

**LAPORAN LATIHAN PRAKTIKUM KE-10**  
**PEMROGRAMAN BERBASIS WEB**  
**KELAS A**

Disusun Untuk Memenuhi Tugas Mata Kuliah Prak. Pemrograman Berbasis Web



Disusun oleh:

Arsya Yan Duribta                            4522210117

Dosen Pengampu:

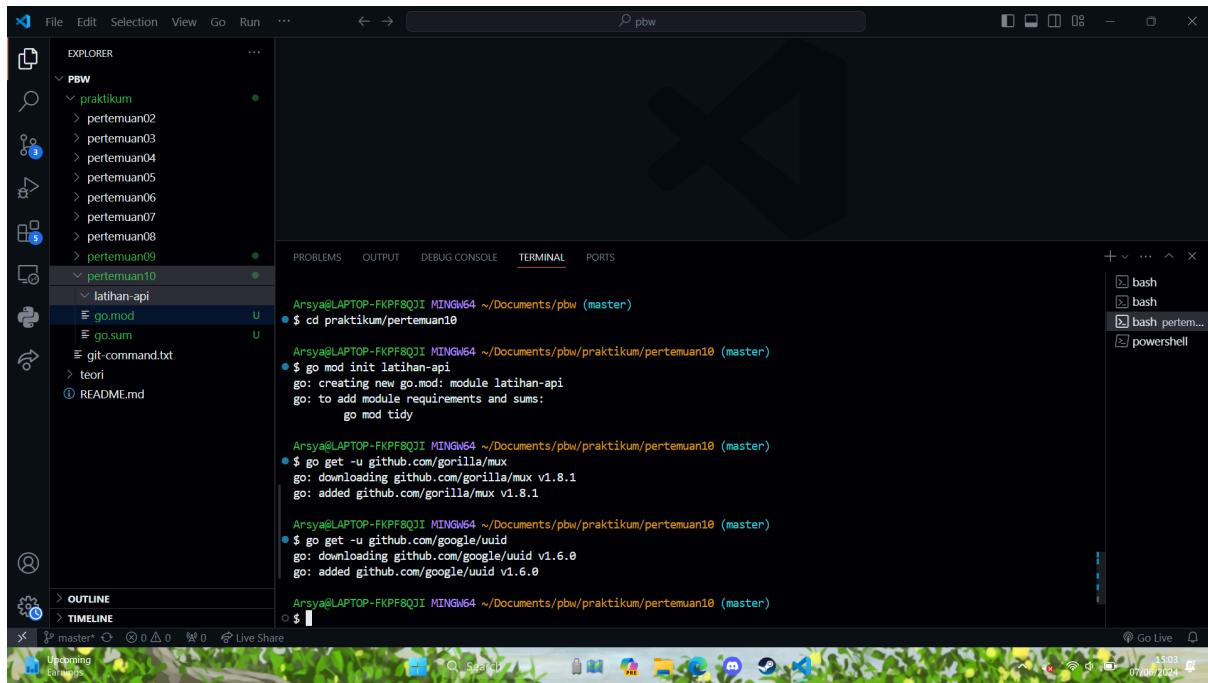
Adi Wahyu Pribadi, S.Si., M.Kom.

**Program Studi Teknik Informatika**  
**Fakultas Teknik Universitas Pancasila**  
**2023/2024**

## Link Repository Github:

<https://github.com/Arsyayd11/pbw/tree/master/praktikum/pertemuan10>

## Latihan API:



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left displays a file tree for a project named 'PBW' containing several 'pertemuan' sub-folders and a 'latihan-api' folder which contains 'go.mod', 'go.sum', 'git command.txt', 'teori', and 'README.md'. The Terminal tab is active at the bottom, showing a series of commands being run in a MinGW64 terminal window. The commands include navigating to the project directory, running 'go mod init latihan-api', adding module requirements with 'go mod tidy', and fetching dependencies from GitHub with 'go get -u github.com/gorilla/mux' and 'go get -u github.com/google/uuid'. The output of these commands is visible in the terminal window.

