Lección 7 - Adaptación de plantilla completa con diferentes páginas

Tabla de contenido

Introducción

En esta clase, exploraremos cómo incorporar datos dinámicos en nuestras aplicaciones web utilizando el poderoso Framework Gin de Go. La capacidad de mostrar información variable y actualizada en nuestras páginas web es esencial para proporcionar experiencias interactivas y personalizadas a los usuarios.

Objetivos

1. **Comprender la Estructura de un Controlador con Datos Dinámicos:**
   - Profundizar en la estructura de un controlador Gin que maneje datos dinámicos. Exploraremos cómo recibir parámetros, procesarlos y pasarlos a las plantillas HTML para un renderizado dinámico.
2. **Implementar una Ruta con Datos Variables:**
   - Aprender a implementar una ruta en Gin que acepte datos variables, como parámetros de URL. Esto permitirá crear páginas dinámicas basadas en la entrada del usuario.
3. **Utilizar el Motor de Renderizado de Gin de Forma Eficiente:**
   - Comprender las mejores prácticas para utilizar el motor de renderizado de Gin de manera eficiente al incorporar datos dinámicos. Exploraremos cómo estructurar las plantillas HTML y pasar datos desde el controlador.

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

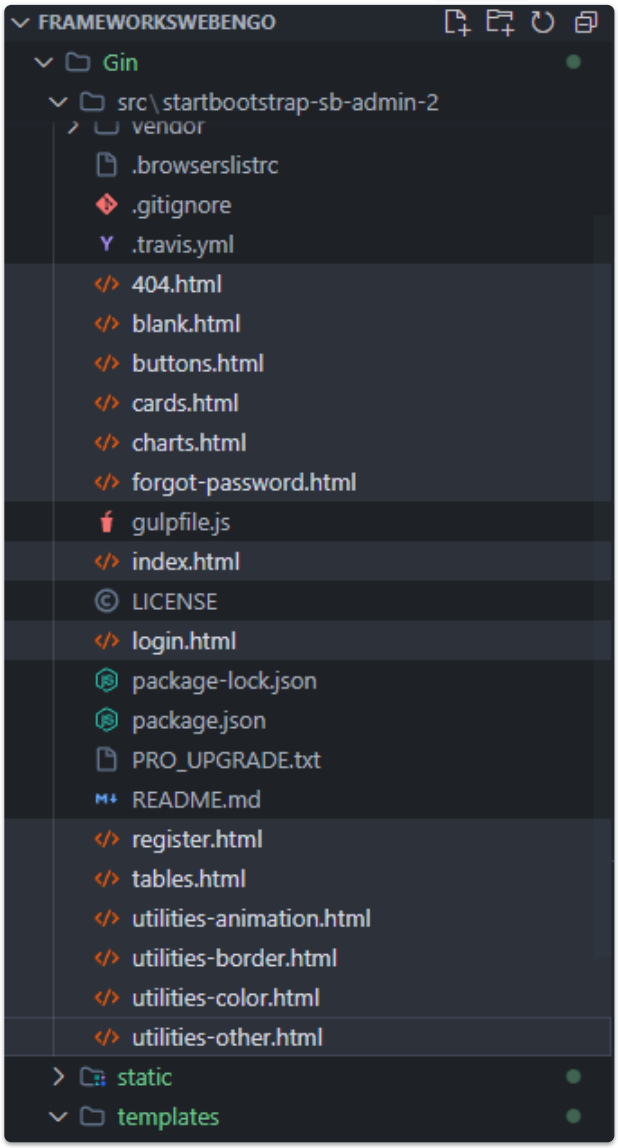
Creamos un directorio `/Gin/src/` para hacer el `git clone` del repositorio.

```
$ 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.
```

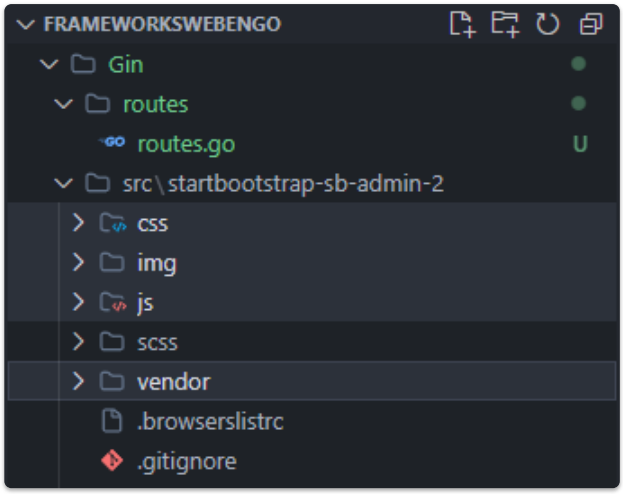
Al `index.html` que teníamos, le llamaremos `index-old.html`, para no colicionar con el index.html de la plantilla.

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



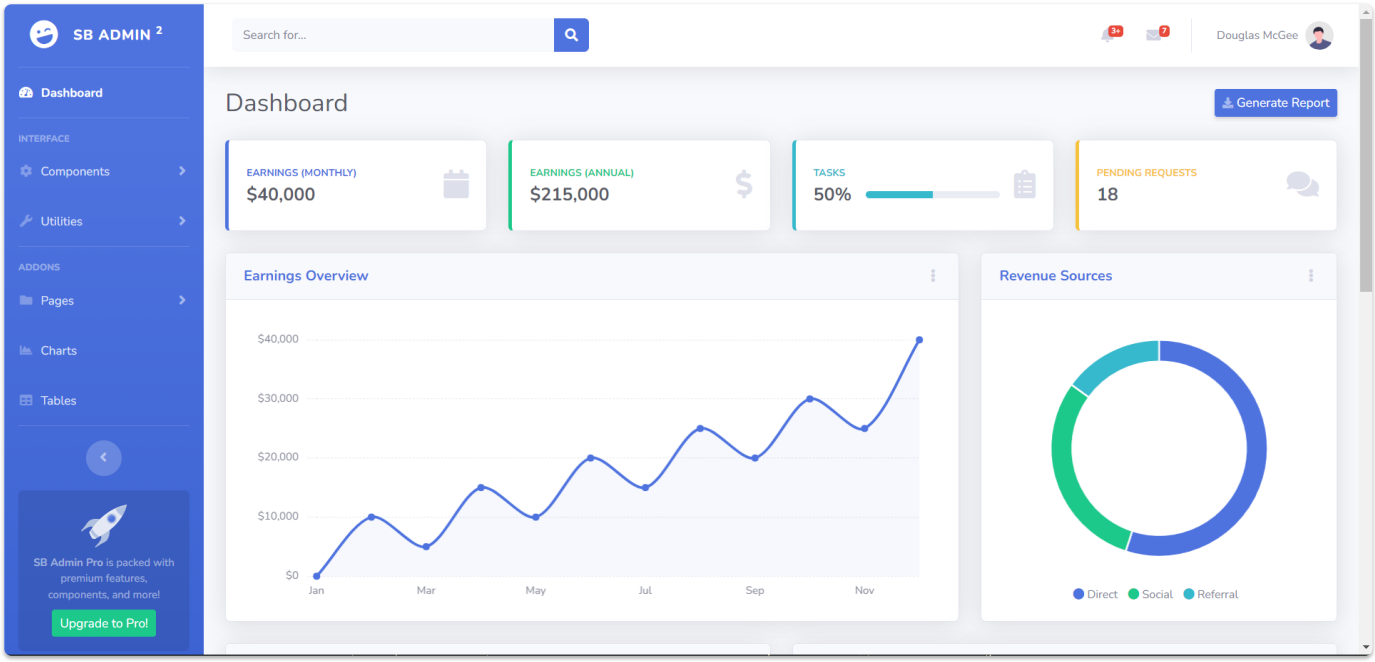
Antes también renombramos los directorios `css` y `js` a `css-old` y `js-old`, para proceder a copiar los demás recursos.

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.

