CRUD RESTFUL API GOLANG DENGAN FRAMEWORK GIN GONIC

UNTUK MEMENUHI ULANGAN TENGAH SEMESTER GENAP

MATA KULIAH PEMROGRAMAN CLOUD COMPUTING

Dosen Pengampu : Sopingi, M.Kom

Disusun Oleh

Zeno Candragufa Muria

170101188

Sistem Informasi 17A1

SISTEM INFORMASI

FAKULTAS ILMU KOMPUTER

UNIVERSITAS DUTA BANGSA SURAKARTA

2020

CRUD RESTFUL API GOLANG

DENGAN FRAMEWORK GIN GONIC

STUDI KASUS "DATA PROGRAM STUDI FAKULTAS ILMU KOMPUTER

UNIVERSITAS DUTA BANGSA SURAKARTA"

A. Membuat Kode RestFul API Golang

1. Setelah meng*install* Golang pada laptop/komputer (sudah diajarkan di awal semester, kalau belum paham bisa cari di google) selanjutnya install driver MySql dan *framework* Gin Gonic (wajib ada koneksi internet, karena ini akan mengunduh paket-paket yang digunakan untuk menjalankannya)
   a. Install MySql, ketikan perintah dibawah ini pada Git Bash atau CMD,

      ```
      go get -u github.com/go-sql-driver/mysql
      ```

   b. Install framework Gin Gonic, ketikan perintah dibawah ini pada Git Bash atau CMD,

      ```
      go get -u github.com/gin-gonic/gin
      ```

2. Buatlah folder bernama UTS, lalu buka folder UTS itu pada Visual Code atau kode editor yang digunakan.
3. Buat database uts dan tabel progdi

Field	Type	Keterangan
id	int (13)	Primary Key
jenjang	Varchar (20)	
nmprogdi	Varchar (50)	

4. Buat file **progdi.go** dan mengisi ***import***, kodenya seperti dibawah :

   package main

```
import (
	"bytes"
	"database/sql"
	"fmt"
	"net/http"

	"github.com/gin-gonic/gin"
	_ "github.com/go-sql-driver/mysql"
)
```

Digunakan untuk menggunakan/menjalankan *framework* Gin Gonic dan MySql

5. Buat **koneksi** di dalam *func main*, dan membuat fungsi menjalankan server :

```
func main() {
	db, err := sql.Open("mysql", "root:@tcp(127.0.0.1:3306)/uts ")
	err = db.Ping()
	if err != nil {
```

```
            panic("Gagal Menghubungkan ke Database...")
      }
      defer db.Close()

       .............
.....................
.....................

      router.Run(":8080")
}
```

Nanti kode/syntax berikutnya di tulis di bagian ini

Digunakan untuk menjalankan server localhost:8080

6. Buat **struct** di dalam *func main*, *struct* ini berisi dari *field* yang ada di tabel progdi :

```
type Progdi struct {
      Id          int          `json: "id"`
      Jenjang     string       `json: "jenjang"`
      NmProgdi    string       `json: "nmprodi"`
}
```

7. Buat perintah untuk melakukan ***routing***, dan buat fungsi untuk menampilkan (GET) data **progdi** berdasarkan **id** di dalam *func main* :

```
router := gin.Default()

router.GET("/:id", func(c *gin.Context) {
      var (
            progdi Progdi
            result gin.H
      )
      id := c.Param("id")
      row := db.QueryRow("select id, jenjang, nmprogdi from progdi where id = ?;", id)
      err = row.Scan(&progdi.Id, &progdi.Jenjang, &progdi.NmProgdi)
      if err != nil {
            result = gin.H{
                  "Hasile" : "Yahh Tidak ada data yang ditemukan",
            }
      } else {
            result = gin.H{
                  "Hasile": progdi,
                  "Jumlahe":  1,
            }
      }
      c.JSON(http.StatusOK, result)
})
```

Menunjukan Gin Gonic yang nantinya akan melakukan Routing

"Progdi" ini mengambil dari Struct di atas

8. Buat fungsi untuk menampilkan (GET) seluruh data **progdi** di dalam *func main* :

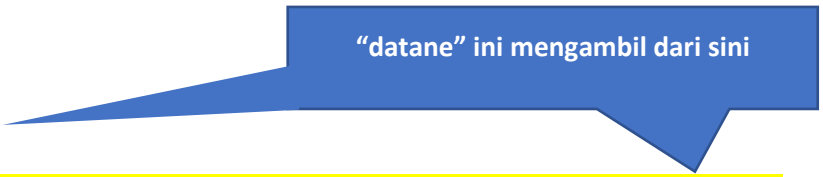
```
router.GET("/", func(c *gin.Context) {
      var (
            progdi  Progdi
            progdis []Progdi
      )
      rows, err := db.Query("select id, jenjang, nmprogdi from progdi;")
      if err != nil {
            fmt.Print(err.Error())
      }
      for rows.Next() {
```

