

**LAPORAN PRAKTIKUM-08**  
**PEMROGRAMAN BERBASIS WEB**  
**KELAS A**

Disusun Untuk Memenuhi Tugas Mata Kuliah Pemrograman Berbasis Web



Disusun oleh:

Gibran Kahlil

4522210128

Dosen Pengampu:

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

**Program Studi Teknik Informatika**  
**Fakultas Teknik Universitas Pancasila**

**2023/2024**

**Link GitHub:** <https://github.com/gibrankahlil260404/PBW>

**Pertanyaan:**

**A. Apa itu AJAX:**

AJAX (Asynchronous JavaScript and XML) adalah teknik pengembangan web yang memungkinkan aplikasi web berkomunikasi dengan server secara asinkron tanpa harus memuat ulang halaman web secara keseluruhan. Teknik ini meningkatkan interaktivitas dan pengalaman pengguna dengan memperbarui hanya bagian tertentu halaman, membuat aplikasi web terasa lebih cepat dan responsif. Komponen utama AJAX antara lain: JavaScript, XMLHttpRequest, XML/JSON, HTML dan CSS

**B. Apa itu JSON? Apa kegunaannya?:**

JSON, atau Notifikasi Objek JavaScript, adalah format data kecil yang digunakan untuk pertukaran data yang mudah dibaca dan ditulis oleh manusia dan mudah diuraikan dan dibuat oleh mesin. JSON terdiri dari array (kumpulan nilai) dan objek (kumpulan pasangan kunci-nilai). Karena mudah diolah dan efisien, JSON banyak digunakan dalam pengembangan web untuk mengirim dan menerima data antara server dan klien. Selain itu, JSON sering digunakan dalam API web, file konfigurasi, dan penyimpanan data di beberapa basis data NoSQL. Ini karena formatnya yang sederhana dan kompatibilitasnya yang luas dengan berbagai bahasa pemrograman, menjadikannya standar untuk pertukaran data di aplikasi modern.

**C. Apa itu ParseMultipartForm dan MultipartReader ? Jelaskan perbedaannya!**

- Fungsi ParseMultipartForm dalam bahasa pemrograman Go (Golang) digunakan untuk mem-parsing dan mengekstrak data dari permintaan HTTP yang memiliki tipe konten multipart/form-data. Fungsi ini memungkinkan aplikasi Go untuk mengakses data formulir, termasuk nilai input dan file yang diunggah.
- Sebaliknya, MultipartReader juga dapat digunakan dalam Go untuk menangani permintaan HTTP multipart/form-data. Namun, dalam metode ini, Anda menggunakan objek MultipartReader untuk membaca dan mem-parsing bagian-bagian permintaan secara manual. Setelah menginisialisasi MultipartReader dengan permintaan HTTP, Anda kemudian membaca setiap bagian permintaan secara terpisah, dan kemudian Anda mengekstrak data form-data multipart.
- Perbedaannya terletak pada tingkat kontrol yang Anda miliki dalam pengolahan data multipart. Dengan ParseMultipartForm, Go secara otomatis menangani proses parsing data multipart, sementara dengan MultipartReader, Anda memiliki kontrol langsung atas membaca dan mem-parsing setiap bagian yang diminta. ParseMultipartForm lebih sederhana dan cocok untuk kebanyakan kasus penggunaan biasa, sementara MultipartReader memberikan fleksibilitas lebih besar dalam pemrosesan.

D. Apa itu JavaScript? Jelaskan kegunaannya dalam frontend !

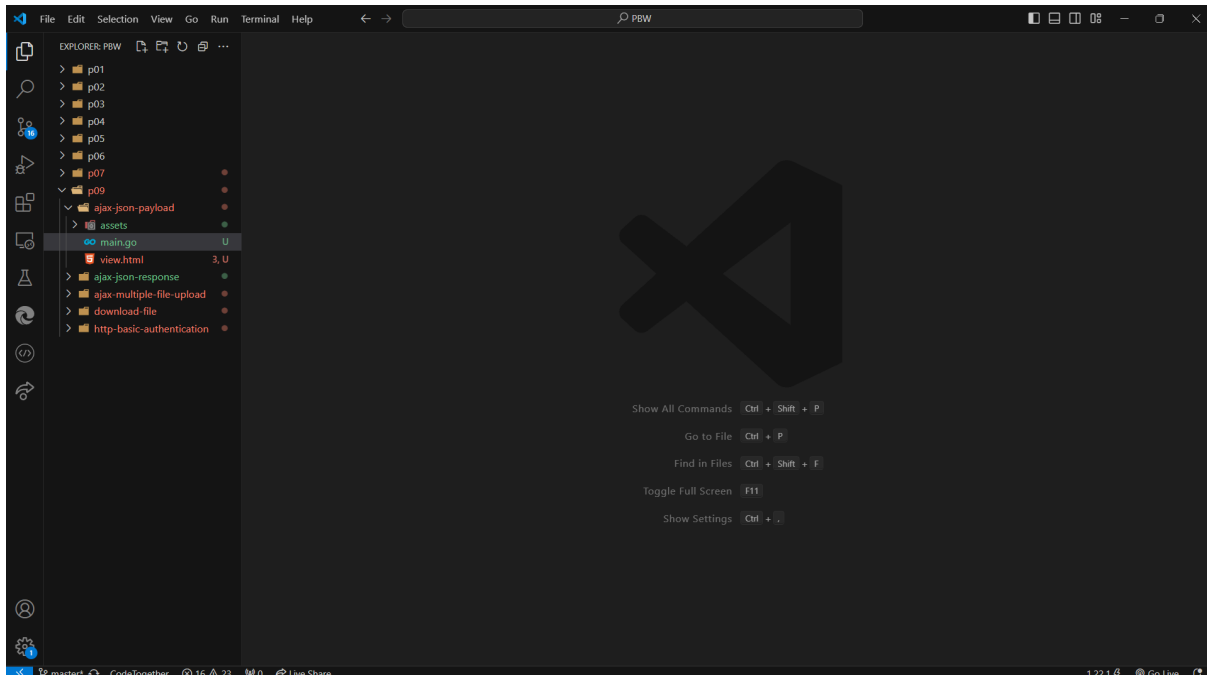
JavaScript adalah bahasa pemrograman yang sangat penting untuk pengembangan frontend karena memungkinkan Anda membuat pengalaman pengguna yang interaktif, dinamis, dan responsif di web. Salah satu kegunaan utamanya adalah untuk menanggapi secara langsung tindakan pengguna, seperti mengklik tombol atau mengisi formulir, dan memperbarui halaman web secara dinamis tanpa memuat ulang. JavaScript memungkinkan pengembang mengintegrasikan API web dan mengambil dan mengirim data secara asinkron untuk meningkatkan interaktivitas aplikasi mereka. Selain itu, JavaScript memungkinkan untuk mengubah struktur HTML dan gaya CSS secara langsung melalui DOM. Ini menghasilkan efek visual yang menarik dan meningkatkan dinamisitas tata letak halaman. JavaScript menjadi fondasi yang kuat dalam pengembangan frontend karena fitur-fitur ini memungkinkan pengembang membuat situs web yang menarik, interaktif, dan ramah pengguna.

