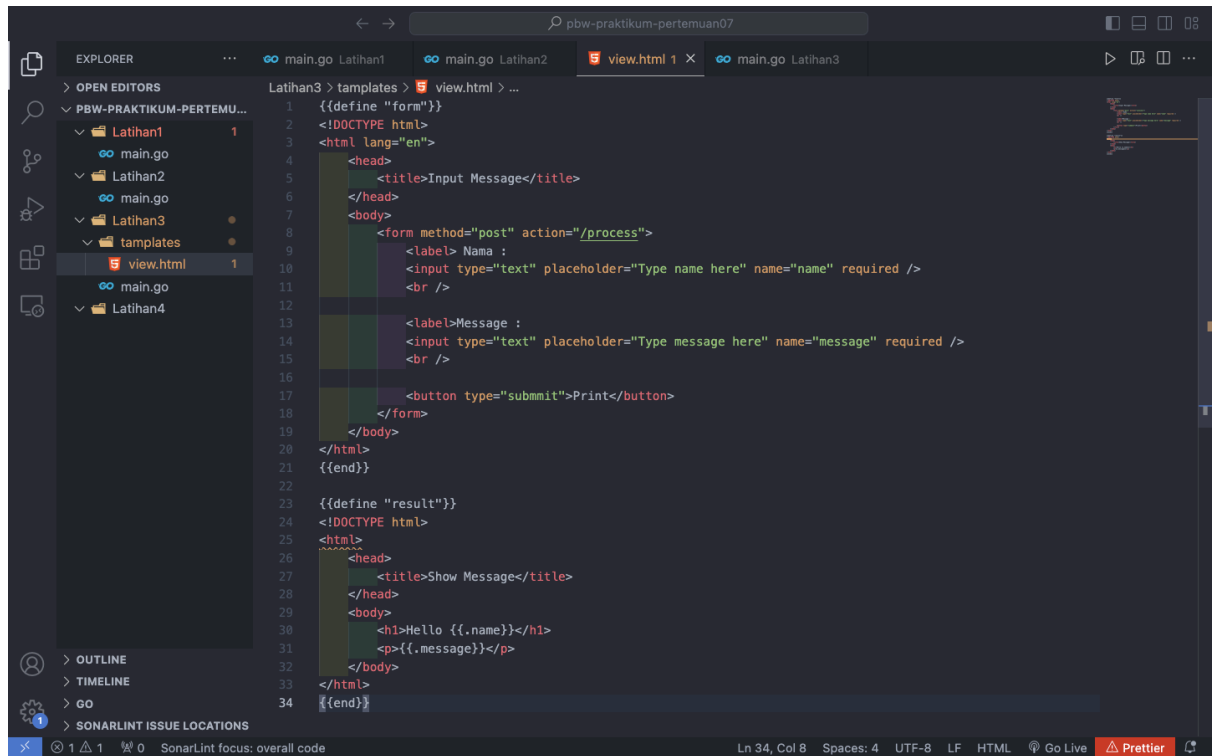
6. Apa itu front end, back end, dan API

- **Front end** : bagian yang terlihat oleh pengguna dalam sebuah aplikasi atau situs web, menangani antarmuka pengguna dan interaksi langsung. Front end biasanya dikembangkan dengan menggunakan bahasa pemrograman seperti HTML, CSS, dan JavaScript, serta kerangka kerja atau library seperti React, Angular, atau Vue.js.
- **Back end** : bagian dari aplikasi atau situs web yang tidak terlihat oleh pengguna, tetapi mengelola logika bisnis, pengolahan data, dan interaksi dengan server. Back end mencakup database, server, dan aplikasi yang menjalankan logika perangkat lunak. Biasanya, back end dikembangkan dengan menggunakan bahasa pemrograman seperti Python, Java, PHP, atau JavaScript
- **API** : serangkaian aturan dan protokol yang memungkinkan berbagai aplikasi dan sistem untuk berkomunikasi satu sama lain. API menyediakan cara bagi pengembang untuk mengakses dan menggunakan fungsionalitas atau data dari aplikasi atau layanan lain tanpa perlu mengetahui detail internalnya. API dapat digunakan untuk mengintegrasikan berbagai sistem, memungkinkan pertukaran data dan interaksi antaraplikasi