## Pertanyaan: JELASKAN KEGUNAAN 2 LIBRARY DIATAS!!

### 1. [github.com/gorilla/mux](https://github.com/gorilla/mux)

Library mux dari Gorilla merupakan sebuah HTTP request router dan dispatcher untuk bahasa pemrograman Go. Kegunaan utamanya adalah untuk membantu dalam menangani routing di aplikasi web. Dengan mux, kita bisa menentukan pola URL yang lebih kompleks dan mengarahkan permintaan HTTP ke handler yang tepat. Kegunaan lainnya:

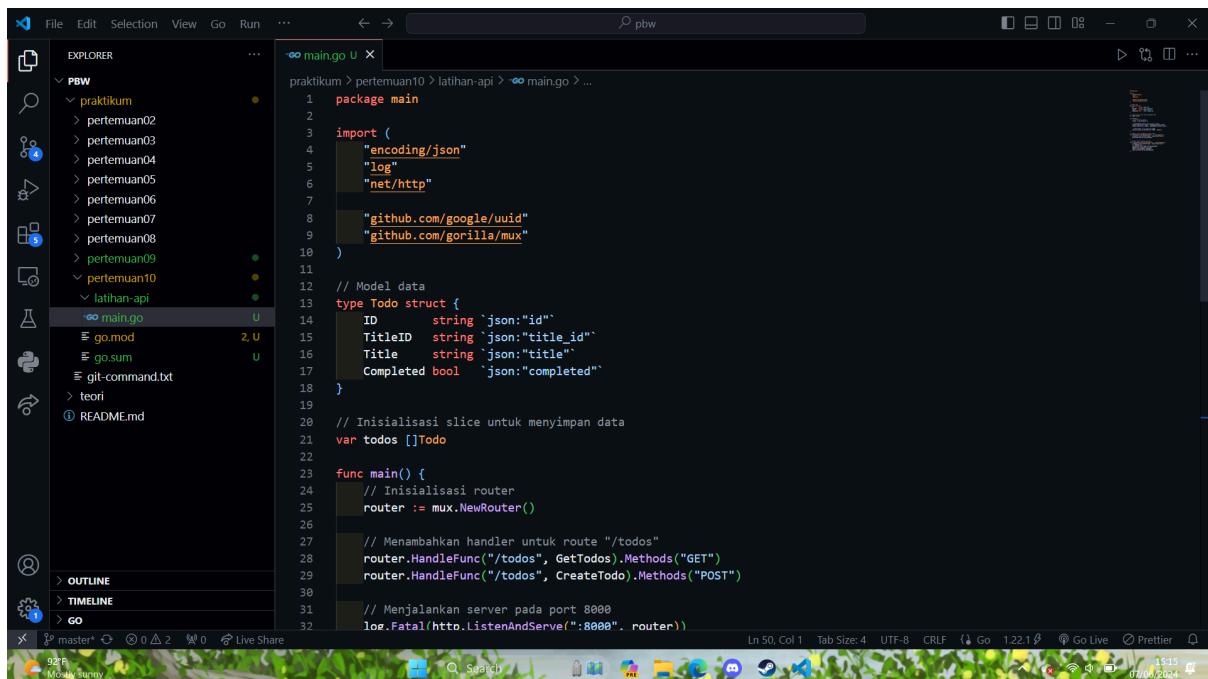
- Routing yang lebih fleksibel: mux memungkinkan untuk menentukan rute yang mendukung parameter, query string, dan metode HTTP yang berbeda (GET, POST, PUT, DELETE, dll).
- Named Parameters: Menangkap bagian dari URL sebagai parameter dan menggunakan dalam handler.
- Host Matching: Dapat memetakan rute berdasarkan nama host, yang memungkinkan aplikasi yang sama melayani beberapa domain.
- Route Priorities: Mendukung urutan penentuan rute sehingga bisa mengatur prioritas rute mana yang harus diproses terlebih dahulu.

