

# Lección 6 - Adaptación de plantilla completa con diferentes páginas

## Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
  - [Manejar Rutas Dinámicas](#)
- [Conclusión](#)
- [Conoce más del autor](#)

## Introducción

En esta lección, exploraremos cómo integrar plantillas HTML y cargar contenido estático en una aplicación web utilizando el framework Echo de Go. La capacidad de mostrar información dinámica y servir recursos estáticos es esencial para crear páginas web completas y atractivas.

---

## Objetivos

1. **Configurar el Motor de Renderizado de Plantillas HTML:**
    - Profundizar en la configuración del motor de renderizado de plantillas HTML en Echo. Aprenderemos cómo cargar y renderizar plantillas HTML para crear páginas dinámicas.
  2. **Servir Contenido Estático:**
    - Aprender a servir contenido estático, como archivos CSS, JavaScript e imágenes, utilizando Echo. Esto mejorará la apariencia y la interactividad de la aplicación.
  3. **Manejar Rutas Dinámicas:**
    - Implementar rutas dinámicas en Echo que acepten diferentes parámetros y carguen páginas específicas según la solicitud del usuario.
- 

## Desarrollo

**Buscar en Google:** StartBootstrap SB Admin 2

Descargar el repositorio:

- [GitHub - StartBootstrap/startbootstrap-sb-admin-2: A free, open source, Bootstrap admin theme created by Start Bootstrap](https://github.com/StartBootstrap/startbootstrap-sb-admin-2)

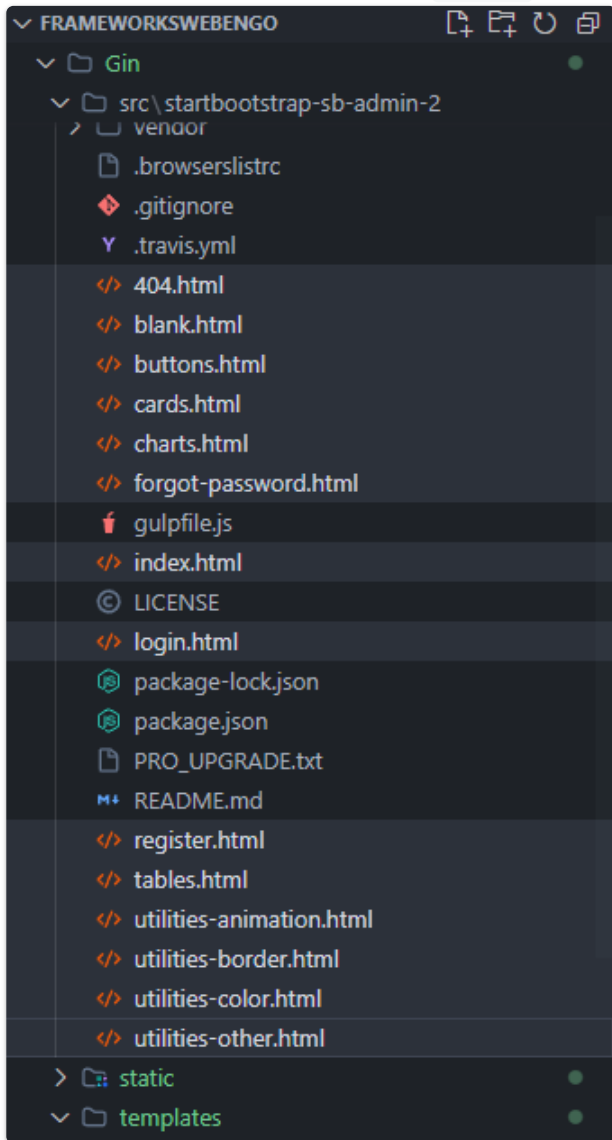
Creamos un directorio `/Echo/src/` para hacer el `git clone` del repositorio.

```
$ mkdir src
$ cd src

$ git clone https://github.com/StartBootstrap/startbootstrap-sb-admin-2.git
Cloning into 'startbootstrap-sb-admin-2'...
remote: Enumerating objects: 8826, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 8826 (delta 8), reused 11 (delta 4), pack-reused 8810
Receiving objects: 100% (8826/8826), 28.10 MiB | 6.81 MiB/s, done.
Resolving deltas: 100% (3300/3300), done.
```

