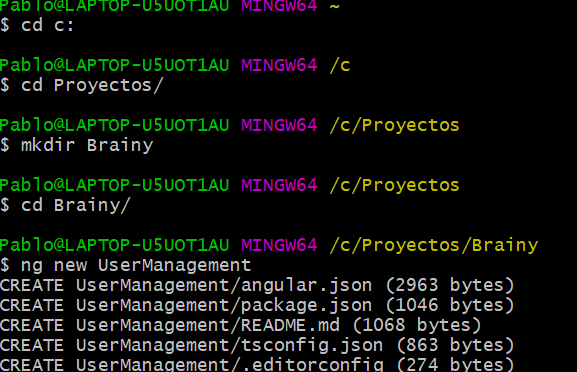
**Proceso de creación UserManagement**

**Planificación de las carpetas**

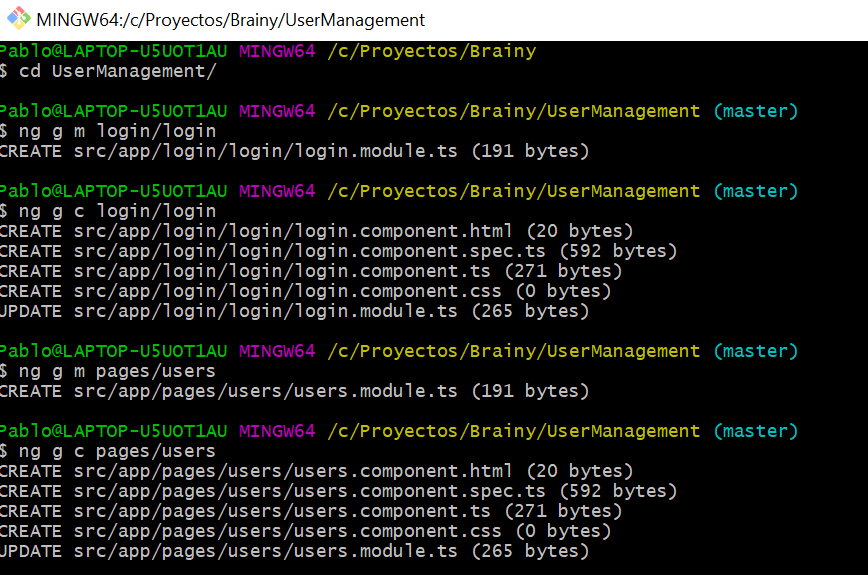
Antes de comenzar la aplicación me he tomado un momento para desglosar la aplicación en componentes.

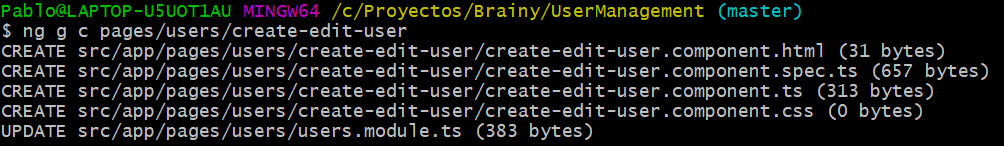
Una vez pensados los componentes creo la aplicación y la estructura de componentes.

Creo la aplicación con el nombre UserManagement:



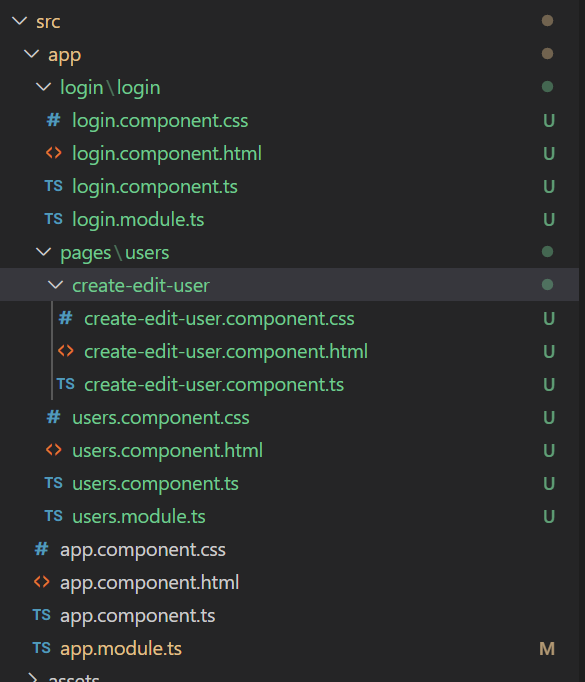
Después creo un primer esqueleto de lo que va a ser la aplicación



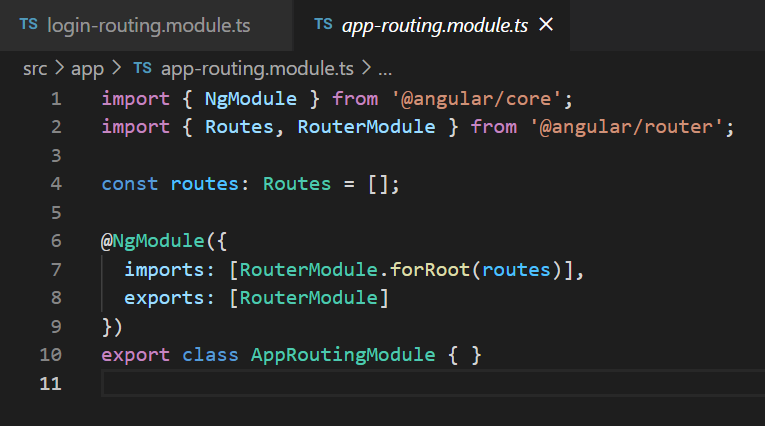
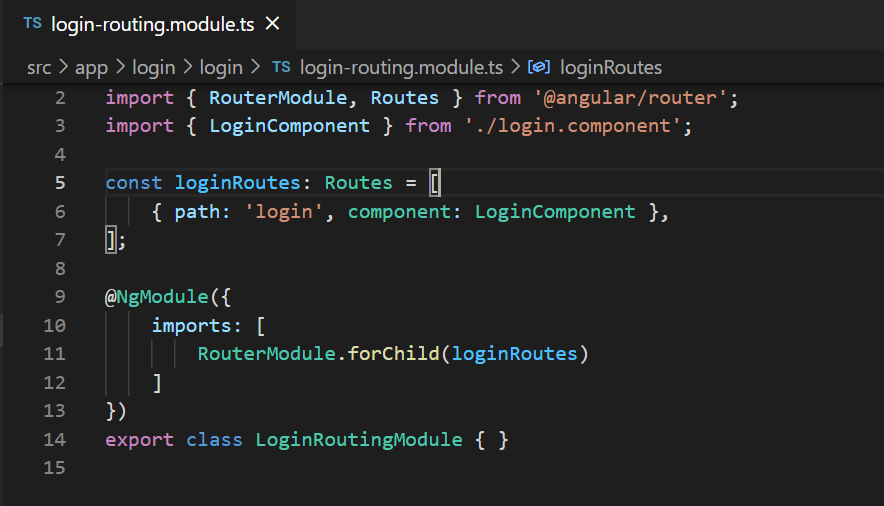


Quedando la estructura tal que asi:

\*Borro los archivos de testing ya que no los voy a utilizar.



Creo manualmente dos archivos de enrutado( routin-module) para el app y para el login y los importo en los módulos correspondientes.