7. Praktek program front end

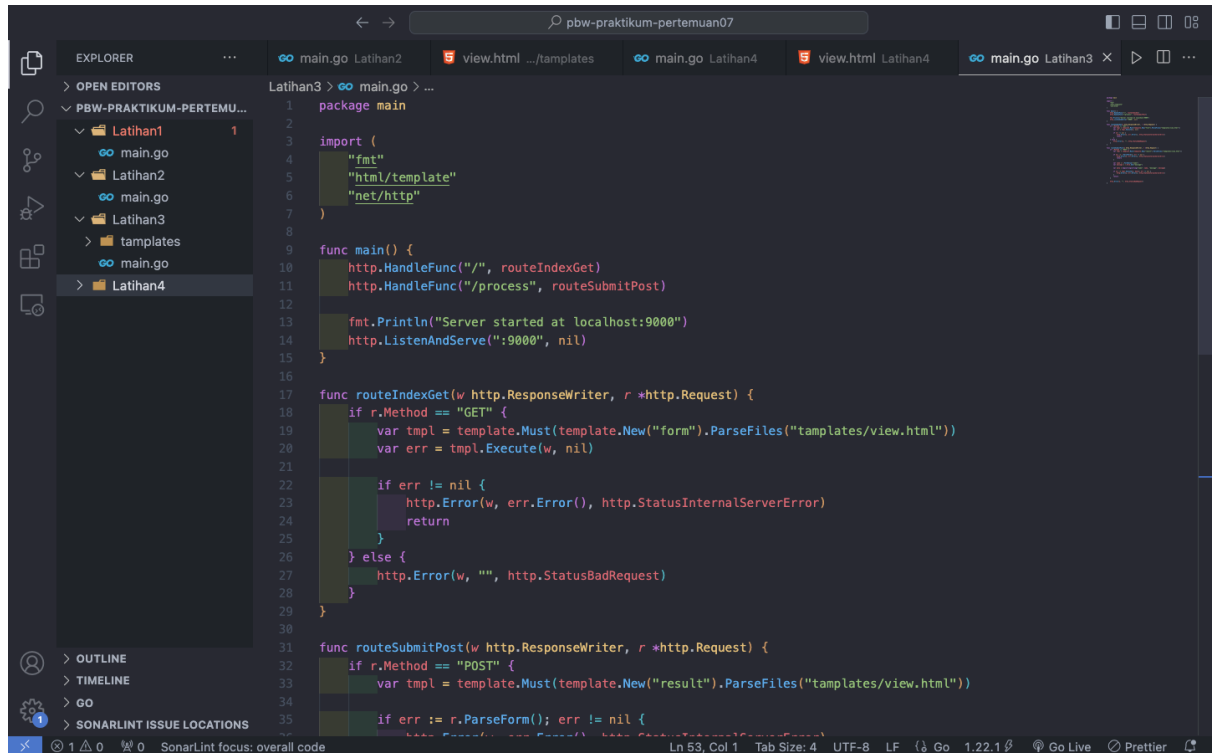
Screenshot Source Code

- view.html

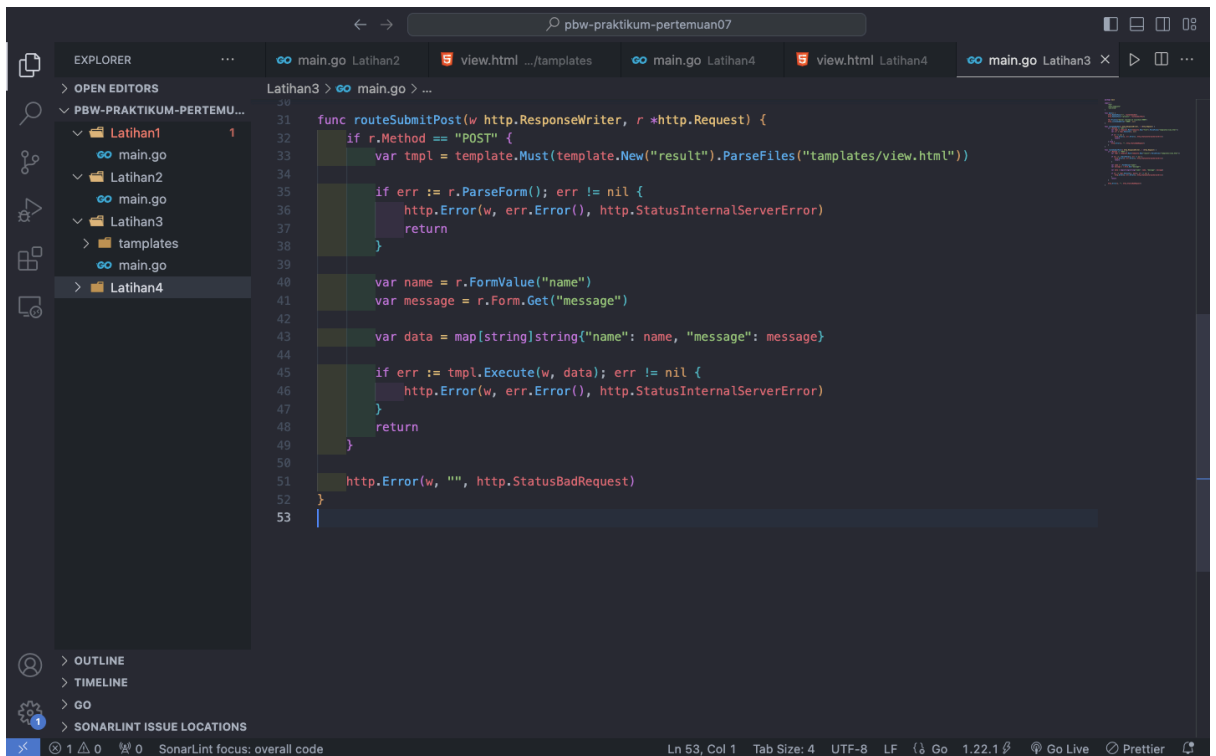


```
1  {{define "form"}}
2  <!DOCTYPE html>
3  <html lang="en">
4  <head>
5  <title>Input Message</title>
6  </head>
7  <body>
8  <form method="post" action="/process">
9  <label> Nama :
10 <input type="text" placeholder="Type name here" name="name" required />
11 <br />
12
13 <label>Message :
14 <input type="text" placeholder="Type message here" name="message" required />
15 <br />
16
17 <button type="submit">Print</button>
18 </form>
19 </body>
20 </html>
21 {{end}}
22
23 {{define "result"}}
24 <!DOCTYPE html>
25 <html>
26 <head>
27 <title>Show Message</title>
28 </head>
29 <body>
30 <h1>Hello {{.name}}</h1>
31 <p>{{.message}}</p>
32 </body>
33 </html>
34 {{end}}
```

