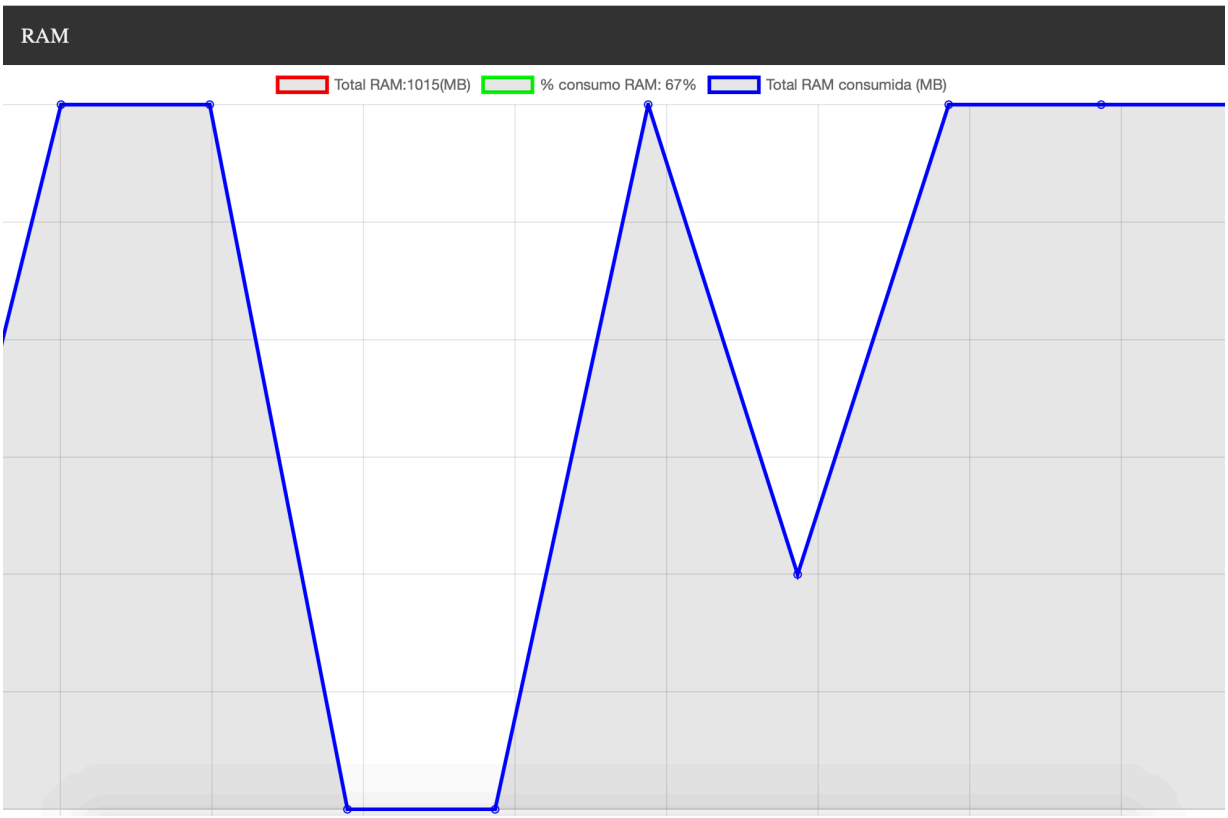
# Manual Tecnico

*Sistemas Operativos 1*



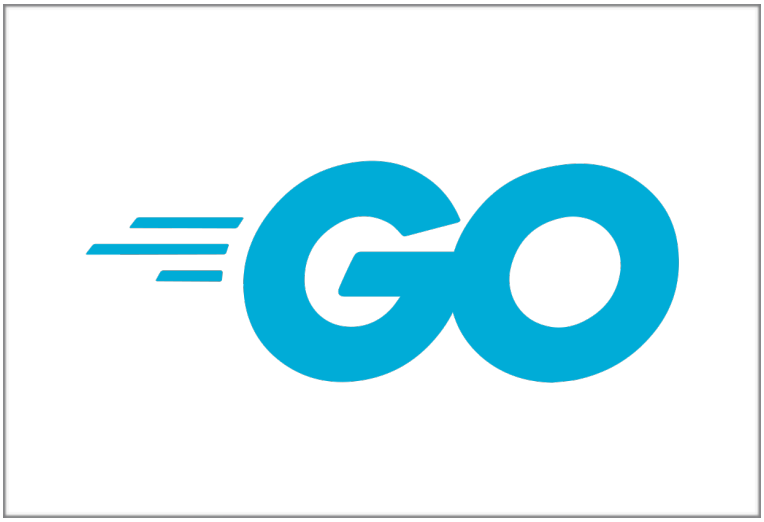

Dennis Alejandro Masaya Nájera

201503413

# Go

*Programming Language*

Es un lenguaje de tapado estatico, compilado y diseñado por Google, por Robert Griesemer, Rob Pike y Ken Thompson. Sintacticamente es parecido a C, pero tiene muchas ventajas como lo son seguridad de memoria, garbage collection, tapado estructuras y concurrencia.




Go fue escrito en Google originalmente en el 2007, y su principal enfoque era mejorar la productividad a la hora de programar en una era donde se tiene multicore, maquinas en red y grandes bases de codigo. Los diseñadores querían eliminar el criticismo que se tenia en Google sobre el uso de otros lenguages, manteniendo las características buenas de dichos lenguajes. Y la motivation principal fue que a los creadores no les gustaba el lenguaje de programación C++.

Ejemplo de un “Hola Mundo” escrito en Go.

```go
package main
import "fmt"
func main() {
    fmt.Println("hello world")
}
```

# Código

main.go

main.go

```go
)

var router = NewRouter()

func NewRouter() *mux.Router {
    router := mux.NewRouter().StrictSlash(true)
    router.PathPrefix("/css/").Handler(http.StripPrefix("/css/",
    http.FileServer(http.Dir("static/css/"))))
    return router
}

func main() {

  //====== GET METHODS
  router.HandleFunc("/", common.LoginPageHandler)
  router.HandleFunc("/adminPage",common.HomePageHandler)
  router.HandleFunc("/cpu_graph", common.CpuProcessHandler)
  router.HandleFunc("/ram_graph", common.RamProcessHandler)

  //====== POST METHODS
  router.HandleFunc("/login", common.LoginHandler).Methods("POST")
  router.HandleFunc("/ram_data",common.RamData).Methods("POST")
  router.HandleFunc("/cpu_data",common.CpuData).Methods("POST")
  router.HandleFunc("/data",common.AdminHandler).Methods("POST")

  //====== CONFIG
  http.Handle("/",router)
  log.Println("Server running on http://localhost:8080")
  log.Fatal(http.ListenAndServe(":8080", nil))
}