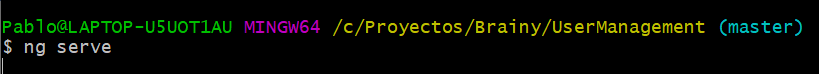


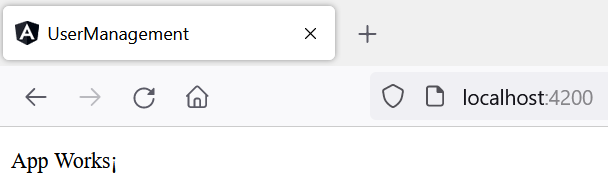
\*Borro el html por defecto del app component y añado un <p> para ver que funciona

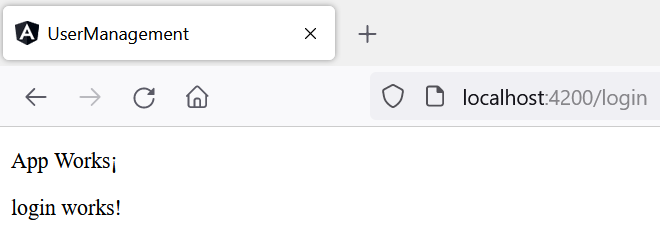
En el login component añado algo de html para probar la ruta /login y después añado en el app component la directiva de rutas:

<router-outlet></router-outlet>

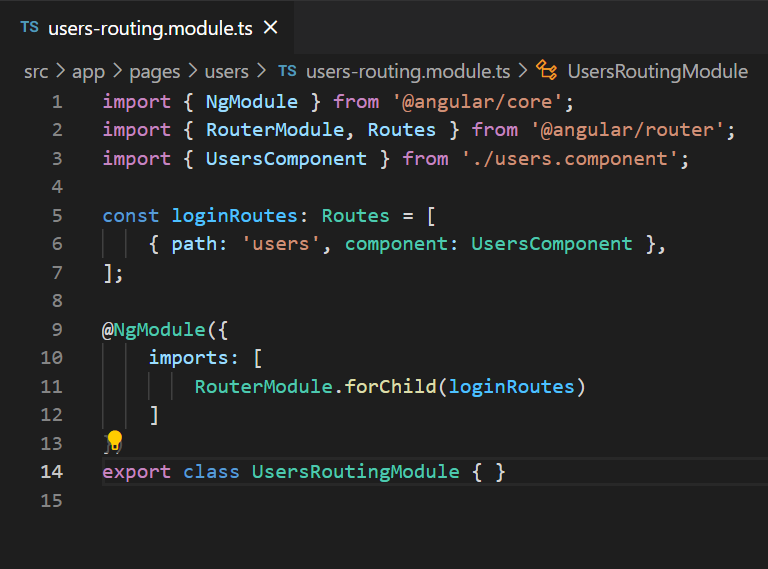
Y pruebo que la aplicación funciona hasta este punto:

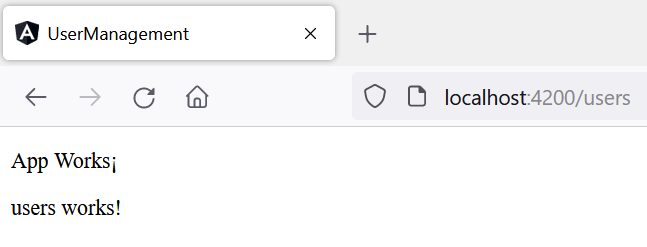






Hago lo mismo con los users: creo un users-routing y añado la ruta /users, añado el usersModule al app module y pruebo la ruta /users



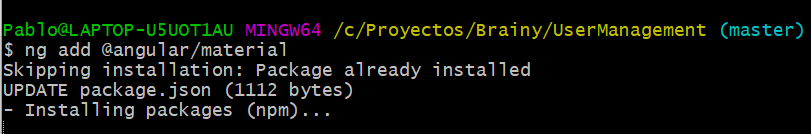


Estoy intentando obtener un token para utilizarlo, pero no consigo que me funcione con las credenciales que me han dado.

Voy a continuar con el desarrollo hasta que me lo aclaren.

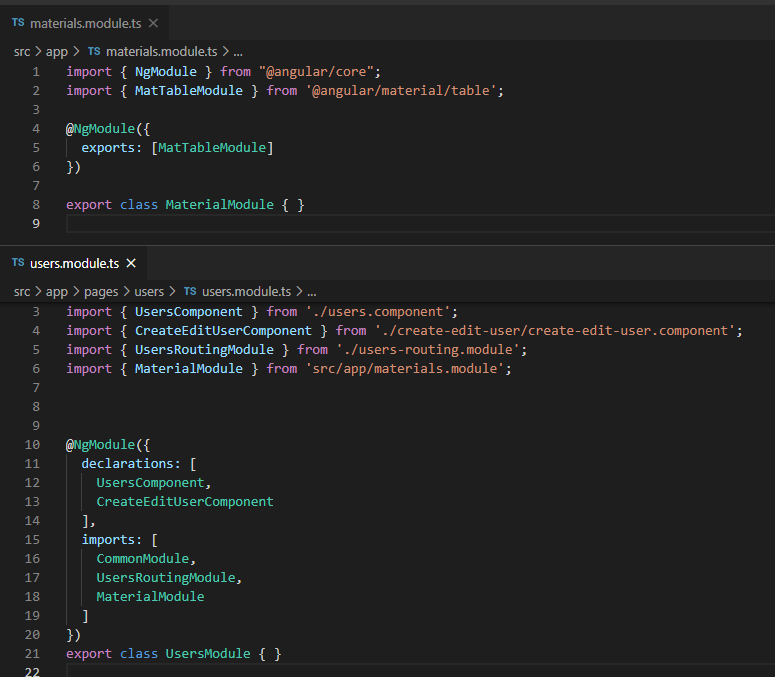
**Angular material**

Ahora voy a instalar Angular material en mi proyecto y crear la tabla de usuarios.



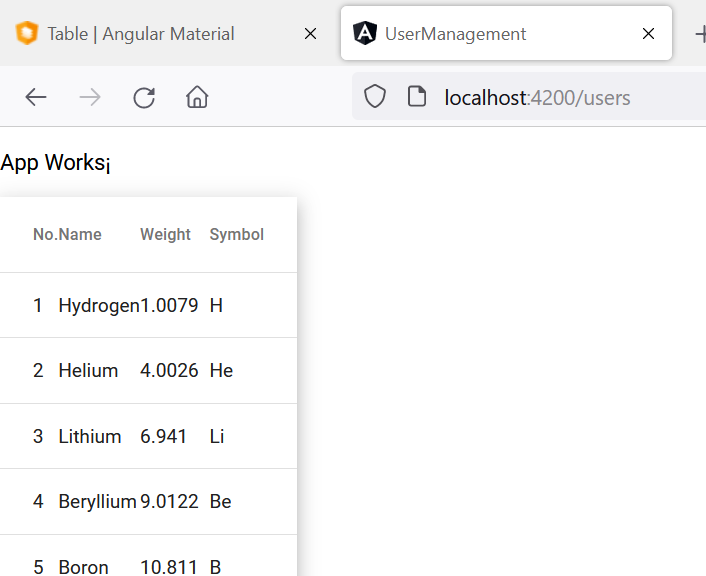
->Creo un módulo autónomo donde importaré todos los materiales que necesite y lo añado para utilizarlo en el users.module.

En materials module importo el material table para utilizarlo en el listado de usuarios:

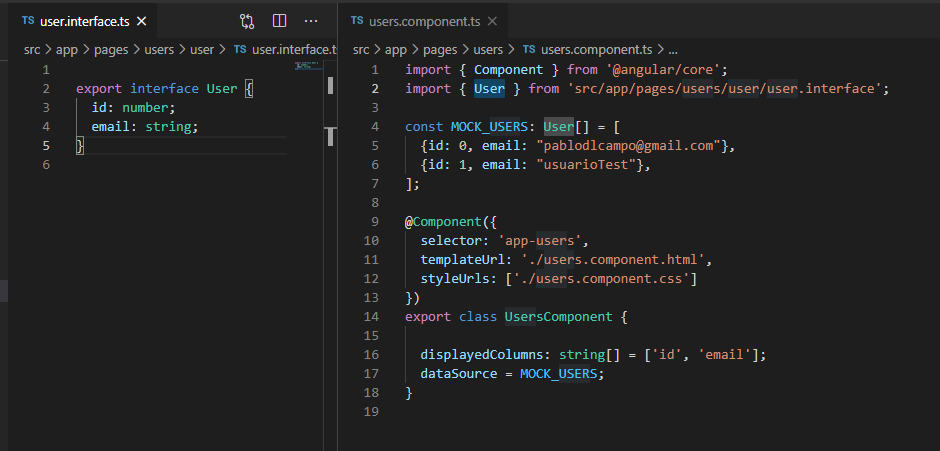


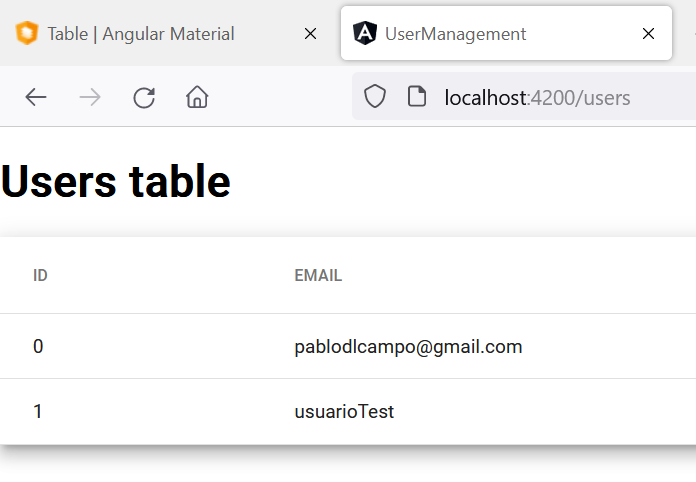
En users-component copio un ejemplo de tabla que aparece en la página de material:  
<https://material.angular.io/components/table/examples>

