INFORME DE PRÁCTICAS

Ingeniería Inversa

Desarrollo de Aplicaciones Web 2

Universidad de Las Palmas de Gran Canaria

Curso 2021-2022

Juan José Bello Santana

ÍNDICE

1. - Introducción

El principal objetivo de este informe es llevar a cabo un proceso de Ingeniería Inversa sobre una aplicación dotada inicialmente desarrollada mediante el uso del Framework Angular [1].

Dicho proceso tiene el principal objetivo de obtener información con el fin de determinar cuáles son los componentes existentes en el proyecto, así como averiguar la manera que interactúan entre sí y determinar el proceso de fabricación del mismo.

Para ello, se analizará en detalle la estructuración del árbol de directorios existente en el proyecto, así como las características de cada uno de los componentes creados en el mismo.

2. - Desarrollo

2.1. - Estructuración del árbol de directorios.

2.1.1.- Estructuración general.

En primer lugar, el esqueleto general del árbol de directorios de la aplicación dotada se trata del siguiente:

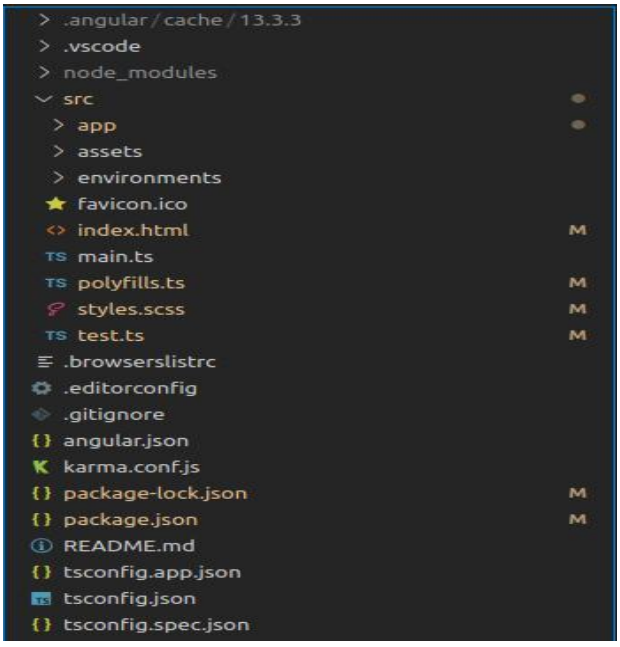

Figura 1: Esqueleto general del árbol de directorios de la aplicación

En el primer nivel del árbol, se pueden observar el directorios básicos de configuración del framework tales como *.angular* y *node_modules,* así como un directorio creado por el propio entorno de programación empleado (Visual Studio Code) *.vscode* . Igualmente, en la parte inferior de la imagen se pueden divisar ficheros de gran importancia para el funcionamiento de dicho framework, tales como *package.json* (contiene la configuración básica de la aplicación), *tsconfig.json* (contiene la configuración TypeScript) o *angular.json* (contiene la configuración básica del proyecto Angular, incluyendo rutas, formato de los estilos empleados, versiones ...).

Por último, se encuentra el directorio *src,* en el cual se halla el contenido principal de la aplicación desarrollada. Dentro del mismo existen tres directorios: *app*, donde se localizan, entre otros aspectos, los componentes implementados que contribuyen al funcionamiento de la aplicación; *assets,* que contiene archivos adicionales, tales como imágenes, para el correcto funcionamiento de la aplicación, y *.environments,* en el que se localizan configuraciones y variables de entorno que permiten establecer el proyecto tanto en desarrollo como en producción.

2.1.2.- Estructuración del directorio *app*.