"Progdi" ini mengambil dari Struct di atas

```
            err = rows.Scan(&progdi.Id, &progdi.Jenjang, &progdi.NmProgdi)
            progdis = append(progdis, progdi)
            if err != nil {
                  fmt.Print(err.Error())
            }
      }
      defer rows.Close()
      c.JSON(http.StatusOK, gin.H{
            "Hasile": progdis,
            "Jumlahe":  len(progdis),
      })
})
```

9. Buat fungsi untuk menambahkan (POST) data ke tabel **progdi** di dalam *func main*:

```
router.POST("/", func(c *gin.Context) {
      var buffer bytes.Buffer
      id := c.PostForm("id")
      jenjang := c.PostForm("jenjang")
      nmprogdi := c.PostForm("nmprogdi")
      stmt, err := db.Prepare("insert into progdi (id, jenjang, nmprogdi)
values(?,?,?); ")
      if err != nil {
            fmt.Print(err.Error())
      }
      _, err = stmt.Exec(id, jenjang, nmprogdi)
      if err != nil {
            fmt.Print(err.Error())
      }

      buffer.WriteString(jenjang)
      buffer.WriteString(" ")
      buffer.WriteString(nmprogdi)
      defer stmt.Close()
      datane := buffer.String()
      c.JSON(http.StatusOK, gin.H{
            "Pesane": fmt.Sprintf("Yeyyy Berhasil menambahkan Progdi %s ", datane),
      })
})
```



10. Buat fungsi untuk merubah (PUT) data dari tabel **progdi** berdasarkan **id** di dalam *func main* :

```
router.PUT("/:id", func(c *gin.Context) {
      var buffer bytes.Buffer
      id := c.Query("id")
      jenjang := c.PostForm("jenjang")
      nmprogdi := c.PostForm("nmprogdi")
      stmt, err := db.Prepare("update progdi set jenjang= ?, nmprogdi= ? where id=
?;")
      if err != nil {
            fmt.Print(err.Error())
      }
      _, err = stmt.Exec(jenjang, nmprogdi, id)
      if err != nil {
            fmt.Print(err.Error())
      }
```

```
        buffer.WriteString(jenjang)
        buffer.WriteString(" ")
        buffer.WriteString(nmprogdi)
        defer stmt.Close()
        datane := buffer.String()
        c.JSON(http.StatusOK, gin.H{
              "Pesane": fmt.Sprintf("Berhasil Merubah Menjadi %s", datane),
        })
})
```

11. Buat fungsi untuk menghapus (DELETE) data dari tabel **progdi** berdasarkan **id** di dalam *func main* :

```
router.DELETE("/:id", func(c *gin.Context) {
        id := c.Query("id")
        stmt, err := db.Prepare("delete from progdi where id= ?;")
        if err != nil {
              fmt.Print(err.Error())
        }
        _, err = stmt.Exec(id)
        if err != nil {
              fmt.Print(err.Error())
        }
        c.JSON(http.StatusOK, gin.H{
        "Pesane": fmt.Sprintf("Berhasil Menghapus"),
        })
})
```

B. Menjalankan RestFul API Menggunakan Postman

Method	Fungsi	*Format Request*	*Format Response*

GET	Tampil data		<pre>{
			  "Hasile": [
			  {
			  "Id": 1,
			  "Jenjang": "S1",
			  "NmProgdi": "Sistem Informasi"
			  }
			  ],
			  "Jumlahe": 1
			}</pre>

POST	Tambah data	Id : 2
		Jenjang : D3
		Nmprogdi : Manajemen Informatika
			<pre>{
			  "Pesane": " Yeyyy Berhasil menambahkan Progdi D3 Manajemen Informatika "
			}</pre>

PUT	Ubah data	http://localhost:8080/2
		Jenjang : D2
		Nmprogdi : Manajemen Informatika
			<pre>{
			  "Pesane": "Berhasil Merubah Menjadi D2 Manajemen Informatika"
			}</pre>

DELETE	Hapus data	http://localhost:8080/2	<pre>{
			  "Pesane": "Berhasil Menghapus "
			}</pre>