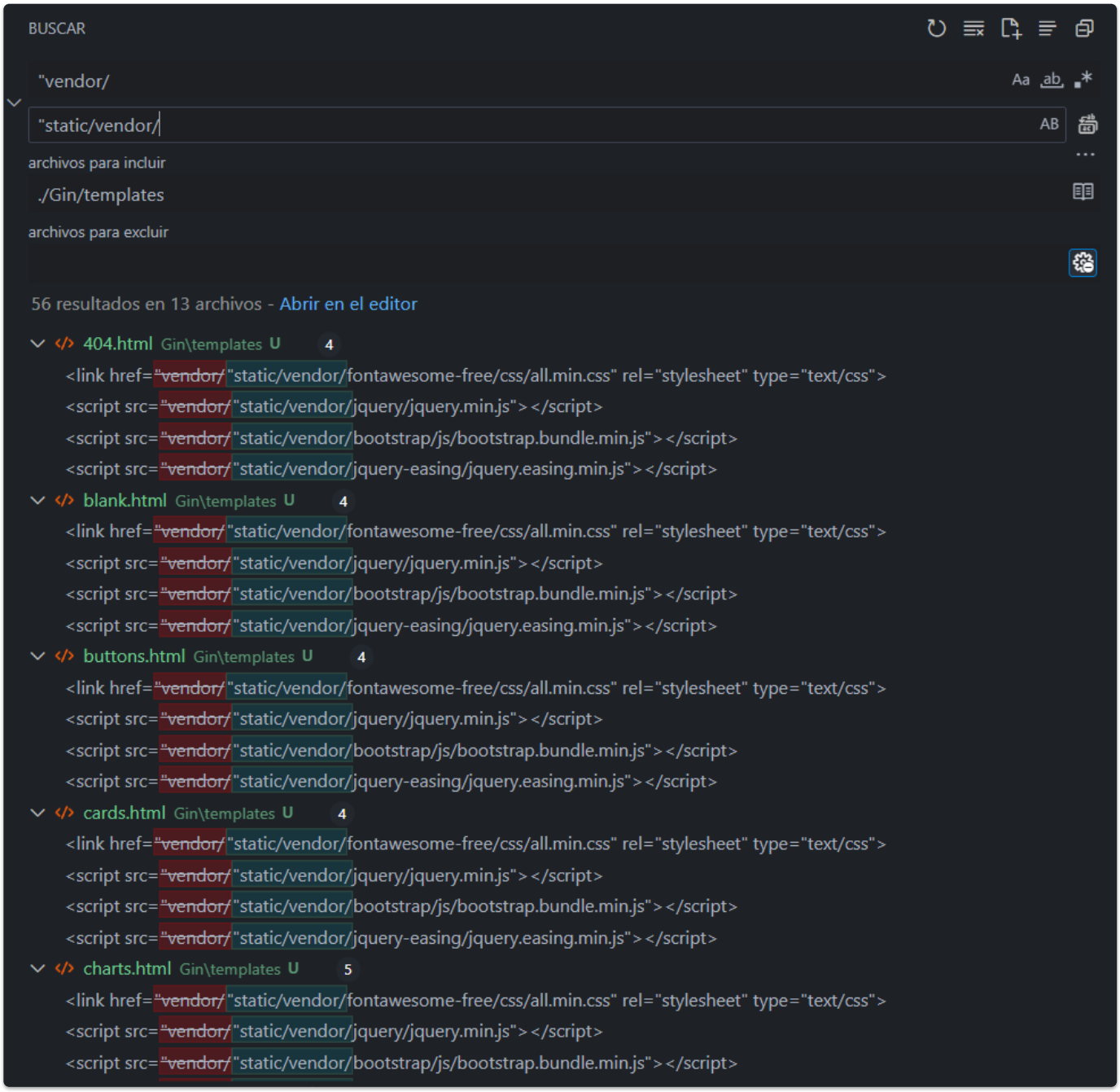
Empezaremos con el `index.html`:Anteponer a todas las direcciones `/static/`, tanto a los `vendor/`, `css/`, `js/`, `img/`.



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
```

Comprender la Estructura de un Controlador con Datos Dinámicos

```
package routes

import (
        // Se importan los paquetes necesarios para el manejo de rutas y
solicitudes HTTP:
    "net/http"  //Para trabajar con protocolo HTTP y códigos de estado.
```

```
    "os" // Para interactuar con el sistema operativo y verificar la
existencia de archivos.
    "strings" //Para manipulación de cadenas.
    "github.com/gin-gonic/gin" //El paquete Gin para configurar y manejar
rutas en la aplicación.
)

// SetupRoutes configura las rutas de la aplicación
func SetupRoutes(r *gin.Engine) {
    // Configurar Gin para servir contenido estático
    r.Static("/static", "./static")

    // Configurar la ubicación de los archivos de plantillas HTML
    r.LoadHTMLGlob("templates/*.html")

    // Configurar la ruta para cargar la página principal

    r.GET("/", func(c *gin.Context) {
        // Renderizar la página de inicio
        c.HTML(http.StatusOK, "index.html", nil)
    })

        // Ruta para manejar todas las solicitudes de página
    r.GET("/:page", func(c *gin.Context) {
        page := c.Param("page") // Se obtiene el parámetro que se ha
pasado en :page

        // Verificar si la página ya tiene la extensión ".html"
        if !strings.HasSuffix(page, ".html") {
            page += ".html" // Agregar la extensión ".html" si no está
presente
        }

        // Comprobar si la plantilla solicitada existe
        if _, err := os.Stat("templates/" + page); err == nil {
            // Si la plantilla existe, renderizarla
            c.HTML(http.StatusOK, page, nil)
        } else {
            // Si la plantilla no existe, mostrar página de error 404
            c.HTML(http.StatusNotFound, "404.html", nil)
        }
    })
}
```

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"
$ 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 configurar rutas dinámicas en una aplicación web utilizando el framework Gin en Go. Hemos explorado cómo servir contenido estático, cargar plantillas HTML y manejar diferentes solicitudes de manera eficiente. Al comprender cómo configurar y gestionar las rutas en Gin, estamos mejor preparados para construir aplicaciones web más complejas y dinámicas en Go. Con estas habilidades, podemos crear aplicaciones web escalables y robustas que cumplan con los requisitos de nuestros usuarios de manera efectiva.

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