

# Lección 6 - Adaptación de plantilla completa con diferentes páginas

## Tabla de contenido

- [Introducción](#)
- [Objetivos](#)
- [Desarrollo](#)
  - [Descargar una Plantilla Completa HTML](#)
- [Conclusión](#)
- [Conoce más del autor](#)

## Introducción

En esta lección, exploraremos cómo integrar una plantilla HTML completa con diferentes páginas en una aplicación web utilizando el framework Gorilla de Go. Esto nos permitirá crear una aplicación web completa con múltiples vistas y un diseño coherente.

---

## Objetivos

### 1. **Configurar el Motor de Renderizado de Plantillas HTML:**

- Profundizar en la configuración del motor de renderizado de plantillas HTML en Gorilla. Aprenderemos cómo cargar y renderizar plantillas HTML para diferentes páginas de nuestra aplicación.

### 2. **Servir Contenido Estático:**

- Aprender a servir contenido estático, como archivos CSS, JavaScript e imágenes, utilizando Gorilla. Esto mejorará la apariencia y la interactividad de la aplicación.

### 3. **Manejar Rutas Dinámicas:**

- Implementar rutas dinámicas en Gorilla que acepten diferentes parámetros y carguen páginas específicas según la solicitud del usuario.

## Desarrollo

### Descargar una Plantilla Completa HTML

- Buscaremos una plantilla HTML completa en Internet que se adapte a nuestras necesidades. En este ejemplo, utilizaremos una plantilla de Start Bootstrap llamada "SB Admin 2".

## 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)

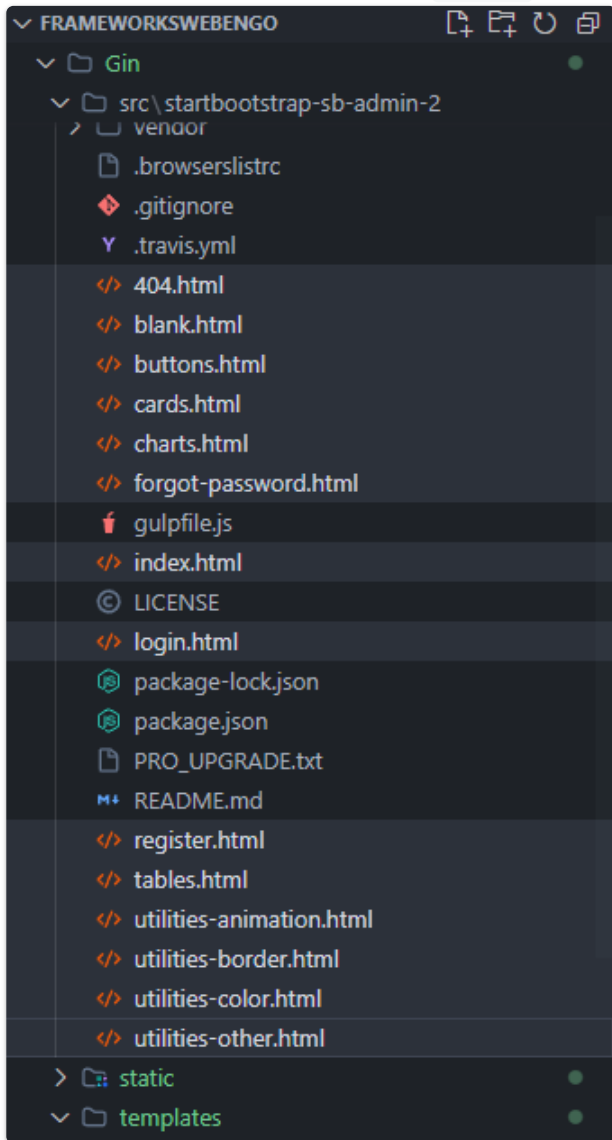
Creamos un directorio `/Gorilla/src/` para hacer el `git clone` del repositorio.

```
$ 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.
```

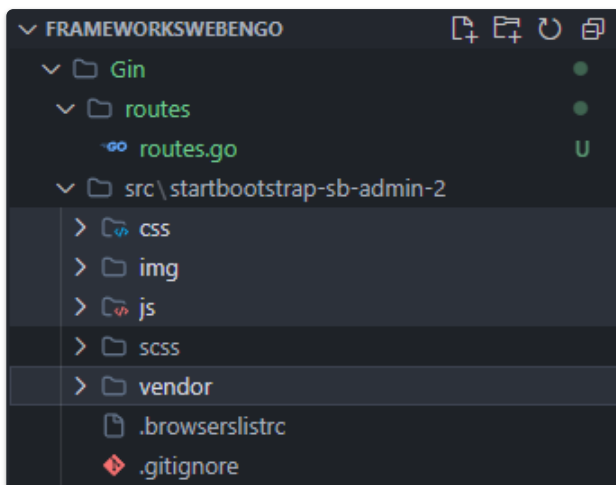
Al `index.html` que teníamos, le llamaremos `index-old.html`, para no colicionar con el `index.html` de la plantilla.