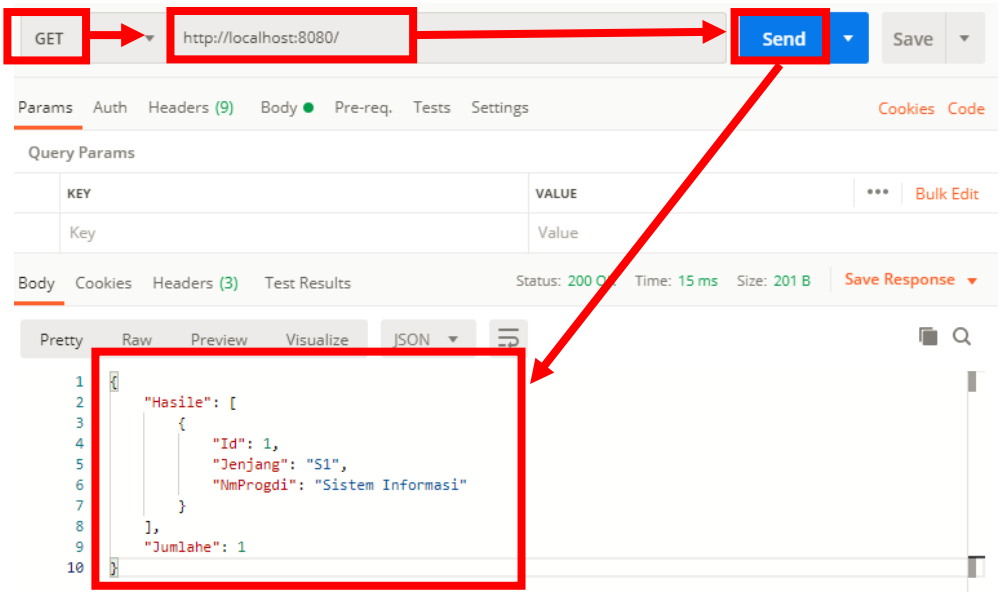
1. Jalankan XAMPP lalu mulai Apache dan MySql
2. Buka folder menyimpan kode go kedalam **Git Bash,** caranya
   a. Sebelumnya harus menginstall **Git Bash** terlebih dahulu, bisa di download di laman ini *git-scm.com/downloads*
   b. Klik kanan folder yang digunakan untuk menyimpan file kode go tadi
   c. Pilih Git Bash Here, maka akan terbuka terminal Git Bash dan langsung masuk ke dalam folder tersebut.
3. Ketikan perintah dibawah untuk menjalankan file go
   `go run progdi.go` lalu klik enter
4. Tunggu hingga muncul jendela *Firewall* lalu klik ***Allow Access***
5. Akan muncul seperti dibawah jika berhasil, disana sudah ada informasi server berjalan dimana dan cara menggunakan method bagaimana :

```
$ go run progdi.go
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /:id                      --> main.main.func1 (3 handlers)
[GIN-debug] GET    /                         --> main.main.func2 (3 handlers)
[GIN-debug] POST   /                         --> main.main.func3 (3 handlers)
[GIN-debug] PUT    /:id                      --> main.main.func4 (3 handlers)
[GIN-debug] DELETE /:id                      --> main.main.func5 (3 handlers)
[GIN-debug] Listening and serving HTTP on :8080
```

6. Untuk melakukan ujicoba API, laptop/komputer harus sudah terinstall aplikasi **Postman** (kalo belum tahu bisa lihat di youtube/google) dan dijalankan.

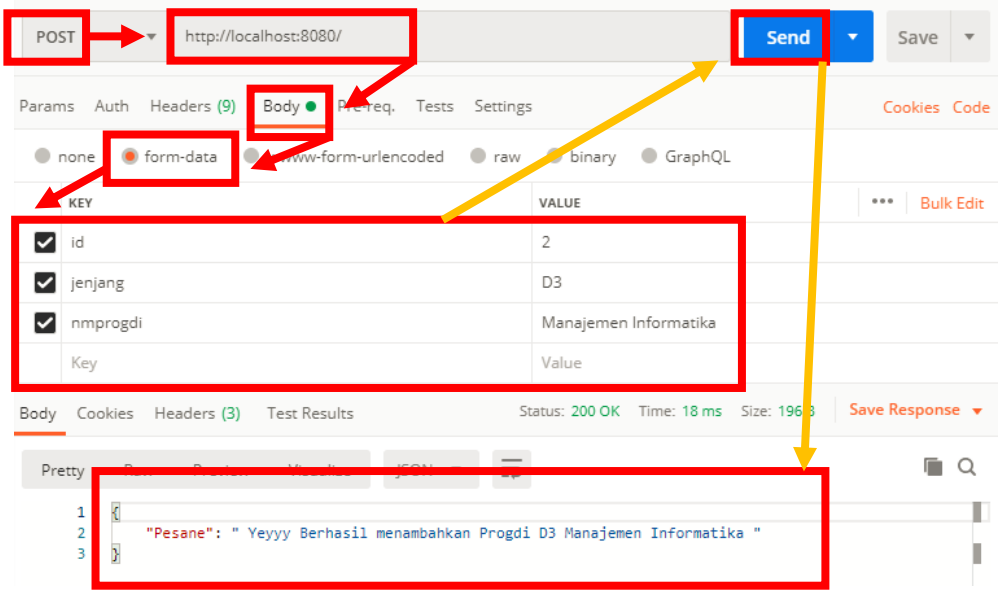
   a. Method GET

      GET untuk menampilkan data – http://localhost:8080/



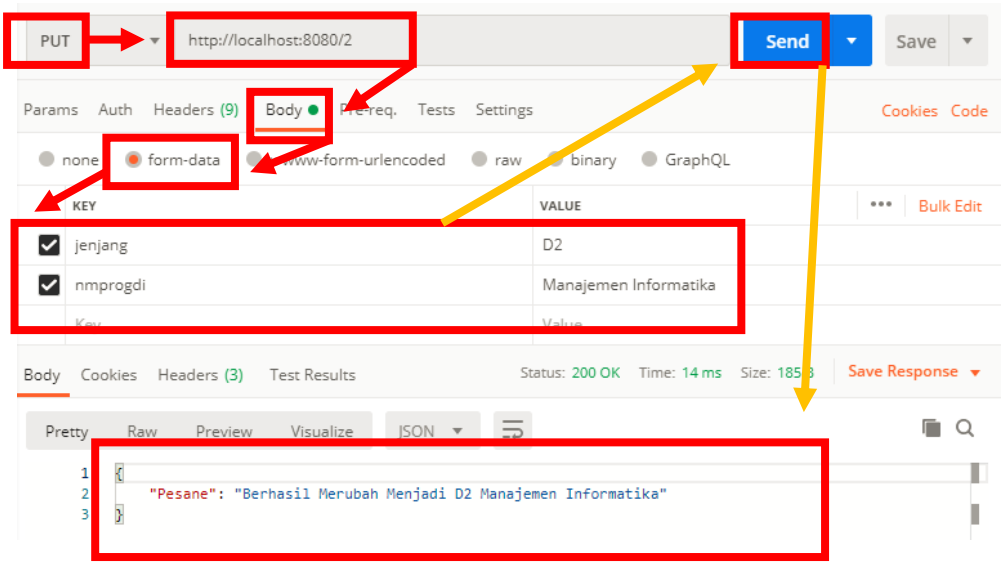
   b. Method POST

      POST untuk menambah data – http://localhost:8080/



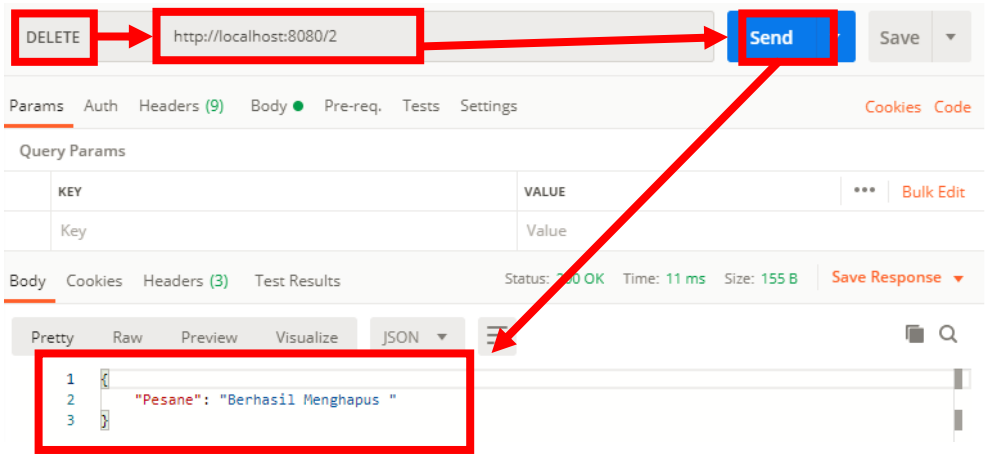
c. Method PUT

PUT untuk merubah data – http://localhost:8080/(id yang ingin di rubah)



d. Method DELETE

DELETE untuk menghapus data – http://localhost:8080/(id yang ingin di hapus)



Versi video ada di Chanel YouTube Zeno Candragufa Muria

https://youtu.be/rZXzhNqAP5I