

LAPORAN LATIHAN PRAKTIKUM KE-7
PEMROGRAMAN BERBASIS WEB
KELAS A

Disusun Untuk Memenuhi Tugas Mata Kuliah Prak. Pemrograman Berbasis Web



Disusun oleh:

Arsya Yan Duribta 4522210117

Dosen Pengampu:

Adi Wahyu Pribadi, S.Si., M.Kom.

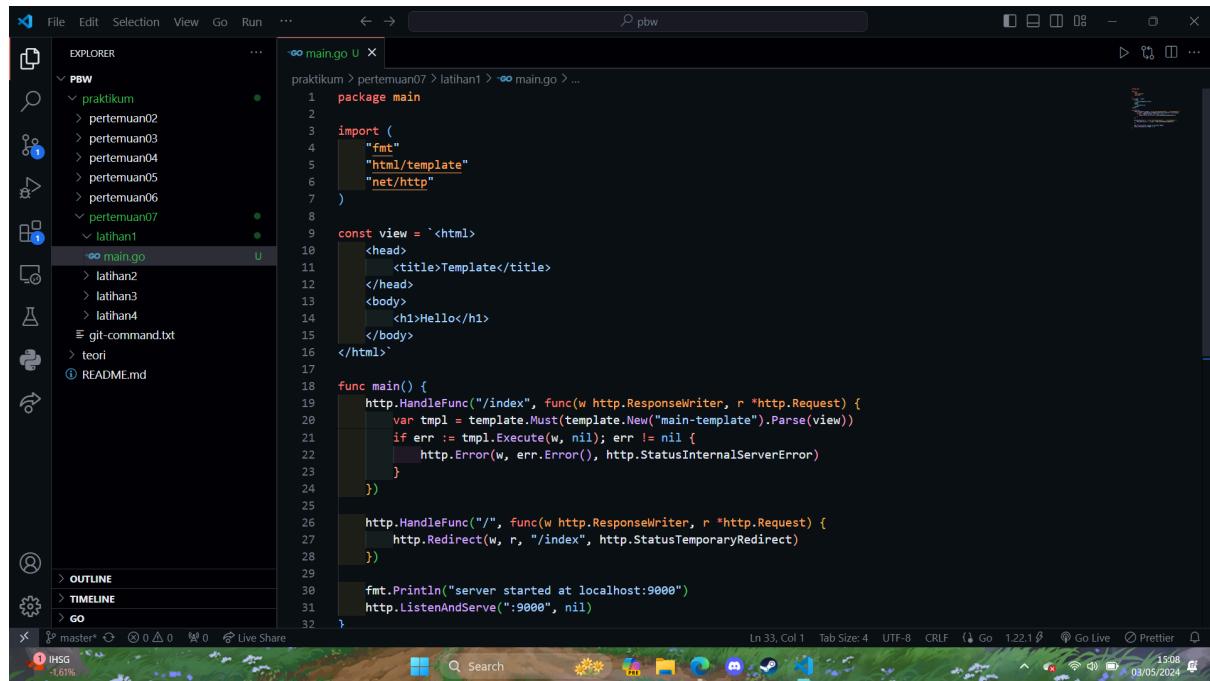
Program Studi Teknik Informatika
Fakultas Teknik Universitas Pancasila
2023/2024

Link Repository Github:

<https://github.com/Arsyayd11/pbw/tree/master/praktikum/pertemuan07>

Latihan 1: TEMPLATE: RENDER HTML STRING

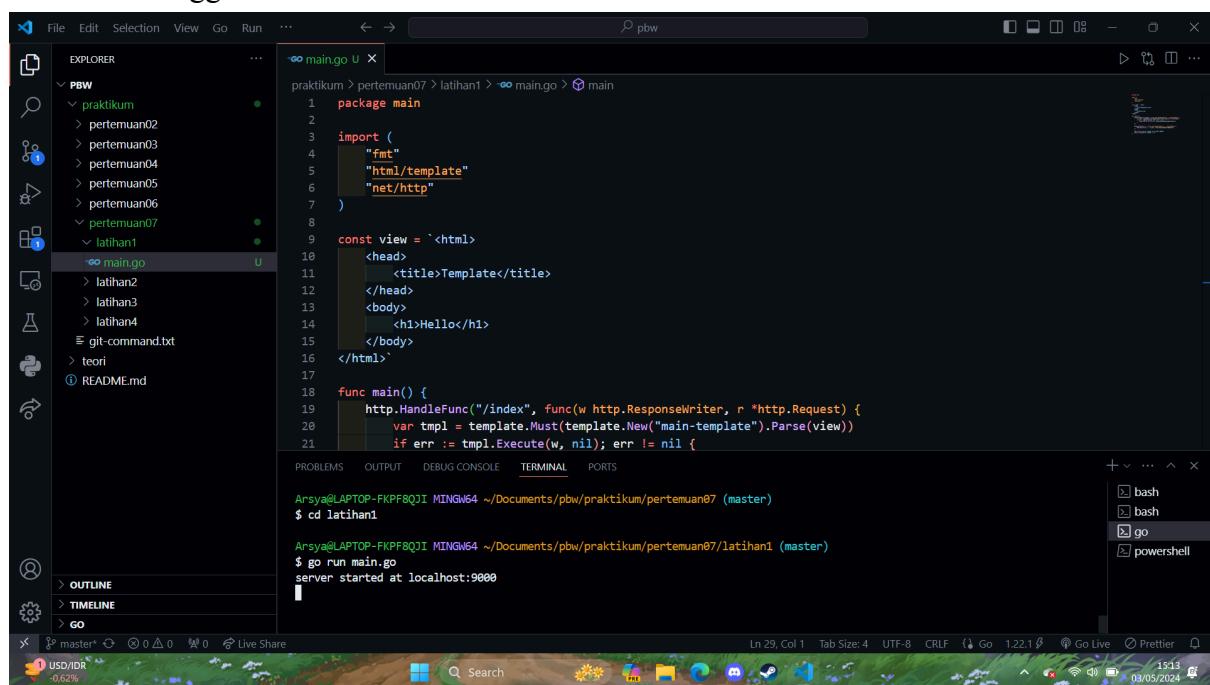
Implementasinya (Source Code):



```
File Edit Selection View Go Run ... ← → pbw

main.go | x
praktikum > pertemuan07 > latihan1 > main.go ...
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 const view = `<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 }
```

Run code menggunakan terminal:



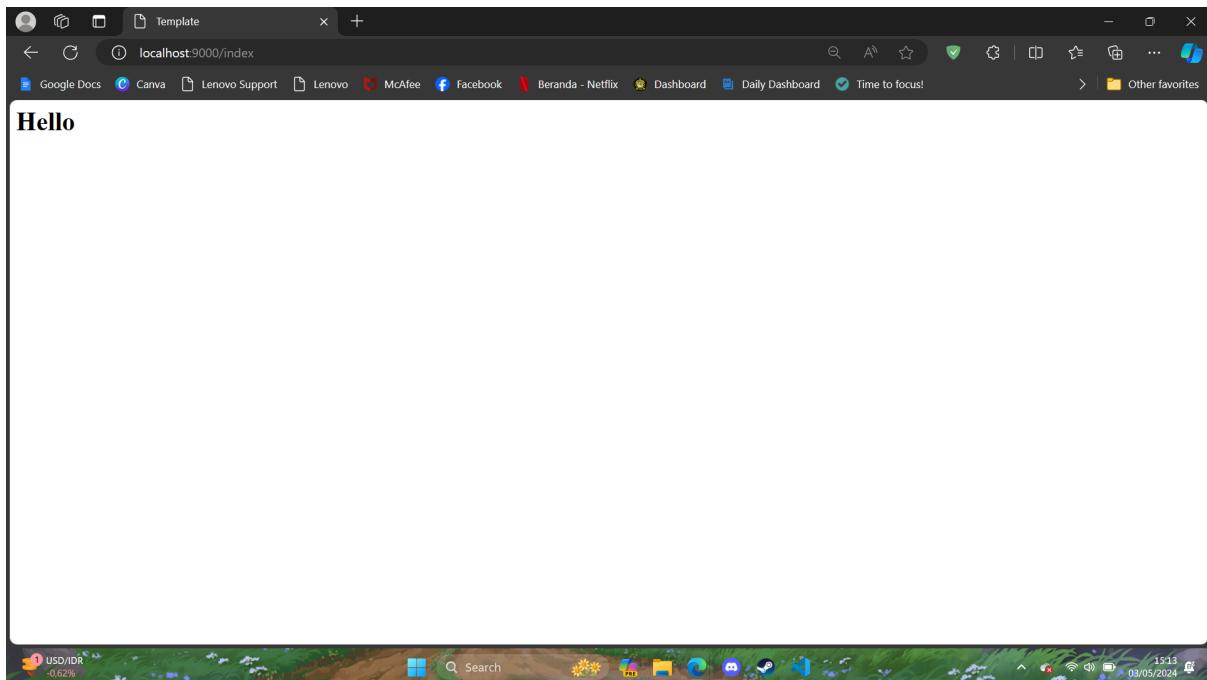
```
File Edit Selection View Go Run ... ← → pbw

main.go | x
praktikum > pertemuan07 > latihan1 > main.go > main
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7 )
8
9 const view = `<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 }

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Arasya@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ cd latihan1
Arasya@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07/latihan1 (master)
$ go run main.go
server started at localhost:9000
```

Tampilan localhost pada website:



Penjelasan:

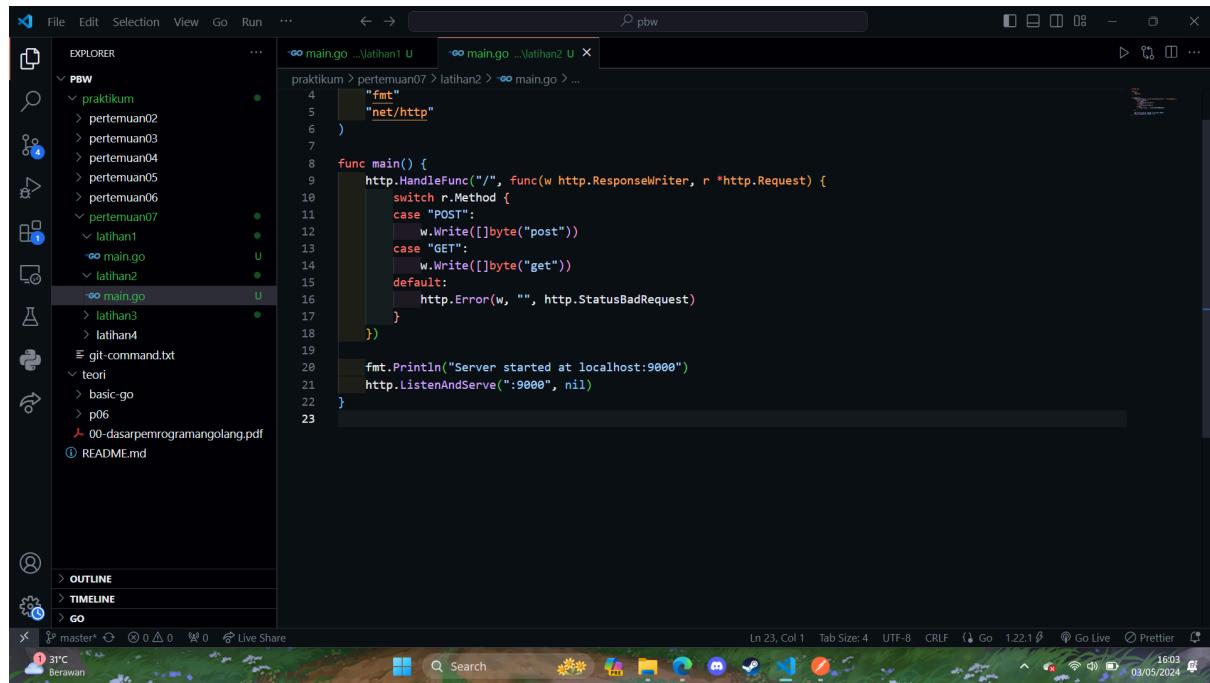
- **JELASKAN HTML, PARSING METHOD!**

HTML (Hypertext Markup Language) merupakan bahasa markup yang digunakan untuk membuat halaman website. HTML berfungsi untuk menentukan struktur dan elemen-elemen suatu halaman web, seperti teks, gambar, link, dan lainnya. HTML terdiri dari serangkaian tag-tag yang menandakan elemen-elemen dalam halaman web.

Parsing adalah proses penguraian atau memproses data dalam bentuk tertentu menjadi struktur data yang lebih terstruktur atau mudah dipahami. Pada kode latihan 1 di atas, parsing digunakan pada template HTML menggunakan package `html/template`. Proses parsing ini menghasilkan halaman web yang siap untuk ditampilkan kepada pengguna.

