Guía Rápida: Introducción a Go y el Framework Gin

# ¿Qué es Go?

Go (también conocido como Golang) es un lenguaje de programación creado por Google. Es compilado, tipado estáticamente y diseñado para el rendimiento, simplicidad y concurrencia.

# ¿Qué es Gin?

Gin es un framework web para Go que permite construir APIs REST de manera rápida, similar a frameworks como Spring Boot en Java, pero con enfoque minimalista y alto rendimiento.

# Estructura Básica de un Proyecto con Gin

package main  
  
import "github.com/gin-gonic/gin"  
  
func main() {  
 r := gin.Default() // Inicializa el router con middleware por defecto  
  
 r.GET("/ping", func(c \*gin.Context) {  
 c.JSON(200, gin.H{"message": "pong"})  
 })  
  
 r.Run(":8080") // Inicia el servidor en el puerto 8080  
}

## Explicación de Componentes

r := gin.Default(): Crea un nuevo router con middleware de logging y recuperación ante pánicos.

r.GET("/ruta", handler): Define una ruta HTTP GET.

\*gin.Context: Objeto que contiene datos de la petición y herramientas para enviar la respuesta.

# Comparación Rápida con Spring Boot

|  |  |  |
| --- | --- | --- |
| Concepto | Spring Boot | Gin (Go) |
| Crear servidor | @RestController | r := gin.Default() |
| Middleware | @Component, Filters | r.Use() |
| Request/Response | HttpServletRequest/Response | \*gin.Context |
| Endpoint | @GetMapping | r.GET |
| JSON | @ResponseBody | c.JSON |

# Ejemplo de Middleware

func AuthMiddleware() gin.HandlerFunc {  
 return func(c \*gin.Context) {  
 token := c.GetHeader("Authorization")  
 if token != "supersecreto" {  
 c.JSON(401, gin.H{"error": "No autorizado"})  
 c.Abort()  
 return  
 }  
 c.Next()  
 }  
}

# Gin Cheat Sheet

- gin.Default() // Crea un router con middleware por defecto  
- r.GET("/path", fn) // Ruta GET  
- r.POST("/path", fn) // Ruta POST  
- r.PUT("/path", fn) // Ruta PUT  
- r.DELETE("/path", fn) // Ruta DELETE  
- c.Param("id") // Parámetro en URL  
- c.Query("q") // Parámetro de query (?q=valor)  
- c.PostForm("name") // Formulario POST  
- c.JSON(200, data) // Responder en JSON  
- c.Abort() // Detener la ejecución del middleware  
- c.Next() // Continuar al siguiente handler