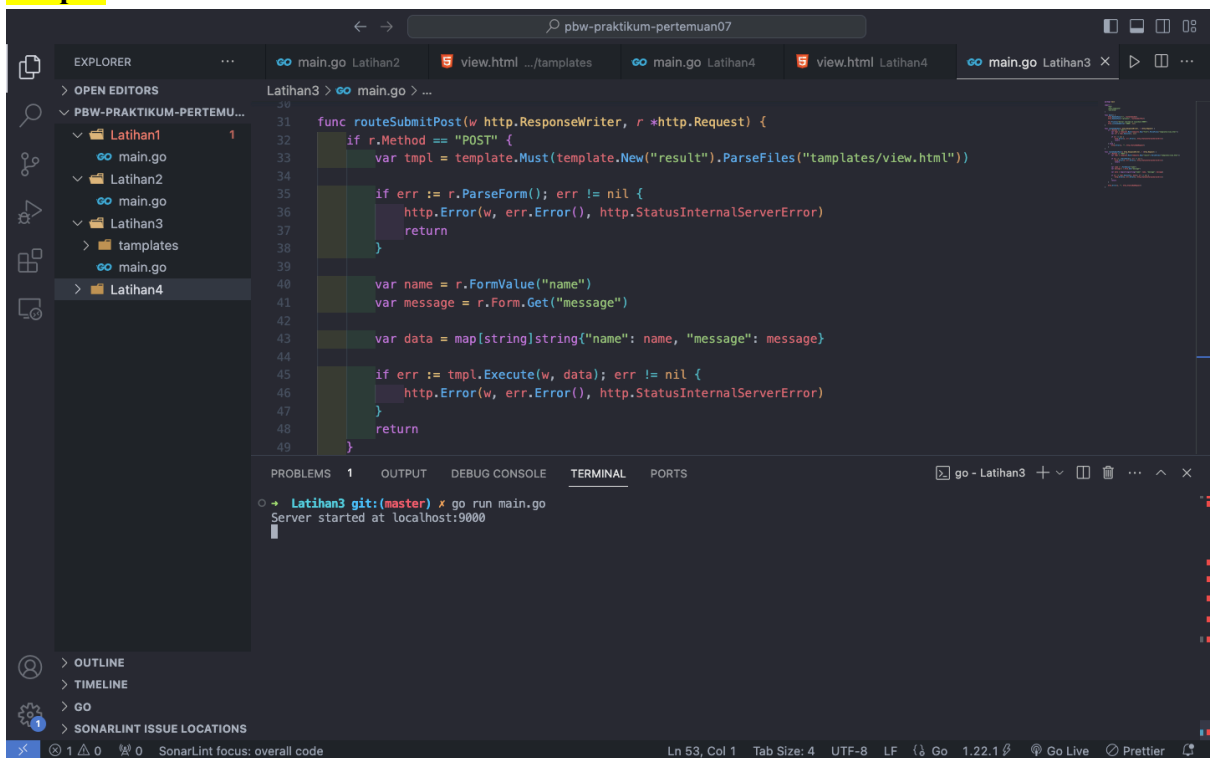
- Main.go



```
1  package main
2
3  import (
4      "fmt"
5      "html/template"
6      "net/http"
7  )
8
9  func main() {
10     http.HandleFunc("/", routeIndexGet)
11     http.HandleFunc("/process", routeSubmitPost)
12
13     fmt.Println("Server started at localhost:9000")
14     http.ListenAndServe(":9000", nil)
15 }
16
17 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
18     if r.Method == "GET" {
19         var tpl = template.Must(template.New("form").ParseFiles("templates/view.html"))
20         var err = tpl.Execute(w, nil)
21
22         if err != nil {
23             http.Error(w, err.Error(), http.StatusInternalServerError)
24             return
25         }
26     } else {
27         http.Error(w, "", http.StatusBadRequest)
28     }
29 }
30
31 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
32     if r.Method == "POST" {
33         var tpl = template.Must(template.New("result").ParseFiles("templates/view.html"))
34
35         if err := r.ParseForm(); err != nil {
```



Output



Golang Form Value - Dasar P... x Input Message x Getting started - My Worksp... x Download Postman | Get Star... x +

localhost:9000

Nama :

Message :