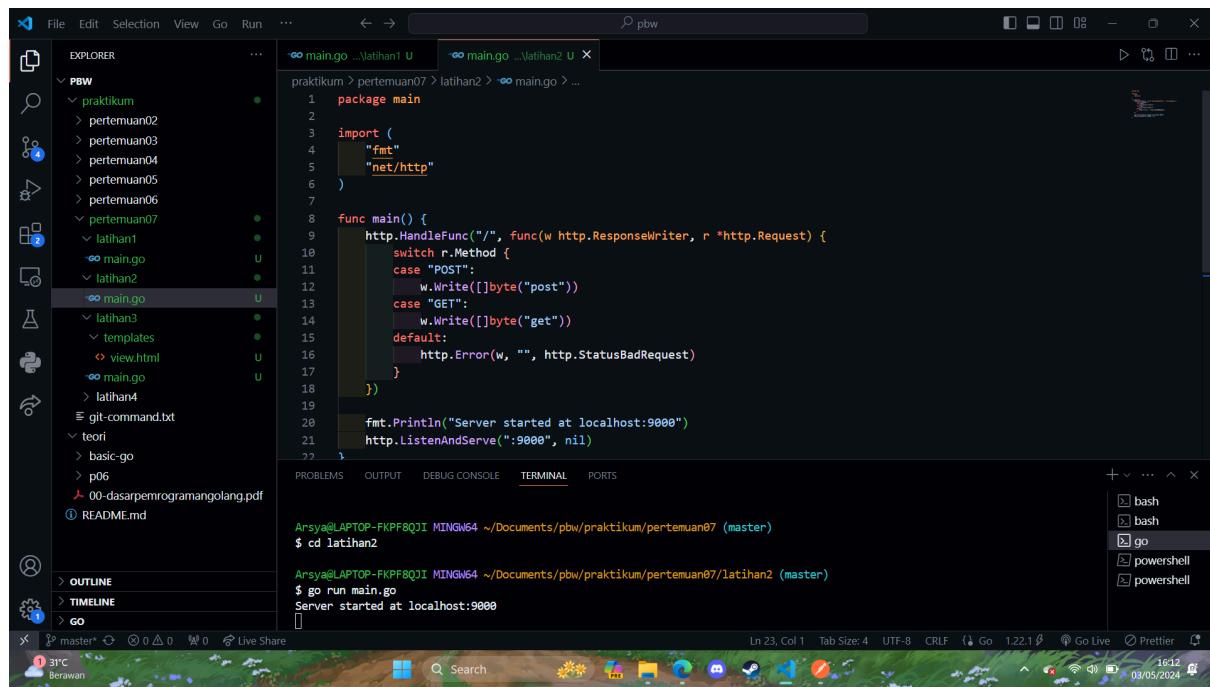
Latihan 2: HTTP METHOD: POST & GET

Implementasinya (Source Code):



```
pbw
main.go ...latihan1 U -> main.go ...latihan2 U x
praktikum > pertemuan07 > latihan2 > main.go > ...
4   "fmt"
5   "net/http"
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 }
```

Run code menggunakan terminal:

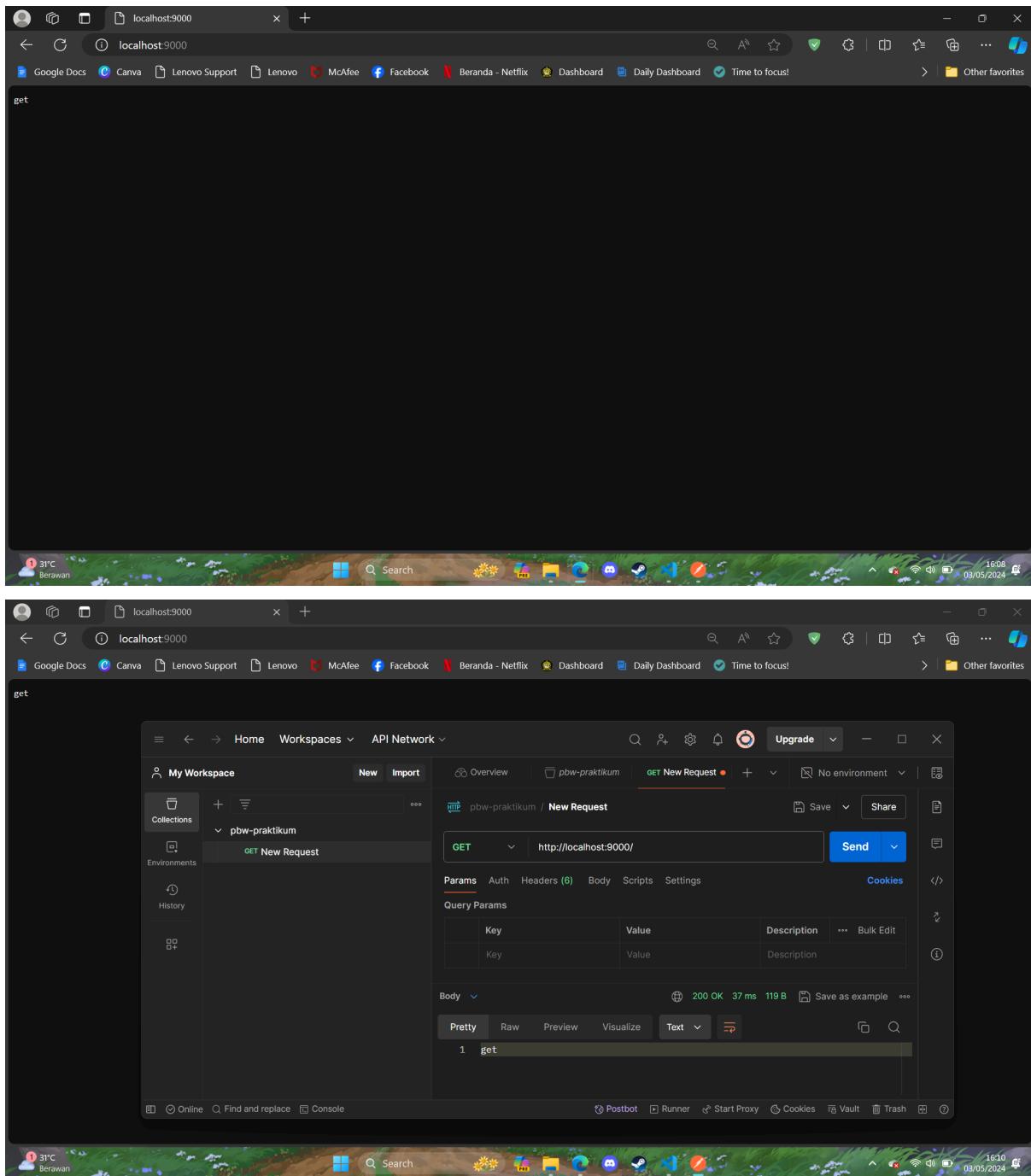


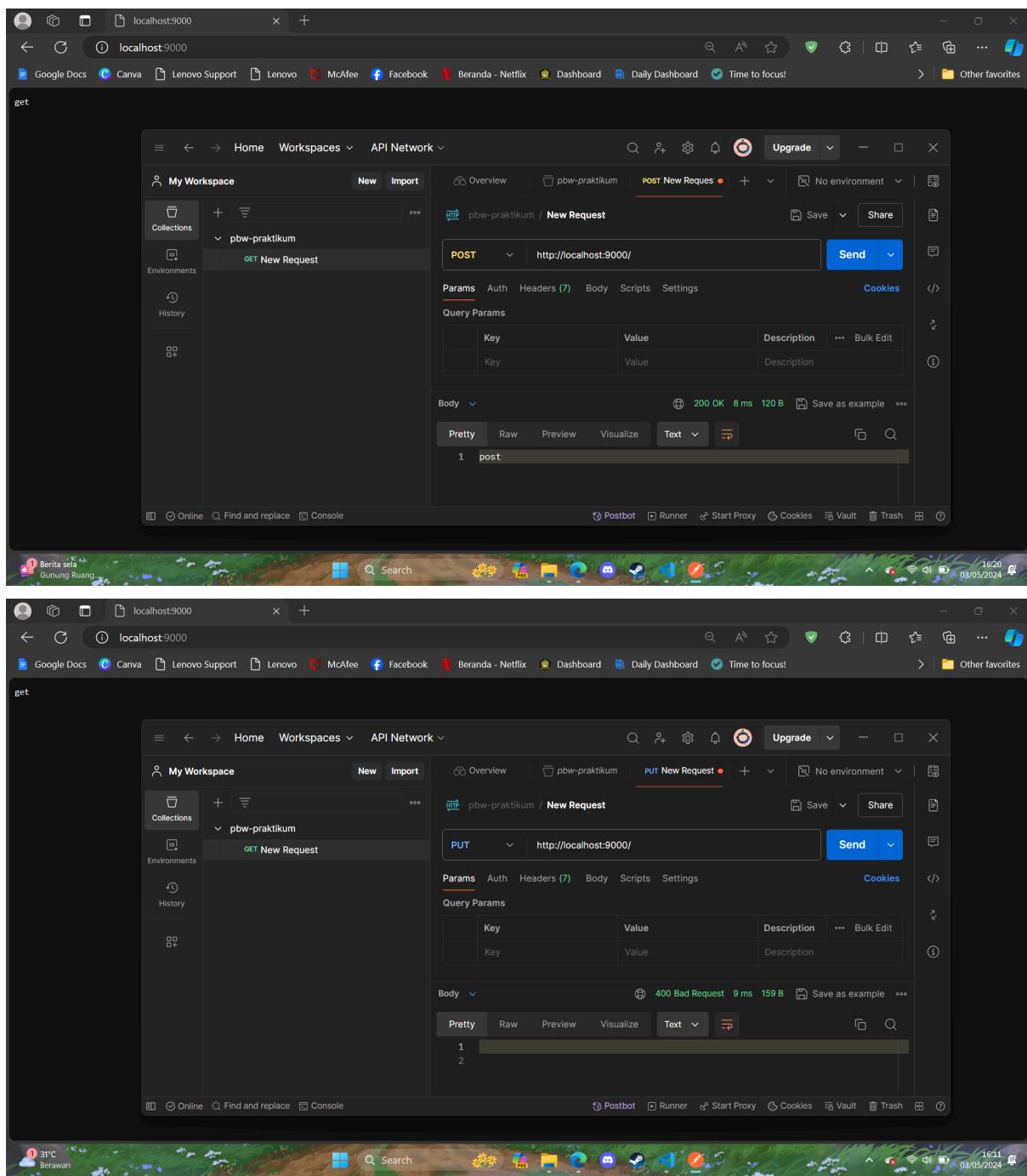
```
pbw
main.go ...latihan1 U -> main.go ...latihan2 U x
praktikum > pertemuan07 > latihan2 > main.go > ...
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func main() {
9     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
10         switch r.Method {
11             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 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Arsya@LAPTOP-FKPF8QJI MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ cd latihan2
Arsya@LAPTOP-FKPF8QJI MINGW64 ~/Documents/pbw/praktikum/pertemuan07/latihan2 (master)
$ go run main.go
Server started at localhost:9000
```

Tampilan localhost pada website:





Penjelasan:

- **APA ITU POSTMAN ?**

Postman adalah sebuah platform yang memungkinkan para pengembang untuk merancang, menguji, dan memecah masalah API secara lebih efisien. Dengan Postman, pengembang dapat membuat permintaan HTTP ke endpoint API, mengirimkan berbagai jenis permintaan seperti GET, POST, PUT, DELETE, dan lainnya, serta menganalisis respon yang diterima. Postman juga menyediakan fitur-fitur seperti pengaturan koleksi permintaan, otomatisasi pengujian, dan dokumentasi API yang memudahkan pengembangan dan kolaborasi tim.

- Kenapa Menggunakan Selain Dua Method Akan Menghasilkan Message 400 Bad Request?

Hal ini terjadi karena server hanya mengizinkan dua jenis method, yaitu "POST" dan "GET". Ketika server menerima permintaan dengan method selain kedua method tersebut, server tidak dapat memproses permintaan tersebut sesuai dengan logika yang telah ditentukan dalam kode. Sebagai respon, server mengirimkan pesan error dengan kode status 400 Bad Request, yang menandakan bahwa permintaan yang diberikan tidak dapat dipenuhi karena kesalahan dari klien.

Latihan 3: FORM VALUE

Implementasinya (Source Code):

- main.go

```
File Edit Selection View Go Run ... ← → pbw
praktikum > pertemuan07 > latihan3 > 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 tmpl = template.Must(template.New("form").ParseFiles("templates/view.html"))
20         var err = tmpl.Execute(w, nil)
21
22         if err != nil {
23             http.Error(w, err.Error(), http.StatusInternalServerError)
24             return
25         } else {
26             http.Error(w, "", http.StatusBadRequest)
27         }
28     }
29 }
30
31 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
32     if r.Method == "POST" {
33         var tmpl = template.Must(template.New("result").ParseFiles("templates/view.html"))
34
35         if err := r.ParseForm(); err != nil {
36             http.Error(w, err.Error(), http.StatusInternalServerError)
37             return
38         }
39
40         var name = r.FormValue("name")
41         var message = r.Form.Get("message")
42
43         var data = map[string]string{"name": name, "message": message}
44
45         if err := tmpl.Execute(w, data); err != nil {
46             http.Error(w, err.Error(), http.StatusInternalServerError)
47         }
48         return
49     }
50
51     http.Error(w, "", http.StatusBadRequest)
52 }
53
```

```
File Edit Selection View Go Run ... ← → pbw
praktikum > pertemuan07 > latihan3 > main.go ...
17 func routeIndexGet(w http.ResponseWriter, r *http.Request) {
18     http.Error(w, err.Error(), http.StatusInternalServerError)
19     return
20 }
21 } else {
22     http.Error(w, "", http.StatusBadRequest)
23 }
24 }
25
26 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
27     if r.Method == "POST" {
28         var tmpl = template.Must(template.New("result").ParseFiles("templates/view.html"))
29
30         if err := r.ParseForm(); err != nil {
31             http.Error(w, err.Error(), http.StatusInternalServerError)
32             return
33         }
34
35         var name = r.FormValue("name")
36         var message = r.Form.Get("message")
37
38         var data = map[string]string{"name": name, "message": message}
39
40         if err := tmpl.Execute(w, data); err != nil {
41             http.Error(w, err.Error(), http.StatusInternalServerError)
42         }
43         return
44     }
45
46     http.Error(w, "", http.StatusBadRequest)
47 }
48
49 }
```