### 2. [github.com/google/uuid](https://github.com/google/uuid)

Library uuid dari Google digunakan untuk menghasilkan Universally Unique Identifier (UUID). UUID adalah 128-bit number yang digunakan untuk mengidentifikasi informasi secara unik di seluruh sistem. UUID sangat berguna untuk membuat ID unik yang hampir tidak mungkin untuk diduplikasi. Kegunaan lainnya:

- Unique Identification: UUID sangat ideal untuk digunakan sebagai kunci unik di basis data atau sebagai ID untuk objek dalam sistem terdistribusi.
- Kombinasi Waktu dan Ruang: UUID menggabungkan informasi dari waktu, ruang (seperti alamat MAC), dan random number untuk menjamin keunikan.
- Versi UUID: Mendukung beberapa versi UUID, termasuk versi 1 (berdasarkan timestamp dan alamat MAC) dan versi 4 (berbasis random number).

## Source Code:



```

package main

import (
    "encoding/json"
    "log"
    "net/http"
    "github.com/google/uuid"
    "github.com/gorilla/mux"
)

type Todo struct {
    ID      string `json:"id"`
    TitleID string `json:"title_id"`
    Title   string `json:"title"`
    Completed bool `json:"completed"`
}

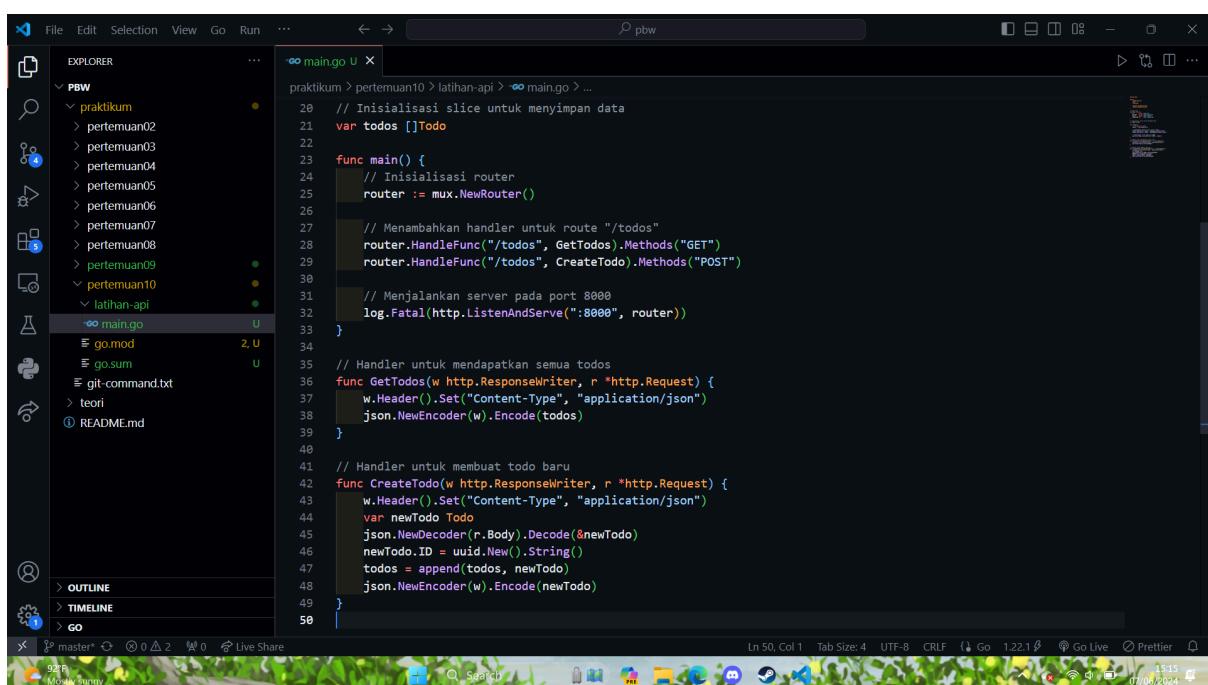
var todos []Todo

func main() {
    // Inisialisasi router
    router := mux.NewRouter()

    // Menambahkan handler untuk route "/todos"
    router.HandleFunc("/todos", GetTodos).Methods("GET")
    router.HandleFunc("/todos", CreateTodo).Methods("POST")

    // Menjalankan server pada port 8000
    log.Fatal(http.ListenAndServe(":8000", router))
}