Que queda tal que así:



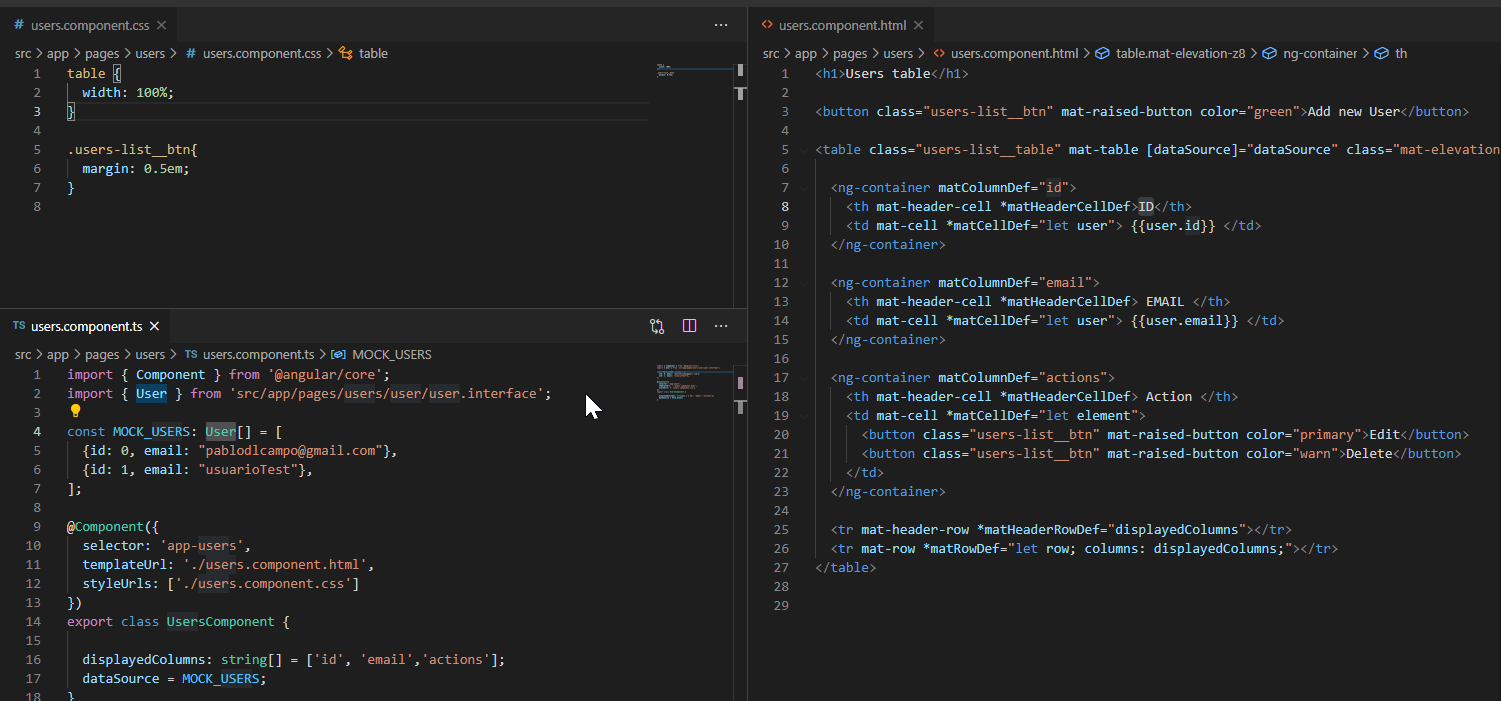
Ahora voy a crear una interfaz de usuario y de momento mockeare los datos del listado hasta poder obtenerlos desde el servicio web.



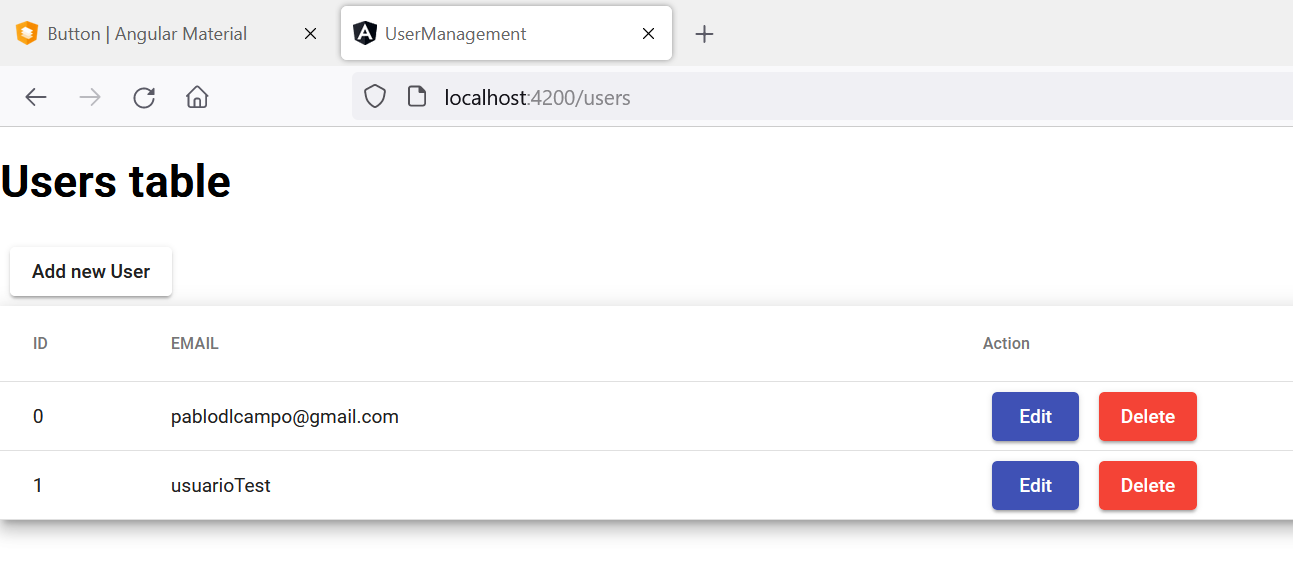
\*He código los datos id/email que aparecen en /users

Ahora voy a añadirle botones de edición, borrado y nuevo usuario.

\*Añado también el material de button.



Queda tal que asi:

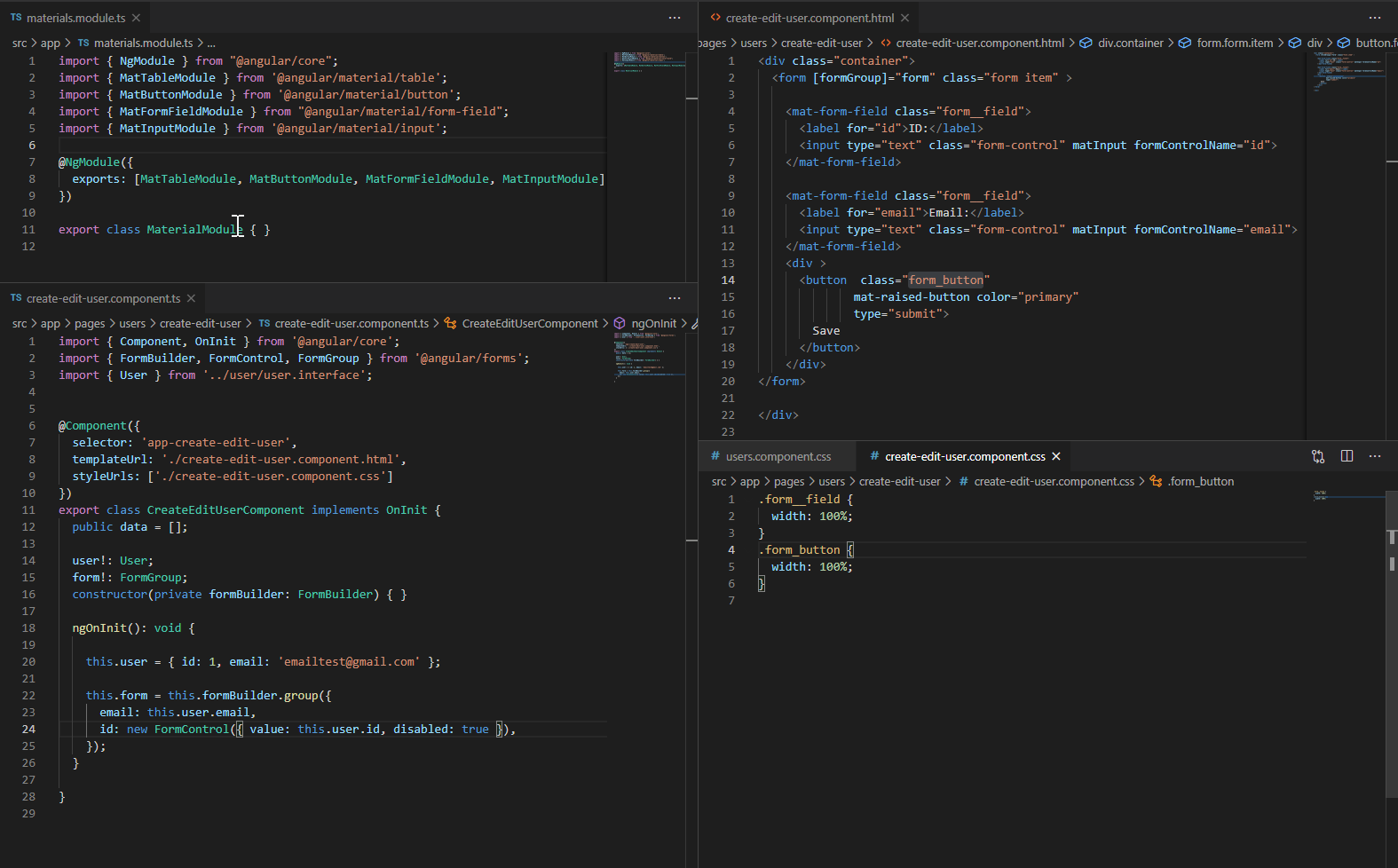


**Formulario creación/edición**

Añado en el modulo materials el formFieldModule y el InputModule

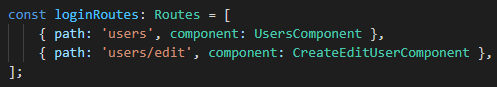
En create-edit-user creo un formulario sencillo y en el componente de créate-edit-user añado unos datos ficticios al formulario de un usuario a editar para mostrarlos por defecto en el formulario

\*añado las herramientas de FormBuilder + FormControl + FormGroup

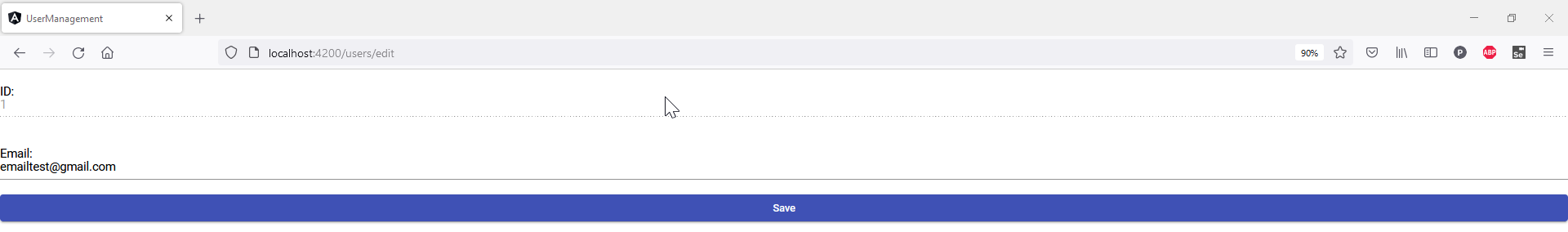


Ahora creo una nueva ruta para la edición de usuarios( mas adelante la modificare para poder enviar la id atraves de la url).

User-routing.module.ts:



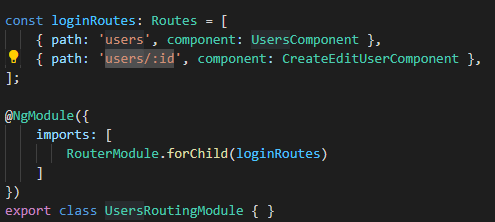
De momento queda así:

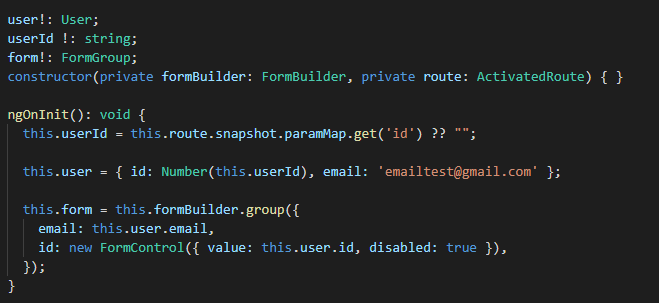


Ahora lo que quiero es pasar la id del usuario como parte de la url y utilizarla en el formulario.

Para ello modifico la ruta de edición y en el componente de edición agrego el servicio ActiveRoute para poder acceder a los parámetros de la url.

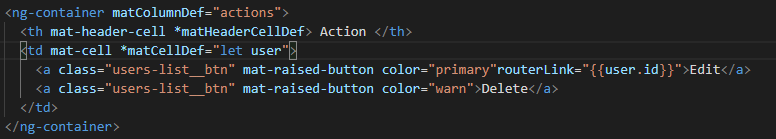
users-routing.module



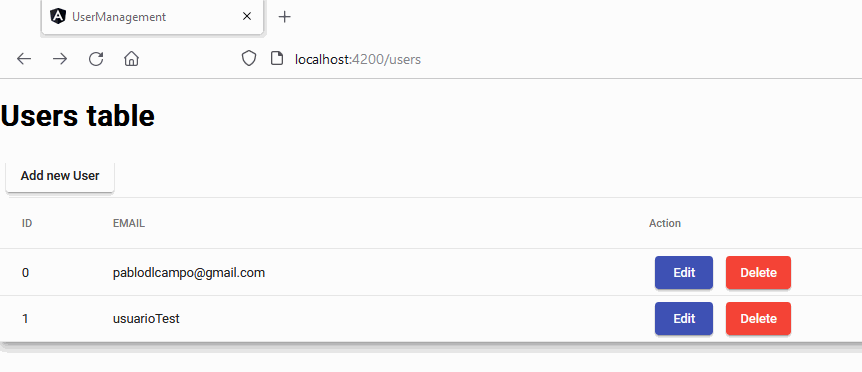
create-edit-user-component  


Después añado a los botones de edición del listado la propiedad routerLink para que redireccione a esta pantalla de edición pasándole la id en la url:

users.component.html

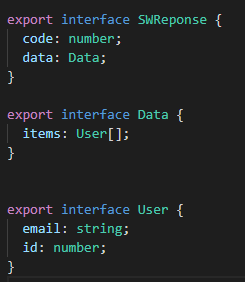


De esta manera los botones de edición ya están enlazados con el formulario de edición, aunque todavía no obtengan usuarios de BD



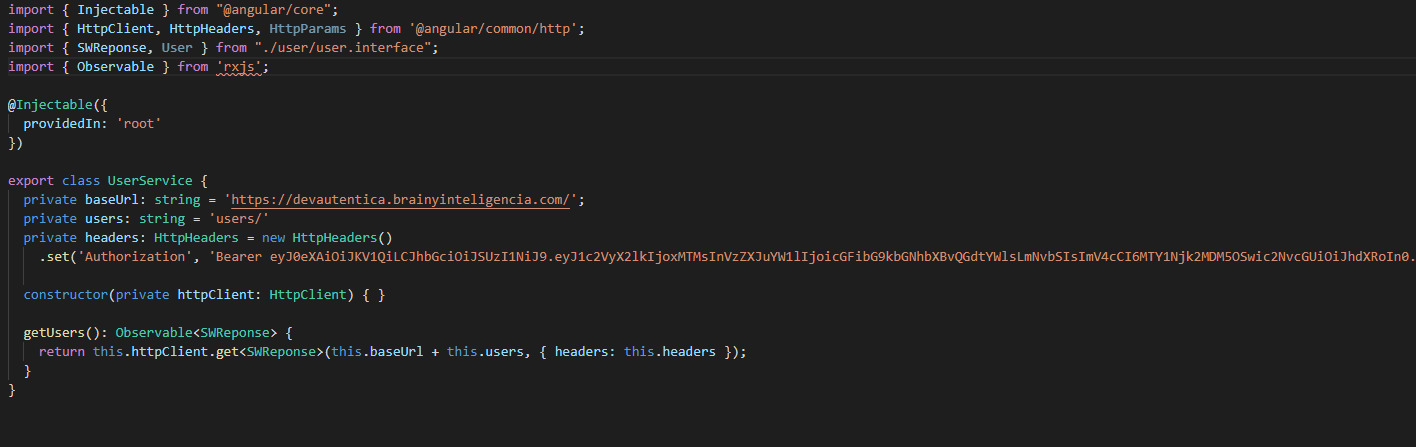
Ahora voy a crear un servicio para obtener los usuarios del servidor tanto para el listado como para la edición.

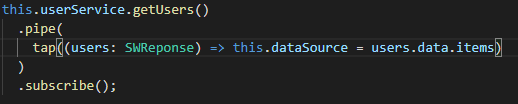
Primero voy a revisar la respuesta del servicio web de usuarios y crear un objeto que pueda contenerla.



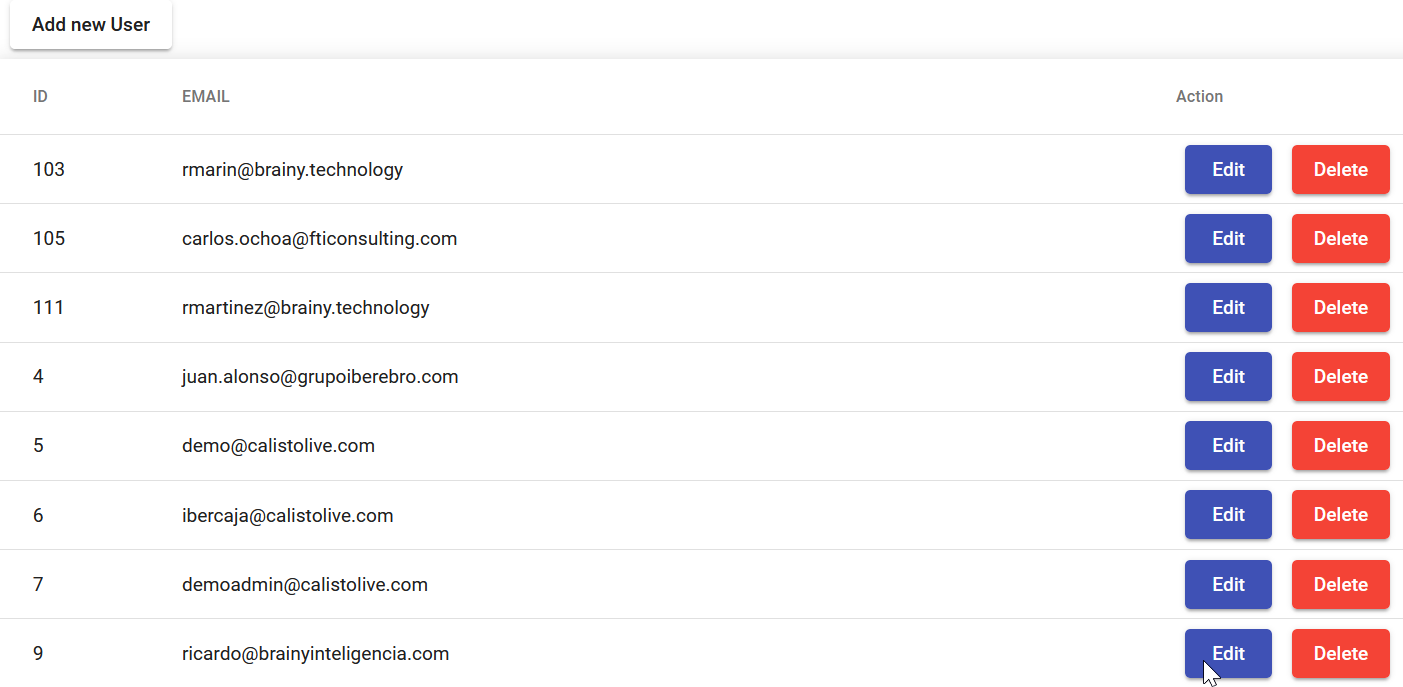
Con esto ya puedo obtener los datos de /users.

Creo un fichero user.service y en el un método obsevable que llame al sw del servidor. Ahora que ya tengo acceso al servicio de tokens puedo usar un token de manera manual para llamar al servicio de usuarios hasta que implemente el login.

  
En el user.component trato la respuesta del servicio para utilizar, de momento, solo los users.



Con esto ya aparecen todos los usuarios, teniendo en cuenta que el token es temporal:

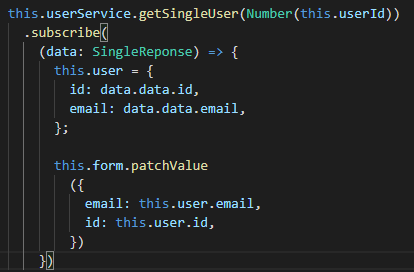


Ahora voy a añadir al servicio de usuarios los demás métodos que necesitamos para hacer el CRUD.

.GetSingleUser para obtener el usuario concreto a editar:



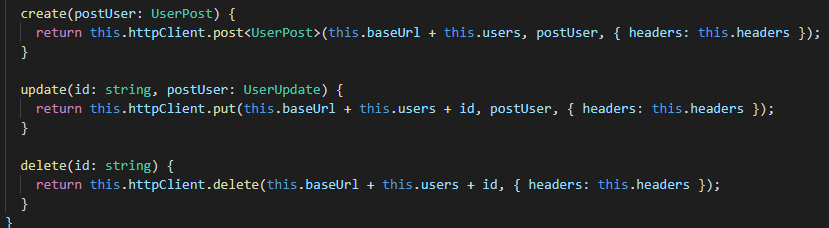
Ademas he modificado el créate-edit-user.component para gestionar la respuesta y pasársela al formulario.

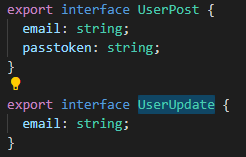


También he creado un botón en el formulario para volver atrás y he habilitado el botón de añadir nuevo usuario.

**Crear editar y borrar**

Métodos para crear, editar y borrar usuarios

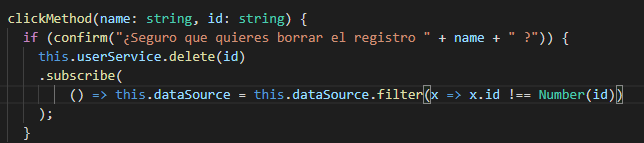




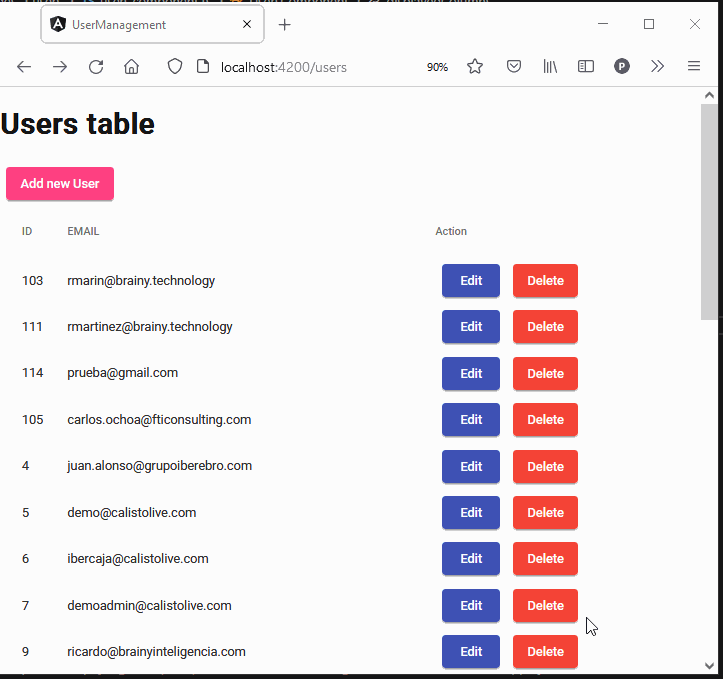
Y en el componente del user:



Para el borrado desde el formulario este método asignado al onclick del botón borrar:



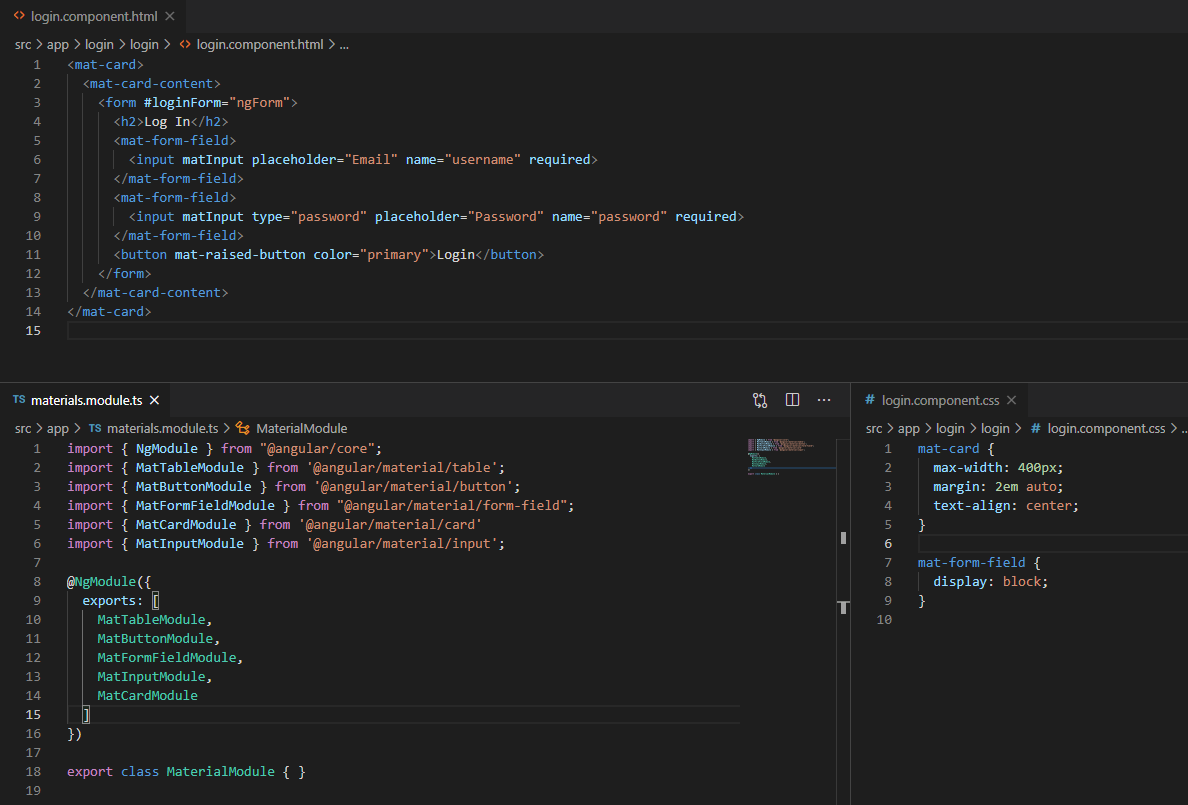
Resultado:

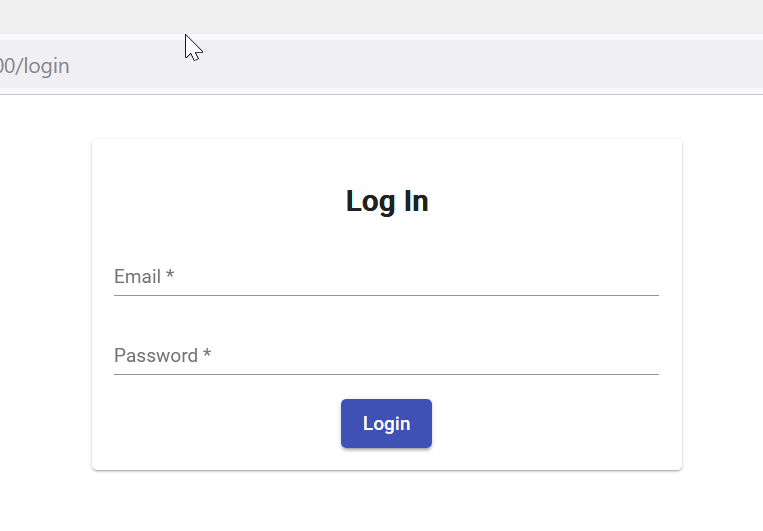
:

\*Faltaría añadir mensajes de confirmación, validaciones, etc...

**Login**

Creo un formulario de login utilizando el material card.

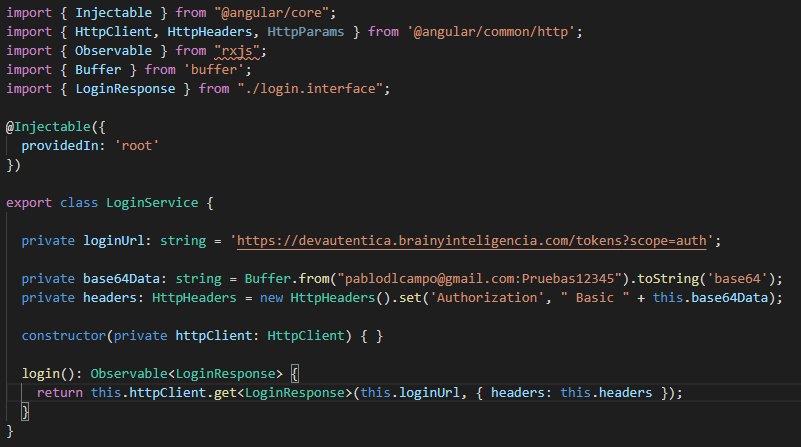




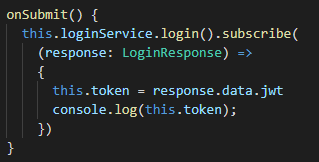
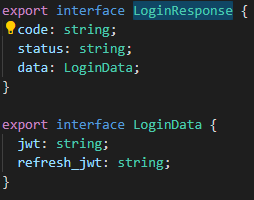
\*Tengo que instalar npm i --save-dev @types/node y añadir al tsconfig.app.json



Creo un servicio para el login que haga una llamada al endpoint /token con mi user/contraseña hardcoded y luego se los pasare desde el formulario.



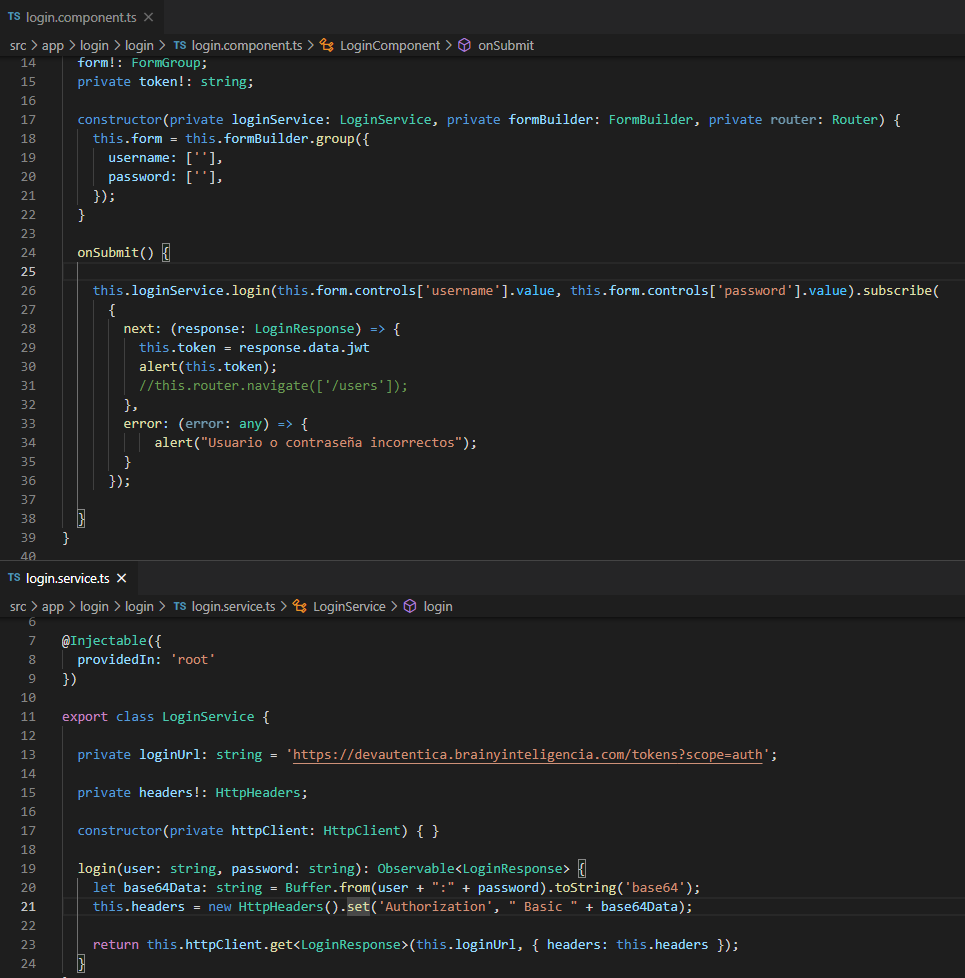
El modelo y el login.component



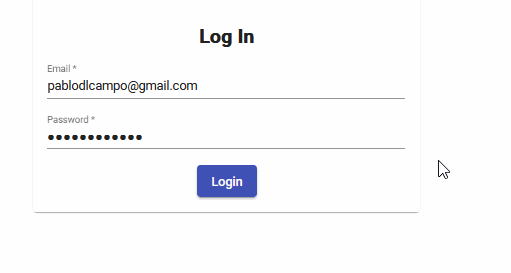
Y con esto el login ya me devuelve un tokenC:\Proyectos\Brainy\UserManagement\src\app\pages\users\create-edit-user\create-edit-user.component.ts



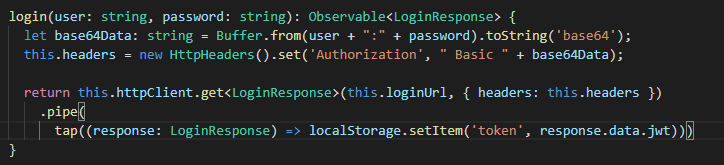
Adapto el componente y servicio para que se envíen usuario y contraseña y se muestre un error si son incorrectos o el token si todo ha ido bien:



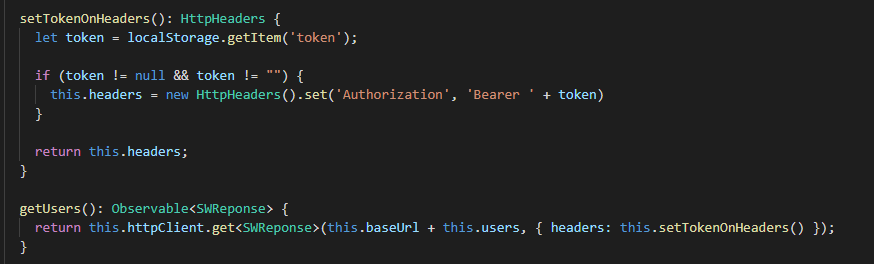
Quedando tal que así:



Ahora voy a guardar el token en localstorage una vez lo reciba y redireccionare desde el login a usuarios en caso de autenticación correcta, una vez allí utilizare el token para hacer las llamadas.



User.service.ts



\*De esta manera ya se puede borrar el token que estaba ‘hardcodeado’.

Ahora voy a crear la función de log out y controlar que solo se pueda acceder a las vistas si se tiene un token.

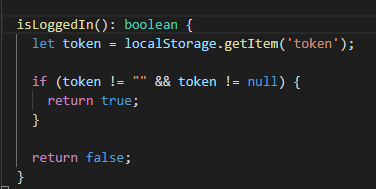
Para controlar las rutas voy a crear una clase de tipo guard.



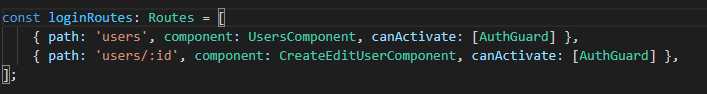
Este guard debe acceder a un método llamado isLoggedIN del servicio de login que comprueba si el token existe o no.



El método isLoggedIn del login.service.ts:



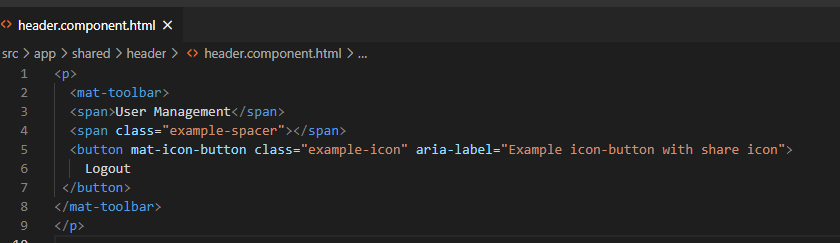
Y en las rutas añado este guard para que no se pueda acceder manualmente desde el navegador. Users-routing,module



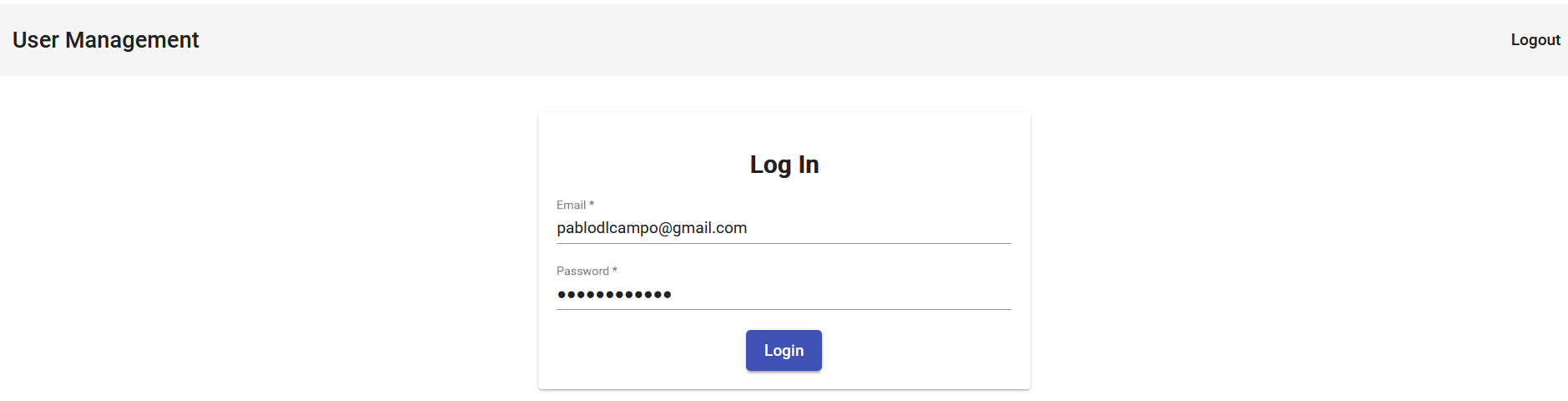
El siguiente paso seria poder cerrar la sesión.

Para ello voy a crear un método en el login.service que limpie el token y redireccione a /login y un header con un botón que llame a este método.

Creo un componente nuevo llamado header y utilizo el componente mat-toolbar de angular material para crear una barra de tareas en el header de la app.

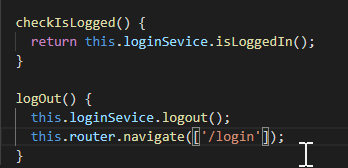


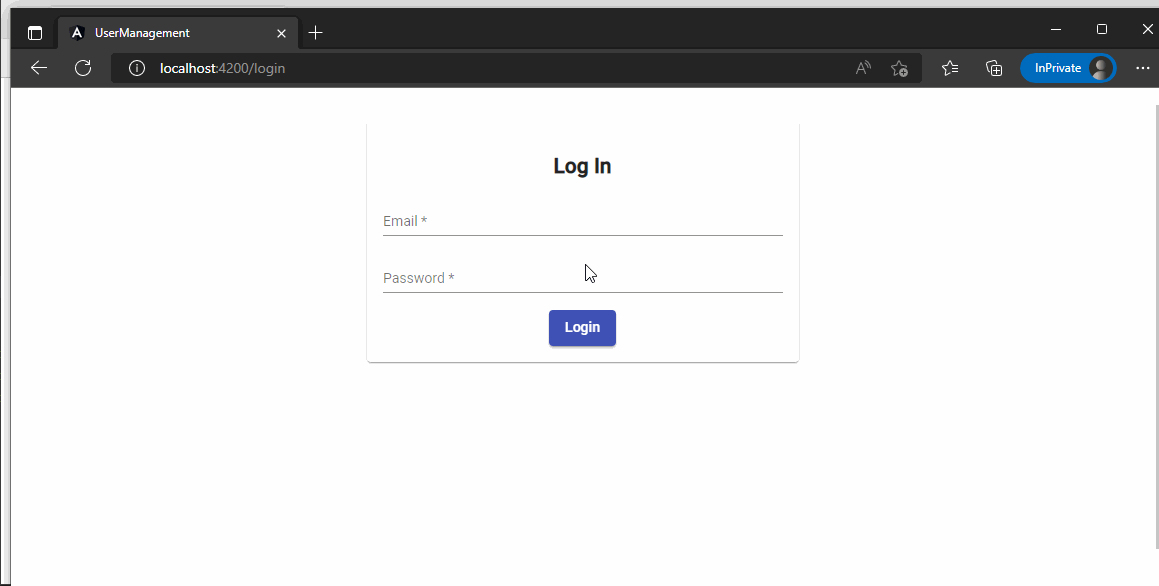
\*Lo añado al app.html y la pantalla se muestra asi:



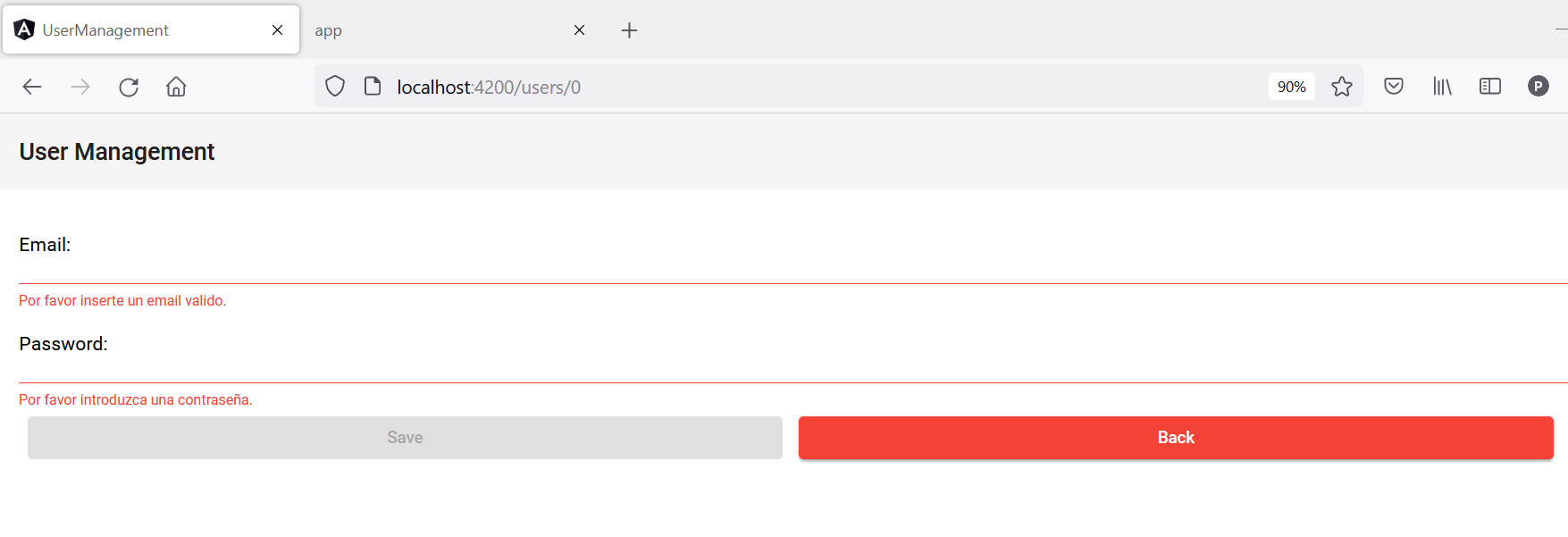
Ahora haré que solo aparezca cuando el usuario esté logeado y voy a crear el evento del botón logout.

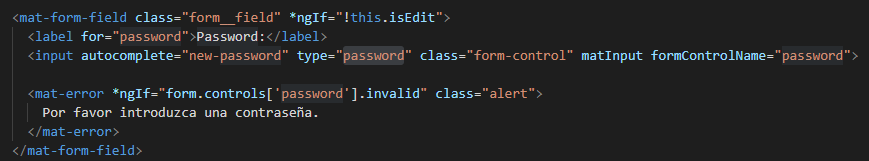




Con esto la aplicación ya controla correctamente el logout y que no se pueda acceder a ninguna ruta sin estar loggueado.

Añadir validaciones en creación de usuario y campo contraseña(en modo creación)



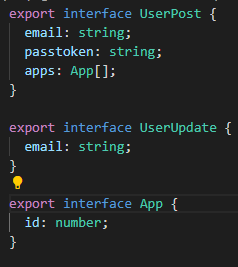


\*Ahora se envía la contraseña en el post.

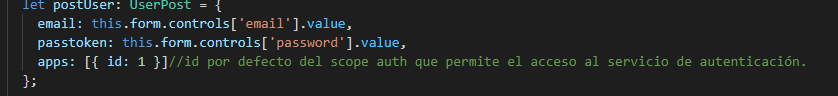
**Añadir a grupo de aplicaciones auth.**

Me he fijado que para poder usar un usuario es necesario que este incluido en la app authentication(scope auth). Voy a añadir esto en el post para poder crear nuevos usuarios funcionales.

Modifico el objeto user-post para incluir un array de objectos app y de momento le incluyo solo el parámetro id.

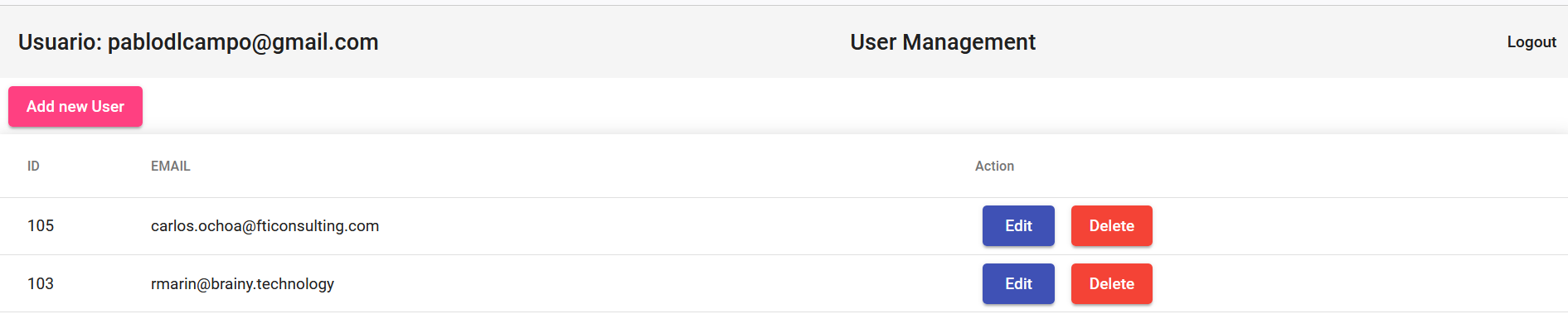


Y en el momento de la creación le añado una app por defecto:

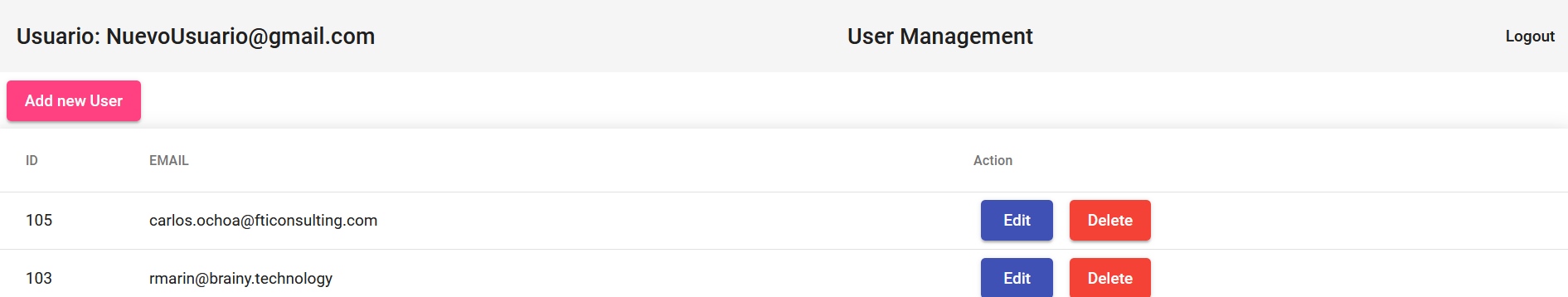


Con esto ya se pueden utilizar los usuarios recién creados.

En la cabecera añado el nombre del usuario que estamos utilizando, que lo guardo en local storage al hacer login.



Al hacer login con otro usuario nuevo:



**Recuperación de contraseña**