● view.html

The screenshot shows the VS Code interface with the 'view.html' file open in the center editor tab. The code is a Go template for a web form. It defines a 'form' with fields for 'name' and 'message', and a submit button. It also defines a 'result' template that prints 'Hello {{.name}}' and 'Message {{.message}}'. The file is part of a project structure under 'praktikum'.

```
praktikum > pertemuan07 > latihan3 > templates > view.html
1 {{define "form"}}
2 <!DOCTYPE html>
3 <html>
4   <head>
5     <title>Input Message</title>
6   </head>
7   <body>
8     <form method="post" action="/process">
9       <label>Name:</label>
10      <input type="text" placeholder="Type name here" name="name" required />
11      <br />
12      <label>Message:</label>
13      <input type="text" placeholder="Type message here" name="message" required />
14      <br />
15      <button type="submit">Print</button>
16    </form>
17  </body>
18 </html>
19 {{end}}
20
21 {{define "result"}}
22 <!DOCTYPE html>
23 <html>
24   <head>
25     <title>Show Message</title>
26   </head>
27   <body>
28     <h1>Hello {{.name}}</h1>
29     <p>{{.message}}</p>
30   </body>
31 </html>
32 {{end}}
```

Run code menggunakan terminal:

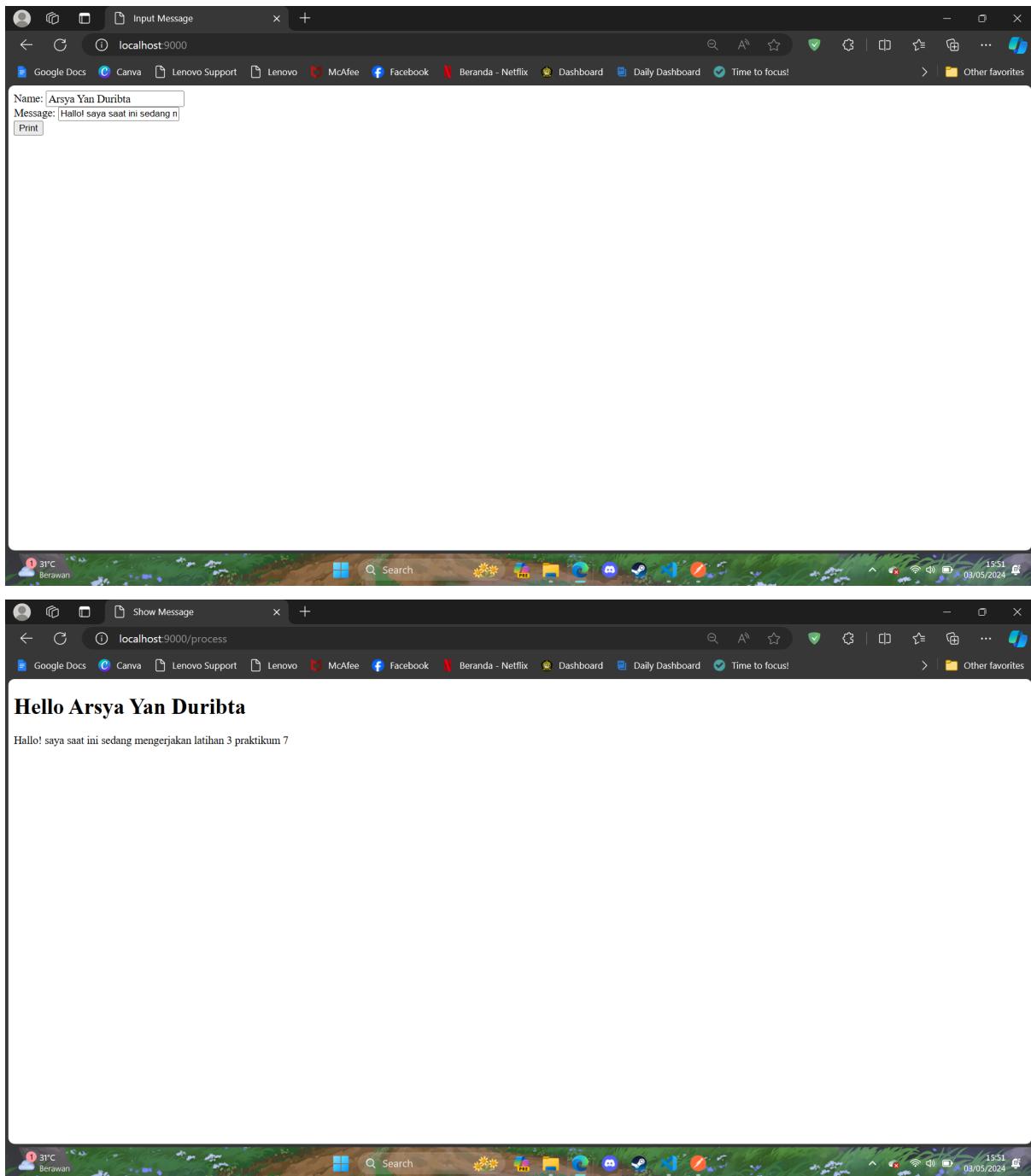
The screenshot shows the VS Code interface with the 'main.go' file open in the center editor tab. The code sets up a simple HTTP server that handles GET and POST requests. It uses a Go template to render 'view.html' for GET requests and prints the message for POST requests. The terminal below shows the command to run the program and the resulting output.

```
praktikum > pertemuan07 > latihan3 > 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 tmpl = template.Must(template.New("form").ParseFiles("templates/view.html"))
20     var err = tmpl.Execute(w, nil)
21
22     if err != nil {
23       log.Println(err)
24     }
25   }
26 }
```

Terminal Output:

```
Arsya@LAPTOP-FKPF8QJI MINGW64 ~/Documents/pbw/praktikum/pertemuan07/latihan3 (master)
$ cd latihan3
$ go run main.go
Server started at localhost:9000
```

Tampilan localhost pada website:



Penjelasan:

- **APA ITU FRONT END, BACK END, API ?**

Front-end adalah bagian dari aplikasi web yang bertanggung jawab untuk tampilan dan interaksi langsung dengan pengguna. Ini mencakup elemen-elemen seperti HTML, CSS, dan JavaScript yang digunakan untuk merancang dan mengatur tampilan serta interface dari halaman web yang dilihat pengguna.

Back-end adalah bagian dari aplikasi web yang tidak terlihat oleh pengguna. Ini mencakup server, database, dan logika aplikasi yang memproses permintaan dari pengguna, melakukan manipulasi data, dan menghasilkan respon yang sesuai.

Back-end biasanya dapat ditulis dengan menggunakan bahasa pemrograman seperti Golang, Python, atau Node.js.

API atau Application Programming Interface merupakan interface yang dapat menghubungkan satu aplikasi dengan aplikasi lainnya. Dengan kata lain, peran API adalah sebagai perantara antar berbagai aplikasi berbeda, baik dalam satu platform yang sama atau pun lintas platform.

