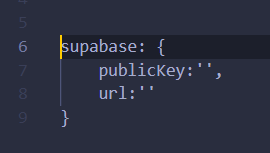
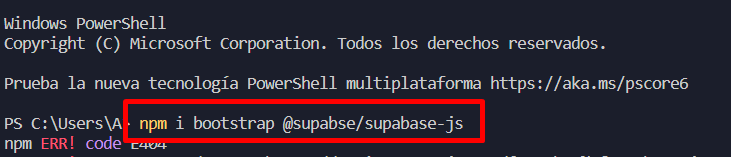
**LOGIN**

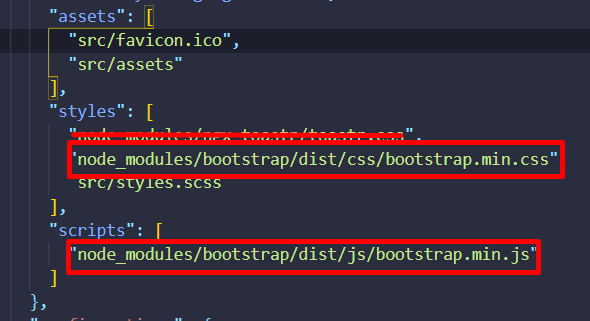
Ir a la carpeta environments y en el archivo environments.ts crearemos una variable llamada Supabase, a la que le pasamos la publickey y la url.



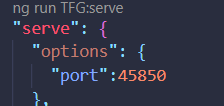
Instalar bootstrap y el paquete de Supabase 🡪 npm i bootstrap @supabase/supabase-js



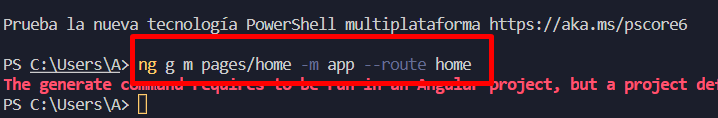
Configurar bootstrap en angular.json:



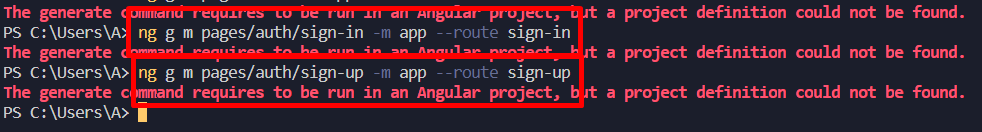
Cambiar el puerto en angular.json:



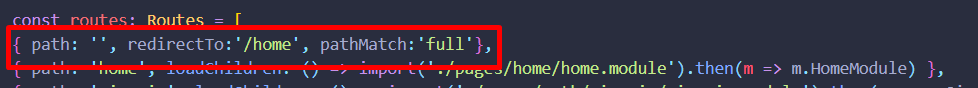
Creación de home dentro de pages:



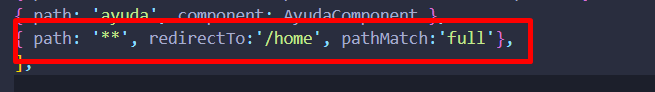
Creación de sign-in y sign-up dentro de auth y dentro de pages:



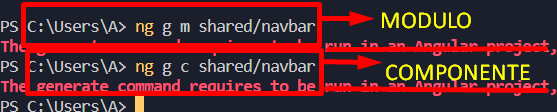
En el app-routing.module.ts redirigir cuando la ruta este vacía me lleve a la página home:



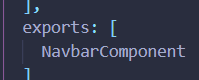
En el app-routing.module.ts redirigir cuando la ruta introducida no existe me lleve a la página home:



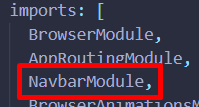
Creación del módulo y del componente del navbar:



Exportar el navbarComponent en el navbar.module para que pueda ser usado en otras partes:

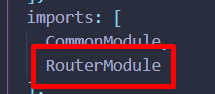


Para que se pueda usar en la aplicación el navbar, solo lo tenemos que importar en el app.module.ts:

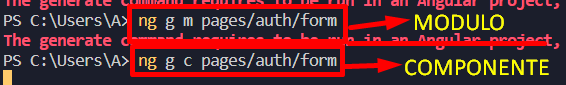


Copiar navbar de la página de bootstrap.

En el navbar.module.ts importar el modulo de rutas para permitir el uso de rutas en el navbar:

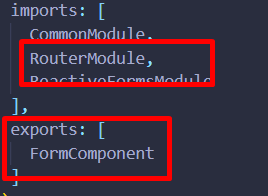


Creación del módulo y del componente del form:

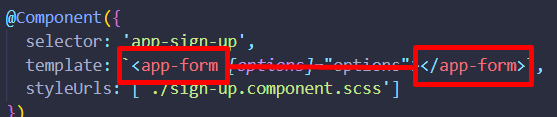
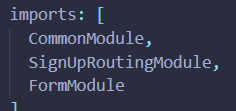


Copiar formulario de la página de bootstrap, que será el mismo para iniciar sesión y para registrarse.

