

Lección 2 - Configuración del entorno

Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
 - [Instalación de Gin y configuración de dependencias](#)
 - [Creando la estructura básica de un proyecto Gin](#)
- [Conclusión](#)
- [Conoce más del autor](#)

Introducción

¡En esta clase, nos sumergiremos en la emocionante tarea de configurar nuestro entorno para comenzar a trabajar con el poderoso Framework Gin de Go!

Objetivos

1. **Instalar y Configurar Gin:**
 - Profundizar en la instalación y configuración de Gin en el entorno de desarrollo. Esto incluirá la instalación del paquete Gin y la inicialización de un nuevo proyecto.
 2. **Explorar la Estructura Básica de un Proyecto Gin:**
 - Aprender a familiarizarse con la estructura básica de un proyecto Gin. Esto incluirá la organización de archivos y directorios para una mejor comprensión del flujo de trabajo.
-

Desarrollo

Instalación de Gin y configuración de dependencias

Creamos el directorio `Gin`

```
mkdir Gin
```

Primero, asegurémonos de tener Go instalado en nuestro sistema. Luego, abrimos la terminal y ejecutamos el siguiente comando:

```
$ go version
go version go1.21.0 windows/amd64

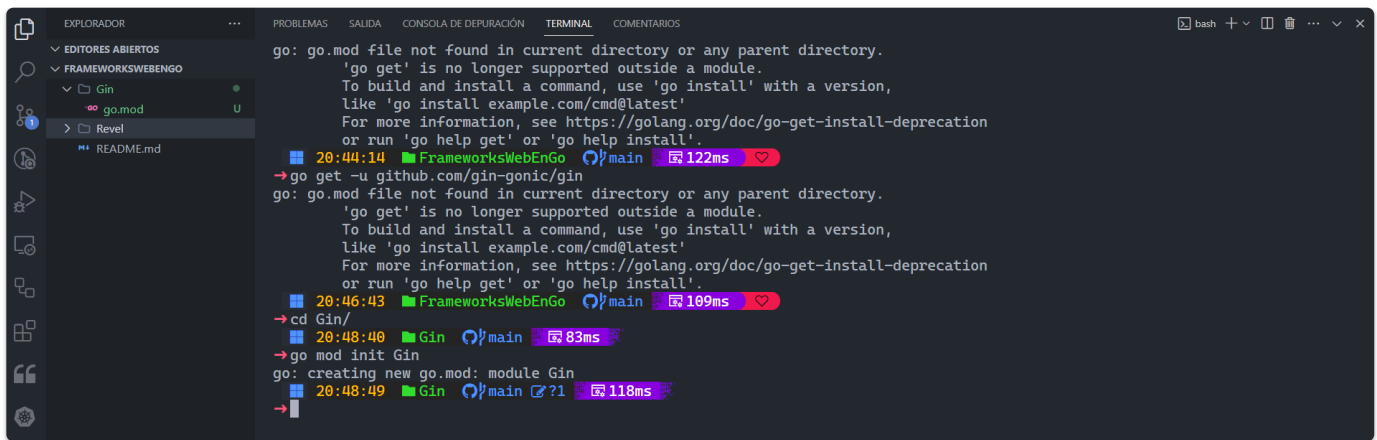
$ go get -u github.com/gin-gonic/gin
go: go.mod file not found in current directory or any parent directory.
      'go get' is no longer supported outside a module.
      To build and install a command, use 'go install' with a version,
      like 'go install example.com/cmd@latest'
      For more information, see https://golang.org/doc/go-get-install-
      deprecation
      or run 'go help get' or 'go help install'.
```

No te preocupes, aquí te explico cómo solucionar ese error: El error indica que estás intentando utilizar el comando `go get` fuera de un módulo Go. A partir de Go 1.16, se recomienda usar `go install` con la especificación de versión para instalar paquetes y comandos.

Para solucionarlo, sigue estos pasos:

- **Crea un módulo Go:**
 - Ubicate en el directorio donde desees crear tu proyecto Gin.
 - ``cd Gin`
 - Ejecuta el comando `go mod init`
`nombre_proyecto` (reemplaza `nombre_proyecto` con el nombre que desees darle a tu módulo). Esto creará un archivo llamado `go.mod` que identifica tu proyecto como un módulo Go.

```
$ cd Gin
$ go mod init Gin
go: creating new go.mod: module Gin
```



```
go: go.mod file not found in current directory or any parent directory.
'go get' is no longer supported outside a module.
To build and install a command, use 'go install' with a version,
like 'go install example.com/cmd@latest'
For more information, see https://golang.org/doc/go-get-install-deprecation
or run 'go help get' or 'go help install'.

20:44:14 FrameworksWebEnGo main 122ms
→go get -u github.com/gin-gonic/gin
go: go.mod file not found in current directory or any parent directory.
'go get' is no longer supported outside a module.
To build and install a command, use 'go install' with a version,
like 'go install example.com/cmd@latest'
For more information, see https://golang.org/doc/go-get-install-deprecation
or run 'go help get' or 'go help install'.

20:46:43 FrameworksWebEnGo main 109ms
→cd Gin/
20:48:40 Gin main 83ms
→go mod init Gin
go: creating new go.mod: module Gin
20:48:49 Gin main 118ms
→
```

- **Instala el comando Gin:**

- Una vez que tienes el módulo Go creado, ejecuta el comando `go install github.com/labstack/echo/v4@latest` para instalar la última versión del comando Gin.

```
$ go install github.com/gin-gonic/gin@latest
go: downloading github.com/gin-gonic/gin v1.9.1
go: downloading github.com/gin-contrib/sse v0.1.0
go: downloading golang.org/x/net v0.10.0
go: downloading github.com/matttn/go-isatty v0.0.19
go: downloading github.com/go-playground/validator/v10 v10.14.0
go: downloading github.com/pelletier/go-toml/v2 v2.0.8
go: downloading github.com/ugorji/go/codec v1.2.11
go: downloading google.golang.org/protobuf v1.30.0
go: downloading gopkg.in/yaml.v3 v3.0.1
go: downloading github.com/gabriel-vasile/mimetype v1.4.2
go: downloading github.com/go-playground/universal-translator v0.18.1
go: downloading github.com/leodido/go-urn v1.2.4
go: downloading golang.org/x/crypto v0.9.0
go: downloading golang.org/x/text v0.9.0
go: downloading github.com/go-playground/locales v0.14.1
package github.com/gin-gonic/gin is not a main package
```

```
$ go get -u github.com/gin-gonic/gin
go: downloading github.com/bytedance/sonic v1.9.1
go: downloading github.com/goccy/go-json v0.10.2
go: downloading github.com/json-iterator/go v1.1.12
go: downloading golang.org/x/sys v0.8.0
...
```

Esto descargará e instalará Gin y sus dependencias en tu proyecto.

Creando la estructura básica de un proyecto Gin

Crear el fichero `main.go` y `routes/routes.go`.

```
$ pwd
/c/Users/Jerson/Documents/FrameworksWebEnGo/Gin

$ touch main.go
$ mkdir routes
$ touch routes/routes.go
```

El fichero `main.go`:

- Se encarga de crear el router.
- Se encarga de iniciar el servidor.
- Importará el paquete `routes` y usará las funciones necesarias para registrar rutas en el router.

El fichero `routes/routes.go`:

- Se encargará de definir las rutas de la aplicación.
- Importa el paquete `gin` y usa sus funciones para definir las rutas.
- Además, define las funciones que se encargan de procesar las solicitudes.

```
Gin
├─ go.mod
├─ go.sum
├─ main.go
└─ routes
    └─ routes.go
```

Conclusión

Con estos pasos, estarán listos para comenzar a trabajar con el Framework Gin de Go. La configuración del entorno es crucial para una experiencia de desarrollo suave, y Gin facilita este proceso para que pueda concentrarse en la creación de aplicaciones web rápidas y eficientes. ¡Disfruten explorando las capacidades de Gin en su próximo proyecto!

Conoce más del autor

¡Encuéntrame en las siguientes redes sociales para estar al tanto de mis proyectos y actividades!

 Red Social	 Enlace
 Página web	jersonmartinez.com
 LinkedIn	Jerson Martínez - DevOps Engineer
 Canales de YouTube	DevOpsea Side Master
 GitHub	Perfil en GitHub
 Twitter (X)	@antoniomorenosm