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



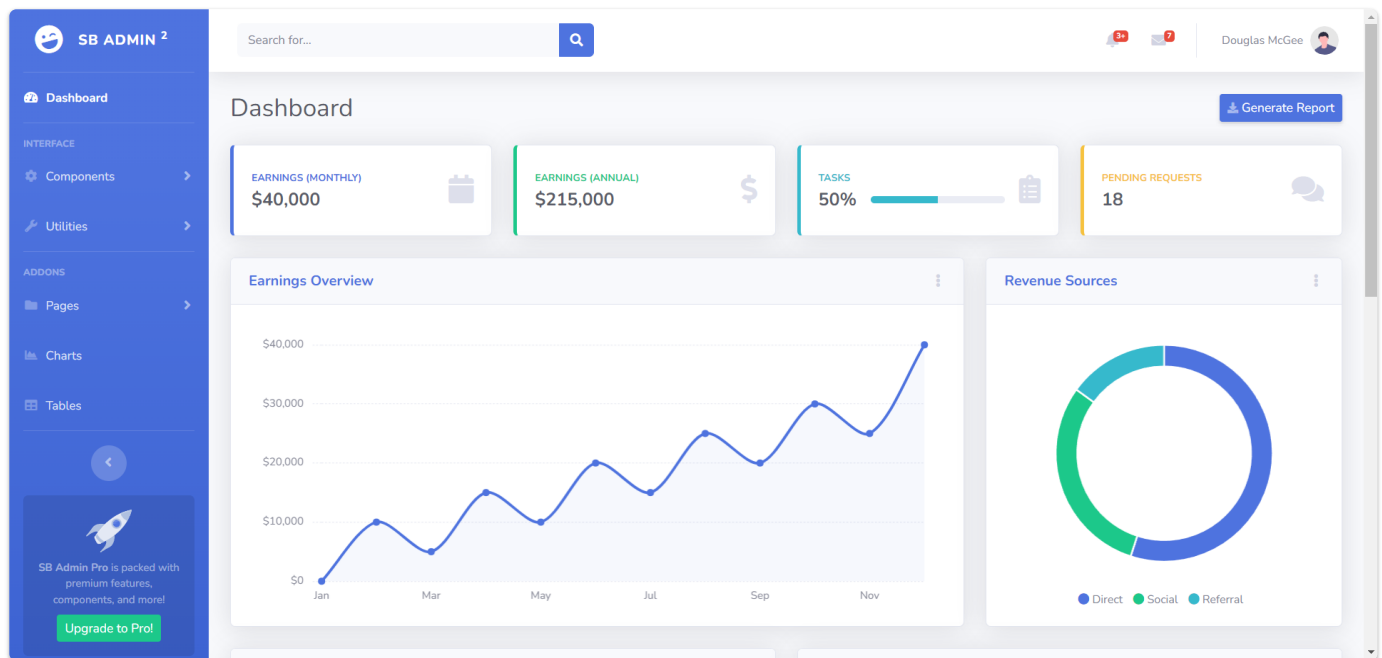
Antes también renombramos los directorios `css` y `js` a `css-old` y `js-old`, para proceder a copiar los demás recursos.

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.