La estructuración del árbol de directorios existente dentro del directorio *app,* se trata de la mostrada en la siguiente figura:

Figura 2: Esqueleto general del árbol de directorios localizado dentro del directorio *app*

Antes de analizar los subdirectorios existentes en el mismo, se pueden encontrar una serie de ficheros generales los cuales son aplicados para todos los componentes de la aplicación , entre los que destacan:

- ***app-routing.module.ts****:* Gestiona el enrutamiento de los componentes de la aplicación, así como controla la carga de dichos componentes de forma que estos sean únicamente cargados en la aplicación cuando estos sean utilizados (lazy-loading).
- ***app.component.html:*** Contiene el código HTML común a toda la aplicación.
- ***app.component.scss:*** Contiene todo el estilo común a la aplicación.
- ***app.component.ts:*** Contiene la configuración y datos generales de la aplicación.
- ***app.module.ts:*** Contiene la importación de todos los componentes existentes y libre.

En cuanto a los subdirectorios existentes, podemos encontrar los cuatro siguientes que se mencionan a continuación:

- ***Components:*** En este directorio se encuentran los componentes comunes a la aplicación, los cuales no pueden relacionarse con ningún tipo de usuario concreto, tales como *forgot-password, login* y *not-found.*

- ***Guards/auth:*** En este directorio destaca la existencia del archivo *auth.guard.ts,* el cual consta de un método que comprueba si existe un usuario con la sesión iniciada en la aplicación. En caso de que este no haya iniciado sesión, la aplicación le redirigirá automáticamente a la página de inicio de sesión, y en caso de que tenga una sesión iniciada, este podrá navegar libremente por las diferentes páginas de las que disponga acceso.

- ***Modules:*** Este directorio se encuentra a su vez subdividido en otros dos directorios, *admin,* en el que se encuentran los componentes gestionados por los administradores y *user*, en el que se encuentran los componentes visibles por los usuarios del sitio web. Si observamos el contenido existente dentro de estos dos subdirectorios hallamos lo siguiente:

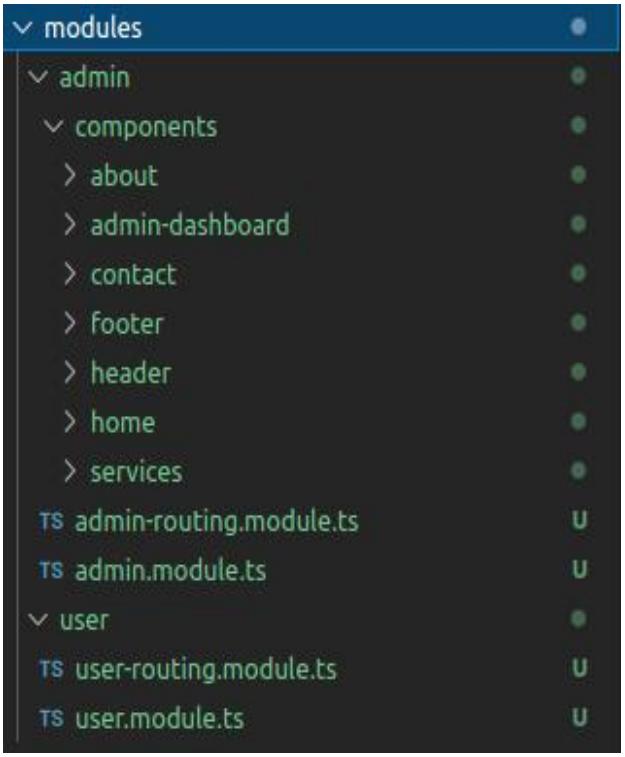

Figura 3: Esqueleto general del árbol de directorios localizado dentro del directorio modules

Tanto el directorio *admin* como *user,* constan de los ficheros (*admin/user)-routing-module.ts* y *(admin/user).module.ts.* En el primero de ellos, se gestiona el enrutamiento entre los componentes pertenecientes a dicho tipo de usuario, así como la carga de los mismos dependiendo del directorio en el que se encuentre situado el usuario. En el segundo de ellos, se importan y configuran cada uno de los componentes pertenecientes al tipo de usuario correspondiente.

Se puede observar, que los usuarios administradores constan de un total de 7 componentes, mientras que los usuarios sin privilegios, no constan de ningún componente en esta aplicación. Dichos componentes se expondrán con mayor detalle en apartados posteriores.

- ***Services/auth:*** Tal y como se puede intuir por su nombre, en dicho directorio se encuentran ficheros encargados de gestionar los servicios y además consta del subdirectorio *auth,* en el que se localizan los ficheros encargados de gestionar el protocolo de autenticación existente en la aplicación. Destaca el fichero *auth.service.ts*, el cual gestiona las acciones de un servicio creado para la gestión del sistema de autenticación de la aplicación. En este se encuentran métodos tales como *setToken()* y *getToken()*, que gestionan el token que consta un usuario una vez se ha autenticado en la aplicación. Luego, encontramos otros métodos tales como *isLoggedIn(),* que comprueba mediante la existencia o no de dicho token, si un usuario se encuentra con una sesión iniciada, o *logout(),* encargado de realizar el cierre de sesión de un usuario mediante la supresión del token de una variable local, *localStorage*, en el que se almacena el mismo. Por último, encontramos el método *login(),* el cual pretende simular el inicio de sesión de un usuario que se encuentra almacenado en una base de datos. En su lugar, dicho método comprueba que las credenciales introducidas en el formulario de inicio de sesión son idénticas a las ya preestablecidas en el método, permitiendo en ese caso el inicio de sesión y el establecimiento de un token también predefinido. En caso contrario, el método emitirá un mensaje de error.

2.2.- Componentes existentes.

De manera a la que se ha podido comprobar en los apartados anteriores, en la aplicación podemos encontrar aquellos componentes no asociados a ningún tipo de usuario en concreto, y aquellos pertenecientes a un usuario administrador, pues tal y como se encuentra desarrollado el servicio de autenticación, únicamente existirá un usuario administrador en la aplicación.

Respecto a los componentes no vinculados con ningún tipo de usuario se hallan:

- ***Forgot-password:*** Este componente se encarga de mostrar al usuario una página en el que puede introducir su correo electrónico para reestablecer la contraseña que ha olvidado.

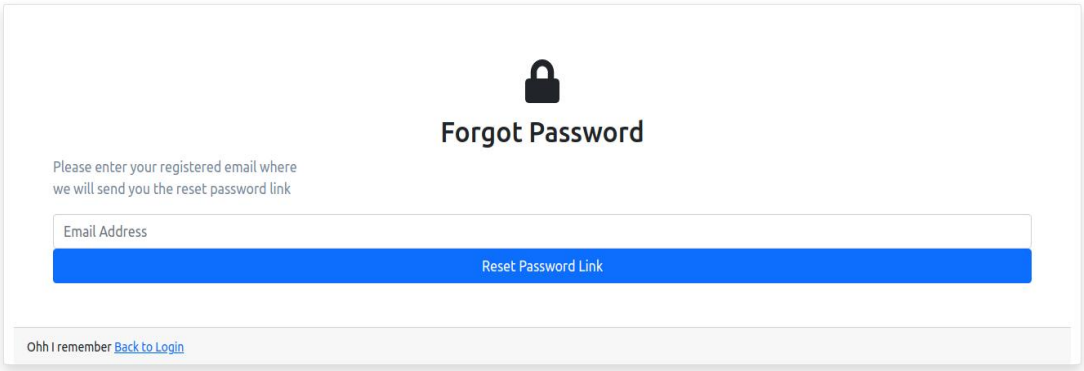

Figura 4: Página que muestra el componente *forgot-password*

- ***Login:*** Este componente se encarga de mostrar al usuario un formulario que permite al mismo autenticarse en la aplicación.