```



```

// Inisialisasi slice untuk menyimpan data
var todos []Todo

func main() {
    // Inisialisasi router
    router := mux.NewRouter()

    // Menambahkan handler untuk route "/todos"
    router.HandleFunc("/todos", GetTodos).Methods("GET")
    router.HandleFunc("/todos", CreateTodo).Methods("POST")

    // Menjalankan server pada port 8000
    log.Fatal(http.ListenAndServe(":8000", router))
}

// Handler untuk mendapatkan semua todos
func GetTodos(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(todos)
}

// Handler untuk membuat todo baru
func CreateTodo(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    var newTodo Todo
    json.NewDecoder(r.Body).Decode(&newTodo)
    newTodo.ID = uuid.New().String()
    todos = append(todos, newTodo)
    json.NewEncoder(w).Encode(newTodo)
}

```

## Run code menggunakan terminal:

```
func main() {
    // Inisialisasi router
    router := mux.NewRouter()

    // Menambahkan handler untuk route "/todos"
    router.HandleFunc("/todos", GetTodos).Methods("GET")
    router.HandleFunc("/todos", CreateTodo).Methods("POST")

    // Menjalankan server pada port 8000
    log.Fatal(http.ListenAndServe(":8000", router))
}

// Handler untuk mendapatkan semua todos
func GetTodos(w http.ResponseWriter, r *http.Request) {
    w.Header().Set("Content-Type", "application/json")
    json.NewEncoder(w).Encode(todos)
}

// Handler untuk membuat todo baru
func CreateTodo(w http.ResponseWriter, r *http.Request) {
```

```
Arsya@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan10/latihan-api (master)
$ cd ..

Arsya@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum/pertemuan10 (master)
$ go run main.go
```

## Tampilan Postman:

My Workspace

New Import

pbw-praktikum / New Request

POST http://localhost:8000/todos

Params

Key	Value	Description
Key	Value	Description

Status: 200 OK Time: 18 ms Size: 197 B

```
1: {
  "id": "6001be31-c9b9-492b-bd32-3e1ab5ae534",
  "title_id": "",
  "title": "",
  "completed": false
}
```

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, API Network, and a search bar labeled "Search Postman". On the right side of the header, there are buttons for "Invite", "Upgrade", and environment selection. The main workspace is titled "pbw-praktikum / New Request". A POST request is being made to "http://localhost:8000/todos". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   ...
3   ...
4     "title_id": "3",
5     "title": "Praktikum PBW",
6     ...
7     ...
8       "completed": false
9     ...
10    ...
11   ...
12 }
```

Below the body, the response status is shown as "Status: 200 OK Time: 5 ms Size: 197 B". The "Pretty" tab in the results panel displays the response in a readable JSON format:

```
1 {
2   "id": "2d6505eb-1867-481c-bac1-a849a2cadd9c",
3   "title_id": "",
4   "title": "",
5   "completed": false
6 }
```

This screenshot shows the Postman application interface again, but this time under the "My Workspace" section. The left sidebar lists "Collections" (pbw-praktikum) and "Environments". The main workspace is titled "pbw-praktikum / POST New Request". A POST request is being made to "http://localhost:8000/todos". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   ...
3   ...
4     "title_id": "1",
5     "title": "Hallo semua",
6     ...
7     ...
8       "completed": false
9     ...
10    ...
11   ...
12 }
```