```

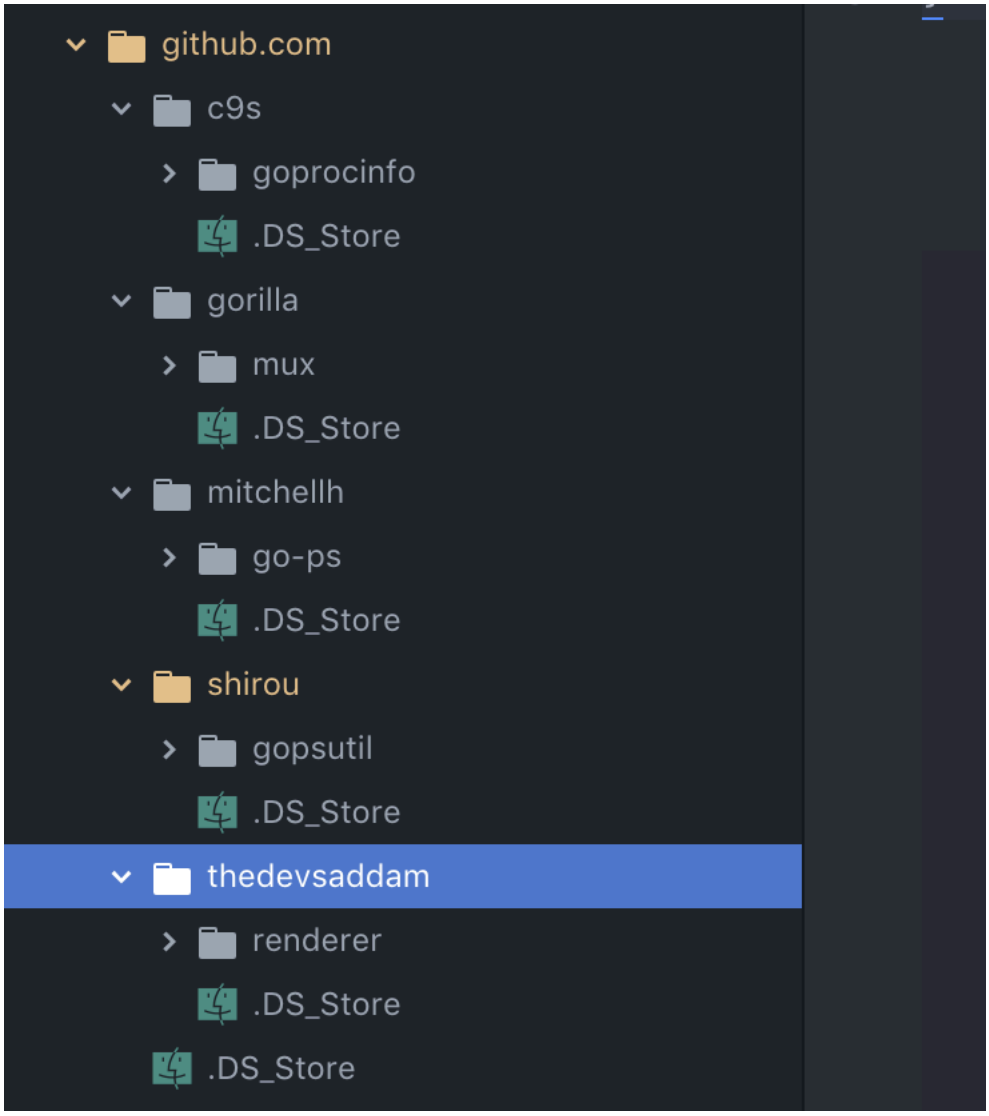
common/Handlers.go

Handlers.go

```go

//======= RAM DATA

type RamStruct struct{
  X int `json:"x"`
  Y int `json:"y"`
  Z int `json:"z"`
}

func RamData(response http.ResponseWriter, request *http.Request){
  memory, _ := linuxproc.ReadMemInfo("/proc/meminfo")
  total:=memory.MemTotal/1000
  consumida:= (memory.MemTotal - memory.MemFree)/1000
  per_consumo := (float64(memory.MemTotal - memory.MemFree)/float64(memory.MemTotal))*100
  ram_data := RamStruct{X:int(total),Y:int(consumida),Z:int(per_consumo)}
  byteArray, _ := json.Marshal(ram_data)
  fmt.Fprintf(response,string(byteArray))
}

//======= CPU DATA

func getCPUSample() (idle, total uint64) {
    contents, err := ioutil.ReadFile("/proc/stat")
    if err != nil {
        return
    }
    lines := strings.Split(string(contents), "\n")
    for _, line := range(lines) {
        fields := strings.Fields(line)
        if fields[0] == "cpu" {
            numFields := len(fields)
```

# Librerias utilizadas

- github.com/gorilla/mux
- github.com/c9s/goprocinfo
- github.com/mitchellh/go-ps
- github.com/thedevsaddam/renderer
- github.com/shirou/gopsutil




# Requerimientos del Sistema

- Procesador: Intel Core 2 Duo o superior.
- Sistema Operativo: Sistema Operativo Windows, Linux o MacOs.
- RAM: 1GB
- Disco Duro: 20GB