

# Lección 4 - Crear rutas y manejar peticiones

## Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
  - [Definir Rutas y Controladores Básicos](#)
  - [Manejar Rutas Dinámicas con Parámetros](#)
  - [Separar el Manejo de Rutas en un Archivo Independiente](#)
  - [Optimización de controladores para las rutas definidas](#)
  - [Llamada de las rutas desde `main.go`](#)
- [Conclusión](#)
- [Conoce más del autor](#)

## Introducción

En esta lección, exploraremos cómo crear rutas y manejar peticiones HTTP utilizando el framework `Fiber` en Go. Aprenderemos a definir rutas estáticas y dinámicas, así como a trabajar con controladores para responder a las solicitudes de manera eficiente.

---

## Objetivos

### 1. Definir Rutas y Controladores Básicos:

- Comenzaremos definiendo rutas estáticas en el archivo `main.go` utilizando las funciones de enrutamiento de `Fiber`. 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

### Definir Rutas y Controladores Básicos

Comencemos definiendo rutas estáticas en nuestro archivo `main.go`. Utilizaremos las funciones de enrutamiento de Fiber para configurar estas rutas y proporcionar controladores básicos para manejar las solicitudes.

Ejemplo básico de definición de rutas y controladores en `main.go`:

```
package main

import (
    "github.com/gofiber/fiber/v2"
)

func main() {
    // Crear una nueva instancia de Fiber
    app := fiber.New()

    // Definir una ruta estática para la página de inicio
    app.Get("/", func(c *fiber.Ctx) error {
        return c.SendString("¡Bienvenido a mi aplicación con Fiber!")
    })

    // Definir una ruta estática para la página "acerca de"
    app.Get("/about", func(c *fiber.Ctx) error {
        return c.SendString("Página de información sobre nuestra aplicación.")
    })

    // Definir una ruta estática para la página de contacto
    app.Get("/contacto", func(c *fiber.Ctx) error {
        return c.SendString("Página de contacto para nuestra aplicación.")
    })
}
```

```
// Iniciar el servidor en el puerto 3000
app.Listen(":3000")
}
```

En este ejemplo, utilizamos el método `Get` de la instancia de Fiber (`app`) para definir rutas estáticas para la página de inicio, la página "acerca de" y la página de contacto. Cada ruta tiene asociada una función de controlador que recibe un contexto (`*fiber.Ctx`) y devuelve un error. Dentro de estas funciones de controlador, utilizamos el método `SendString` del contexto para enviar una respuesta al cliente con un mensaje de texto plano.

## Manejar Rutas Dinámicas con Parámetros

Ahora, avanzaremos hacia la definición de rutas dinámicas que acepten parámetros en la URL. Esto nos permitirá crear rutas más flexibles que puedan responder a diferentes solicitudes basadas en la entrada del usuario.

```
// Definir una ruta dinámica que acepte un parámetro "nombre"
app.Get("/saludo/:nombre", func(c *fiber.Ctx) error {
    // Obtener el parámetro "nombre" de la URL
    nombre := c.Params("nombre")

    // Enviar un saludo personalizado al usuario
    return c.SendString("¡Hola, " + nombre + "!")
})
```

En este ejemplo, utilizamos el prefijo `:` en el segmento de la ruta para indicar que estamos definiendo un parámetro dinámico llamado "nombre". Dentro de la función de controlador, utilizamos el método `Params` del contexto para obtener el valor del parámetro "nombre" de la URL y luego enviamos una respuesta al cliente con un saludo personalizado.

## 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.

```
// routes/routes.go
package routes

import "github.com/gofiber/fiber/v2"
```

```
// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(app *fiber.App) {
    // Definir las rutas estáticas en el servidor Fiber
    app.Get("/", func(c *fiber.Ctx) error {
        return c.SendString("¡Bienvenido a mi aplicación Fiber!")
    })

    app.Get("/about", func(c *fiber.Ctx) error {
        return c.SendString("Página de información sobre nuestra
aplicación.")
    })

    app.Get("/contacto", func(c *fiber.Ctx) error {
        return c.SendString("Página de contacto para nuestra
aplicación.")
    })

    // Definir la ruta dinámica para el saludo
    app.Get("/saludo/:nombre", func(c *fiber.Ctx) error {
        nombre := c.Params("nombre")
        return c.SendString("¡Hola, " + nombre + "!")
    })
}
```

## Optimización de controladores para las rutas definidas

```
package routes

import "github.com/gofiber/fiber/v2"

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(app *fiber.App) {
    // Definir rutas aquí...
    app.Get("/", handlerInicio)
    app.Get("/about", handlerAcercaDe)
    app.Get("/saludo/:nombre", handlerSaludo)
}

// Controladores para las rutas definidas

func handlerInicio(c *fiber.Ctx) error {
    return c.SendString("¡Bienvenido a mi aplicación con Fiber!")
}
```

```
func handlerAcercaDe(c *fiber.Ctx) error {
    return c.SendString("Acerca de nosotros: Somos una aplicación
construida con Fiber.")
}

func handlerSaludo(c *fiber.Ctx) error {
    nombre := c.Params("nombre")
    return c.SendString("¡Hola, " + nombre + "!")
}
```

Definimos funciones de controlador separadas para cada ruta, lo que hace que nuestro código sea más modular y fácil de mantener.

### Llamada de las rutas desde `main.go`

```
package main

import (
    "github.com/gofiber/fiber/v2"
    "Fiber/routes" // Importa el paquete de rutas
)

func main() {
    // Crear una nueva instancia de Fiber
    app := fiber.New()

    // Configurar las rutas utilizando el paquete de rutas
    routes.SetupRoutes(app)

    // Iniciar el servidor en el puerto 3000
    app.Listen(":3000")
}
```

## Conclusión








En esta lección, hemos aprendido cómo crear rutas y manejar peticiones HTTP utilizando el framework `Fiber` en Go. A través de ejemplos prácticos, hemos explorado cómo definir rutas estáticas y dinámicas, así como separar el manejo de rutas en un archivo independiente para

mejorar la organización de nuestro código. Estas habilidades son fundamentales para construir aplicaciones web escalables y mantenibles con `Fiber`.

---

## Conoce más del autor

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

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