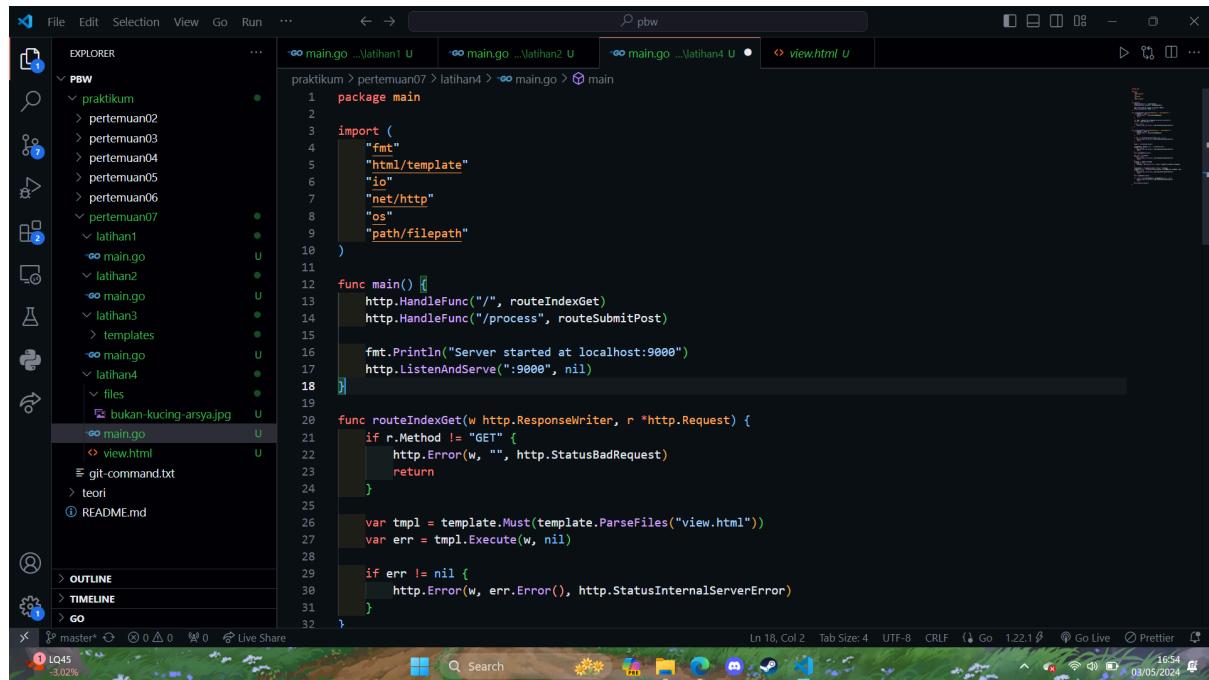
Berikut adalah beberapa manfaat dari API:

1. Memudahkan Pembuatan Aplikasi yang Fungsional
2. Efisiensi Pengembangan Aplikasi
3. Meringankan Beban Server

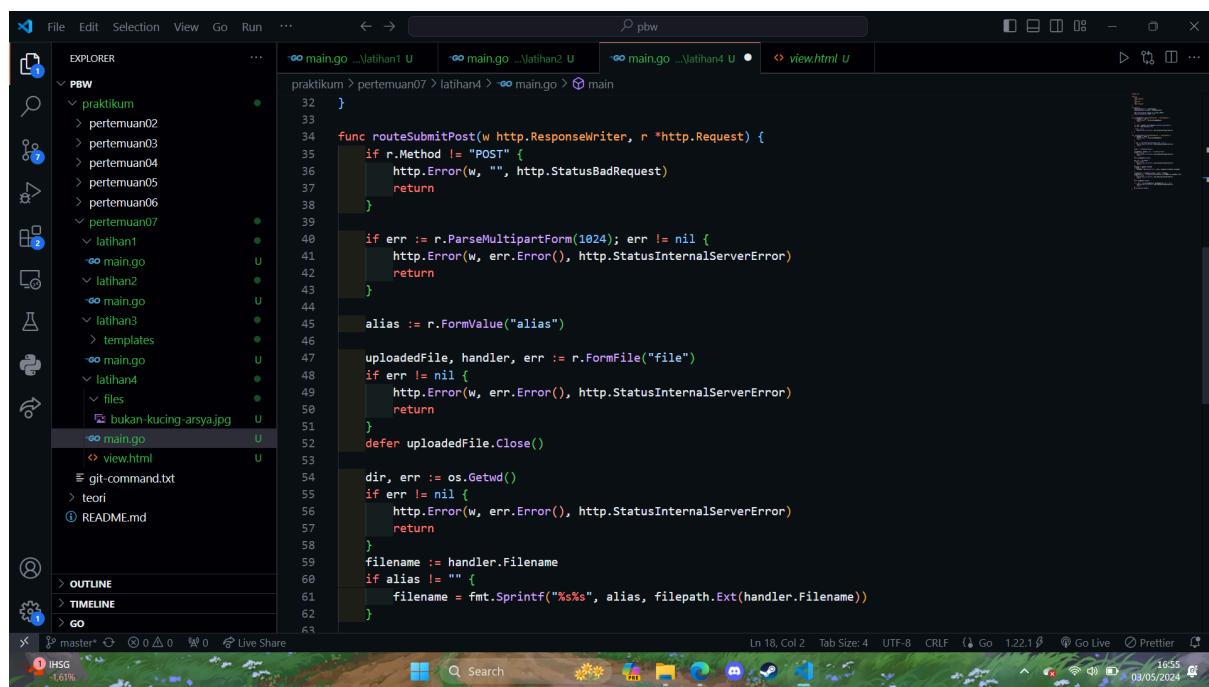
Latihan 4: FORM UPLOAD FILE

Implementasinya (Source Code):

- main.go



```
File Edit Selection View Go Run ... ← → pbw
praktikum > pertemuan07 > latihan4 > main.go > main
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 tmpl = template.Must(template.ParseFiles("view.html"))
27     var err = tmpl.Execute(w, nil)
28
29     if err != nil {
30         http.Error(w, err.Error(), http.StatusInternalServerError)
31     }
32 }
```



```
File Edit Selection View Go Run ... ← → pbw
praktikum > pertemuan07 > latihan4 > main.go > view.html
32 }
33
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 }
```

```
pbw
File Edit Selection View Go Run ... ← → ⌂ pbw
praktikum > pertemuan07 > latihan4 > main.go > ...
34 func routeSubmitPost(w http.ResponseWriter, r *http.Request) {
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 }
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 }
```

master* 0 0 0 0 Live Share 1655 03/05/2024

● view.html

```
pbw
File Edit Selection View Go Run ... ← → ⌂ pbw
praktikum > pertemuan07 > latihan4 > view.html > ...
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            <label>Rename to:</label>
11            <input type="text" name="alias" /><br />
12            <button type="submit">Submit</button>
13        </form>
14    </body>
15 </html>
16
```

master* 0 0 0 0 Live Share 1708 03/05/2024

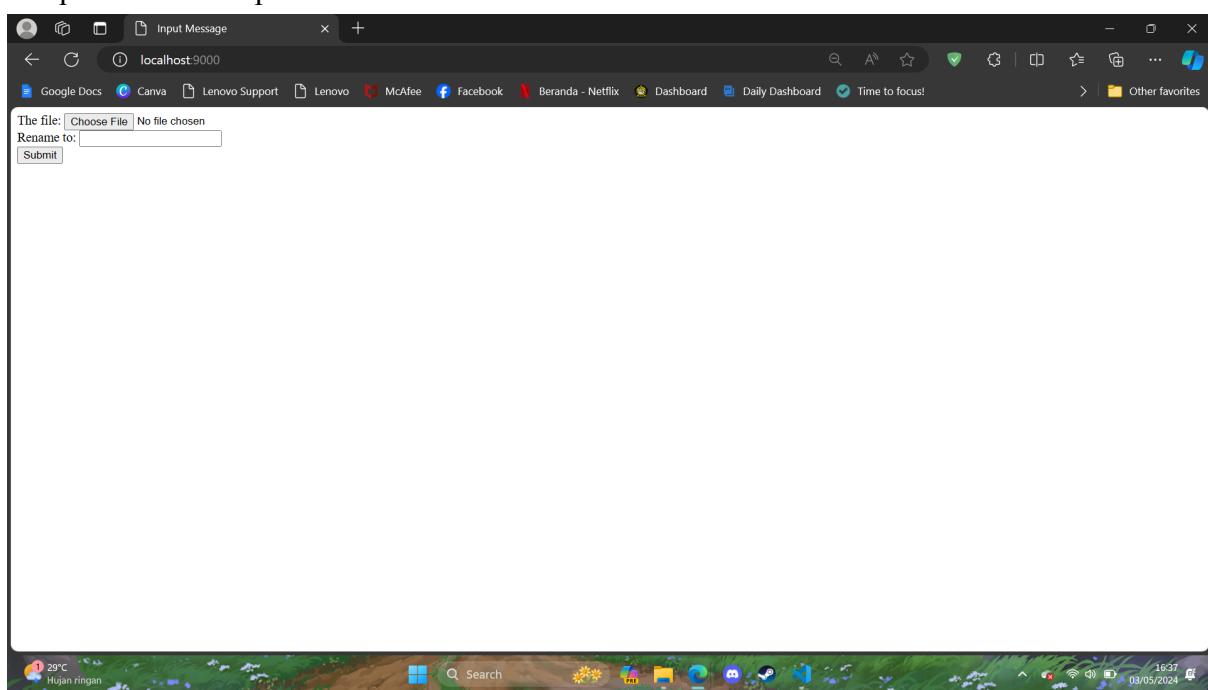
Run code menggunakan terminal:

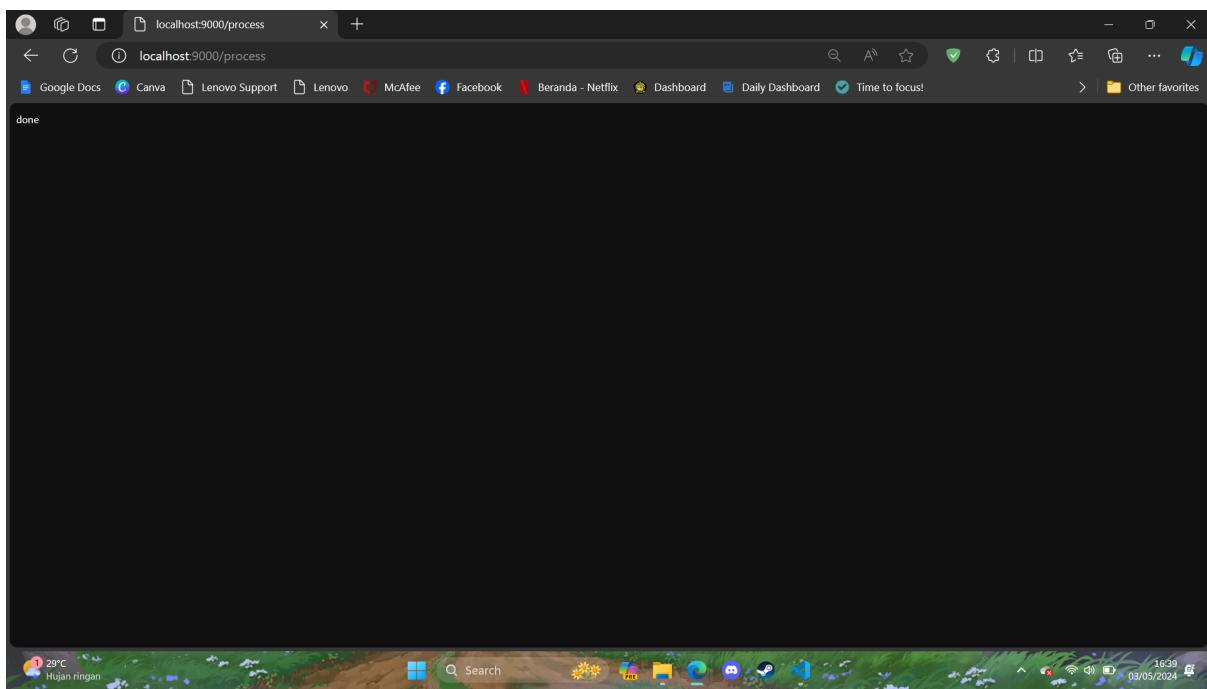
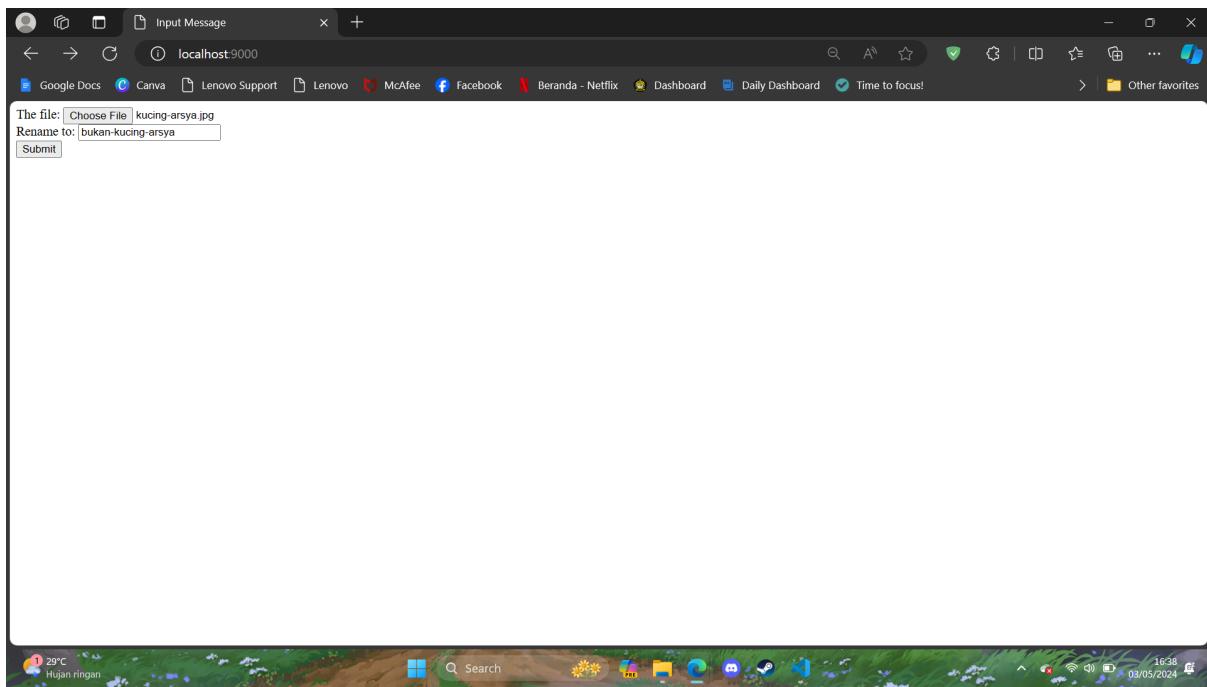
A screenshot of the Visual Studio Code interface. The Explorer sidebar shows a project structure under 'PBW' with folders like 'praktikum', 'pertemuan02', 'pertemuan03', etc., and files such as 'main.go', 'view.html', and 'git-command.txt'. The 'TERMINAL' tab is active, displaying the command-line output of a Go application. The terminal window shows the user navigating to the 'latihan4' directory and running the command '\$ go run main.go'. The response 'Server started at localhost:9000' is visible. The status bar at the bottom indicates the file is 18 lines long, has 2 tabs open, and the current tab is 'TERMINAL'. The system tray shows the date as 03/05/2024 and the time as 16:49.