E. Jelaskan apa itu EventHandler!

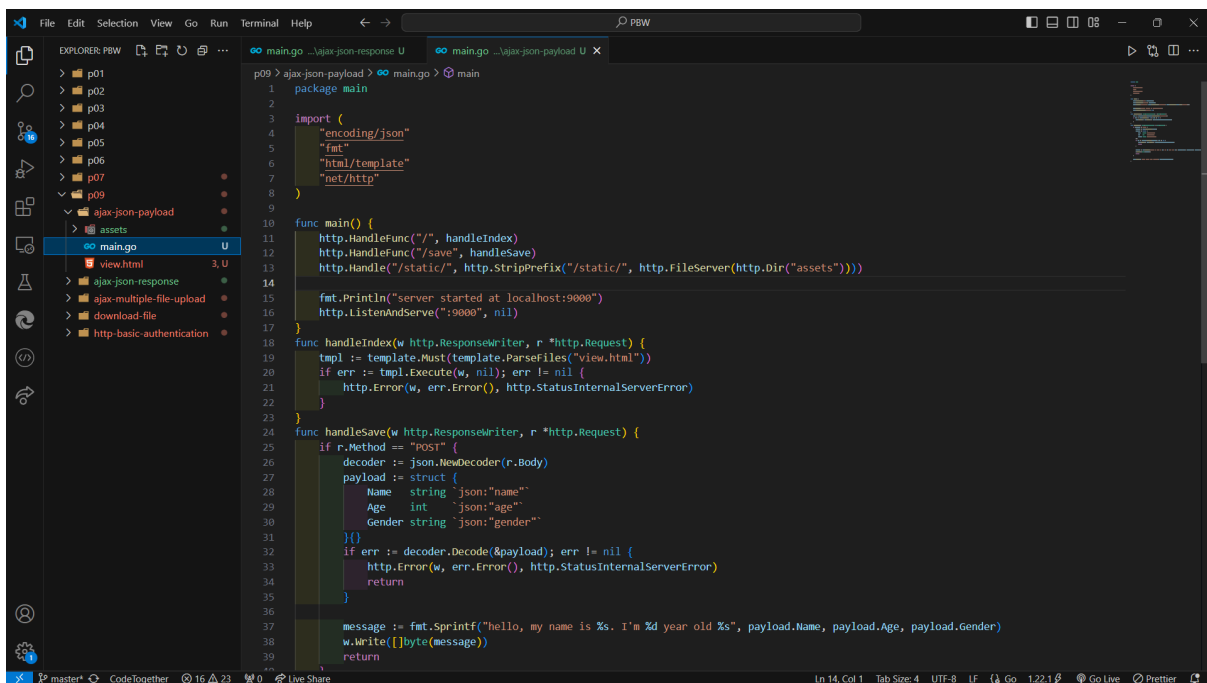
Fungsi atau mekanisme pemrograman yang dikenal sebagai Event Handler menangani dan merespons kejadian atau interaksi yang terjadi dalam aplikasi atau lingkungan perangkat lunak. Dalam pengembangan web, EventHandler biasanya merujuk pada fungsi JavaScript yang dipasang untuk menangani tindakan pengguna seperti klik mouse, penekanan tombol keyboard, atau mengubah nilai formulir. Ketika suatu kejadian terjadi, EventHandler akan dieksekusi, memungkinkan aplikasi web untuk memberikan tanggapan yang sesuai. Misalnya, ketika pengguna mengklik tombol "Submit" di formulir, EventHandler dapat diprogram untuk memvalidasi input, mengirim data ke server, dan menampilkan pesan sukses atau gagal kepada pengguna. Pengembang dapat menggunakan EventHandler untuk meningkatkan interaksi aplikasi-pengguna menjadi lebih dinamis dan responsif.

## 1. AJAX JSON Payload

- Buatlah struktur folder sebagai berikut

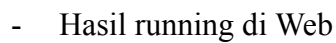
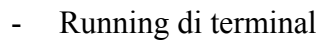


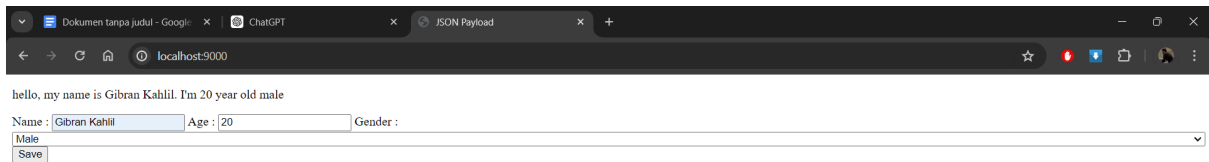
- Download file jquery  
”<https://drive.google.com/file/d/1hWXFf058m2k68gopnNKEjIWgUgUmMoR5/view?usp=sharing>”
- Source Code main.go



- Source Code View.go

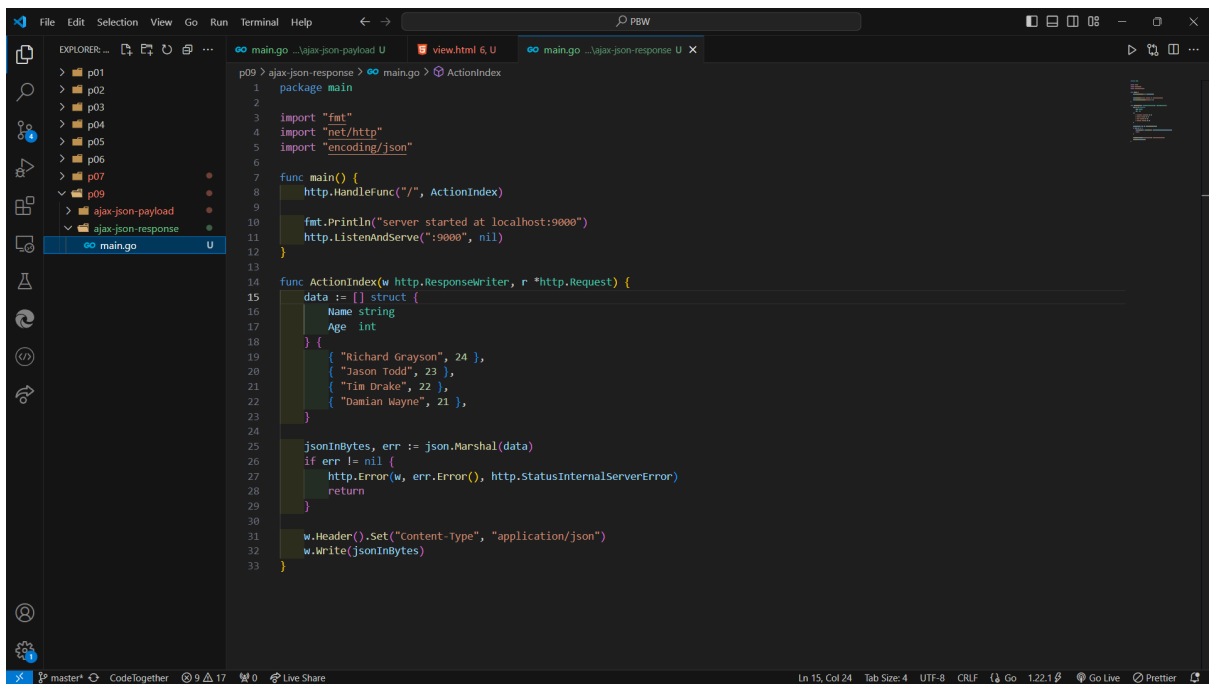






## 2. AJAX JSON Response

### - Source Code main.go



### - Hasil Running di terminal

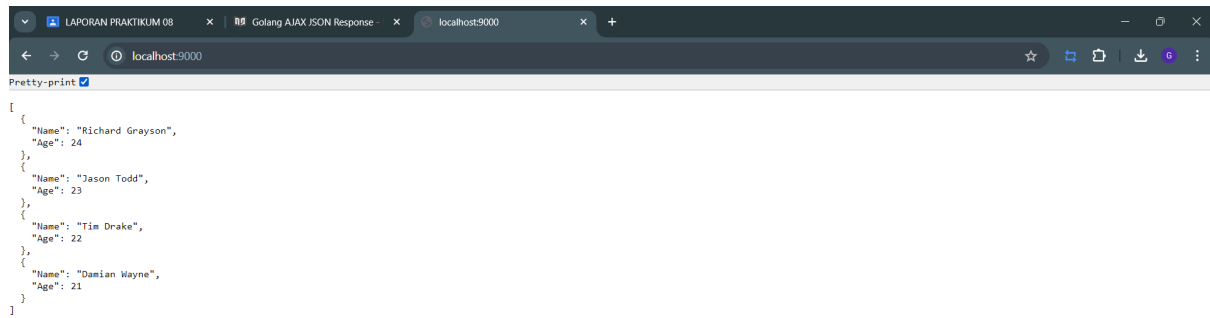
```
1 package main
2
3 import "fmt"
4 import "net/http"
5 import "encoding/json"
6
7 func main() {
8     http.HandleFunc("/", ActionIndex)
9
10    fmt.Println("server started at localhost:9000")
11    http.ListenAndServe(":9000", nil)
12 }
13
14 func ActionIndex(w http.ResponseWriter, r *http.Request) {
15     data := [] struct {
16         Name string
17         Age  int
18     } {
19         { "Richard Grayson", 24 },
20         { "Jason Todd", 23 },
21         { "Tim Drake", 22 },
22         { "Damian Wayne", 21 }
23     }
24     w.Header().Set("Content-Type", "application/json")
25     json.NewEncoder(w).Encode(data)
26 }
```

PS D:\SNT\_4\PBW\p09\ajax-json-response> go run .\main.go  
server started at localhost:9000

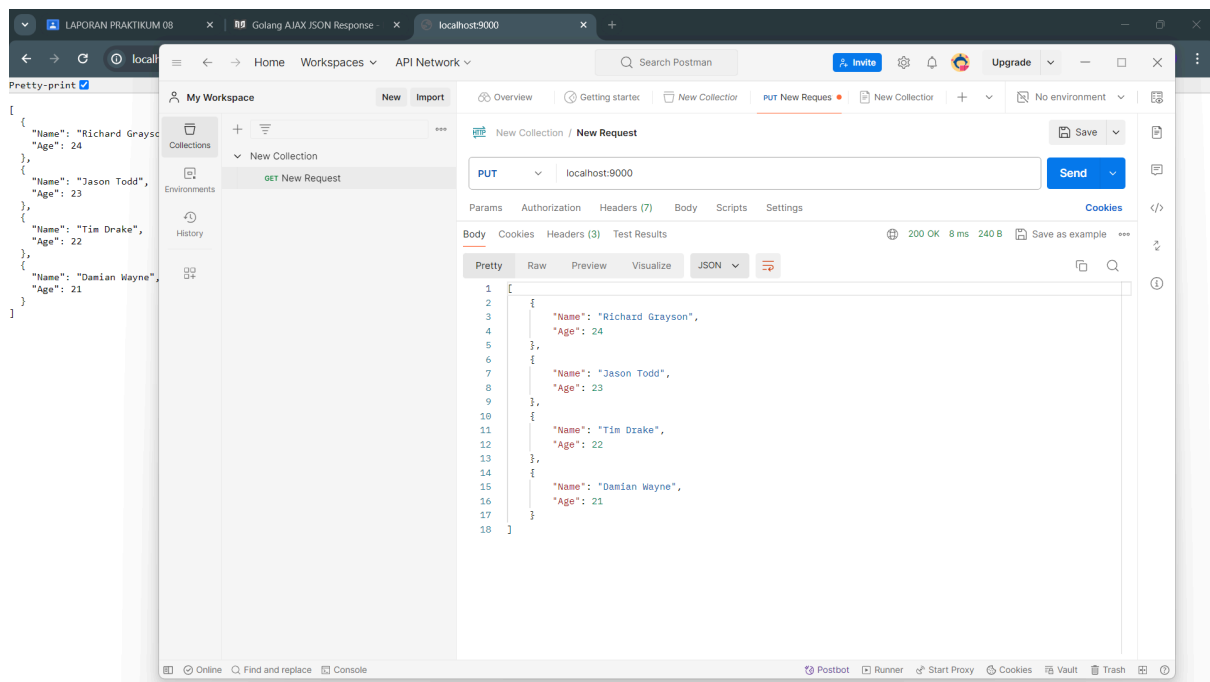
## - Hasil running di Web

```
[[{"Name": "Richard Grayson", "Age": 24}, {"Name": "Jason Todd", "Age": 23}, {"Name": "Tim Drake", "Age": 22}, {"Name": "Damian Wayne", "Age": 21}]]
```



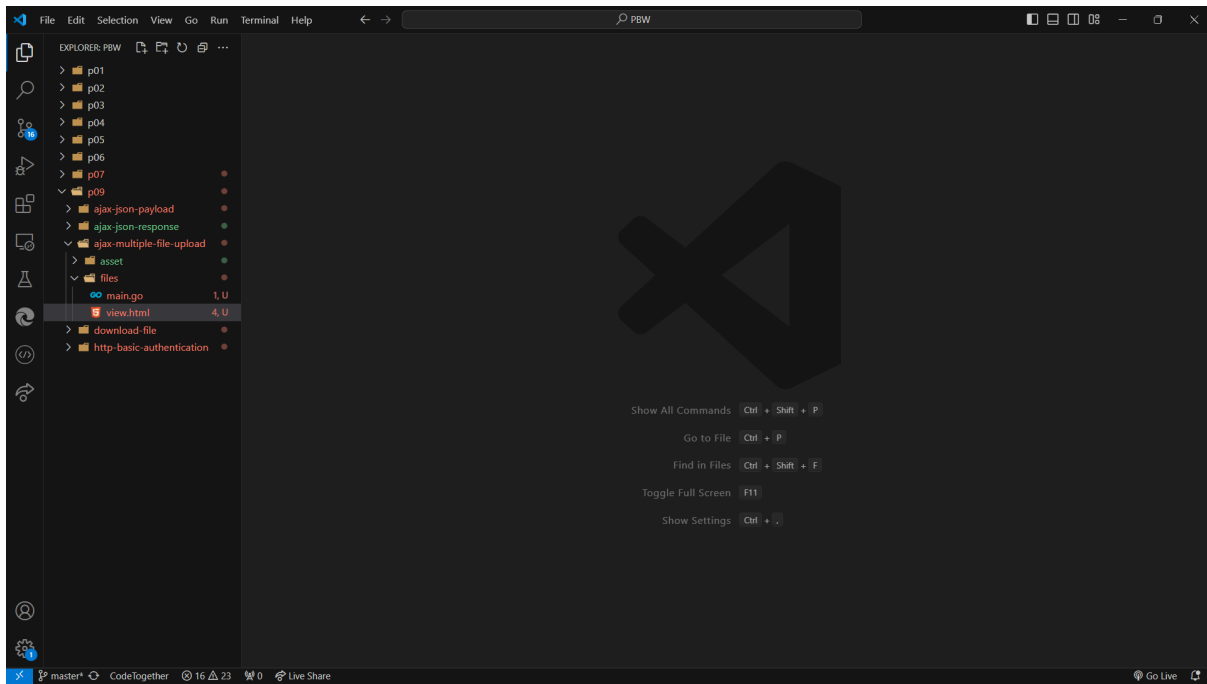


## - Lakukan test API

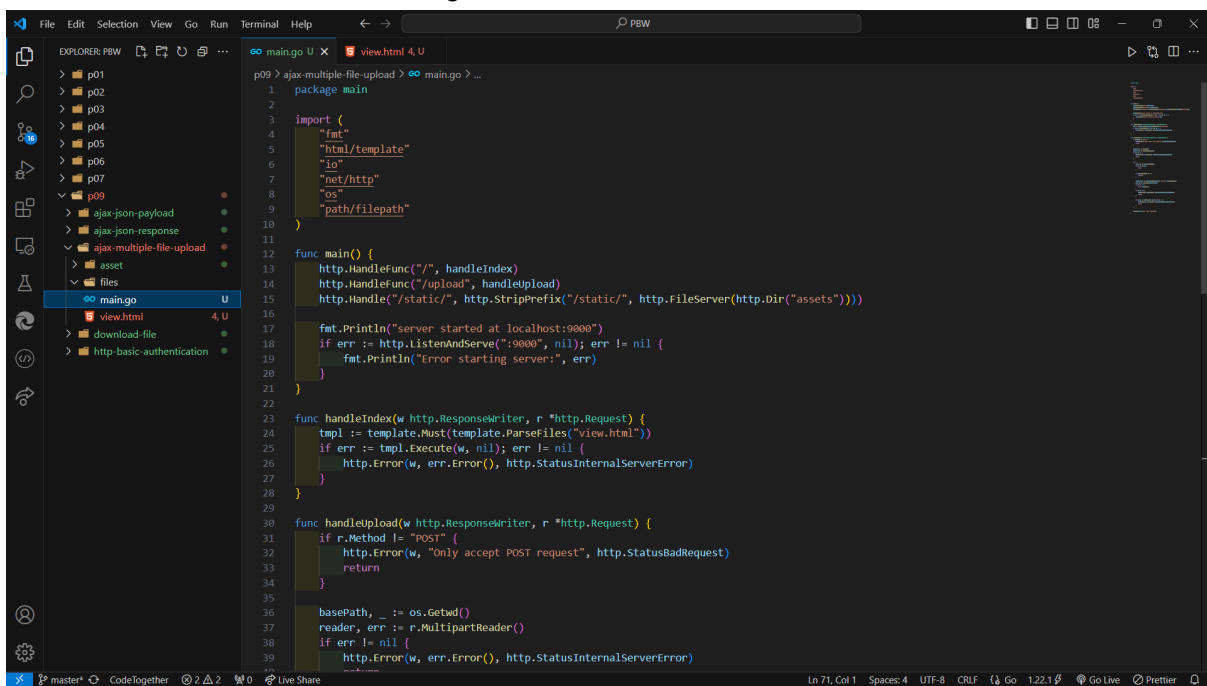


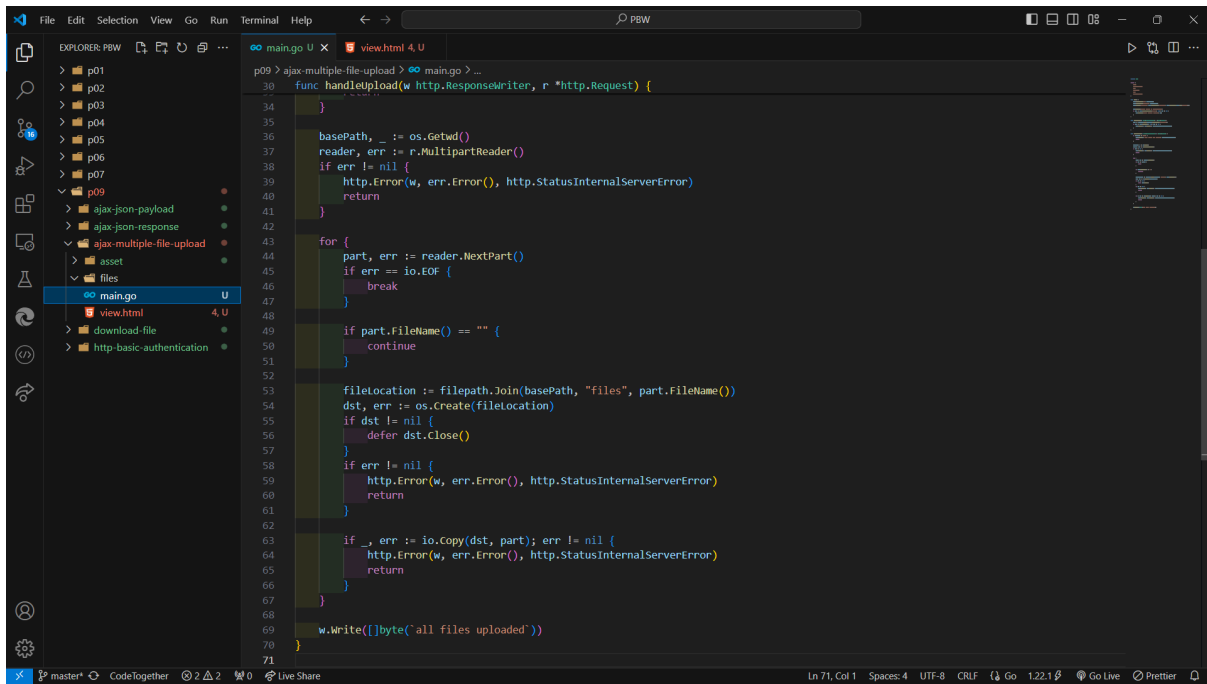
## 3. AJAX Multiple File Upload

- Buat Stuktur folder sebagai berikut



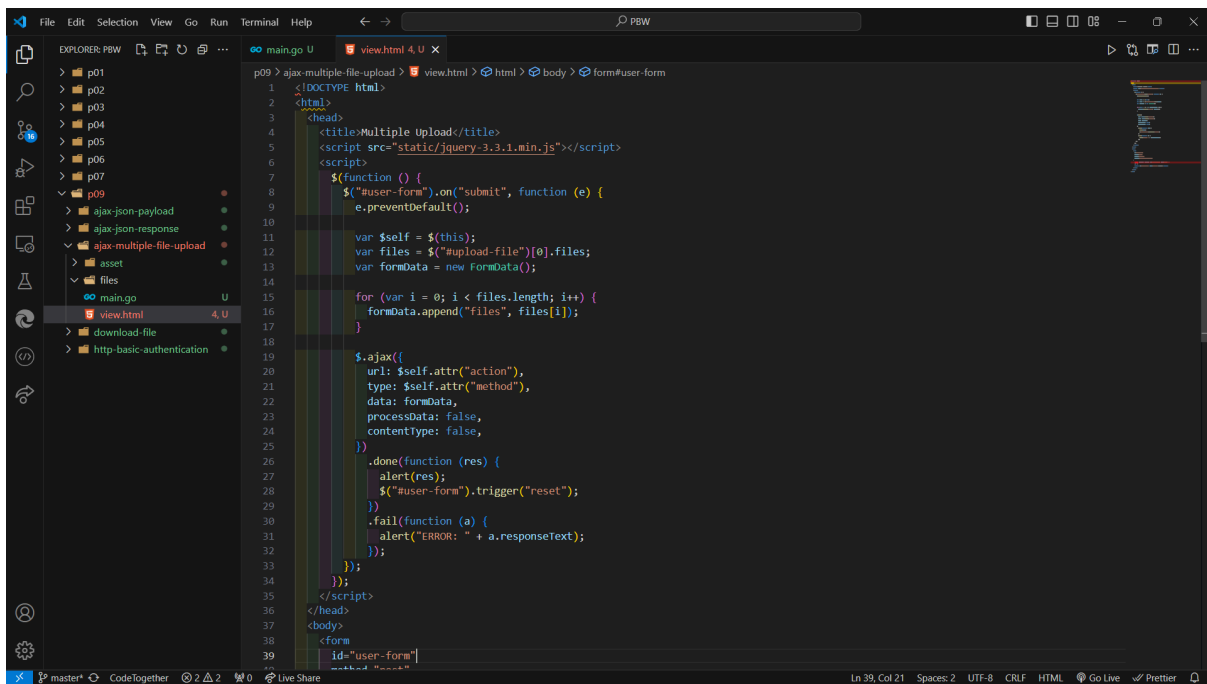
## - Source Code main.go





```
p09 > ajax-multiple-file-upload > main.go > ...
30 func handleupload(w http.ResponseWriter, r *http.Request) {
31
32 }
33
34 basePath, _ := os.Getwd()
35 reader, err := r.MultipartReader()
36 if err != nil {
37     http.Error(w, err.Error(), http.StatusInternalServerError)
38     return
39 }
40
41 for {
42     part, err := reader.NextPart()
43     if err == io.EOF {
44         break
45     }
46
47     if part.FileName() == "" {
48         continue
49     }
50
51     filelocation := filepath.Join(basePath, "files", part.FileName())
52     dst, err := os.Create(filelocation)
53     if dst != nil {
54         defer dst.Close()
55     }
56     if err != nil {
57         http.Error(w, err.Error(), http.StatusInternalServerError)
58         return
59     }
60
61     if _, err := io.Copy(dst, part); err != nil {
62         http.Error(w, err.Error(), http.StatusInternalServerError)
63         return
64     }
65 }
66
67 w.Write([]byte("all files uploaded"))
68
69 }
70
71 }
```

- Source Code view.go



```
p09 > ajax-multiple-file-upload > view.html > html > body > form#user-form
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Multiple Upload</title>
5 <script src="static/jquery-3.3.1.min.js"></script>
6 <script>
7     $(function () {
8         $("#user-form").on("submit", function (e) {
9             e.preventDefault();
10
11             var $self = $(this);
12             var files = $("#upload-file")[0].files;
13             var formData = new FormData();
14
15             for (var i = 0; i < files.length; i++) {
16                 formData.append("files", files[i]);
17             }
18
19             $.ajax({
20                 url: $self.attr("action"),
21                 type: $self.attr("method"),
22                 data: formData,
23                 processData: false,
24                 contentType: false,
25             })
26             .done(function (res) {
27                 alert(res);
28                 $("#user-form").trigger("reset");
29             })
30             .fail(function (a) {
31                 alert("ERROR: " + a.responseText);
32             });
33         });
34     });
35 </script>
36 </head>
37 <body>
38 <form
39     id="user-form"
40     method="POST">
```

The screenshot shows the Visual Studio Code editor with a project named 'ajax-multiple-file-upload'. The Explorer sidebar on the left shows the file structure, including folders for 'p01' through 'p09', 'asset', 'files', 'main.go', 'download-file', and 'http-basic-authentication'. The 'view.html' file is selected and open in the editor. The code in 'view.html' is as follows:

```
1 <html>
2 <head>
3   <script>
4     $(function () {
5       $('#user-form').on('submit', function (e) {
6         $.ajax({
7           url: $('#user-form').attr('action'),
8           type: $('#user-form').attr('method'),
9           data: $('#user-form').serialize(),
10          processData: false,
11          contentType: false,
12        })
13        .done(function (res) {
14          alert(res);
15          $('#user-form').trigger('reset');
16        })
17        .fail(function (a) {
18          alert("ERROR: " + a.responseText);
19        });
20      });
21    });
22  </script>
23 </head>
24 <body>
25   <form
26     id="user-form"
27     method="post"
28     action="/upload"
29     enctype="multipart/form-data"
30   >
31     <input required multiple id="upload-file" type="file" />
32     <br />
33     <button id="btn-upload" type="submit">Upload!</button>
34   </form>
35 </body>
36 </html>
```

## - Running di terminal

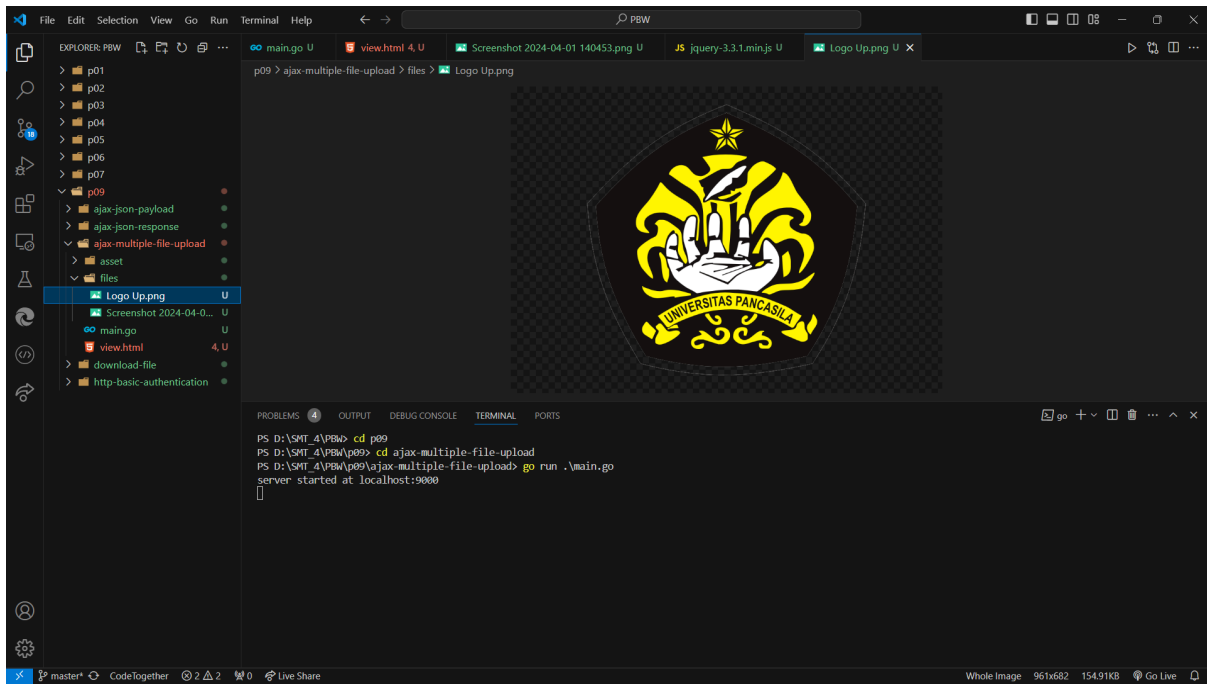
The screenshot shows the Visual Studio Code editor with the same project. The terminal window at the bottom is active, showing the following commands and output:

```
PS D:\SMT_4\PBW> cd p09
PS D:\SMT_4\PBW\p09> cd ajax-multiple-file-upload
PS D:\SMT_4\PBW\p09\ajax-multiple-file-upload> go run .\main.go
server started at localhost:9000
```

## - Hasil Running di Web

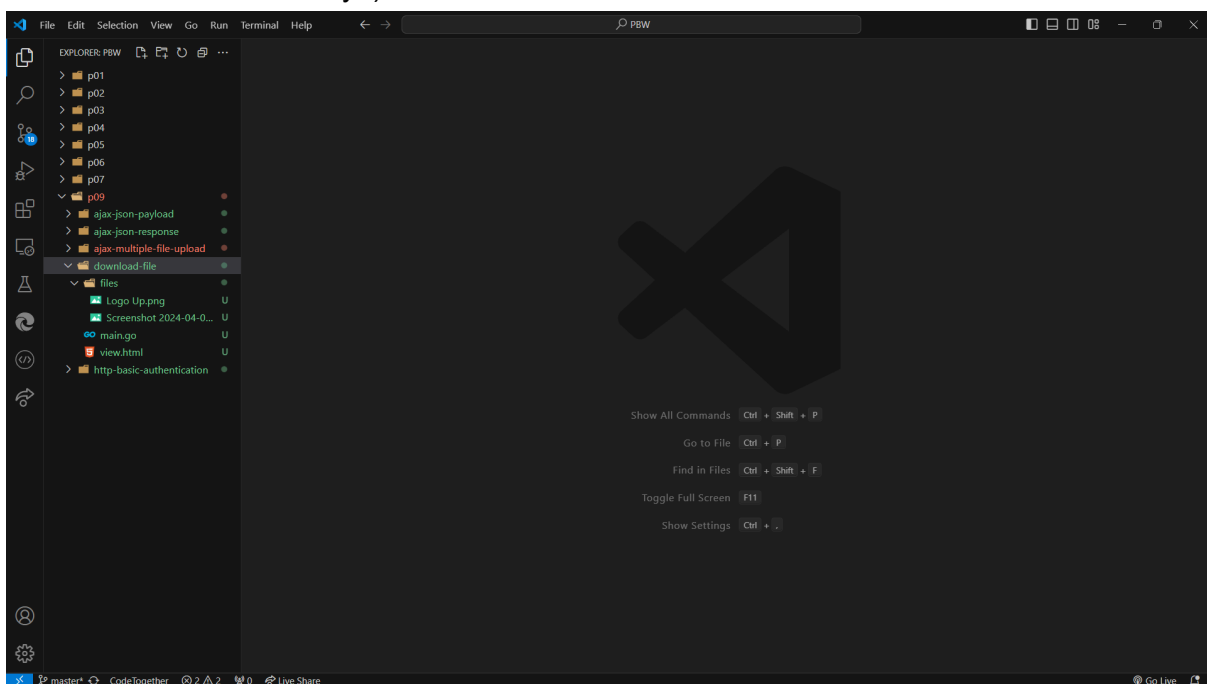


- 
- Hasil upload di folder files



#### 4. Download Files

- Buat Struktur sebagai berikut: (note isi file bebas untuk didownload di dalam Web nantinya)



- Source Code main.go

```
1 package main
2
3 import "fmt"
4 import "net/http"
5 import "html/template"
6 import "path/filepath"
7 import "io"
8 import "encoding/json"
9 import "os"
10
11 type M map[string]interface{}
12
13 func main() {
14     http.HandleFunc("/", handleIndex)
15     http.HandleFunc("/list-files", handleListFiles)
16     http.HandleFunc("/download", handleDownload)
17
18     fmt.Println("server started at localhost:9000")
19     http.ListenAndServe(":9000", nil)
20 }
21
22 func handleIndex(w http.ResponseWriter, r *http.Request) {
23     tmpl := template.Must(template.ParseFiles("view.html"))
24     if err := tmpl.Execute(w, nil); err != nil {
25         http.Error(w, err.Error(), http.StatusInternalServerError)
26     }
27 }
28
29 func handleListFiles(w http.ResponseWriter, r *http.Request) {
30     files := []M{}
31     basePath, _ := os.Getwd()
32     filesLocation := filepath.Join(basePath, "files")
33
34     err := filepath.Walk(filesLocation, func(path string, info os.FileInfo, err error) error {
35         if err != nil {
36             return err
37         }
38
39         if info.IsDir() {
40             return nil
41         }
42
43         files = append(files, M{"filename": info.Name(), "path": path})
44     })
45
46     if err != nil {
47         http.Error(w, err.Error(), http.StatusInternalServerError)
48         return
49     }
50
51     res, err := json.Marshal(files)
52     if err != nil {
53         http.Error(w, err.Error(), http.StatusInternalServerError)
54         return
55     }
56
57     w.Header().Set("Content-type", "application/json")
58     w.Write(res)
59 }
60
61 func handleDownload(w http.ResponseWriter, r *http.Request) {
62     if err := r.ParseForm(); err != nil {
63         http.Error(w, err.Error(), http.StatusInternalServerError)
64         return
65     }
66
67     filename := r.FormValue("filename")
68     path := filepath.Join(basePath, "files", filename)
69
70     data, err := os.ReadFile(path)
71     if err != nil {
72         http.Error(w, err.Error(), http.StatusInternalServerError)
73         return
74     }
75
76     w.Write(data)
77 }
```

```
22 func handleIndex(w http.ResponseWriter, r *http.Request) {
23     tmpl := template.Must(template.ParseFiles("view.html"))
24     if err := tmpl.Execute(w, nil); err != nil {
25         http.Error(w, err.Error(), http.StatusInternalServerError)
26     }
27 }
28
29 func handleListFiles(w http.ResponseWriter, r *http.Request) {
30     files := []M{}
31     basePath, _ := os.Getwd()
32     filesLocation := filepath.Join(basePath, "files")
33
34     err := filepath.Walk(filesLocation, func(path string, info os.FileInfo, err error) error {
35         if err != nil {
36             return err
37         }
38
39         if info.IsDir() {
40             return nil
41         }
42
43         files = append(files, M{"filename": info.Name(), "path": path})
44     })
45
46     if err != nil {
47         http.Error(w, err.Error(), http.StatusInternalServerError)
48         return
49     }
50
51     res, err := json.Marshal(files)
52     if err != nil {
53         http.Error(w, err.Error(), http.StatusInternalServerError)
54         return
55     }
56
57     w.Header().Set("Content-type", "application/json")
58     w.Write(res)
59 }
60
61 func handleDownload(w http.ResponseWriter, r *http.Request) {
62     if err := r.ParseForm(); err != nil {
63         http.Error(w, err.Error(), http.StatusInternalServerError)
64         return
65     }
66
67     filename := r.FormValue("filename")
68     path := filepath.Join(basePath, "files", filename)
69
70     data, err := os.ReadFile(path)
71     if err != nil {
72         http.Error(w, err.Error(), http.StatusInternalServerError)
73         return
74     }
75
76     w.Write(data)
77 }
```

```
29 func handleListFiles(w http.ResponseWriter, r *http.Request) {
47     if err != nil {
48         http.Error(w, err.Error(), http.StatusInternalServerError)
49         return
50     }
51
52     res, err := json.Marshal(files)
53     if err != nil {
54         http.Error(w, err.Error(), http.StatusInternalServerError)
55         return
56     }
57
58     w.Header().Set("Content-Type", "application/json")
59     w.Write(res)
60 }
61
62 func handleDownload(w http.ResponseWriter, r *http.Request) {
63     if err := r.ParseForm(); err != nil {
64         http.Error(w, err.Error(), http.StatusInternalServerError)
65         return
66     }
67
68     path := r.FormValue("path")
69     f, err := os.Open(path)
70     if f != nil {
71         defer f.Close()
72     }
73     if err != nil {
74         http.Error(w, err.Error(), http.StatusInternalServerError)
75         return
76     }
77
78     contentDisposition := fmt.Sprintf("attachment; filename=%s", f.Name())
79     w.Header().Set("Content-Disposition", contentDisposition)
80
81     if _, err := io.Copy(w, f); err != nil {
82         http.Error(w, err.Error(), http.StatusInternalServerError)
83         return
84     }
}
```

## - Source Code view.go

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title>Download file</title>
5 <script>
6     function Yo() {
7         var self = this;
8         var $ul = document.getElementById("list-files");
9
10        var renderData = function (res) {
11            res.forEach(function (each) {
12                var $li = document.createElement("li");
13                var $a = document.createElement("a");
14
15                $li.innerText = "download ";
16                $li.appendChild($a);
17                $ul.appendChild($li);
18
19                $a.href = "/download?path=" + encodeURIComponent(each.path);
20                $a.innerText = each.filename;
21                $a.target = "_blank";
22            });
23        };
24
25        var getAllListFiles = function () {
26            var xhr = new XMLHttpRequest();
27            xhr.open("GET", "/list-files");
28            xhr.onreadystatechange = function () {
29                if (xhr.readyState == 4 && xhr.status == 200) {
30                    var json = JSON.parse(xhr.responseText);
31                    renderData(json);
32                }
33            };
34            xhr.send();
35        };
36
37        self.init = function () {
38            getAllListFiles();
39        };
40    };
}
```

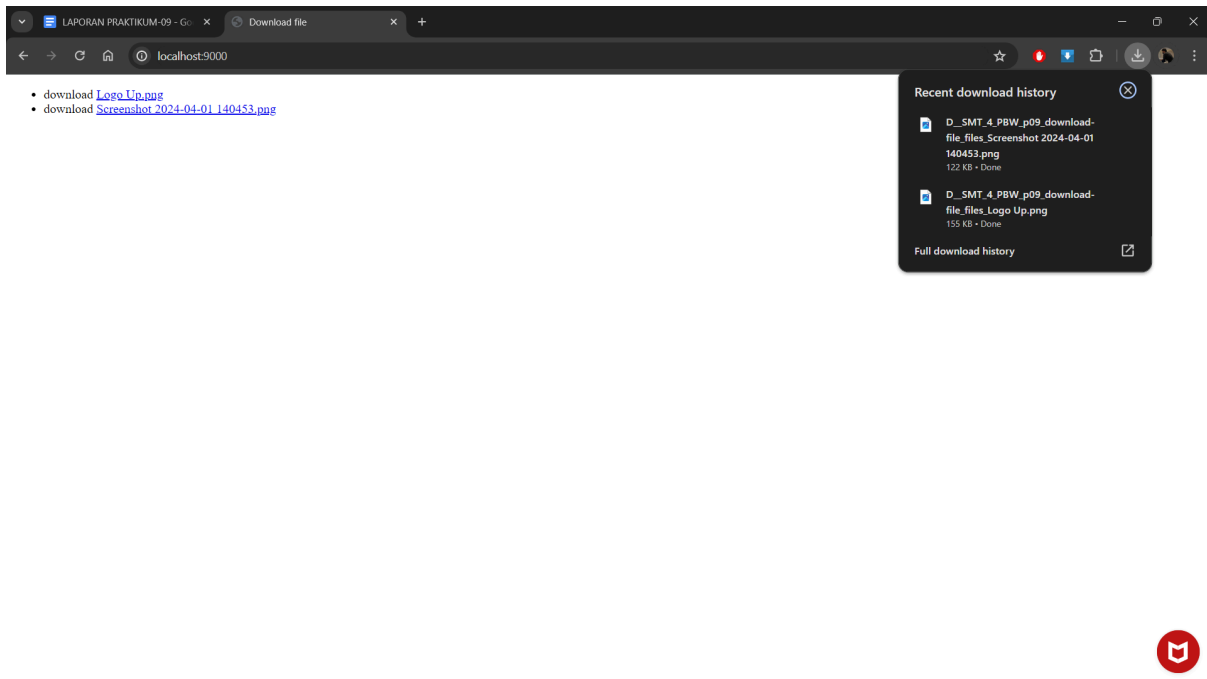


