

Resumen

Es una aplicación para gestionar una red social entre cooperativas utilizando Ionic/Angular y Firebase.

Requisitos Previos

Instalación de Node.js , Angular CLI , Firebase CLI

Instalación de dependencias

Dependencias Angular:

- **@angular/animations**
Descripción: Proporciona soporte para animaciones en aplicaciones Angular.
- **@angular/common**
Descripción: Contiene funcionalidades comunes para todas las aplicaciones Angular.
- **@angular/compiler**
Descripción: Compila los componentes y plantillas de Angular en código JavaScript.
- **@angular/core**
Descripción: Contiene las clases esenciales para construir aplicaciones Angular.
- **@angular/fire**
Descripción: Integración de Angular con Firebase.
- **@angular/forms**
Descripción: Proporciona soporte para construir y validar formularios en Angular.
- **@angular/platform-browser**
Descripción: Contiene las implementaciones comunes para el navegador.
- **@angular/platform-browser-dynamic**
Descripción: Proporciona el código necesario para iniciar la aplicación Angular en un navegador.
- **@angular/router**
Descripción: Implementa la navegación y la administración de rutas para aplicaciones Angular.
- **rxjs**
Descripción: Biblioteca para programación reactiva en JavaScript.
- **Tslib**
Descripción: Biblioteca de utilidades para TypeScript.
- **zone.js**
Descripción: Biblioteca que proporciona zona para Angular.

Dependencias Ionic:

- **@ionic/angular**
Descripción: Framework de IU basado en componentes para la construcción de aplicaciones móviles y web con Angular.
- **@ionic/pwa-elements**
Descripción: Proporciona elementos web que se pueden utilizar en aplicaciones progresivas de Ionic.
- **ionicons**
Descripción: Conjunto de iconos de IU para aplicaciones web y móviles.

Dependencias Firebase:

- **Firebase**

- **Descripción:** Plataforma de desarrollo de aplicaciones móviles y web que proporciona servicios en la nube, incluido el backend para aplicaciones.

Estructura del Proyecto

app: Aquí se encuentran los componentes, módulos y servicios principales.

components: Contiene componentes reutilizables, como custom-input, header, nuevaPublicacion y terminosycondiciones.

- **custom-input:** Componente para entradas personalizadas.
- **header:** Componente para el encabezado de la aplicación.
- **nuevaPublicacion:** Componente para crear nuevas publicaciones.
- **terminosycondiciones:** Componente que muestra los términos y condiciones.

guards:

- **auth.guard.ts:** Asegura que solo los usuarios autenticados puedan acceder a ciertas rutas.
- **no-auth.guard.ts:** Asegura que los usuarios no autenticados puedan acceder a ciertas rutas.

models:

- Contiene modelos como galeria.model.ts, grupo.model.ts, publicacion.model.ts y user.model.ts que definen las estructuras de datos.

pages:

Contiene subcarpetas como auth y main, que agrupan componentes específicos de la página.

auth:

- forgot-password: Página para recuperar contraseñas olvidadas.
- login: Página de inicio de sesión.
- register: Página de registro.

main:

- about: Página acerca del proyecto o empresa.
- cooperatives: Página con información sobre cooperativas.
- home: Página principal.
- lista-usuarios: Lista o directorio de usuarios.
- portada: Portada o landing page.
- profile: Perfil del usuario.
- publicar: Herramienta o interfaz para publicar contenido.

Configuración Firebase

Instalamos las dependencias con: `npm install firebase`

Añadimos la configuración de Firebase en environment:

```
3  export const environment = {
4    production: false,
5
6    firebaseConfig : {
7      apiKey: "AIzaSyCK_zDeNewKpKolzbCa-UvOMN8c0h9VnZA",
8      authDomain: "redsocal-fb3a3.firebaseio.com",
9      projectId: "redsocal-fb3a3",
10     storageBucket: "redsocal-fb3a3.appspot.com",
11     messagingSenderId: "334481362015",
12     appId: "1:334481362015:web:4f653d10d3484e11341fea"
13   }
14 };
15
16
```

Añadimos las importaciones necesarias:

```
TS app.module.ts X
src > app > TS app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3  import { RouteReuseStrategy } from '@angular/router';
4
5  import { IonicModule, IonicRouteStrategy } from '@ionic/angular';
6
7  import { AppComponent } from './app.component';
8  import { AppRoutingModule } from './app-routing.module';
9
10 //FIREBASE
11 import { AngularFireModule } from '@angular/fire/compat';
12 import { environment } from 'src/environments/environment';
13
```