Golang Form Value - Dasar P... x Show Message x Getting started - My Worksp... x Download Postman | Get Star... x +

localhost:9000/process

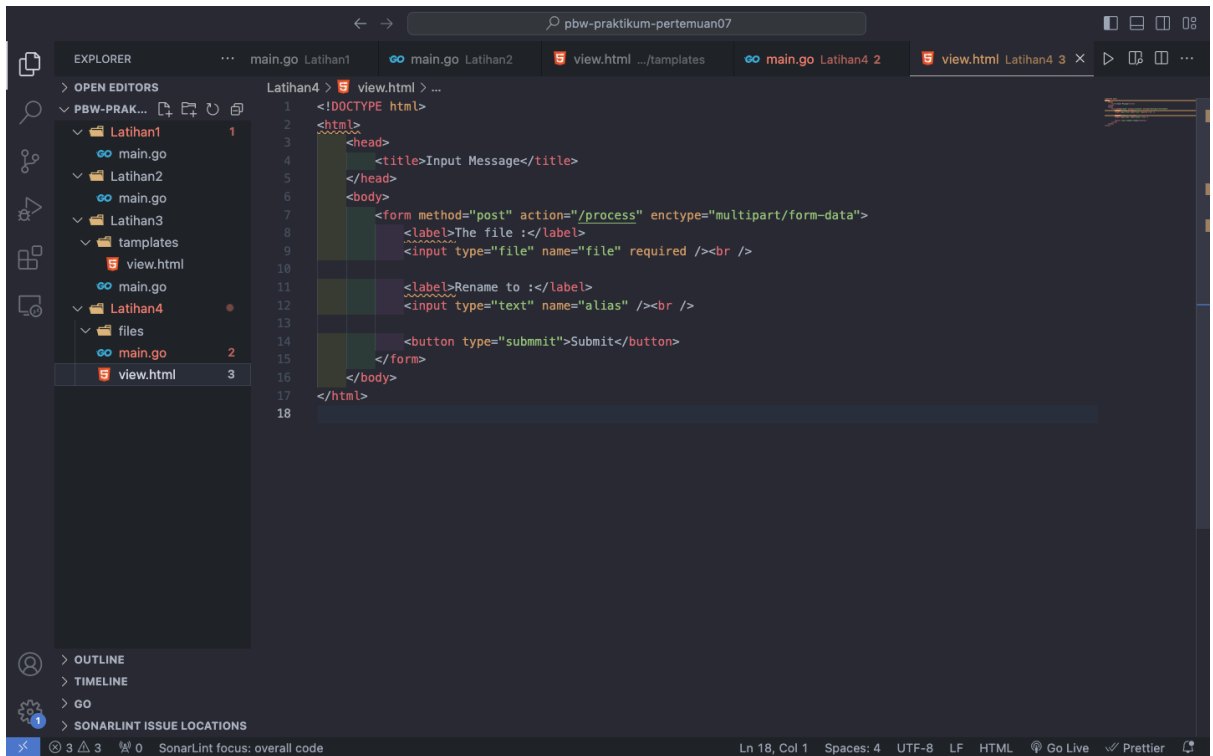
Hello Siti Ghumaisa

Hallo Prak PBW

8. Praktek program hal 27

Screenshot Source Code

- view.html

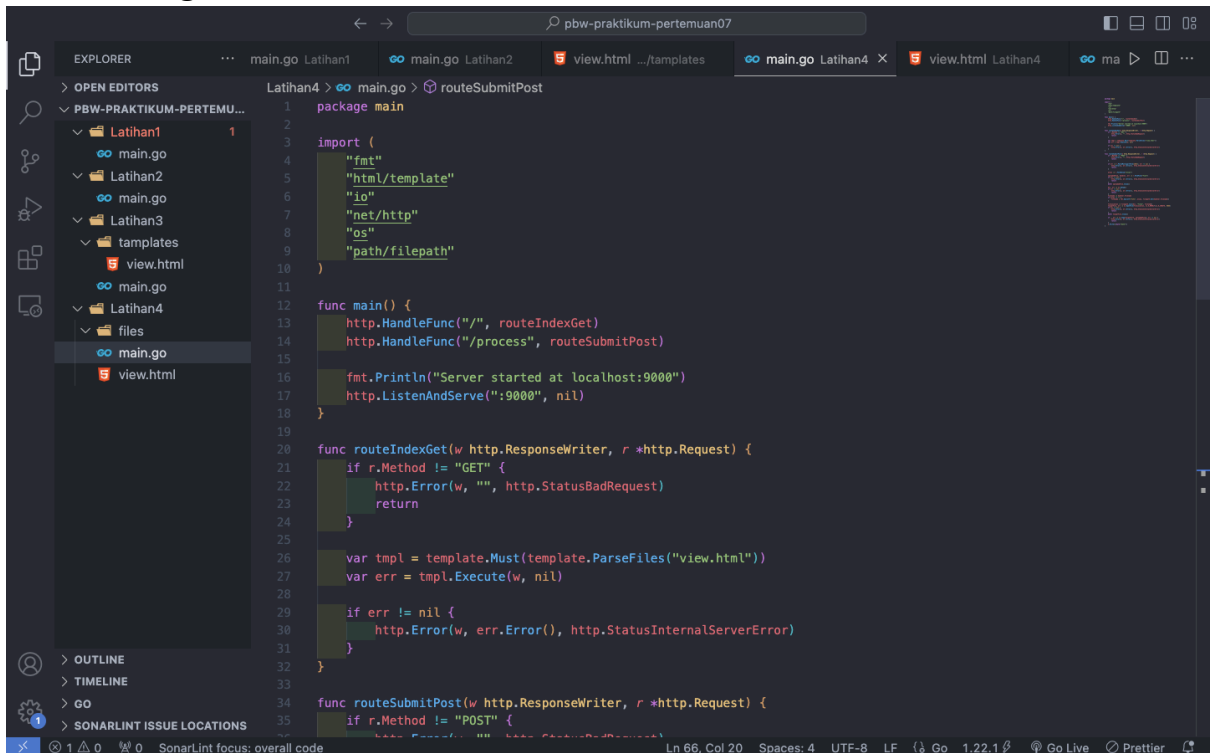


The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The project structure includes folders for 'Latihan1', 'Latihan2', 'Latihan3', 'templates', and 'Latihan4'. The 'view.html' file is selected in the Explorer and is also open in the editor. The editor displays the following HTML code:

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Input Message</title>
5   </head>
6   <body>
7     <form method="post" action="/process" enctype="multipart/form-data">
8       <label>The file :</label>
9       <input type="file" name="file" required /><br />
10
11       <label>Rename to :</label>
12       <input type="text" name="alias" /><br />
13
14       <button type="submit">Submit</button>
15     </form>
16   </body>
17 </html>
18
```

The status bar at the bottom indicates the file is at Line 18, Column 1, with 4 spaces, UTF-8 encoding, LF line endings, and HTML language. It also shows 'Go Live' and 'Prettier' icons.

- main.go