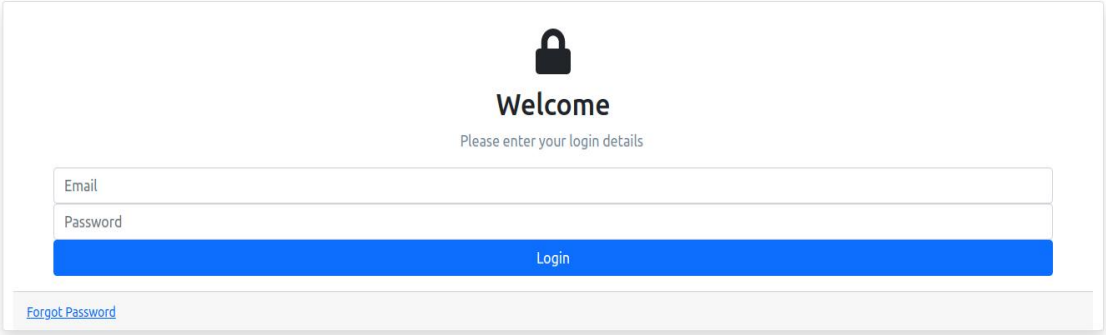

Figura 5: Página que muestra el componente *login*

- ***Not-found:*** Este componente se encarga de mostrar al usuario una página de error en caso de que la URL introducida por el mismo sea incorrecta.

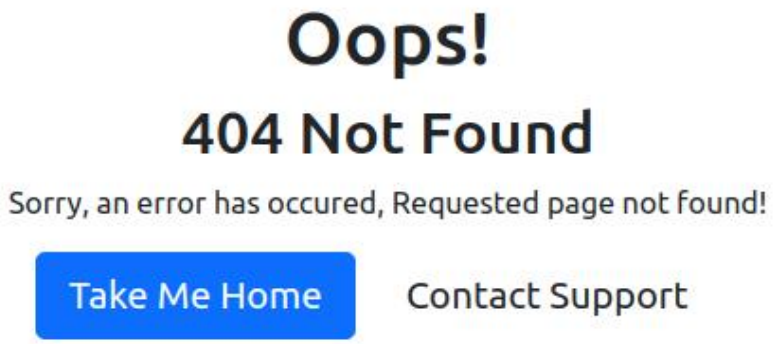

Figura 6: Página que muestra el componente *not-found*

En cuanto a los componentes vinculados con el usuario administrador se encuentran:

- ***about:*** Este componente se encarga de mostrar la pestaña "About" de la página, en la que la organización puede exponer información sobre la misma.

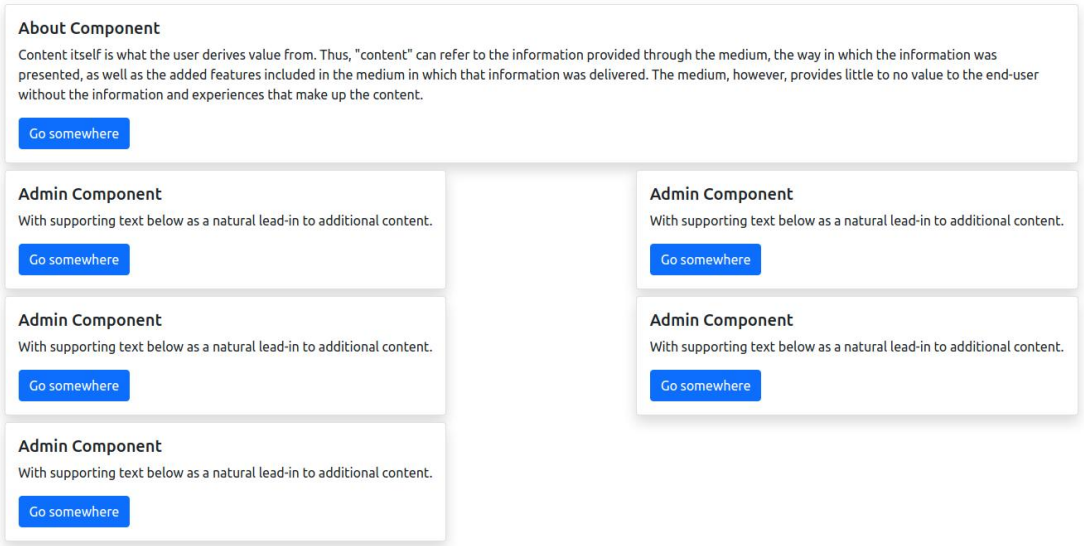

Figura 7: Página que muestra el componente *about*

- ***admin-dashboard:*** Este componente se encarga de anexar a cualquier pestaña que sea accedida a través de la navegación, los componentes que muestran el footer y header creado para dicho sitio web, tal y como se puede observar en el caso de la pestaña “About”.

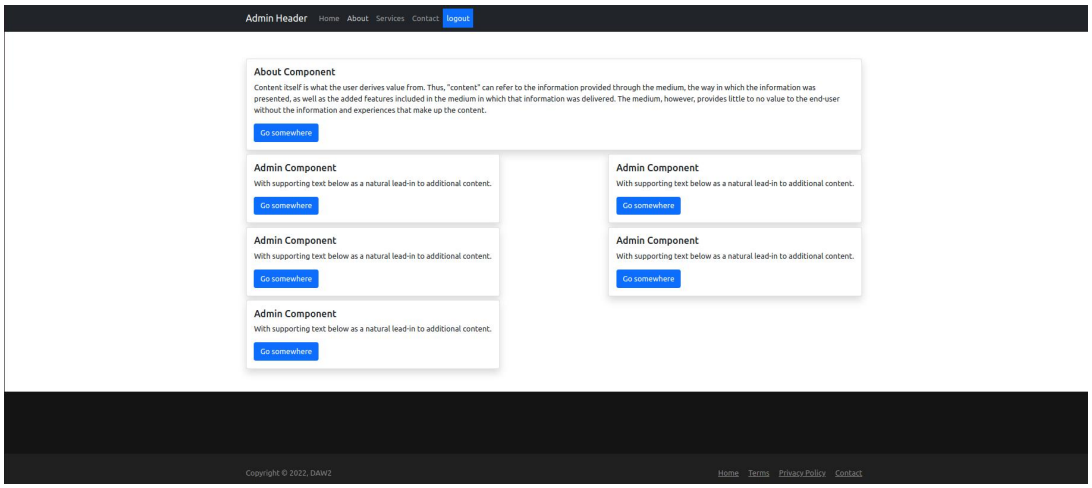

Figura 8: Ejemplo de página en el que se aprecia la acción realizado por el componente *admin-dashboard*

- **contact:** Este componente se encarga de mostrar la pestaña “Contact” de la página, en la que la organización puede incluir un formulario de contacto de usuarios hacia la misma.

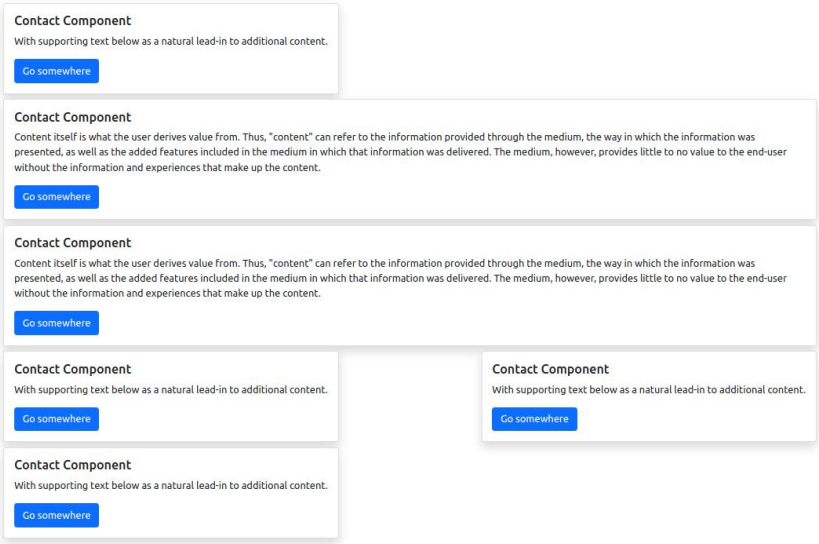

Figura 9: Página que muestra el componente *contact*

- **footer:** Este componente se encarga de mostrar el pie de página que posee el sitio web, el cual aparecerá en todas las páginas navegables existentes del mismo.

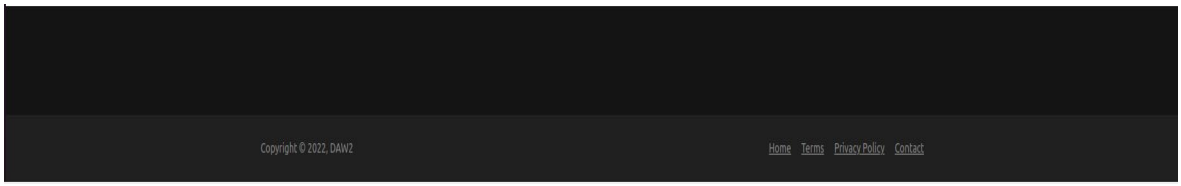

Figura 10: Pie de página existente en el sitio web mostrado por el componente *footer*

- ***header:*** Este componente se encarga de mostrar el encabezado que posee el sitio web, que aparecerá en todas las páginas navegables existentes del mismo, las cuales en esta aplicación todas pertenecen al usuario administrador.

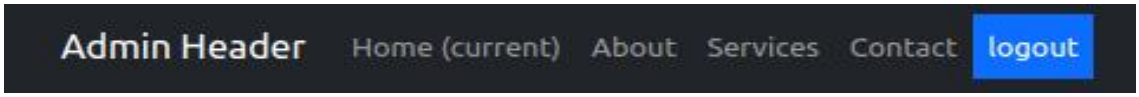

Figura 11: Encabezado existente en el sitio web mostrado por el componente heade*r*

- ***home:*** Este componente se encarga de mostrar la pestaña "Home" de la página, en la que la organización puede exponer información general sobre la misma.

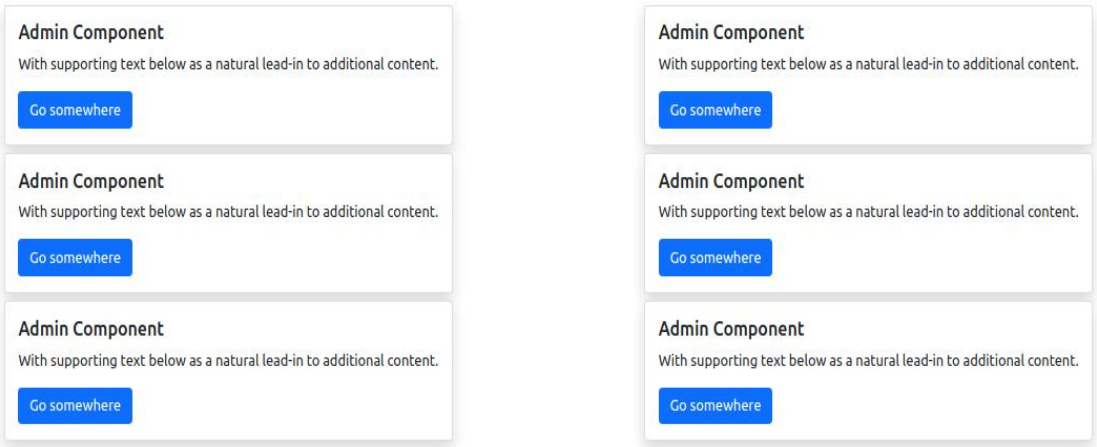

Figura 12: Página que muestra el componente *home*

- **services:** Este componente se encarga de mostrar la pestaña “Services” de la página, en la que la organización puede exponer información sobre los diferentes servicios que esta ofrece