Creamos un Servicio que se encargara de toda la interacción con Firebase:

```
2 import { Injectable, inject } from '@angular/core';
3 import { AngularFireAuth } from '@angular/fire/compat/auth';
4 import { getAuth, signInWithEmailAndPassword, createUserWithEmailAndPassword, updateProfile, sendPasswordResetEmail } from 'firebase/auth';
5 import { User } from '../models/user.model';
6 import { AngularFireStore } from '@angular/fire/compat/firestore';
7 import { getFirestore, setDoc, doc, getDoc, addDoc, collection, collectionData, query, updateDoc, deleteDoc } from '@angular/fire/firestore';
8 import { UtilsService } from './utils.service';
9 import { AngularFireStorage } from '@angular/fire/compat/storage';
10 import { getStorage, uploadString, ref, getDownloadURL, deleteObject } from 'firebase/storage';
11 import { Observable, catchError, map } from 'rxjs';
12 import { Grupo } from '../models/grupo.model';
13 import { QueryConstraint, getDocs } from 'firebase/firestore';
14
15 @Injectable({
16   providedIn: 'root'
17 })
18 export class FirebaseService {
19
20   constructor(
21     private utilsSvc: UtilsService,
22     private firestore: AngularFireStore,
23     private storage: AngularFireStorage,
24     private auth: AngularFireAuth
25   ) {}
26 }
```

Estructura de Datos

La bbdd consta de 3 colecciones, users, groups y publicaciones.

users

9fqhppHeyMNTzqgIGnMtRSyRrg12

+ Iniciar colección

groups

users >

+ Agregar documento

1PnNXSUSiCP0zDXAP04Ls...

4DCdHLnZctRgGTWrLdoJa...

8MIrhW890gMXBmfTHJ8HU...

9fqhppHeyMNTzqgIGnMtR... >

N6YN4ymsB4XRMEfAx6QgX...

NYr27TbrUGY85MI3dtqwx...

QbbdRWBZ7SMWdcse3MSVw...

SAT7VceUfmdKkFS2rpCLb...

Sx65L2HcB108EXL6JBiz2

+ Iniciar colección

publicaciones

+ Agregar campo

email: "alf@gmail.com"

image: "https://firebasestorage.googleapis.com/v0/b/redsocie

name: "Alf80"

uid: "9fqhppHeyMNTzqgIGnMtRSyRrg12"

Inicio de Sesión y Registro

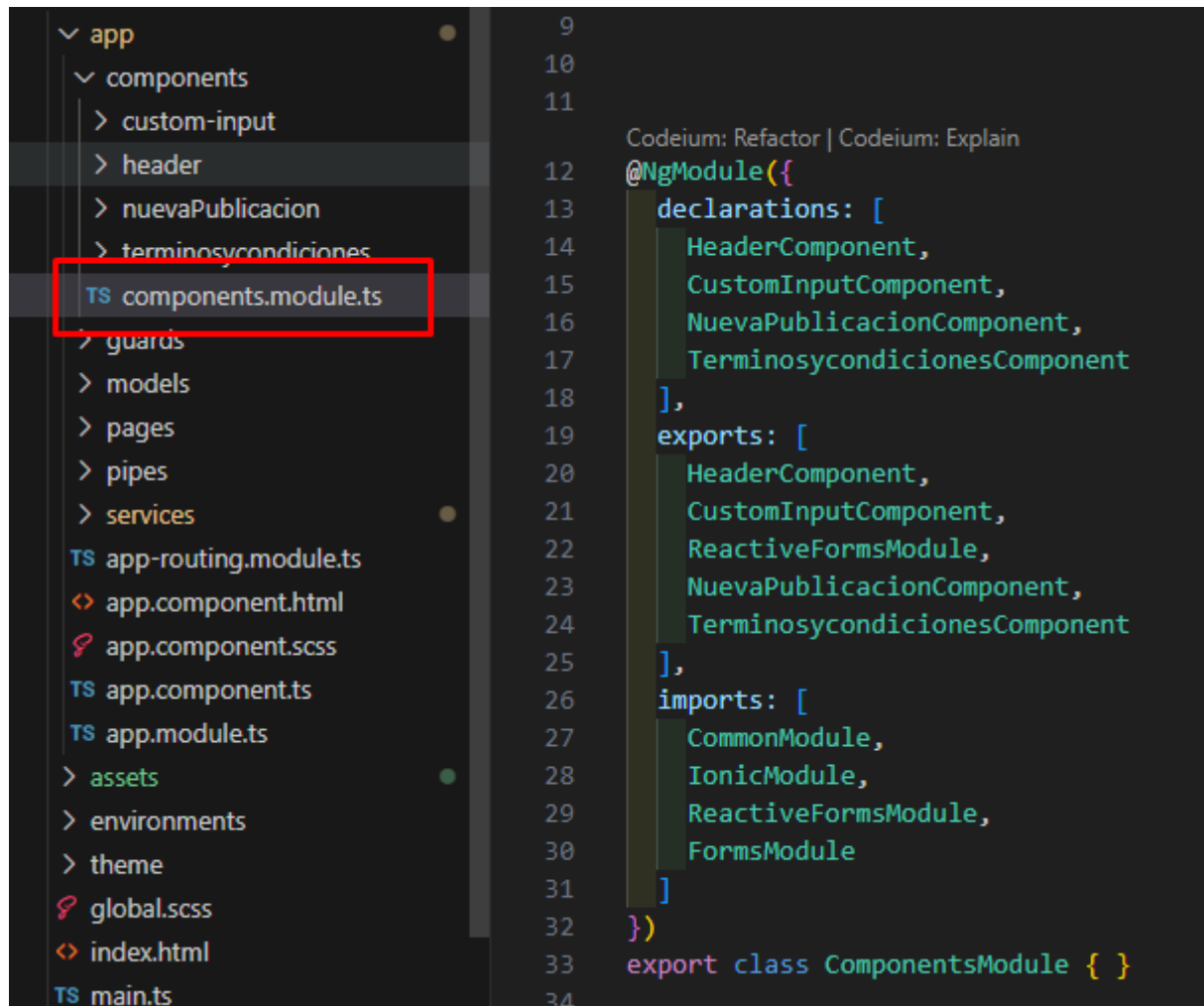
- Para las paginas de Registro (Login, Register y forgot-password) estan dentro de la carpeta Auth para poder evitar que un usuario sin registrar acceda al resto de paginas gracias al uso de guards.

```
const routes: Routes = [  
  {  
    path: '',  
    redirectTo: 'auth',  
    pathMatch: 'full'  
  },  
  {  
    path: 'auth',  
    loadChildren: () => import('./pages/auth/auth.module').then( m => m.AuthPageModule), canActivate: [NoAuthGuard]  
  },  
  {  
    path: 'main',  
    loadChildren: () => import('./pages/main/main.module').then( m => m.MainPageModule), canActivate: [AuthGuard]  
  },  
];
```

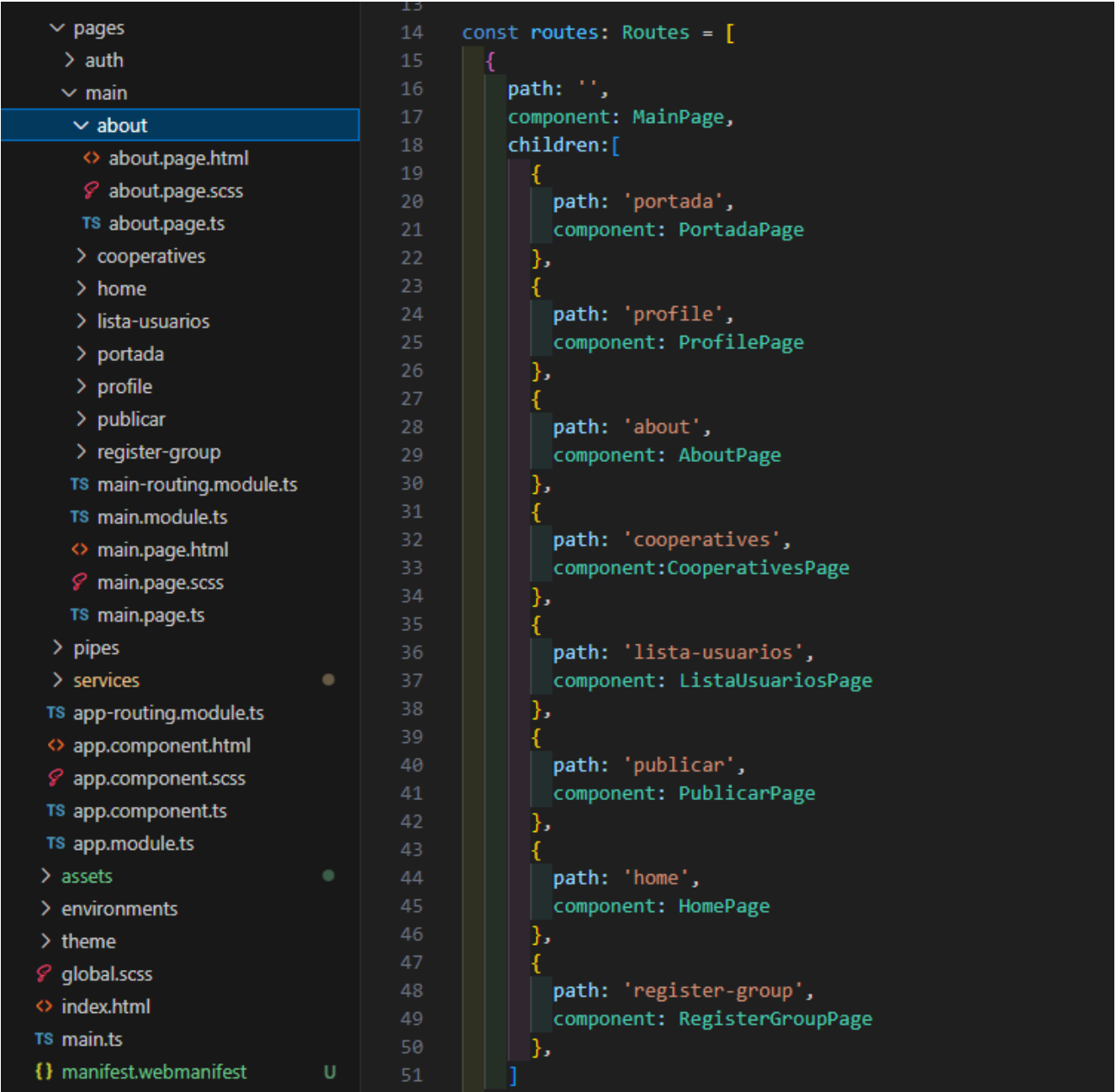
```
10 export class AuthGuard implements CanActivate {  
11  
12   firebaseSvc = inject(FirebaseService);  
13   utilsSvc = inject(UtilsService);  
14  
15   canActivate(  
16     route: ActivatedRouteSnapshot,  
17     state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {  
18  
19     let user = localStorage.getItem('user');  
20  
21     return new Promise((resolve) => {  
22       this.firebaseSvc.getAuth().onAuthStateChanged((auth) => {  
23         if(auth){  
24           if(user){  
25             resolve(true);  
26           }  
27         }else{  
28           // this.utilsSvc.routerLink('/auth');  
29           this.firebaseSvc.signOut();  
30           resolve(false);  
31         }  
32       })  
33     });  
34   }  
35 }
```

Paginas y Componentes:

- Para los componentes he generado un único modulo que se encarga de exportarlos todos.



- En cuanto a las paginas he modificado la estructura para tratarlos también como componentes, eliminando los modulos y routings.



The image shows a code editor interface. On the left is a file explorer with a tree view. The 'pages' folder is expanded, showing subfolders 'auth' and 'main'. The 'main' folder is further expanded, showing 'about' (selected), 'cooperatives', 'home', 'lista-usuarios', 'portada', 'profile', 'publicar', and 'register-group'. Below these are various files including HTML, SCSS, and TypeScript files. On the right, a TypeScript file contains a routing configuration. The code defines a 'routes' array with a root route and several child routes, each mapping a path to a specific component.

```
13
14 const routes: Routes = [
15   {
16     path: '',
17     component: MainPage,
18     children: [
19       {
20         path: 'portada',
21         component: PortadaPage
22       },
23       {
24         path: 'profile',
25         component: ProfilePage
26       },
27       {
28         path: 'about',
29         component: AboutPage
30       },
31       {
32         path: 'cooperatives',
33         component: CooperativesPage
34       },
35       {
36         path: 'lista-usuarios',
37         component: ListaUsuariosPage
38       },
39       {
40         path: 'publicar',
41         component: PublicarPage
42       },
43       {
44         path: 'home',
45         component: HomePage
46       },
47       {
48         path: 'register-group',
49         component: RegisterGroupPage
50       },
51     ]
52   }
53 ]
```