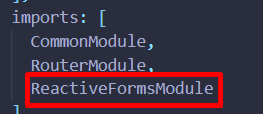
En form.module.ts importamos el modulo de rutas para el uso del formulario y exportar el componente del formulario para usarlo en la aplicación:



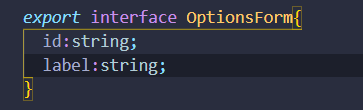
Cambiar en el sign-up.components.ts, el template usar el app-form, además de importar en el sign-up.module.ts el Form Module:

Importar en form.module.ts el modulo para hacer formularios reactivos:



En form.component.ts creamos una interfaz:



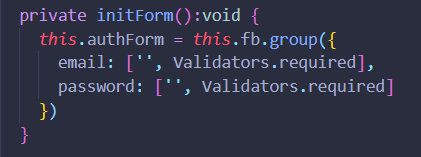
Creamos el formulario en form.component.ts:



Uso de la exclamación porque no se está inicializando

Inyectar FormBuilder en el constructor del form.component.ts.

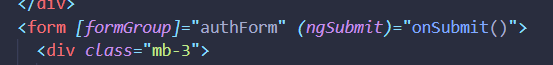
Creación del método initForm() en form.component.ts:



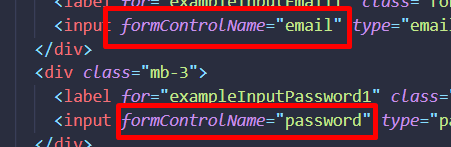
Llamar método initForm en el ngOnInit()

Creación del método onSubmit() en form.component.ts

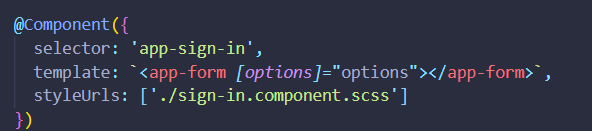
Llamar método onSubmit() y el authForm en el form.component.html:

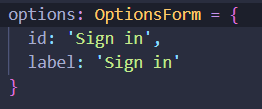
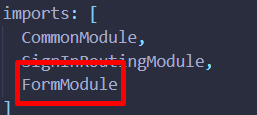


Llamar campo email y password del método initForm() en form.component.html:

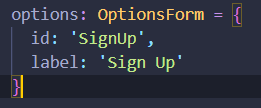


Cambiar en el sign-in.components.ts, el template usar el app-form, además de importar en el sign-in.module.ts el Form Module:



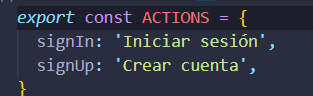
Crear en sign-up.component.ts la variable options:



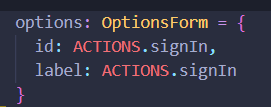
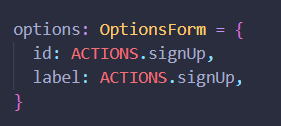
En el form.component.ts usamos el decorador @Input para traer las variables creadas anteriormente:



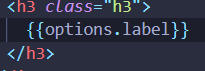
Crear un fichero de constantes para iniciar sesión y registarse:



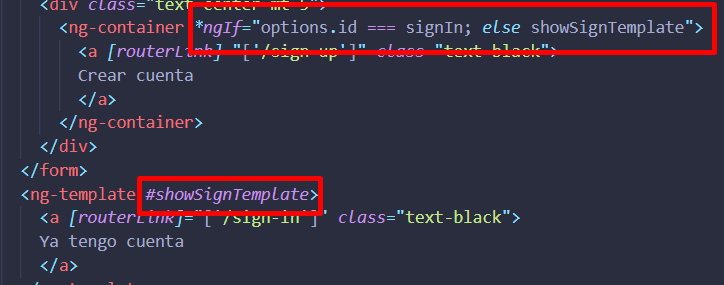
Una vez creado, cambiamos el contenido de la variable options tanto en sign-in como en sign-up:

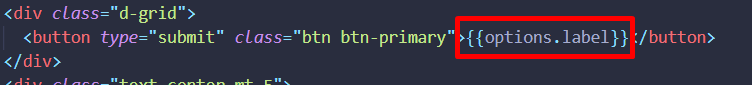
Crear en el form.component.html para mostar la opcion correcta en cuanto si el usuario está iniciando sesión o registrandose:



Hacer que se muestre la opción del login correspondiente y si no que muestre la otra opción mediante el uso de la directiva \*ngIf:

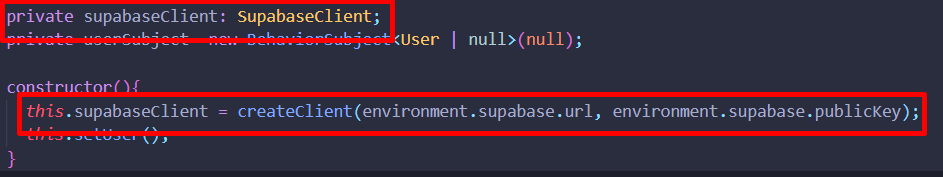


Cambiar el texto que saque el botón mediante el uso de la variable options:

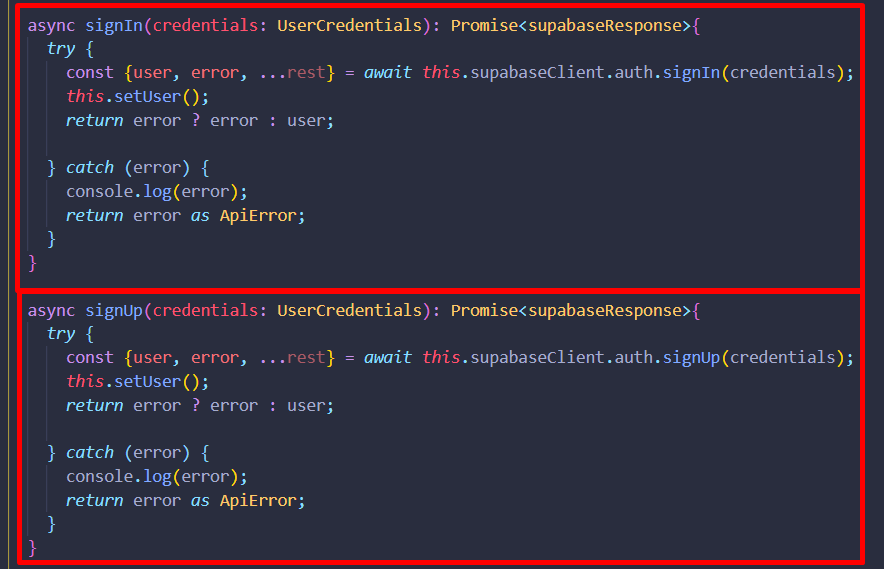


Crear el arhcivo auth.service.ts, en la carpeta services dentro de auth.

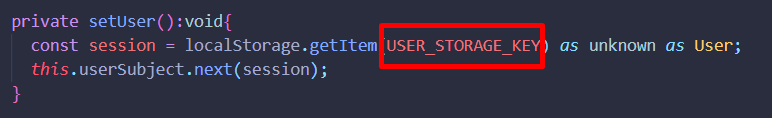
Crear variable SupabaseClient y en el constructos al crear el cliente se le pasa la publickey y la url:



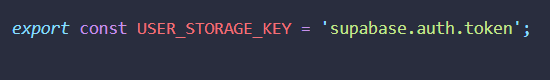
Crear método signIn() y sign Up():



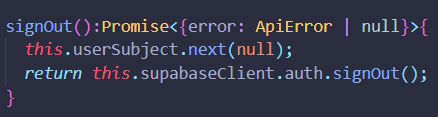
Crear el método setUser(), que se encarga de almacenar la sesión temporal cuando el usuario inicia sesión:



Crear variable USER\_STORAGE\_KEY en el archivo de las constantes



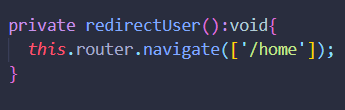
Crear el método signOut(), para cerrar sesión:



Inyectar AuthService en el form.component.ts.

Hacer la lógica del método onSubmit(), para la acción de pulsar se crea una variable actionToCall para hacer la ejecución:

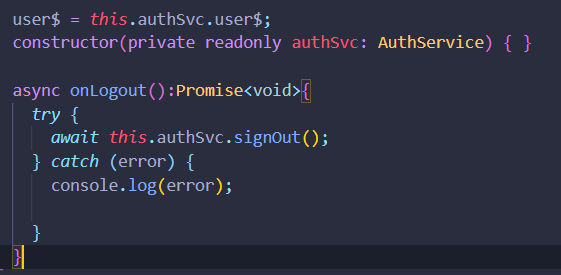


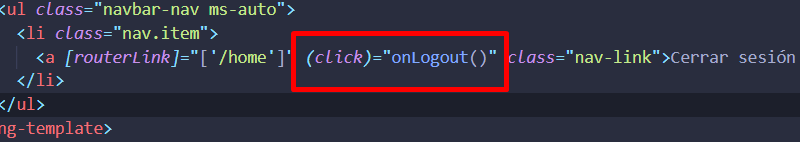


En caso de que funcione se redirige a la home y si no sale un mensaje

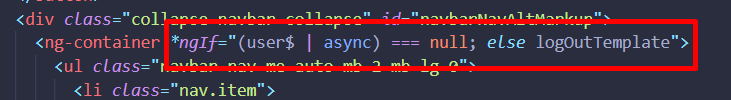
Inyectar Router en el form.component.ts en el constructor para que redirija.

