



DESARROLLO DE APLICACIONES WEB II

INGENIERÍA INVERSA

Javier Suárez Suárez
javier.suarez113@alu.ulpgc.es

ÍNDICE

1.-INTRODUCCIÓN	2
2.-DESARROLLO	2
2.1-ESTRUCTURA DE DIRECTORIOS DE LA APLICACIÓN	2
2.2-DEFINICIÓN DE DIRECTORIOS Y FICHEROS	4
3.-CONCLUSIÓN.....	8
4.-BIBLIOGRAFÍA	8

1.-INTRODUCCIÓN

En este documento se mostrará el proceso de ingeniería inversa realizado sobre una aplicación web desarrollada en Angular [1] con el objetivo de estudiar su estructura de directorios y documentarla.

En primer lugar, se mostrará toda la estructura de directorios de la aplicación existente bajo el directorio fuente (*src/*)

Posteriormente, se explicará para qué funciona cada uno de los componentes hallados y que razón de existencia tienen.

A partir de la aplicación estudiada, se desarrollará una aplicación para gestionar la plantilla de jugadores de un equipo deportivo. Esta aplicación a desarrollar es la que se corresponde con la existente en el repositorio en *GitHub* donde se halla este documento.

2.-DESARROLLO

Como anotación previa, se destaca la necesidad de instalar los paquetes de Angular correspondientes a *Fort-Awesome* y *Bootstrap* para que la aplicación proporcionada se ejecute correctamente. Para ello, ejecutar las siguientes órdenes dentro del proyecto:

```
-ng add @fortawesome/angular-fontawesome@0.10.1 -> Esta versión como mínimo  
-ng add @ng-bootstrap/ng-bootstrap
```

2.1-ESTRUCTURA DE DIRECTORIOS DE LA APLICACIÓN

La estructura de directorios de la aplicación es la siguiente:

```
src  
|  
|-app  
|  
|-components  
|   |-forgot-password  
|       |-forgot-password.component.html  
|       |-forgot-password.component.scss  
|       |-forgot-password.component.spec.ts  
|       |-forgot-password.component.ts  
|  
|   |-login  
|       |-login.component.html  
|       |-login.component.scss  
|       |-login.component.spec.ts  
|       |-login.component.ts  
|  
|   |-not-found  
|       |-not-found.component.html  
|       |-not-found.component.scss  
|       |-not-found.component.spec.ts  
|       |-not-found.component.ts  
|  
|-guards/auth  
|   |-auth.guard.spec.ts  
|   |-auth.guard.ts
```

Figura 1: Estructura de directorios de la aplicación (I).

```

|-guards/auth
  |-auth.guard.spec.ts
  |-auth.guard.ts

|-modules

  |-admin
    |-components
      |-about
        |-about.component.html
        |-about.component.scss
        |-about.component.spec.ts
        |-about.component.ts
      |-admin-dashboard
        |-admin-dashboard.component.html
        |-admin-dashboard.component.scss
        |-admin-dashboard.component.spec.ts
        |-admin-dashboard.component.ts
      |-contact
        |-contact.component.html
        |-contact.component.scss
        |-contact.component.spec.ts
        |-contact.component.ts
      |-footer
        |-footer.component.html
        |-footer.component.scss
        |-footer.component.spec.ts
        |-footer.component.ts
      |-header
        |-header.component.html
        |-header.component.scss
        |-header.component.spec.ts
        |-header.component.ts
      |-home
        |-home.component.html
        |-home.component.scss
        |-home.component.spec.ts
        |-home.component.ts
      |-services
        |-services.component.html
        |-services.component.scss
        |-services.component.spec.ts
        |-services.component.ts

    |-admin-routing.module.ts
    |-admin.module.ts

  |-user
    |-user-routing.module.ts
    |-user.module.ts

```

Figura 2: Estructura de directorios de la aplicación (II).

```

  |-user
    |-user-routing.module.ts
    |-user.module.ts

|-services/auth
  |-auth.service.spec.ts
  |-auth.service.ts

-app-routing.module.ts

-app.component.html

-app.component.scss

-app.component.spec.ts

-app.component.ts

-app.module.ts

-index.html

-styles.scss

```

Figura 3: Estructura de directorios de la aplicación (III).

2.2-DEFINICIÓN DE DIRECTORIOS Y FICHEROS

La definición de cada directorio y/o fichero relevante es la siguiente:

-**Directorio *components***: Es un subdirectorio de **app** que contiene 3 componentes: *forgot-password*, *login*, *not-found*

-**Componente *forgot-password***: Ubicado en **/forgot-password** es un componente que se corresponde con un formulario en el que se le pide al usuario que introduzca su dirección de correo electrónico para hacer reset de su contraseña. Además, ofrece navegación hacia el componente *login*.

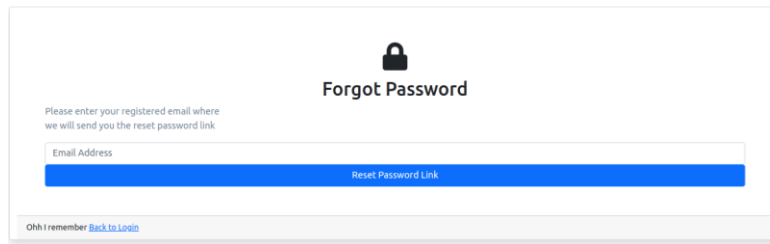
The screenshot shows a web form titled "Forgot Password" with a lock icon. Below the title, it says "Please enter your registered email where we will send you the reset password link". There is a text input field labeled "Email Address". Below the input field is a blue button labeled "Reset Password Link". At the bottom, there is a link that says "Ohh I remember [Back to Login](#)".

Figura 4: Componente *forgot-password*.

-**Componente *login***: Ubicado en **/login** es un componente que se corresponde con un formulario en el que se le pide al usuario que introduzca sus credenciales para iniciar sesión en el sitio web. Además, ofrece navegación hacia el componente *forgot-password*.

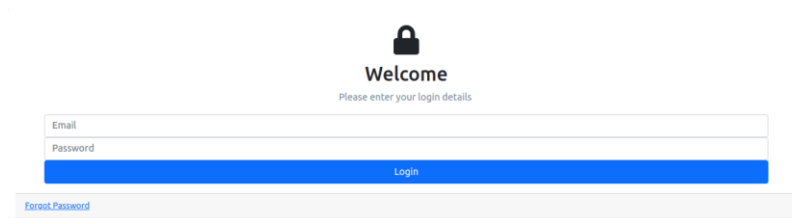
The screenshot shows a web form titled "Welcome" with a lock icon. Below the title, it says "Please enter your login details". There are two text input fields, one labeled "Email" and one labeled "Password". Below the input fields is a blue button labeled "Login". At the bottom, there is a link that says "Forgot Password".

Figura 5: Componente *login*.

-**Componente *not-found***: Ubicado en cualquier ruta no registrada en la aplicación, es un componente que se corresponde con un mensaje que indica al usuario que la página solicitada no existe. Además, ofrece navegación por vía de 2 enlaces hacia el componente *home* que se comentará posteriormente. Si el usuario no ha iniciado sesión, se le redirige al componente *login*.

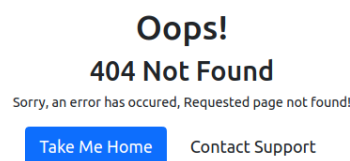
The screenshot shows a 404 error page. At the top, it says "Oops!" in a large font, followed by "404 Not Found". Below that, it says "Sorry, an error has occurred, Requested page not found!". At the bottom, there are two buttons: a blue button labeled "Take Me Home" and a text link labeled "Contact Support".

Figura 6: Componente *not-found*.

