

# Lección 4 - Crear rutas y manejar peticiones

## Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
  - [Crear Rutas y Manejar Peticiones](#)
  - [Separar el Manejo de Rutas](#)
- [Conclusión](#)
- [Conoce más del autor](#)

## Introducción

En esta lección, nos enfocaremos en cómo crear rutas y manejar peticiones utilizando las funcionalidades proporcionadas por el framework Gorilla en Go. Empezaremos definiendo rutas estáticas y avanzaremos hacia rutas dinámicas que acepten parámetros. También aprenderemos a modularizar el manejo de rutas en un archivo independiente para mantener nuestro código organizado y escalable.

---

## Objetivos

### 1. Definir Rutas y Controladores Básicos:

- Comenzaremos definiendo rutas estáticas en el archivo `main.go` utilizando las funciones de enrutamiento de Gorilla. Esto establecerá el flujo básico de nuestra aplicación y nos permitirá responder a las solicitudes HTTP de manera simple.

### 2. Manejar Rutas Dinámicas con Parámetros:

- Avanzaremos hacia la definición de rutas dinámicas que acepten parámetros en la URL. Aprenderemos a extraer estos parámetros y utilizarlos en nuestros controladores para generar respuestas personalizadas basadas en la entrada del usuario.

### 3. Separar el Manejo de Rutas en un Archivo Independiente:

- Para mejorar la organización y mantenibilidad de nuestro código, moveremos el manejo de rutas a un archivo separado llamado `routes/routes.go`. Esto nos permitirá tener un mejor control sobre nuestras rutas y facilitará la expansión de nuestra aplicación en el futuro.

## Desarrollo

### Crear Rutas y Manejar Peticiones

En el archivo `main.go`, definiremos rutas y controladores utilizando las funciones proporcionadas por Gorilla. Por ejemplo:

```
package main

import (
    "fmt"
    "net/http"

    "github.com/gorilla/mux"
)

func main() {
    // Crear un enrutador Gorilla
    r := mux.NewRouter()

    // Ruta básica
    r.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintln(w, "¡Hola, mundo!")
    })

    // Ruta con parámetros
    r.HandleFunc("/saludo/{nombre}", func(w http.ResponseWriter, r
    *http.Request) {
        vars := mux.Vars(r)
        nombre := vars["nombre"]
        fmt.Fprintf(w, "¡Hola, %s!", nombre)
    })

    // Iniciar el servidor en el puerto 8080
    http.ListenAndServe(":8080", r)
}
```

### Separar el Manejo de Rutas

Trabajaremos sobre el archivo `routes/routes.go`:

```

package routes

import (
    "fmt"
    "net/http"

    "github.com/gorilla/mux"
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(r *mux.Router) {
    // Ruta básica
    r.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        fmt.Fprintln(w, "¡Hola, mundo!")
    })

    // Ruta con parámetros
    r.HandleFunc("/saludo/{nombre}", func(w http.ResponseWriter, r
    *http.Request) {
        vars := mux.Vars(r)
        nombre := vars["nombre"]
        fmt.Fprintf(w, "¡Hola, %s!", nombre)
    })
}

```

Modificar el archivo `main.go` para usar el manejador de rutas separado:

```

package main

import (
    "net/http"
    "github.com/gorilla/mux"
    "Gorilla/routes" // Importar el paquete de rutas
)

func main() {
    // Crear un enrutador Gorilla
    r := mux.NewRouter()

    // Configurar rutas
    routes.SetupRoutes(r)
}

```

```
// Iniciar el servidor en el puerto 8080
http.ListenAndServe(":8080", r)
}
```

```
$ go run main.go
```

Probar en el navegador:

```
127.0.0.1:8080/saludo/Jerson
```

### Subir los cambios a GitHub:

```
$ git add .
$ git commit -m "Crear rutas y manejar peticiones"
$ git push
```

Revisar en la web de GitHub que las actualizaciones estén disponibles. Detallaré mayor contenido sobre cada lección.

---

## Conclusión


En esta lección, hemos aprendido cómo crear rutas y manejar peticiones utilizando el framework Gorilla en Go. Desde rutas estáticas hasta rutas dinámicas que aceptan parámetros, hemos explorado diferentes aspectos del enrutamiento web. Además, hemos modularizado el manejo de rutas para mejorar la organización de nuestro código y facilitar futuras expansiones de la aplicación. Con estos conocimientos, estamos preparados para desarrollar aplicaciones web más complejas y escalables con Gorilla.

---

## Conoce más del autor

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

 Red Social	 Enlace
 Página web	<a href="https://jersonmartinez.com">jersonmartinez.com</a>
 LinkedIn	<a href="#">Jerson Martínez - DevOps Engineer</a>
 Canales de YouTube	<a href="#">DevOpsea</a>   <a href="#">Side Master</a>

 <b>Red Social</b>	 <b>Enlace</b>
 GitHub	<a href="#">Perfil en GitHub</a>
 Twitter (X)	<a href="#">@antoniomorenosm</a>