Below the body, the response status is shown as "Status: 200 OK Time: 3 ms Size: 197 B". The "Pretty" tab in the results panel displays the response in a readable JSON format:

```
1 {
2   "id": "fiffe9e23-588c-4f32-80ed-599b4c4bdbda",
3   "title_id": "",
4   "title": "",
5   "completed": false
6 }
```

The screenshot shows the Postman application interface. In the left sidebar, there's a 'My Workspace' section with a collection named 'pbw-praktikum'. A 'POST New Request' card is selected. The main workspace shows a POST request to 'http://localhost:8000/todos'. The 'Body' tab is active, showing a JSON payload:

```
1 {  
2 ... "title_id": "2",  
3 ... "title": "Saya Arsy Yan Duribta",  
4 ... "completed": false  
5 }
```

The response status is 200 OK, Time: 10 ms, Size: 221 B. The response body is identical to the request body.

This screenshot is nearly identical to the one above, showing a POST request to 'http://localhost:8000/todos'. The JSON body is different:

```
1 {  
2 ... "title_id": "3",  
3 ... "title": "Praktikum PBW",  
4 ... "completed": false  
5 }
```

The response status is 200 OK, Time: 5 ms, Size: 212 B. The response body is identical to the request body.

The screenshot shows the Postman application interface. In the top navigation bar, 'pbw-praktikum' is selected under 'Workspaces'. A POST request is being made to 'http://localhost:8000/todos'. The 'Body' tab is active, showing a JSON payload:

```
1 {  
2 ... "title_id": "4",  
3 ... "title": "Dosen Pengampu Pak Adi",  
4 ... "completed": false  
5 }
```

The response status is 200 OK, and the JSON response body is displayed below:

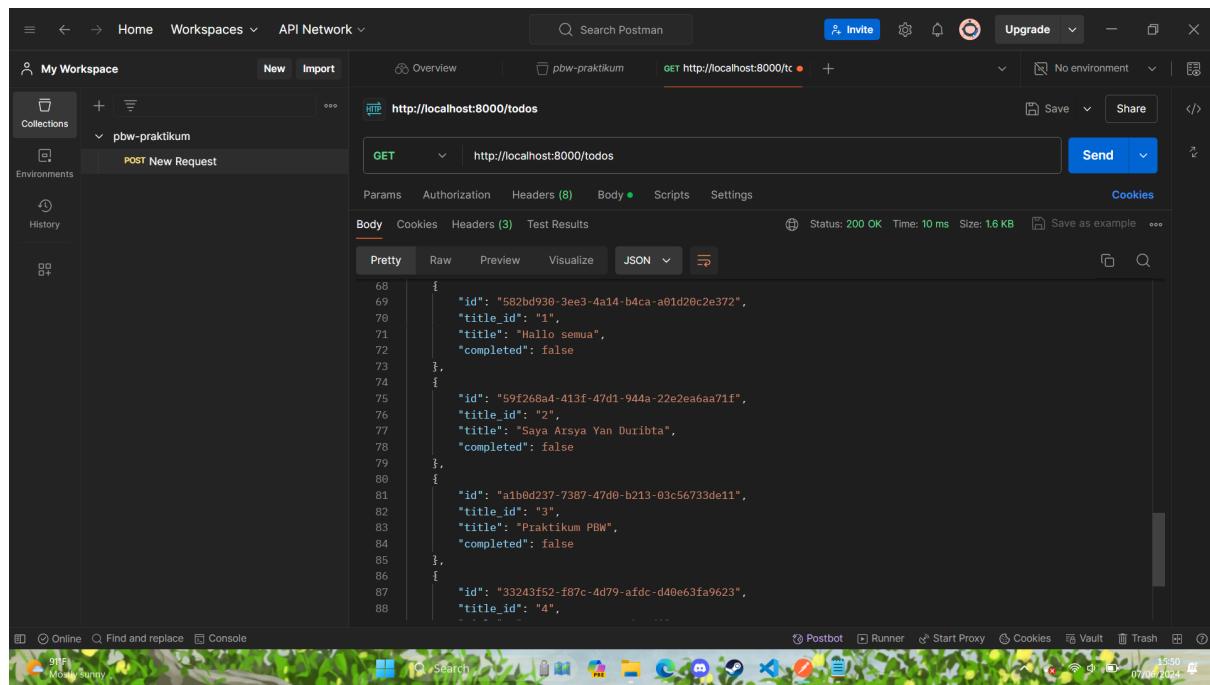
```
1 {  
2 ... "id": "33243ff52-f87c-4d79-afdc-d40e63fa9623",  
3 ... "title_id": "4",  
4 ... "title": "Dosen Pengampu Pak Adi",  
5 ... "completed": false  
6 }
```

The screenshot shows the Postman application interface. In the top navigation bar, 'pbw-praktikum' is selected under 'Workspaces'. A POST request is being made to 'http://localhost:8000/todos'. The 'Body' tab is active, showing a JSON payload:

```
1 {  
2 ... "title_id": "5",  
3 ... "title": "Teknik Informatika",  
4 ... "completed": false  
5 }
```

The response status is 200 OK, and the JSON response body is displayed below:

```
1 {  
2 ... "id": "4d276fb5-7bb4-4310-aec -ad33954131fb",  
3 ... "title_id": "5",  
4 ... "title": "Teknik Informatika",  
5 ... "completed": false  
6 }
```



My Workspace

pbw-praktikum

POST New Request

http://localhost:8000/todos

GET

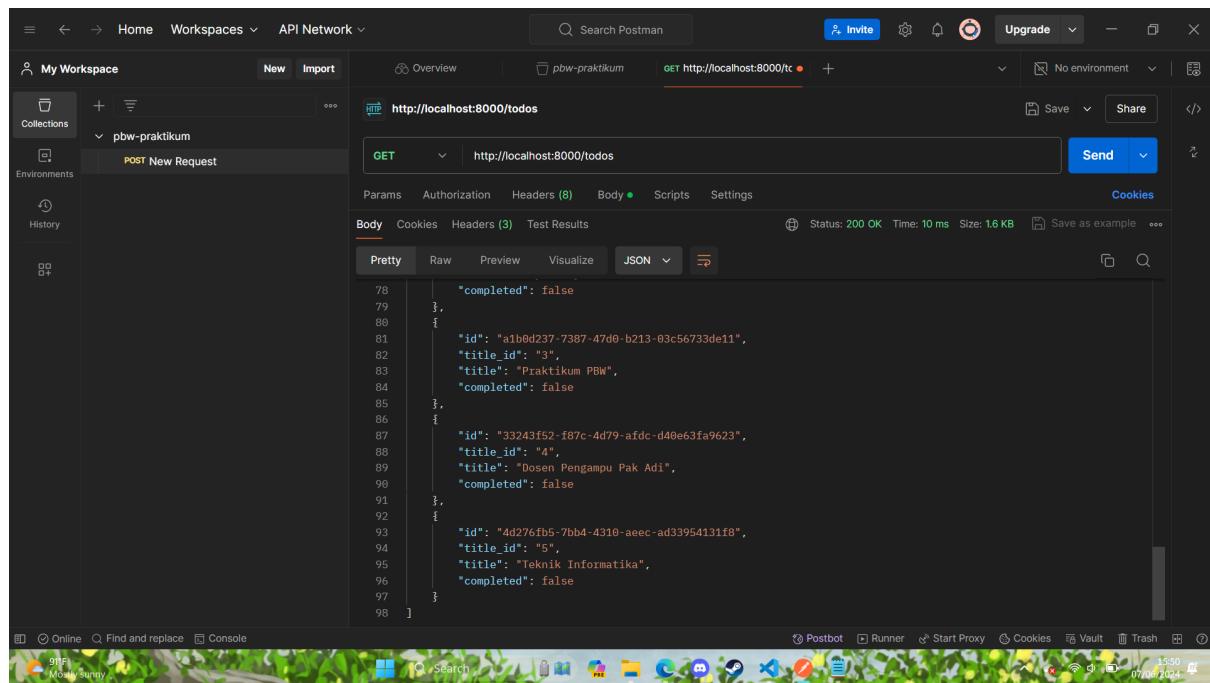
Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
```

```
[{"id": "582bd930-see3-4a14-b4ca-a01d20c2e372", "title_id": "1", "title": "Hallo semua", "completed": false}, {"id": "59f268a4-413f-47d1-944a-22e2ea6aa71f", "title_id": "2", "title": "Saya Arsyia Yan Duribta", "completed": false}, {"id": "a1bdd237-7387-47d0-b213-03c56733de11", "title_id": "3", "title": "Praktikum PBW", "completed": false}, {"id": "33243f52-f87c-4d79-afdc-d40e63fa9623", "title_id": "4", "completed": false}, {"id": "4d276fb5-7bb4-4310-aec -ad33954131f8", "title_id": "5", "title": "Teknik Informatika", "completed": false}]
```



My Workspace

pbw-praktikum

POST New Request

http://localhost:8000/todos

GET

Params Authorization Headers (8) Body Scripts Settings

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
```

```
[{"completed": false}, {"id": "a1bdd237-7387-47d0-b213-03c56733de11", "title_id": "3", "title": "Praktikum PBW", "completed": false}, {"id": "33243f52-f87c-4d79-afdc-d40e63fa9623", "title_id": "4", "title": "Dosen Pengampu Pak Adi", "completed": false}, {"id": "4d276fb5-7bb4-4310-aec -ad33954131f8", "title_id": "5", "title": "Teknik Informatika", "completed": false}]
```

## Upload Ke Git

The screenshot shows the Visual Studio Code interface with the following details:

- EXPLORER:** Shows a file tree for a project named "pbw". The "praktikum" folder contains sub-folders "pertemuan02" through "pertemuan10", and files "go.mod", "go.sum", and "main.go".
- TERMINAL:** Displays a terminal session with the following commands and output:

```
Arsya@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum (master)
$ git add .
warning: in the working copy of 'praktikum/pertemuan10/go.mod', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'praktikum/pertemuan10/go.sum', LF will be replaced by CRLF the next time Git touches it
Arsa@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum (master)
$ git commit -m "upload latihan praktikum ke 10"
[master 92d5204] upload latihan praktikum ke 10
 4 files changed, 61 insertions(+)
   create mode 100644 praktikum/pertemuan09/quiz.txt
   create mode 100644 praktikum/pertemuan10/go.mod
   create mode 100644 praktikum/pertemuan10/go.sum
   create mode 100644 praktikum/pertemuan10/main.go
Arsa@LAPTOP-FKPF8QJ1 MINGW64 ~/Documents/pbw/praktikum (master)
$ git push -f origin master
Enumerating objects: 10, done.
Counting objects: 1000, 10/10, done.
Delta compression using up to 12 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.43 KiB | 487.00 KiB/s, done.
Total 8 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Arsyayd11/pbw.git
 3647645..92d5204 master -> master
```
- SIDE BAR:** Shows a sidebar with sections like OUTLINE, TIMELINE, and GO.
- STATUS BAR:** Shows the current file path (~/Documents/pbw/praktikum), the terminal command (\$), and the status bar with various icons and text.

## Kesimpulan

Dalam praktikum Go latihan API, library `github.com/gorilla/mux` digunakan untuk menangani routing HTTP yang fleksibel dan memungkinkan pengaturan endpoint dengan parameter dan metode yang berbeda, sementara library `github.com/google/uuid` digunakan untuk menghasilkan UUID unik yang dapat digunakan sebagai identifier dalam endpoint API, seperti ID untuk resource yang dibuat. Kombinasi kedua library ini memungkinkan pembuatan API yang robust, dengan routing yang terstruktur dan identifier yang unik, meningkatkan efisiensi dari aplikasi web yang dibangun.

## Link Repository Github:

<https://github.com/Arsyayd11/pbw/tree/master/praktikum/pertemuan10>