-Directorio *guards/auth*: Contiene los usuales ficheros con extensión *.ts* y *.spec.ts* y se corresponden con un *Guard* de Angular, *middleware* destinado a actuar como intermediario cuando el usuario accede a una ruta específica de la aplicación determinando si el usuario, dependiendo de sus permisos, puede acceder a esa ruta o se le redirige a otra. Posee un método **canActivate()** que determina, a través de un servicio (**auth.service**), a donde se redirige al usuario si ha iniciado sesión e intenta acceder a una ruta para usuarios registrados o, si no lo ha hecho, a donde se le redirige que es hacia el componente *login*.

-Directorio *modules*: Contiene dos subdirectorios (**admin** y **user**) que poseen toda la lógica de módulos y componentes a cargar en diferido (**lazy load**) según el tipo de usuario registrado.

-Subdirectorio *admin*: Contiene los componentes y ficheros de modulo y routing a usar cuando el usuario registrado posee perfil de administrador.

-Subdirectorio *components*: Contiene los componentes *about*, *admin-dashboard*, *contact*, *footer*, *header*, *home* y *services* destinados a un usuario administrador.

-Componente *about*: Ubicado en **/admin/about** es un componente cuyo contenido es de ejemplo y se corresponde con el de la siguiente imagen:

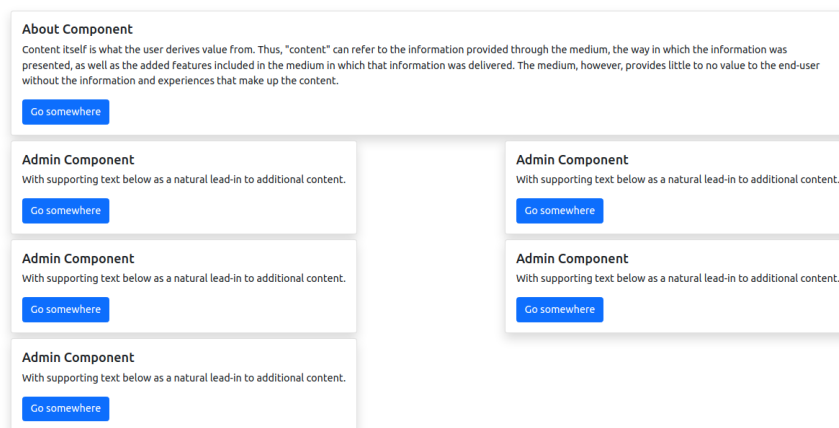


Figura 7: Componente *about*.

-Componente *admin-dashboard*: Es el componente base del usuario administrador y declara al componente *footer* y *header* estableciendo entre los dos el **router outlet** que permite cargar en su interior el componente accedido por vía de la navegación y routing de la aplicación.

-Componente *contact*: Ubicado en **/admin/contact** es un componente cuyo contenido es de ejemplo y se corresponde con el de la siguiente imagen:

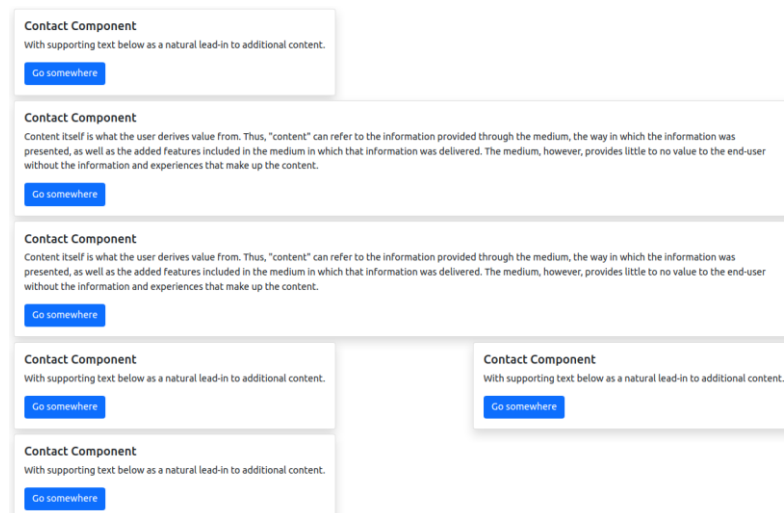


Figura 8: Componente *contact*.

-**Componente *footer***: Ubicado en todas las páginas destinadas al usuario administrador, sirve de componente estético y se corresponde con la siguiente imagen:



Figura 9: Componente *footer*.

-**Componente *header***: Ubicado en todas las páginas destinadas al usuario administrador, sirve de componente estético y se corresponde con la siguiente imagen:

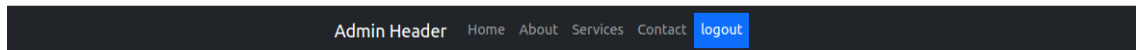


Figura 10: Componente *header*.

-**Componente *home***: Ubicado en `/admin/home` es un componente cuyo contenido es de ejemplo y se corresponde con el de la siguiente imagen:

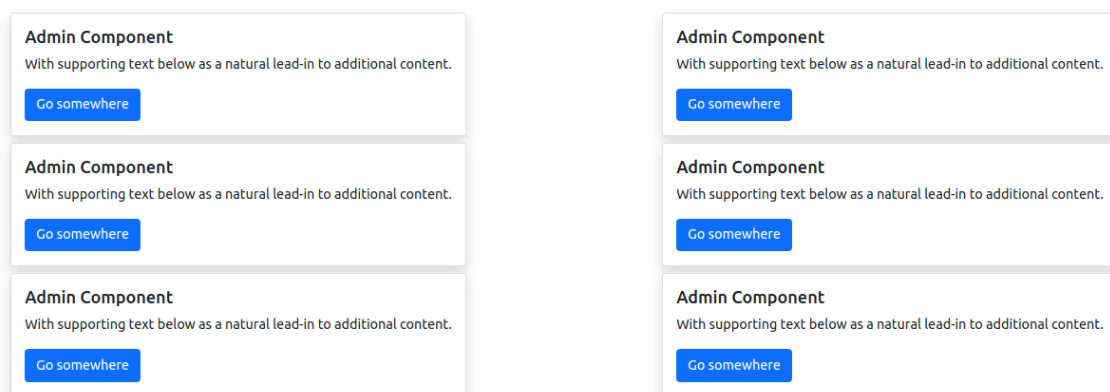


Figura 11: Componente *home*.

-**Componente *services***: Ubicado en `/admin/services` es un componente cuyo contenido es de ejemplo y se corresponde con el de la siguiente imagen:

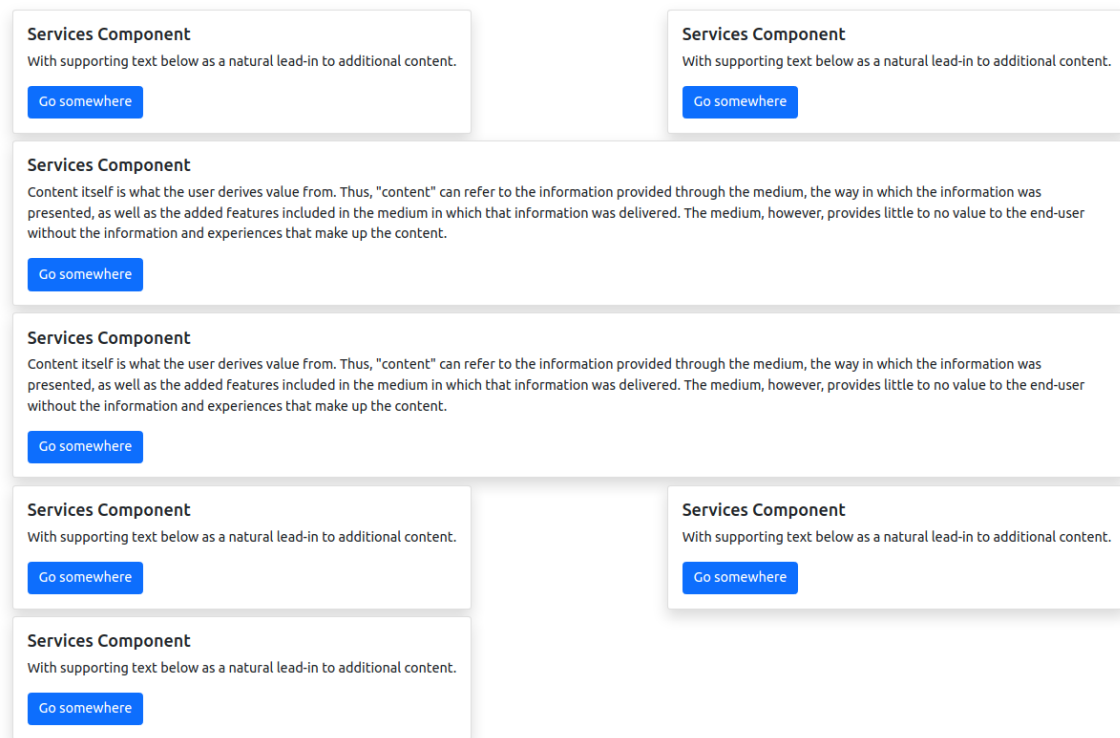


Figura 12: Componente *services*.

-**Fichero *admin-routing.module.ts***: Contiene definidas las rutas hacia los componentes *admin-dashboard*, *about*, *contact*, *footer*, *header*, *home* y *services*.

-**Fichero *admin.module.ts***: Contiene declarados los componentes *admin-dashboard*, *about*, *contact*, *footer*, *header*, *home* y *services*.

-**Subdirectorío *user***: Contiene los ficheros *user-routing.module.ts* y *user.module.ts* destinados a declarar los módulos y rutas para un usuario registrado no administrador.

-**Directorío *services/auth***: Contiene los ficheros con extensión *.ts* y *.spec.ts* destinados a implementar un servicio de autenticación en la aplicación por vía de la habilitación y deshabilitación de un token en el *localStorage* del navegador. Para ello, posee los métodos *setToken()*, *getToken()*, *isLoggedIn()*, *logout()* y *login()*.

-**Fichero *app-routing.module.ts***: Contiene definidas las rutas hacia los componentes *forgot-password*, *login*, *not-found* y, además, carga de manera diferida (*lazy load*) el módulo *admin* si el *Guard Auth* por vía de su método *canActivate()* así lo indica.

-**Fichero *app.module.ts***: Contiene declarados los componentes *forgot-password*, *login*, *not-found*, así como módulos para implementar *Bootstrap* y *Font-Awesome*.

3.-CONCLUSIÓN

El trabajo realizado en este informe ha permitido conocer como son las labores de ingeniería inversa en el sector de la informática.

Es un ejercicio innovador, pues hasta ahora solo se había trabajado con ejercicios de desarrollo y esta actividad pone de manifiesto lo importante que es estudiar bien las aplicaciones que se desarrollan para poder conocer cómo solucionar los errores que se puedan presentar.

Por todo ello, se considera que el trabajo realizado es de utilidad ya que permite conocer algo muy útil como es el estudio de un desarrollo completado algo importante en las organizaciones del mundo profesional, pues el proceso de control de calidad es esencial para que un producto salga con los mínimos errores posibles al mercado.

4.-BIBLIOGRAFÍA

[1] Carpeta de la aplicación en el Campus Virtual de la ULPGC. [En línea]. Disponible en:

<https://aep22.ulpgc.es/mod/folder/view.php?id=1712293>

[Accedido: 21/04/2022]