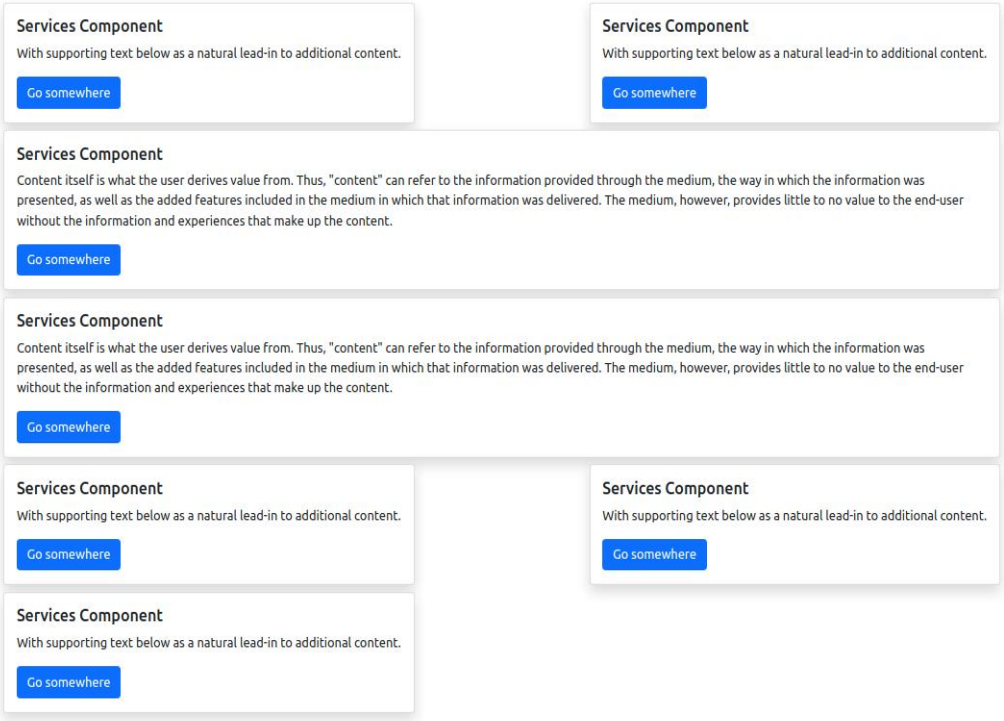

Figura 12: Página que muestra el componente *services*

3. - Conclusiones

En definitiva, la realización de este trabajo ha resultado de gran utilidad, pues gracias al mismo se ha podido apreciar la importancia e utilidad que tiene un proceso tal y como se trata del de Ingeniería Inversa.

El hecho de observar y analizar la manera en el que se encuentra diseñado e implementado un determinado producto finalizado, contribuye notoriamente no sólo a la comprensión del funcionamiento del mismo, sino a desarrollar desde cero un producto similar utilizando como referencia la manera que emplea el producto original para relacionar sus componentes.

Es por ello, que considero que el empleo de esta práctica es fundamental para numerosas organizaciones y en nuestro futuro laboral, pues en el mercado surgirán por parte de empresas, aplicaciones innovadoras, a las cuales deben hacer competencia el resto de organizaciones para seguir siendo competitivas en el mercado, siendo fundamental en estos casos aplicar un proceso de ingeniería inversa sobre dichas aplicaciones.

4. - Referencias

[1] Afonso Suárez, Maria Dolores. “Ingeniería Inversa”. [En línea]. Disponible en el Campus Virtual de la asignatura : https://aep22.ulpgc.es/mod/folder/view.php?id=1712293 [accedido el 22/04/2022]