```
p09 > download-file > view.html > html > head > script > Yo > renderData > res.forEach() callback
2 <html>
3 <head>
4 <script>
5   function Yo() {
6     var renderData = function (res) {
7       // ...
8     };
9   };
10
11   var getAllListFiles = function () {
12     var xhr = new XMLHttpRequest();
13     xhr.open("GET", "/list-files");
14     xhr.onreadystatechange = function () {
15       if (xhr.readyState == 4 && xhr.status == 200) {
16         var json = JSON.parse(xhr.responseText);
17         renderData(json);
18       }
19     };
20     xhr.send();
21   };
22
23   self.init = function () {
24     getAllListFiles();
25   };
26
27   window.onload = function () {
28     new Yo().init();
29   };
30 </script>
31 </head>
32 <body>
33 <ul id="list-files"></ul>
34 </body>
35 </html>
```

- Running di terminal

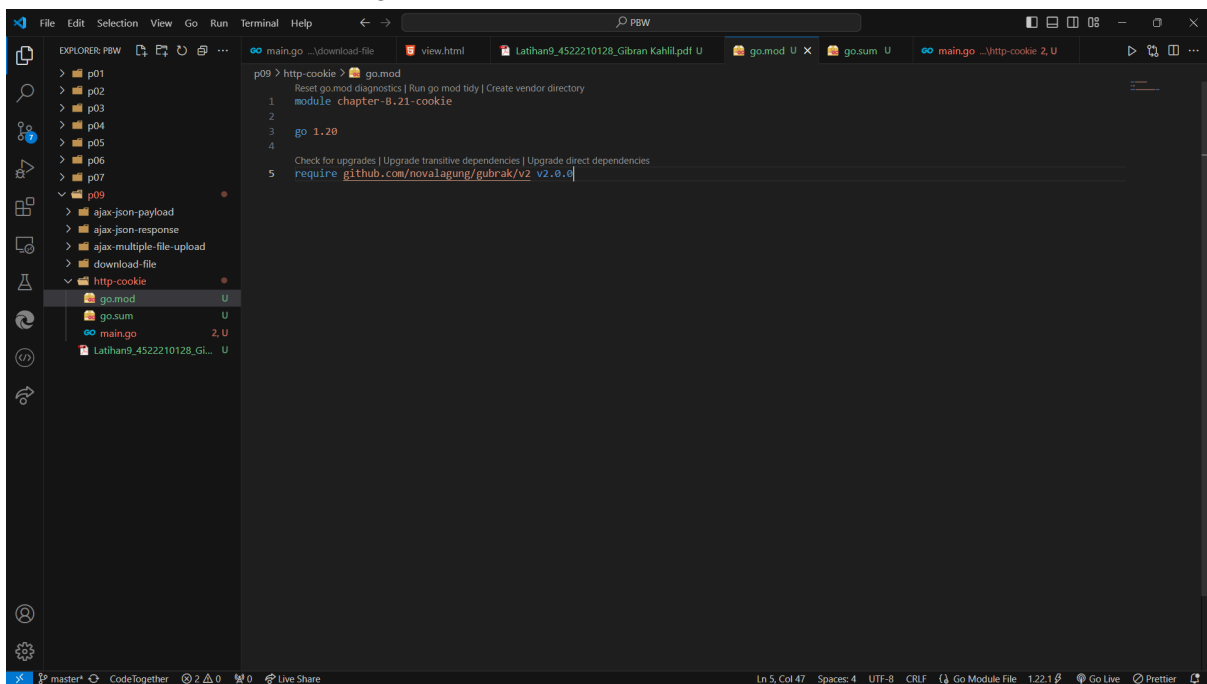
```
PS D:\SWT_4\VBW\p09> cd download-file
PS D:\SWT_4\VBW\p09\download-file> go run .\main.go
server started at localhost:9000
```

- Hasil running di Web dan download file



## 5. http-cookie

- Buatlah 3 file dengan nama “main.go, go.sum, go.mod”
- Source Code go.mod



- Source Code go.sum

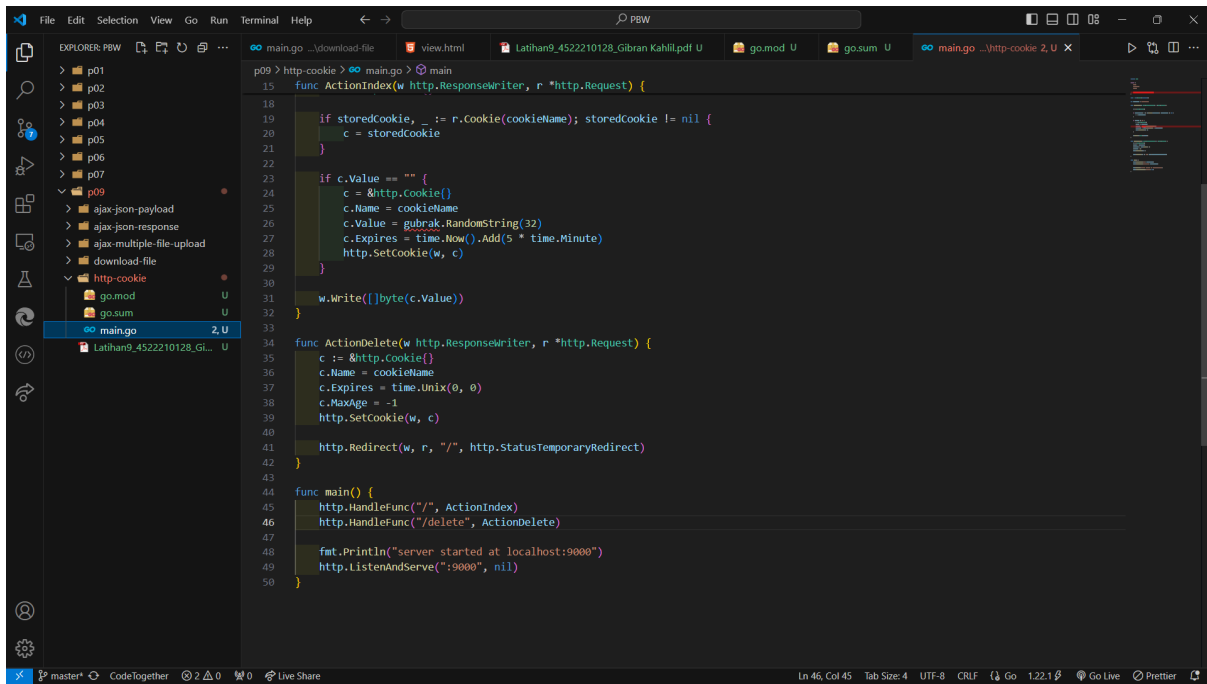
The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like p01, p02, p03, p04, p05, p06, p07, p09, ajax-json-payload, ajax-json-response, ajax-multiple-file-upload, download-file, http-cookie, go.mod, go.sum, main.go, and Latihan9\_4522210128\_GI... The code editor shows the content of main.go, which is a list of cookies. The cookies are listed in a table format with columns for the cookie name, value, and domain. The cookies are:

Cookie Name	Value	Domain
github.com/definitelyTyped/gocsv	v0.0.0-20181205141819-acfa5f112b45	h1:009vawoB08Hk84zWokunBCSEeAb88VAHPUMg+UE=
github.com/definitelyTyped/gocsv	v0.0.0-20181205141819-acfa5f112b45	go.mod h1:1rAb8au591C1KUSRA1h1ndIOQY1N0BrtE1Plqht4=
github.com/davecgh/go-spew	v1.1.0	h1:ZDRjV015cmhc3fIQnie+owkZQ0ADaZdg9rXUJ11z8=
github.com/davecgh/go-spew	v1.1.0	go.mod h1:7Y9Vcm2NhsgmW/mv3lAw1/skM41LHj5SI+c5H38=
github.com/novalagung/gubrak/v2	v2.0.0	h1:7wq2X242cLUTHyVw3ASivRSf1w08h3m0XPFULQMA=
github.com/novalagung/gubrak/v2	v2.0.0	go.mod h1:zi111H12P28SD27527Mq14hg3352UTFR34tUXopA0=
github.com/pmezard/go-difflib	v1.0.0	h1:4DBwDE0NGyQ8HBLQYPvSUpOCMWSBEzIK/f112BAQM=
github.com/pmezard/go-difflib	v1.0.0	go.mod h1:IKH77koFHYXTK1pcRnkKqfTogsbg7g2NNV4sRDV2/4=
github.com/stretchr/objx	v0.1.0	go.mod h1:HFkY9161F+rwDfMAKV70tWuqBVzrE8GR6GFX+WEKME=
github.com/stretchr/testify	v1.4.0	h1:2E45XV/wtOKTonXsotY41iezWxvY1ZuVNCe9XR3yk=
github.com/stretchr/testify	v1.4.0	go.mod h1:77eGeouHqKXV5pUUK4zz7dfj8NFuz+81PSLYec5m4=
gopkg.in/check.v1	v0.0.0-20161208181325-20d25e288405	go.mod h1:Co61bVJAzNAaIkop8shuTw13QC7016jof/cbM4VW5Yz0=
gopkg.in/yaml.v2	v2.2.2	h1:ZCjp+Egi0T71HqUv23862kp8Qj64J6aaz82+3Td9dZw=
gopkg.in/yaml.v2	v2.2.2	go.mod h1:h193XBmqTisBFMUT0bBFm-jr3Dg1Nkqmp+5A1VGuI=

## - Source Code main.go

The screenshot shows a VS Code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like p01, p02, p03, p04, p05, p06, p07, p09, ajax-json-payload, ajax-json-response, ajax-multiple-file-upload, download-file, http-cookie, go.mod, go.sum, main.go, and Latihan9\_4522210128\_GI... The code editor shows the source code of main.go, which is a Go program that handles cookies. The code is as follows:

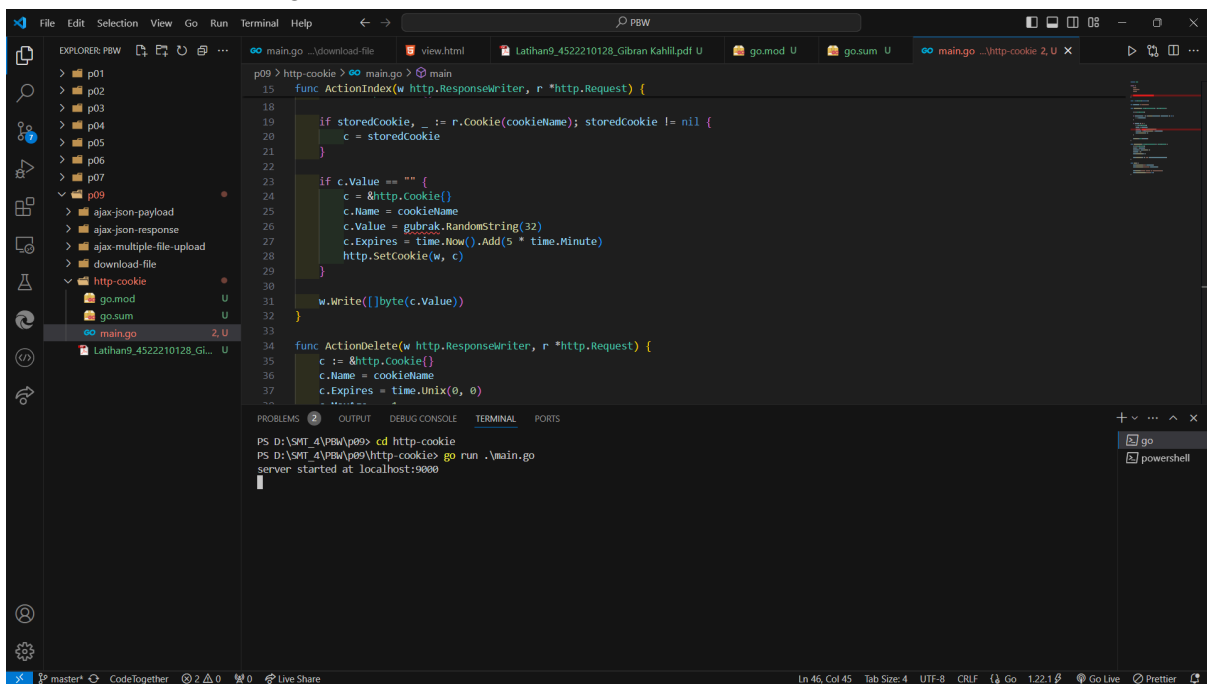
```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6     "time"
7
8     "github.com/novalagung/gubrak/v2"
9 )
10
11 type M map[string]interface{}
12
13 var cookieName = "cookieData"
14
15 func ActionIndex(w http.ResponseWriter, r *http.Request) {
16
17     c := &http.Cookie{}
18
19     if storedCookie, _ := r.Cookie(cookieName); storedCookie != nil {
20         c = storedCookie
21     }
22
23     if c.Value == "" {
24         c = &http.Cookie{}
25         c.Name = cookieName
26         c.Value = gubrak.RandomString(32)
27         c.Expires = time.Now().Add(5 * time.Minute)
28         http.SetCookie(w, c)
29     }
30
31     w.Write([]byte(c.Value))
32 }
33
34 func ActionDelete(w http.ResponseWriter, r *http.Request) {
35     c := &http.Cookie{}
36     c.Name = cookieName
37     c.Expires = time.Unix(0, 0)
38     c.MaxAge = -1
39     http.SetCookie(w, c)
40 }
```



The screenshot shows the VS Code editor with a Go project. The Explorer sidebar on the left shows a file tree with folders p01 through p09, and files go.mod, go.sum, and main.go. The main.go file is open in the editor, showing the following code:

```
15 func ActionIndex(w http.ResponseWriter, r *http.Request) {
18     if storedCookie, _ := r.Cookie(cookieName); storedCookie != nil {
19         c = storedCookie
20     }
21
22     if c.Value == "" {
23         c = &http.Cookie{
24             c.Name = cookieName
25             c.Value = gubrak.RandomString(32)
26             c.Expires = time.Now().Add(5 * time.Minute)
27             http.SetCookie(w, c)
28         }
29     }
30     w.Write([]byte(c.Value))
31 }
32
33 func ActionDelete(w http.ResponseWriter, r *http.Request) {
34     c := &http.Cookie{
35         c.Name = cookieName
36         c.Expires = time.Unix(0, 0)
37         c.MaxAge = -1
38         http.SetCookie(w, c)
39     }
40     http.Redirect(w, r, "/", http.StatusTemporaryRedirect)
41 }
42
43 func main() {
44     http.HandleFunc("/", ActionIndex)
45     http.HandleFunc("/delete", ActionDelete)
46
47     fmt.Println("server started at localhost:9000")
48     http.ListenAndServe(":9000", nil)
49 }
50 }
```

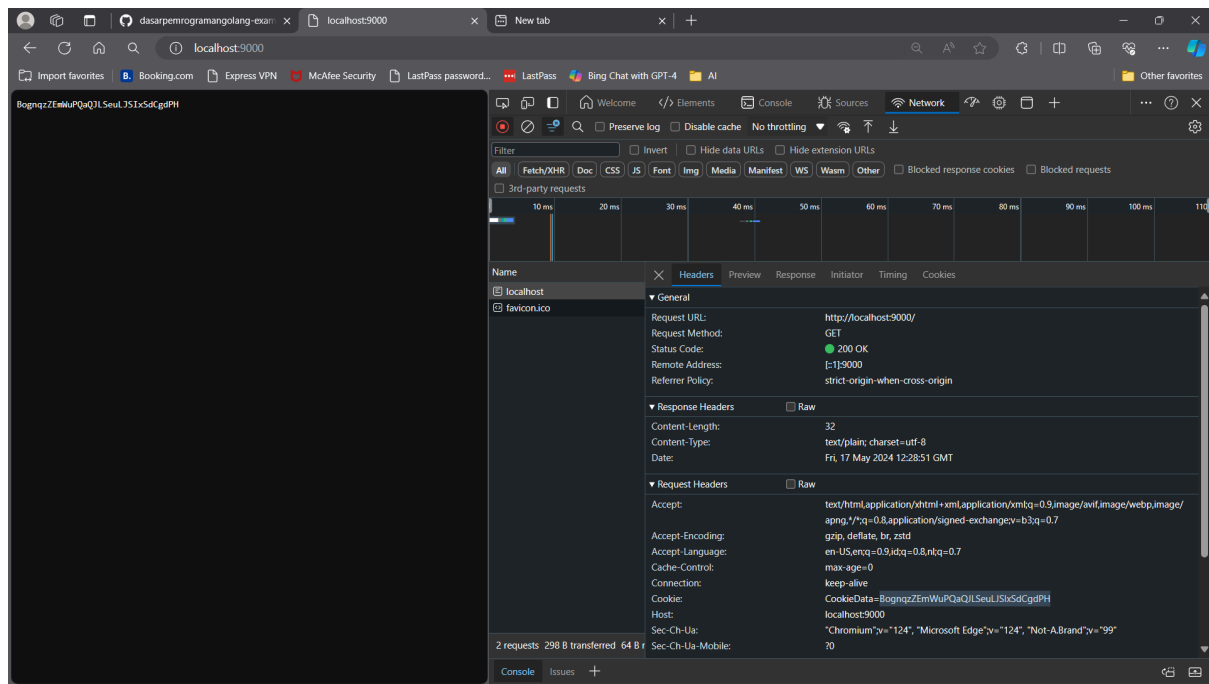
## - Running di terminal



The screenshot shows the VS Code editor with the same Go project. The main.go file is open in the editor, showing the same code as the previous screenshot. The terminal window at the bottom shows the following output:

```
PS D:\SMT_4\VBW\p09> cd http-cookie
PS D:\SMT_4\VBW\p09\http-cookie> go run .\main.go
server started at localhost:9000
```

## - Running di Web



### Kesimpulan:

Teknologi web seperti AJAX, JavaScript, JSON, dan fitur seperti ParseMultipartForm dan MultipartReader sangat penting untuk pengembangan web kontemporer. AJAX memungkinkan klien dan server berinteraksi secara asinkron, sementara JavaScript memungkinkan situs web menjadi interaktif dan dinamis. JSON digunakan untuk pertukaran data yang mudah dan mudah dipahami, sementara fitur seperti ParseMultipartForm dan MultipartReader membantu menangani data berbagai bagian pada server. Secara keseluruhan, teknologi ini memungkinkan pengembang membuat aplikasi web yang responsif, dinamis, dan ramah pengguna.