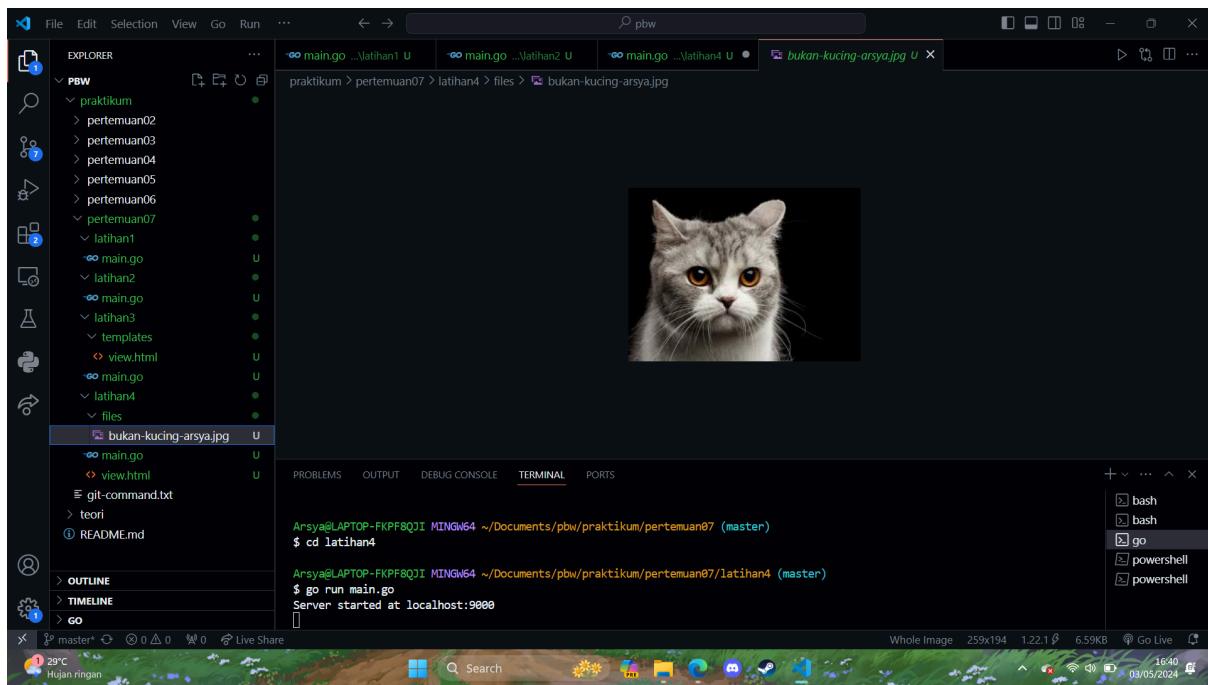
```
praktikum > pertemuan07 > latihan4 > main.go > main
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "io"
7     "net/http"
8     "os"
9     "path/filepath"
10 )
11
12 func main() {
13     http.HandleFunc("/", routeIndexGet)
14     http.HandleFunc("/process", routeSubmitPost)
15
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 }
```

```
Arsyah@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ cd latihan4
$ go run main.go
Server started at localhost:9000
```

Tampilan localhost pada website:







Penjelasan:

- Perlu diperhatikan, pada tag <form> perlu ditambahkan atribut enctype="multipart/form-data" kenapa perlu ditambahkan ? Jelaskan di laporan!

Pada tag <form>, perlu ditambahkan atribut enctype="multipart/form-data". Hal ini perlu ditambahkan karena attribute enctype ini menentukan cara browser mengenkapsulasi data form sebelum mengirimkannya ke server. Saat mengunggah file, kita perlu menggunakan multipart/form-data agar browser dapat mengirimkan file yang dipilih oleh pengguna dalam format yang sesuai.

Upload Ke Git

The screenshot shows a terminal window in Visual Studio Code with the following command history:

```
Arsyah@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ git add .
Arsyah@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ git commit -m "upload latihan praktikum ke 7"
[master 063bb4d] upload latihan praktikum ke 7
 7 files changed, 238 insertions(+)
    create mode 100644 praktikum/pertemuan07/latihan1/main.go
    create mode 100644 praktikum/pertemuan07/latihan2/main.go
    create mode 100644 praktikum/pertemuan07/latihan3/main.go
    create mode 100644 praktikum/pertemuan07/latihan3/templates/view.html
    create mode 100644 praktikum/pertemuan07/latihan4/files/bukan-kucing-arsya.jpg
    create mode 100644 praktikum/pertemuan07/latihan4/main.go
    create mode 100644 praktikum/pertemuan07/latihan4/view.html
Arsyah@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan07 (master)
$ git push -f origin master
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (17/17), 9.34 KiB | 735.00 KiB/s, done.
Total 17 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To https://github.com/Arsyahd11/pbw.git
 d9e9c19..063bb4d master -> master
```

Kesimpulan

Praktikum ini telah memberikan wawasan tentang pengembangan web menggunakan GoLang. Dari memahami dasar-dasar HTML dan penggunaan package html/template untuk membuat halaman web yang dinamis, hingga pemahaman tentang peran masing-masing komponen dalam arsitektur web seperti front-end, back-end, dan API. Dengan menggunakan GoLang, kita dapat dengan mudah merancang tampilan yang menarik di front-end dan mengimplementasikan logika bisnis yang kuat di back-end. Selain itu, praktikum ini juga memperkenalkan penggunaan Postman sebagai alat bantu yang efektif dalam pengujian dan debugging API. Terakhir, praktikum ini juga memberikan pemahaman tentang penggunaan form untuk mengunggah file dalam pengembangan web dengan GoLang. Dengan memperhatikan penggunaan atribut enctype="multipart/form-data" pada tag <form>, kita dapat memastikan bahwa aplikasi web yang kita kembangkan mampu menerima dan mengelola unggahan file dari pengguna dengan baik.

Link Repository Github:

<https://github.com/Arsyahd11/pbw/tree/master/praktikum/pertemuan07>