Creación y Gestión de Usuarios

Desde firebaseSvc.ts manejo la interacción de los usuarios con la BBDD:

```
// CREAR USUARIO
Codeium: Refactor | Explain | X
singUp(user: User) {
  return createUserWithEmailAndPassword(getAuth(), user.email, user.password)
}

// ACTUALIZAR USUARIO
Codeium: Refactor | Explain | X
updateUser(displayName: string) {
  return updateProfile(getAuth().currentUser, { displayName })
}

// OBTENER TODOS LOS USUARIOS
Codeium: Refactor | Explain | X
getAllUsers(): Observable<any[]> {
  const path = 'users';
  return this.getCollectionData(path);
}

// OBTENER UN USUARIO POR ID
Codeium: Refactor | Explain | X
getUserById(userId: string) {
  return this.getDocument(`users/${userId}`);
}
```

Y desde utilsSvc.service.ts el almacenamiento local y uso de la cámara:

<div><div>models</div><div>galeria.model.ts</div><div>grupo.model.ts</div><div>publicacion.model.ts</div><div>user.model.ts</div><div>pages</div><div>auth</div><div>forgot-password</div><div>forgot-password.page.html</div><div>forgot-password.page.scss</div><div>forgot-password.page.ts</div><div>login</div><div>login.page.html</div><div>login.page.scss</div><div>login.page.ts</div><div>register</div><div>register.page.html</div><div>register.page.scss</div><div>register.page.ts</div><div>auth-routing.module.ts</div><div>auth.module.ts</div><div>main</div><div>pipes</div><div>filtrado.pipe.ts</div><div>services</div><div>firebase.service.ts</div><div>utils.service.ts</div></div>	<div>70</div> <div>71</div> <div>Codeium: Refactor Explain X</div> <div>72</div> <div>73</div> <div>74</div> <div>75</div> <div>76</div> <div>77</div> <div>Codeium: Refactor Explain X</div> <div>78</div> <div>79</div> <div>80</div> <div>81</div> <div>82</div> <div>Codeium: Refactor Explain X</div> <div>83</div> <div>84</div> <div>85</div> <div>86</div> <div>87</div> <div>88</div> <div>89</div> <div>90</div> <div>91</div> <div>92</div> <div>93</div> <div>94</div> <div>95</div> <div>96</div> <div>97</div> <div>98</div>	<pre>// OBTENER USUARIO ACTUAL getCurrentUser() { const user = this.getFromLocalStorage('user'); return user ? user : null; } // ACTUALIZAR USUARIO ACTUAL updateCurrentUser(updatedUser: any) { this.saveInLocalStorage('user', updatedUser); } // MOSTRAR FOTO DEL USUARIO ACTUAL async showCurrentUserPhoto() { const currentUser = this.getCurrentUser(); if (currentUser && currentUser.photoUrl) { const alertOptions: AlertOptions = { header: 'Foto de Perfil', message: ``, buttons: ['Cerrar'] }; await this.presentAlert(alertOptions); } else { await this.presentToast({ message: 'No hay foto de perfil disponible.', duration: 2000, position: 'top' }); } }</pre>
--	--	--

Creación y Gestión de Grupos y Publicaciones

- Genero un modelo en donde defino las propiedades de cada uno, (al igual que para usuarios).
- Los nuevos grupos se muestran en formato lista en la página de Cooperativas.
- Las publicaciones pueden verse desde la pagina de portada, se añaden en orden inverso para que siempre salga primero la ultima publicación.

Uso de pipe para filtrar:

```
import { Pipe, PipeTransform } from '@angular/core';
import { Publicacion } from '../models/publicacion.model';

Codeium: Explain
@Pipe({
  name: 'filtrado'
})
export class FiltradoPipe implements PipeTransform {

  Codeium: Refactor | Explain | Generate JSDoc | X
  transform(items: any[], filtro:string, campoFiltrado:string = ''): any[] {
    if(!items || (!filtro && !campoFiltrado)){
      return items;
    }
    filtro = filtro.toLowerCase();
    return items.filter(item => {
      if (campoFiltrado && item[campoFiltrado]) {
        return item[campoFiltrado].toLowerCase().includes(filtro);
      } else {
        for (const key in item) {
          if (item.hasOwnProperty(key) && typeof item[key] === 'string') {
            if (item[key].toLowerCase().includes(filtro)) {
              return true;
            }
          }
        }
        return false;
      }
    });
  }
}
```

Utilizo el filtrado tanto para los Usuarios, grupos como publicaciones.