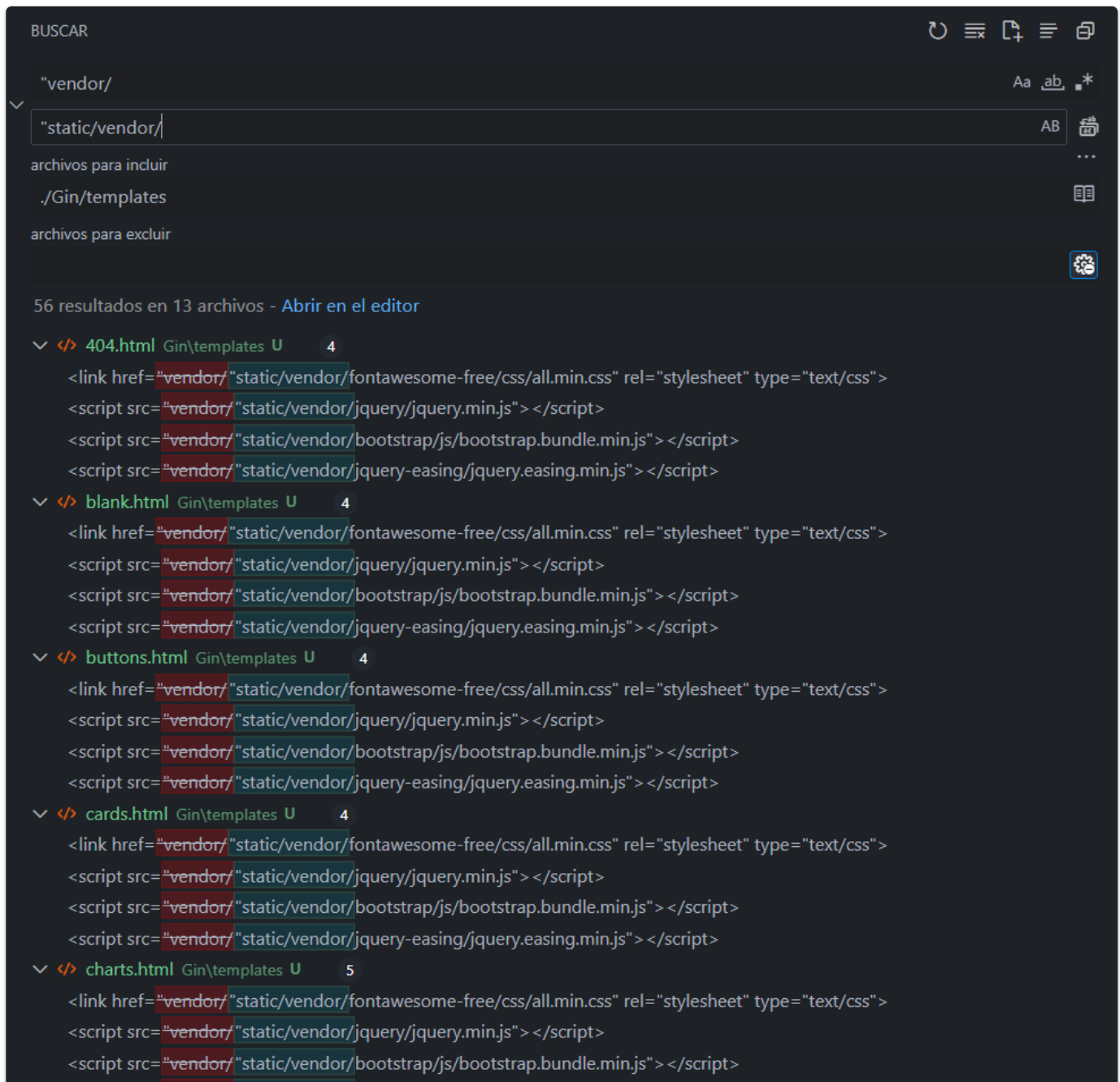
Empezaremos con el `index.html`:Anteponer a todas las direcciones `/static/`, tanto a los `vendor/`, `css/`, `js/`, `img/`.



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
```

```
package routes
```

```
import (  
    "html/template"  
    "io"  
    "net/http"  
    "os"  
    "strings"
```

```

    "github.com/gorilla/mux"
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(r *mux.Router) {
    // Configurar el motor de renderizado de plantillas HTML
    renderer := Renderer()

    // Configurar la ruta para cargar la página principal
    r.HandleFunc("/", renderTemplate(renderer, "index.html"))

    // Configurar Gorilla para manejar rutas dinámicas
    r.HandleFunc("/{page}", dynamicPageHandler(renderer)).Methods("GET")

    // Configurar Gorilla para servir contenido estático
    r.PathPrefix("/static/").Handler(http.StripPrefix("/static/",
    http.FileServer(http.Dir("static"))))
}

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

// Función para renderizar una plantilla HTML
func renderTemplate(renderer *template.Template, templateName string)
http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        err := renderer.ExecuteTemplate(w, templateName, nil)
        if err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
        }
    }
}

// Función para manejar páginas dinámicas
func dynamicPageHandler(renderer *template.Template) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        vars := mux.Vars(r)
        page := vars["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
        err := renderer.ExecuteTemplate(w, page, nil)
        if err != nil {
            http.Error(w, err.Error(), http.StatusInternalServerError)
        }
        return
    }

    // Si la plantilla no existe, mostrar página de error 404
    http.Error(w, "Página no encontrada", http.StatusNotFound)
}
}

```

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 Gorilla"
$ 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 adaptar una plantilla HTML completa con diferentes páginas utilizando el framework Gorilla de Go. Configuramos Gorilla para manejar las rutas dinámicas y servir contenido estático, lo que nos permitió crear una aplicación web completa con múltiples vistas. Con este conocimiento, estamos preparados para desarrollar aplicaciones web más complejas y personalizadas en Go utilizando Gorilla.

---

## 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>