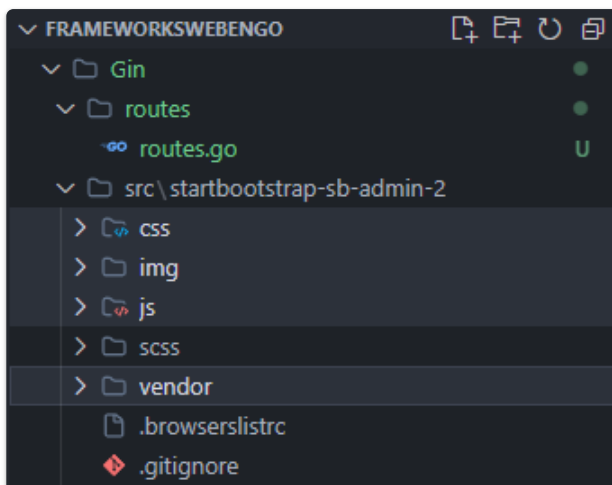
Al `index.html` que teníamos, le llamaremos `index-old.html`, para no colicionar con el `index.html` de la plantilla.

Seleccionar todos los ficheros `.html` y copiarlos al directorio `templates\`:



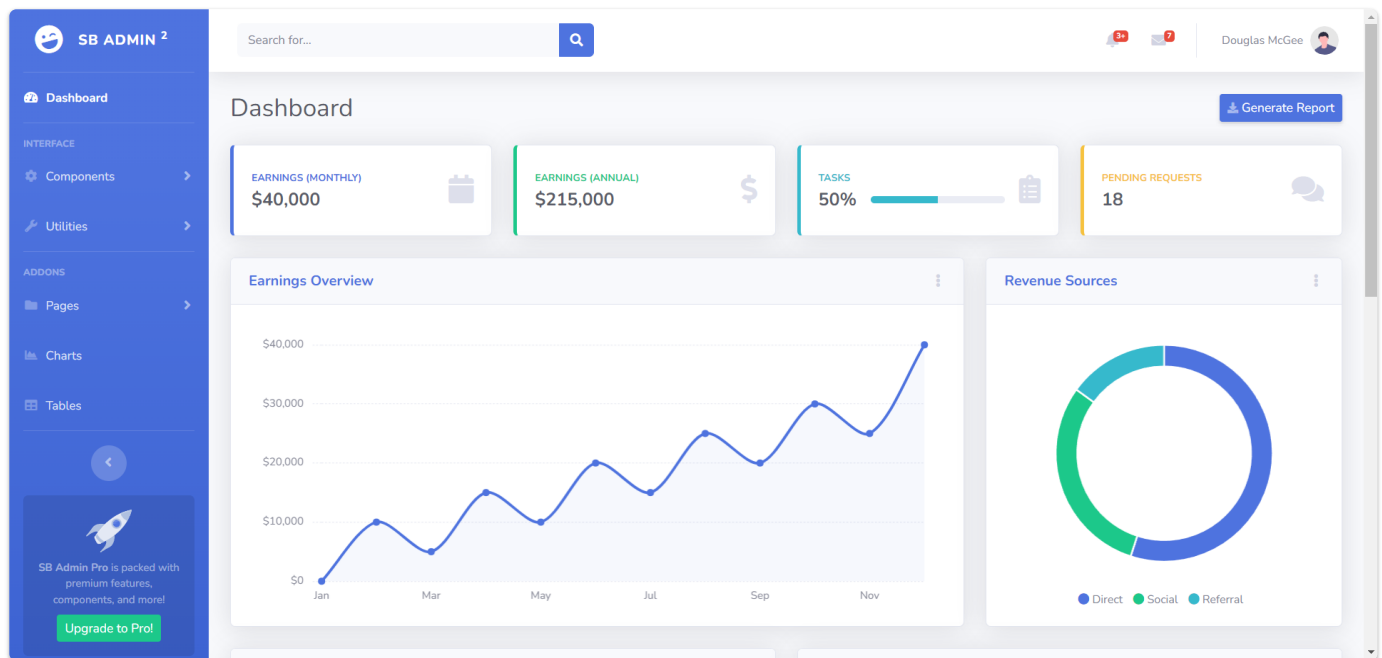
Antes también renombramos los directorios `css` y `js` a `css-old` y `js-old`, para proceder a copiar los demás recursos.

Copiar los directorios `css`, `img`, `js` y `vendor` al directorio `static`.



En todos los ficheros `.html` que se han colocado dentro del directorio templates, hay que modificar las rutas de llamadas de los recursos estáticos.

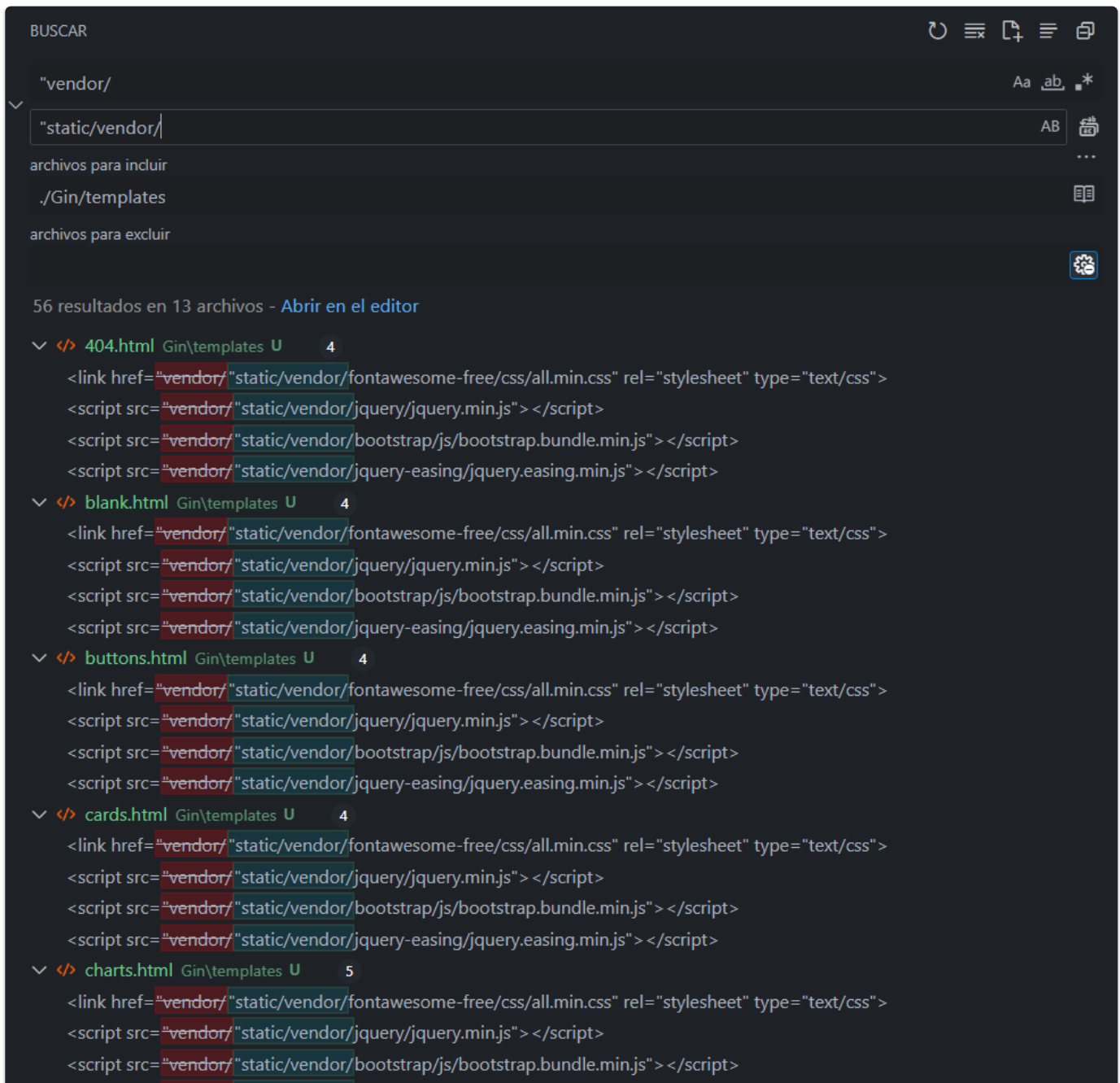
Empezaremos con el `index.html`:Anteponer a todas las direcciones `/static/`, tanto a los `vendor/`, `css/`, `js/`, `img/`.



Ahora hay que hacer lo mismo para los demás.

Buscar en todos los recursos dentro del directorio templates el siguiente patrón:

- `"vendor/` y reemplazar por `"/static/vendor/`.
- `"css/` y reemplazar por `"/static/css/`.
- `"js/` y reemplazar por `"/static/js/`.
- `"img/` y reemplazar por `"/static/img/`.



Finalmente, eliminamos el directorios de recursos, src, ya que no es necesario mantenerlo.

```
rm -rf src
```

## Manejar Rutas Dinámicas

Ahora, configuraremos Echo para manejar rutas dinámicas que acepten diferentes parámetros y carguen páginas específicas según la solicitud del usuario.

```
package routes
```

```
import (
```

```

"net/http"
"os"
"strings"

"github.com/labstack/echo/v4"
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(e *echo.Echo) {
    // Configurar el motor de renderizado de plantillas HTML
    e.Renderer = Renderer()

    // Configurar Echo para servir contenido estático
    e.Static("/static", "static")

    // Configurar la ruta para cargar la página principal
    e.GET("/", func(c echo.Context) error {
        // Renderizar la plantilla index.html con los datos proporcionados
        return c.Render(http.StatusOK, "index.html",
            map[string]interface{}{
                "Title":    "Mi Aplicación",
                "Heading": "¡Hola, mundo!",
                "Message": "Bienvenido a mi aplicación web con Echo y
plantillas HTML.",
            })
    })

    // Configurar Echo para manejar rutas dinámicas
    e.GET("/:page", func(c echo.Context) error {
        page := c.Param("page")

        // Verificar si la página ya tiene la extensión ".html"
        if !strings.HasSuffix(page, ".html") {
            page += ".html"
        }

        // Comprobar si la plantilla solicitada existe
        if _, err := os.Stat("templates/" + page); err == nil {
            // Si la plantilla existe, renderizarla
            return c.Render(http.StatusOK, page, nil)
        }

        // Si la plantilla no existe, mostrar página de error 404
    })
}

```

```

        return c.Render(http.StatusNotFound, "404.html", nil)
    })
}

// Renderer configura el motor de renderizado de plantillas HTML
func Renderer() *TemplateRenderer {
    return &TemplateRenderer{
        templates: template.Must(template.ParseGlob("templates/*.html")),
    }
}

// TemplateRenderer es una estructura que se utiliza para renderizar
plantillas HTML
type TemplateRenderer struct {
    templates *template.Template
}

// Render es un método de TemplateRenderer que renderiza una plantilla
HTML
func (t *TemplateRenderer) Render(w io.Writer, name string, data
interface{}, c echo.Context) error {
    return t.templates.ExecuteTemplate(w, name, data)
}

```

Comprobamos que todas las páginas estén funcionando correctamente.

### Subir los cambios a GitHub:

```

$ git add .
$ git commit -m "Adaptación de plantilla completa con diferentes páginas
utilizando Echo"
$ 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 integrar plantillas HTML, cargar contenido estático y manejar rutas dinámicas en una aplicación web utilizando el framework Echo de Go. Estas

habilidades nos permiten crear páginas web completas y personalizadas que pueden adaptarse a las necesidades y preferencias de nuestros usuarios. Con estas bases, estamos preparados para desarrollar aplicaciones web más complejas y funcionales en Go utilizando Echo.

---

## 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>