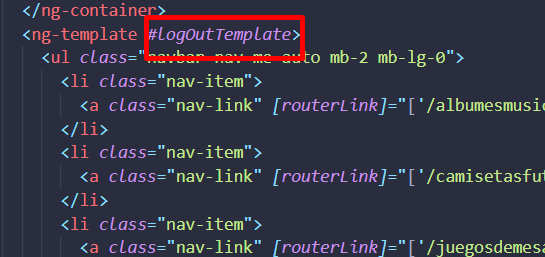
Crear método onLogout() en navbar.component.ts, para implementarlo ahora en navbar.component.html





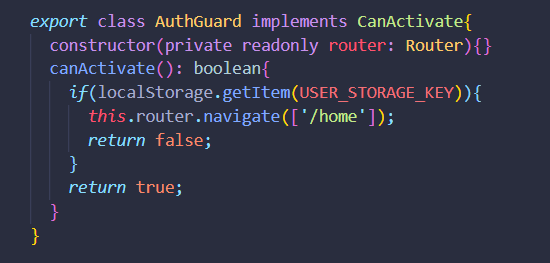
Uso de la directiva \*ngIf para que muestre iniciar sesión y registrarse si no hay ningún usuario, y en caso de que exista se mostrará el cerrar sesión, en el navbar.component.html:



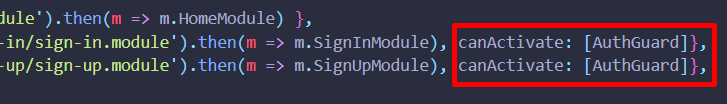


Crear un archivo que se llame auth.guards.ts en la carpeta guards que encuentra en auth.

Si se encuentra una sesión abierta, lo redirigirá a la home:



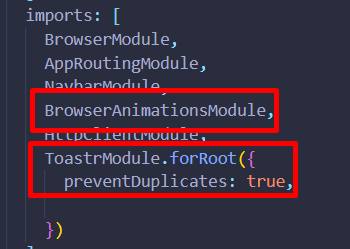
Ahora en app-routing.module.ts declaramos en que rutas queremos pasar los guards:



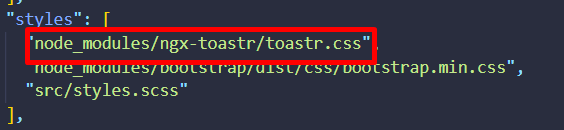
Instalar el paquete ngx-toastr, para mostrar un mensaje en caso de que se haya iniciado sesión de manera errónea:



En el app.module.ts es necesario importar las cosas necesarias para que se muestre el mensaje:

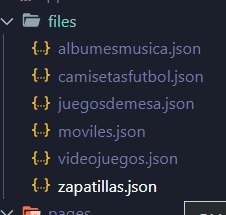


En el angular.json añadir en los estilos donde habíamos añadido el estilo de bootstrap añadir el de tostr:



**JSON**

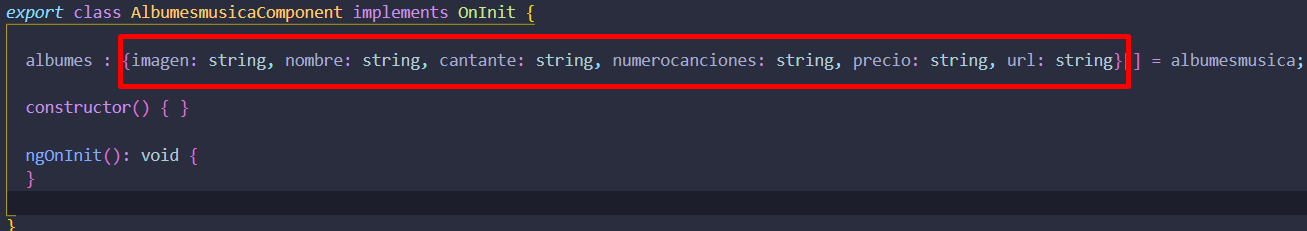
Creación del archivo json que se encuentra en la carpeta files:



Crea el componente necesario:



En el component.ts elegimos que es lo que queremos sacar del json:



Copiamos las cartas de bootstrap en el component.html se mostrará el contenido del json, sacando los datos de cada elemento del array mediante la directiva \*ngFor:

