

**LAPORAN PRAKTIKUM-10**  
**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/tree/master/p10>

## Latihan Web API menggunakan Gorilla

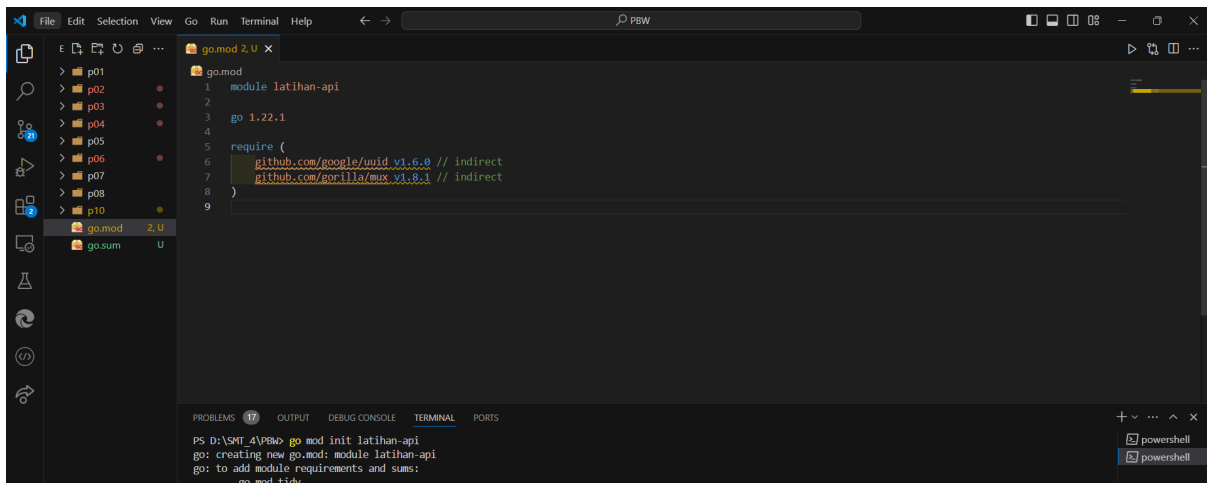
- Apa Kegunaan library “go mod init latihan-api” ?

Jawaban: Library bahasa Go "go mod init latihan-api" memungkinkan pengaktifan modul baru "latihan-api" dan mengelola dependensi proyek dengan membuat file "go.mod", yang menyimpan informasi versi dan ketergantungan paket-paket lainnya. Memastikan versi yang tepat, untuk mencegah masalah kompatibilitas, dan memungkinkan lingkungan pengembangan yang konsisten di berbagai mesin.

- Apa Kegunaan library “go get [github.com/google/uuid](https://github.com/google/uuid)” ?

Jawaban: Library Go bernama "github.com/google/uuid" berfungsi untuk menghasilkan UUID (Universally Unique Identifier), angka unik digunakan untuk mengidentifikasi informasi dalam sistem komputasi. Library ini menyediakan fungsi untuk membuat UUID versi 1 dan 4, digunakan dalam berbagai aplikasi yang menjamin keunikan pengidentifikasi, seperti database, sistem distribusi, dan berbagai aplikasi lainnya. Pengembang dapat membuat pengidentifikasi unik yang hampir tidak dapat diulang dengan menggunakan "github.com/google/uuid", yang membantu mengelola data dan entitas secara aman dan efisien.

- Initiate Project



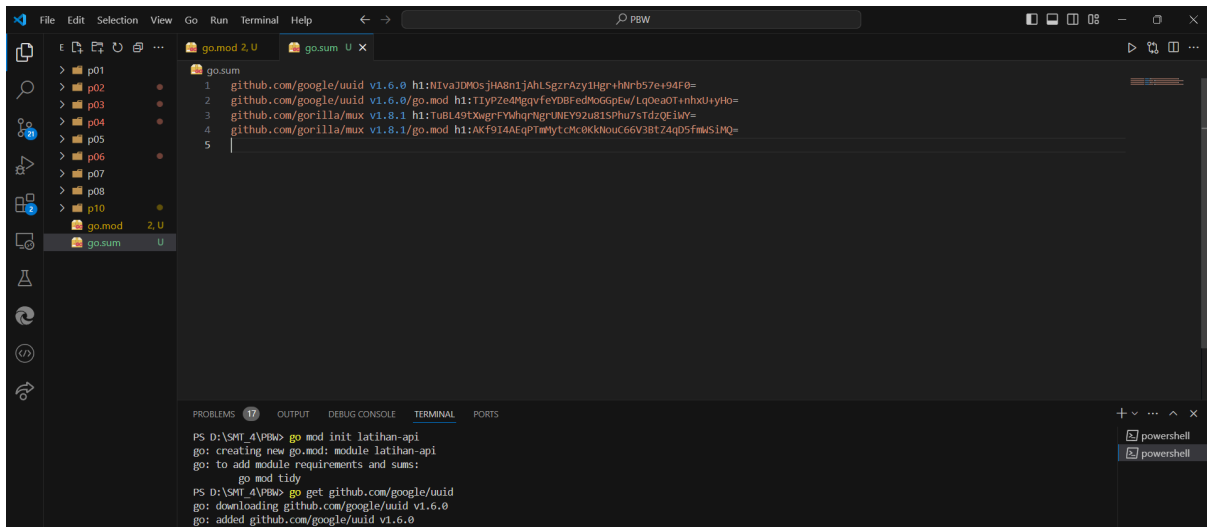
The screenshot shows a Visual Studio Code editor window with a Go project. The file explorer on the left shows a directory structure with folders p01 through p10. The main editor area shows a file named `go.mod` with the following content:

```
1 module latihan-api
2
3 go 1.22.1
4
5 require (
6     github.com/google/uuid v1.6.0 // indirect
7     github.com/gorilla/mux v1.8.1 // indirect
8 )
9
```

Below the editor, the terminal shows the output of the `go mod init latihan-api` command:

```
PS D:\SMT_4\PBW> go mod init latihan-api
go: creating new go.mod: module latihan-api
go: to add module requirements and sums:
go mod tidy
```

- Install Library Gorilla dan UUID



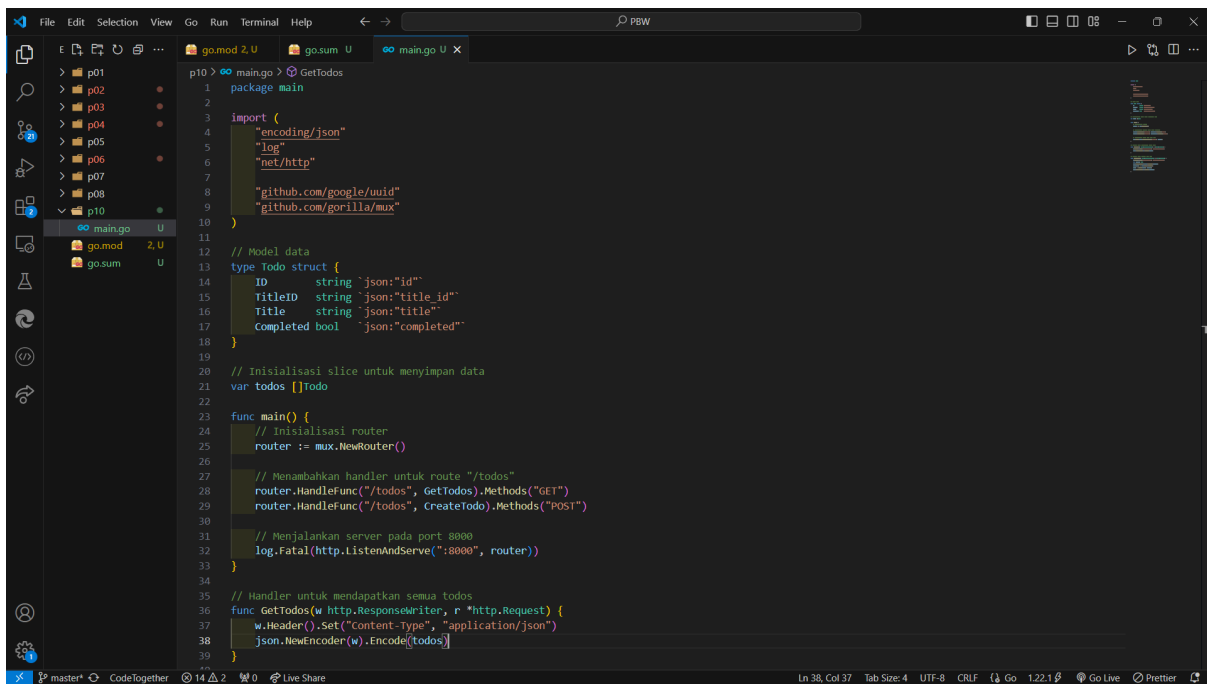
The screenshot shows the VS Code interface with a file explorer on the left and a terminal at the bottom. The file explorer shows a project structure with folders p01 through p10, and files go.mod and go.sum. The go.mod file contains the following content:

```
1 github.com/google/uuid v1.6.0 h1:NIva3DMQsJH8n1jAhlSgzrAzyIHgr+hNrb57e+94F0=
2 github.com/google/uuid v1.6.0/go.mod h1:TiiPZ64NgqvfeYDBFedMogGpEW/LQ0eaOT+nhxU+yyHo=
3 github.com/gorilla/mux v1.8.1 h1:TubL49TwwgrFVwhqNgrUNEY92u81SPhu7sTdZQEilMY=
4 github.com/gorilla/mux v1.8.1/go.mod h1:AKf914AEqPTmMytcMc0KklouC66V3BtZ4qD5fakS1MQ=
5
```

The terminal shows the following commands and output:

```
PS D:\SWT_4\PBW> go mod init latihan-api
go: creating new go.mod: module latihan-api
go: to add module requirements and sums:
  go mod tidy
PS D:\SWT_4\PBW> go get github.com/google/uuid
go: downloading github.com/google/uuid v1.6.0
go: added github.com/google/uuid v1.6.0
```

- Source Code main.go



The screenshot shows the VS Code interface with a file explorer on the left and a code editor in the center. The code editor shows the source code of main.go, which is a simple REST API using gorilla/mux and google/uuid. The code is as follows:

```
1 package main
2
3 import (
4     "encoding/json"
5     "log"
6     "net/http"
7
8     "github.com/google/uuid"
9     "github.com/gorilla/mux"
10 )
11
12 // Model data
13 type Todo struct {
14     ID          string `json:"id"`
15     TitleID     string `json:"title_id"`
16     Title       string `json:"title"`
17     completed   bool  `json:"completed"`
18 }
19
20 // Inisialisasi slice untuk menyimpan data
21 var todos []Todo
22
23 func main() {
24     // inisialisasi router
25     router := mux.NewRouter()
26
27     // Menambahkan handler untuk route "/todos"
28     router.HandleFunc("/todos", GetTodos).Methods("GET")
29     router.HandleFunc("/todos", CreateTodo).Methods("POST")
30
31     // Menjalankan server pada port 8000
32     log.Fatal(http.ListenAndServe(":8000", router))
33 }
34
35 // Handler untuk mendapatkan semua todos
36 func GetTodos(w http.ResponseWriter, r *http.Request) {
37     w.Header().Set("Content-Type", "application/json")
38     json.NewEncoder(w).Encode(todos)
39 }
40
```

The screenshot shows the Visual Studio Code editor with a Go web application. The file explorer on the left shows a project structure with folders p01 through p10, and files go.mod, go.sum, and main.go. The main.go file is open in the editor, showing the following code:

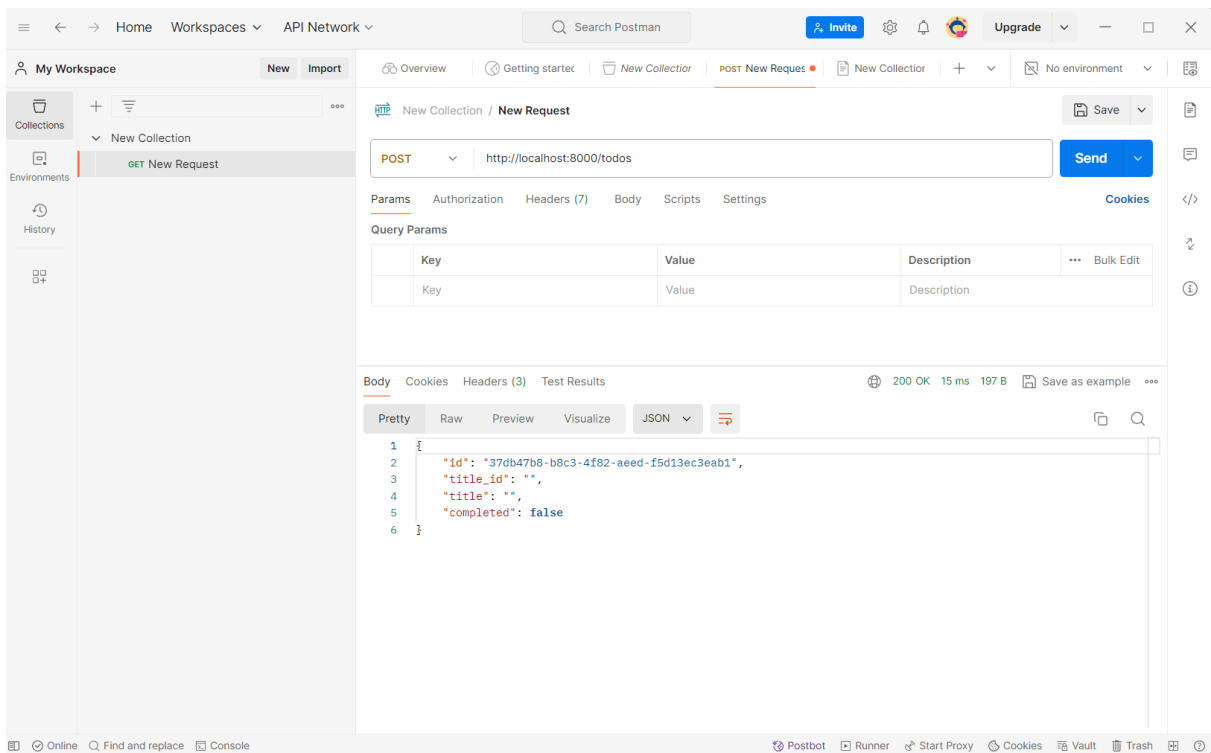
```
23 func main() {
24     // Menjalankan server pada port 8000
25     log.Fatal(http.ListenAndServe(":8000", router))
26 }
27
28 // Handler untuk mendapatkan semua todos
29 func GetTodos(w http.ResponseWriter, r *http.Request) {
30     w.Header().Set("Content-Type", "application/json")
31     json.NewEncoder(w).Encode(todos)
32 }
33
34 // Handler untuk membuat todo baru
35 func CreateTodo(w http.ResponseWriter, r *http.Request) {
36     w.Header().Set("Content-Type", "application/json")
37     var newTodo Todo
38     json.NewDecoder(r.Body).Decode(&newTodo)
39     newTodo.ID = uuid.New().String()
40     todos = append(todos, newTodo)
41     json.NewEncoder(w).Encode(newTodo)
42 }
43
44 // Menjalankan server pada port 8000
45 log.Fatal(http.ListenAndServe(":8000", router))
46 }
```

The terminal window at the bottom shows the command `go run main.go` being executed.

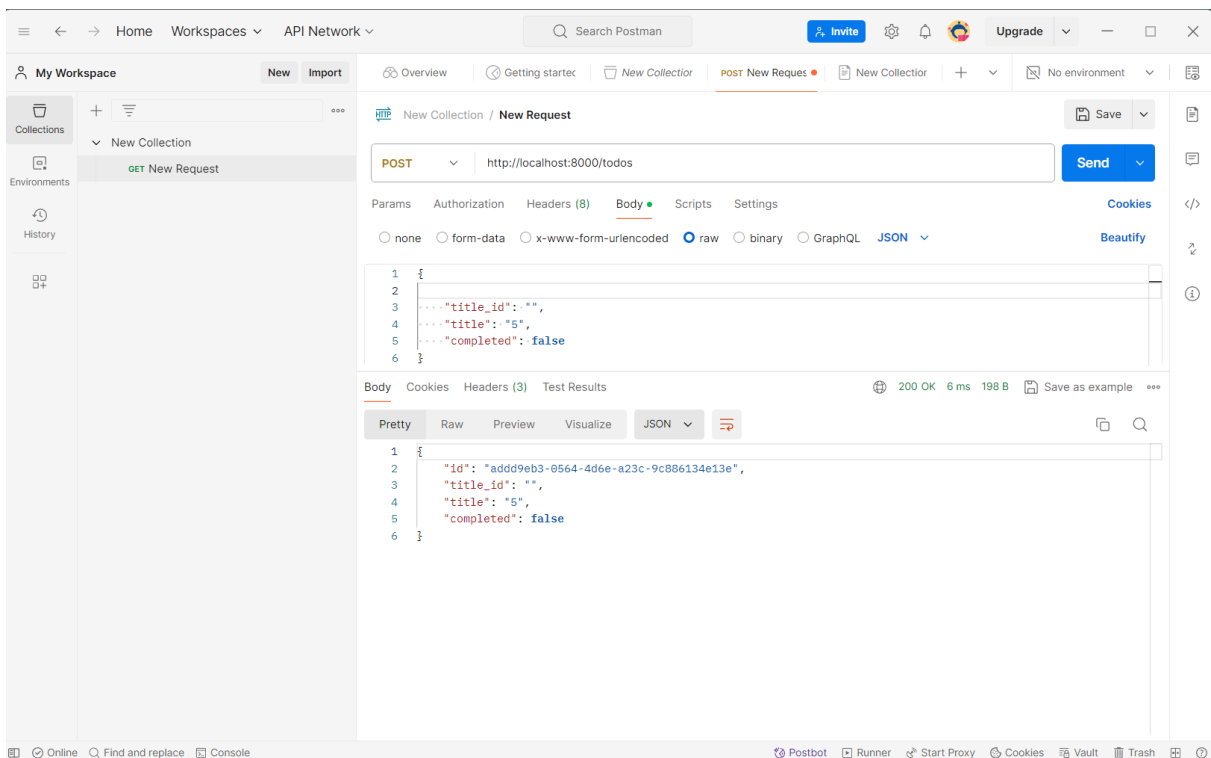
- Jalankan localhost dan buka postman

The screenshot shows the Visual Studio Code editor with the same Go web application. The file explorer on the left shows the same project structure. The main.go file is open in the editor, showing the same code as in the previous screenshot. The terminal window at the bottom shows the command `go run main.go` being executed, and the output shows the server starting on port 8000.

- Buka postman



- Masukan title dengan body JSON dengan metode POST



- Post dengan minimal 5

The screenshot shows the Postman interface with a new POST request configured. The URL is `http://localhost:8000/todos`. The request body is set to raw JSON. The JSON body contains a single todo item:

```
1 {
2   "title_id": "1",
3   "title": "Gibran Kahlil",
4   "completed": false
5 }
```

The response is displayed in the bottom pane, showing a 200 OK status with a response time of 4 ms and a body size of 212 B. The response body is a JSON object:

```
1 {
2   "id": "2f47ea2f-1011-4713-b2a1-05cf42d957f2",
3   "title_id": "1",
4   "title": "Gibran Kahlil",
5   "completed": false
6 }
```

The screenshot shows the Postman interface with a new POST request configured. The URL is `http://localhost:8000/todos`. The request body is set to raw JSON. The JSON body contains two todo items:

```
1 {
2   "title_id": "2",
3   "title": "Universitas Pancasila",
4   "completed": false
5 }
```

The response is displayed in the bottom pane, showing a 200 OK status with a response time of 3 ms and a body size of 220 B. The response body is a JSON object:

```
1 {
2   "id": "2522dd00-3fd5-4685-8a47-e74d6a31d917",
3   "title_id": "2",
4   "title": "Universitas Pancasila",
5   "completed": false
6 }
```