The screenshot shows the Visual Studio Code editor with the Explorer sidebar on the left. The project structure is the same as the previous screenshot. The 'main.go' file is selected in the Explorer and is also open in the editor. The editor displays the following Go code:

```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "io"
7     "net/http"
8     "os"
9     "path/filepath"
10 )
11
12 func main() {
13     http.HandleFunc("/", routeIndexGet)
14     http.HandleFunc("/process", routeSubmitPost)
15
16     fmt.Println("Server started at localhost:9000")
17     http.ListenAndServe(":9000", nil)
18 }
19
20 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
21     if r.Method != "GET" {
22         http.Error(w, "", http.StatusBadRequest)
23         return
24     }
25
26     var tpl = template.Must(template.ParseFiles("view.html"))
27     var err = tpl.Execute(w, nil)
28
29     if err != nil {
30         http.Error(w, err.Error(), http.StatusInternalServerError)
31     }
32 }
33
34 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
35     if r.Method != "POST" {
36         http.Error(w, "", http.StatusBadRequest)
37     }
38 }
```

The status bar at the bottom indicates the file is at Line 66, Column 20, with 4 spaces, UTF-8 encoding, LF line endings, and Go language. It also shows 'Go Live' and 'Prettier' icons.

pbw-praktikum-pertemuan07

EXPLORER

- main.go Latihan1
- main.go Latihan2
- view.html .../templates
- main.go Latihan4
- view.html Latihan4

OPEN EDITORS

- Latihan4 > main.go > routeSubmitPost

```
34 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
35     if r.Method != "POST" {
36         http.Error(w, "", http.StatusBadRequest)
37         return
38     }
39
40     if err := r.ParseMultipartForm(1024); err != nil {
41         http.Error(w, err.Error(), http.StatusInternalServerError)
42         return
43     }
44
45     alias := r.FormValue("alias")
46
47     uploadedFile, handler, err := r.FormFile("file")
48     if err != nil {
49         http.Error(w, err.Error(), http.StatusInternalServerError)
50         return
51     }
52     defer uploadedFile.Close()
53
54     dir, err := os.Getwd()
55     if err != nil {
56         http.Error(w, err.Error(), http.StatusInternalServerError)
57         return
58     }
59     filename := handler.Filename
60     if alias != "" {
61         filename = fmt.Sprintf("%s%s", alias, filepath.Ext(handler.Filename))
62     }
63
64     fileLocation := filepath.Join(dir, "files", filename)
65     targetFile, err := os.OpenFile(fileLocation, os.O_WRONLY|os.O_CREATE, 0666)
66     if err != nil {
67         http.Error(w, err.Error(), http.StatusInternalServerError)
68         return
69     }
```

Ln 66, Col 20 Spaces: 4 UTF-8 LF Go 1.22.1 Go Live Prettier

pbw-praktikum-pertemuan07

EXPLORER

- main.go Latihan1
- main.go Latihan2
- view.html .../templates
- main.go Latihan4
- view.html Latihan4

OPEN EDITORS

- Latihan4 > main.go > routeSubmitPost

```
34 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
64     fileLocation := filepath.Join(dir, "files", filename)
65     targetFile, err := os.OpenFile(fileLocation, os.O_WRONLY|os.O_CREATE, 0666)
66     if err != nil {
67         http.Error(w, err.Error(), http.StatusInternalServerError)
68         return
69     }
70     defer targetFile.Close()
71
72     if _, err := io.Copy(targetFile, uploadedFile); err != nil {
73         http.Error(w, err.Error(), http.StatusInternalServerError)
74         return
75     }
76     w.Write([]byte("done"))
77 }
```

Ln 66, Col 20 Spaces: 4 UTF-8 LF Go 1.22.1 Go Live Prettier

Output

The screenshot shows a VS Code editor with a Go web application. The Explorer sidebar on the left shows a project structure with folders 'Latihan1', 'Latihan2', 'Latihan3', 'Latihan4', and 'templates'. The 'Latihan4' folder is expanded, showing 'files' and 'main.go'. The main editor displays the code in 'main.go' for 'Latihan4', which includes imports for 'fmt', 'html/template', 'io', 'net/http', 'os', and 'path/filepath'. The 'main' function sets up a web server on localhost:9000 with routes for '/' (routeIndexGet) and '/process' (routeSubmitPost). The terminal at the bottom shows the command 'go run main.go' being executed, resulting in the output 'Server started at localhost:9000'.

```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "io"
7     "net/http"
8     "os"
9     "path/filepath"
10 )
11
12 func main() {
13     http.HandleFunc("/", routeIndexGet)
14     http.HandleFunc("/process", routeSubmitPost)
15
16     fmt.Println("Server started at localhost:9000")
17     http.ListenAndServe(":9000", nil)
18 }
19
20 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Latihan4 git:(master) x go run main.go
Server started at localhost:9000

The file Screen Shot...6.26.51.png

Rename to :