Home Workspaces API Network Search Postman

My Workspace New Import

Collections + New Collection GET New Request

Overview Getting started New Collector New Collector POST New Request No environment

New Collection / New Request Save

POST http://localhost:8000/todos Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ...title_id: "3",
3   ...title: "4522210128",
4   ...completed: false
5 }
```

Body Cookies Headers (3) Test Results 200 OK 4 ms 209 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "9757b181-fa2c-4927-aff3-70de305154a8",
3   "title_id": "3",
4   "title": "4522210128",
5   "completed": false
6 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash

Home Workspaces API Network Search Postman

My Workspace New Import

Collections + New Collection GET New Request

Overview Getting started New Collector New Collector POST New Request No environment

New Collection / New Request Save

POST http://localhost:8000/todos Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

none form-data x-www-form-urlencoded raw binary GraphQL JSON

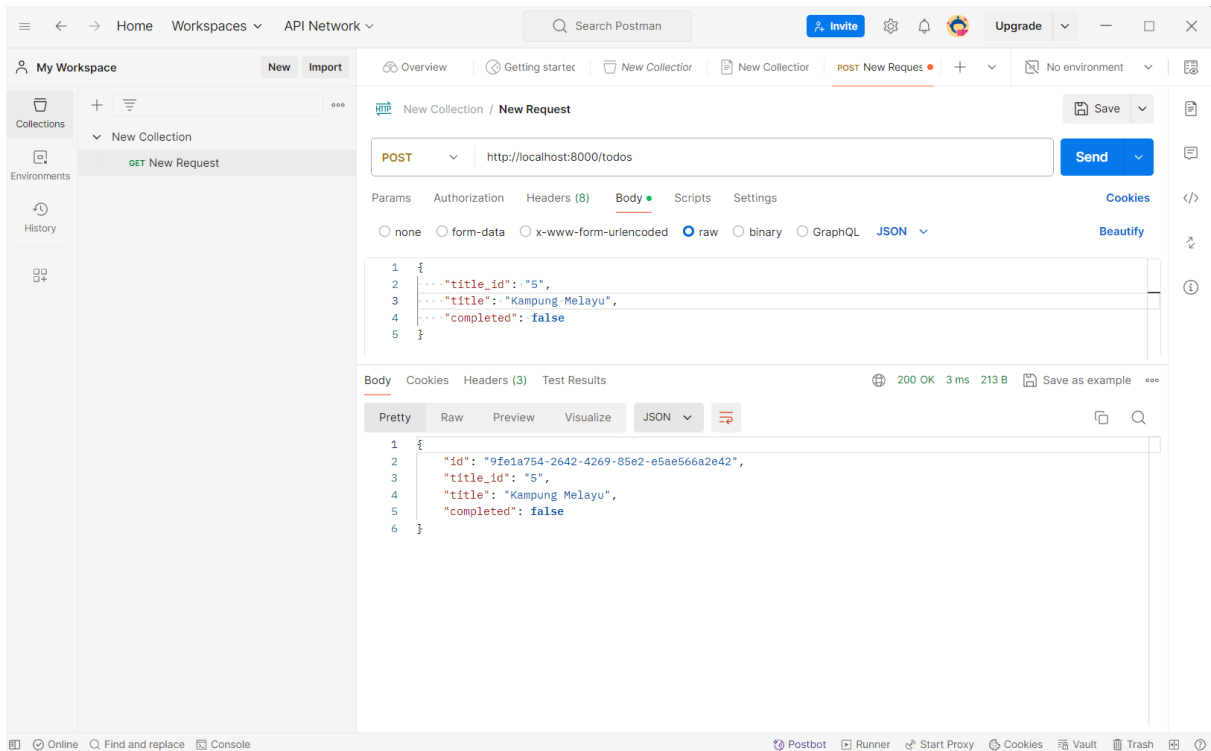
```
1 {
2   ...title_id: "4",
3   ...title: "Teknik Informatika",
4   ...completed: false
5 }
```

Body Cookies Headers (3) Test Results 200 OK 4 ms 217 B Save as example

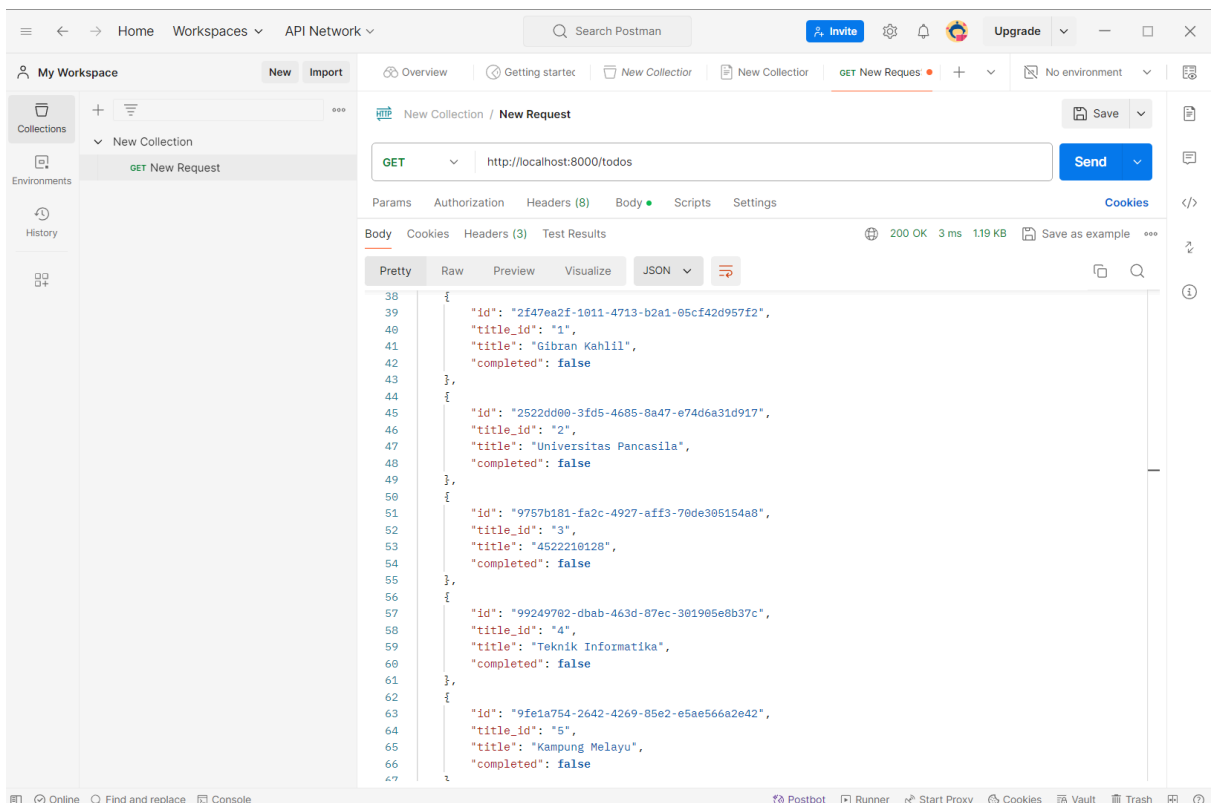
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": "99249702-dbab-463d-87ec-301905e8b37c",
3   "title_id": "4",
4   "title": "Teknik Informatika",
5   "completed": false
6 }
```

Online Find and replace Console Postbot Runner Start Proxy Cookies Vault Trash



- Gunakan Get untuk melihat title yang dibuat





Kesimpulan: Library `'github.com/google/uuid'` dalam Go digunakan untuk menghasilkan UUID yang unik, untuk mengidentifikasi informasi secara unik dalam sistem komputasi. Dengan menyediakan fungsi untuk membuat UUID versi 1 dan versi 4, library ini membantu pengembang memastikan keunikan data dalam berbagai aplikasi, seperti database dan sistem distribusi, sehingga dapat meningkatkan keamanan pengolahan data.