

## **PROYECTO FINAL HLC (COOPERATIVAS)**

### **Introducción:**

La aplicación permite el registro y autenticación de los usuarios, con una opción para recuperar contraseña en caso de olvidarla. Los usuarios pueden añadir publicaciones que se muestran tanto en portada como en la página de perfil, los usuarios pueden añadir grupos nuevos.

Tiene un sistema de filtrado para la búsqueda de usuarios, grupos y publicaciones.

Toda la información se almacena a través de Firebase.

También hago uso de los guardianes para no permitir el acceso a los usuarios no registrados.

Puedes ver el proyecto desde:

<https://cooperativas.vercel.app/auth>

### **INSTALACIÓN Y CONFIGURACIÓN**

Añadimos ionic de forma global: `npm install -g @ionic/cli`

Comenzamos un nuevo proyecto con: `ionic start cooperativa`

Añado las siguientes dependencias:

```
npm install yarn
npm install @angular/core
npm install firebase
npm install @angular/fire
npm install @asymmetrik/ngx-leaflet
npm install @types/leaflet
npm install @capacitor/android
npm install @capacitor/camera
```

`npx cap add Android`

**FIREBASE:**

**ENVIRONMENT**

```
export const environment = {  
  production: true,  
  
  firebaseConfig : {  
    apiKey: "AIzaSyCK_zDeNewKpKolzbCa-UvOMN8c0h9VnZA",  
    authDomain: "redsocal-fb3a3.firebaseio.com",  
    projectId: "redsocal-fb3a3",  
    storageBucket: "redsocal-fb3a3.appspot.com",  
    messagingSenderId: "334481362015",  
    appId: "1:334481362015:web:4f653d10d3484e11341fea"  
  }  
};
```

## APP.MODULE

```
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
```

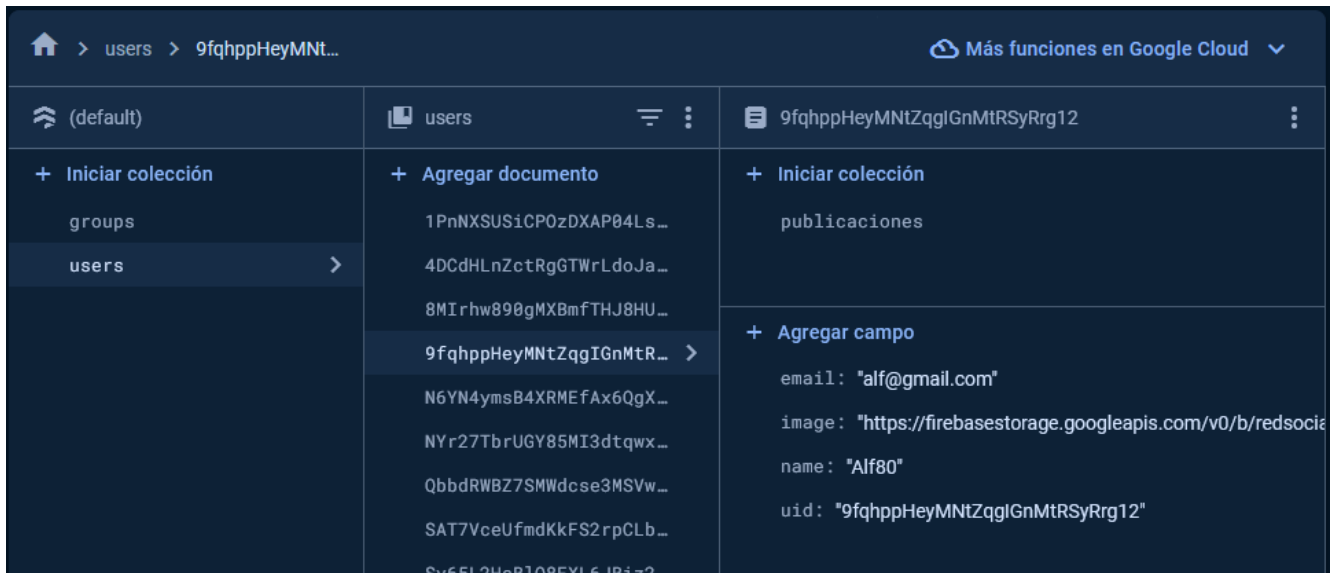
## SERVICIO PARA GESTIONAR LA COMUNICACIÓN CON FIREBASE

```
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 Codeium: Explain
16 @Injectable({
17   providedIn: 'root'
18 })
19 export class FirebaseService {
20
21   Codeium: Refactor | Explain | Generate JSDoc | X
22   constructor(
23     private utilsSvc: UtilsService,
24     private firestore: AngularFireStore,
25     private storage: AngularFireStorage,
26     private auth: AngularFireAuth
27   ) {}
28 }
```

```
TS firebase.service.ts X
src > app > services > TS firebase.service.ts > ...
18 export class FirebaseService {
19
20   // AUTENTICACION FIREBASE
21   Codeium: Refactor | Explain | X
22   getAuth() {
23     return getAuth();
24   }
25
26   // INICIAR SESION
27   Codeium: Refactor | Explain | X
28   signIn(user: User) {
29     return signInWithEmailAndPassword(user.email, user.password);
30   }
31
32   // CREAR USUARIO
33   Codeium: Refactor | Explain | X
34   signUp(user: User) {
35     return createUserWithEmailAndPassword(user.email, user.password);
36   }
37
38   // ACTUALIZAR USUARIO
39   Codeium: Refactor | Explain | X
40   updateUser(displayName: string) {
41     return updateProfile(getAuth().currentUser, { displayName });
42   }
43
44   // OBTENER TODOS LOS USUARIOS
45   Codeium: Refactor | Explain | X
46   getAllUsers(): Observable<any[]> {
47     const path = 'users';
48     return this.getCollectionData(path);
49   }
50
51   // OBTENER UN USUARIO POR ID
52   Codeium: Refactor | Explain | X
53   getUserById(userId: string) {
54     return this.getDocument(`users/${userId}`);
55   }
56
57   Codeium: Refactor | Explain | Generate JSDoc | X
58   sendRecoveryEmail(email: string) {
59     return sendPasswordResetEmail(email);
60   }
61
62   Codeium: Refactor | Explain | Generate JSDoc | X
63 }
```

```
TS firebase.service.ts X
src > app > services > TS firebase.service.ts > ...
18 export class FirebaseService {
19
20   Codeium: Refactor | Explain | Generate JSDoc | X
21   sendRecoveryEmail(email: string) {
22     return sendPasswordResetEmail(email);
23   }
24
25   Codeium: Refactor | Explain | Generate JSDoc | X
26   signInOut() {
27     getAuth().signInOut();
28     localStorage.removeItem('user');
29     this.utilsSvc.routerLink('/auth');
30   }
31
32   // BASE DE DATOS
33
34   // CREAR UNA COLECCION
35   Codeium: Refactor | Explain | X
36   async createCollection(path: string) {
37     const firestore = getFirestore();
38     const docRef = await addDoc(collection(firestore, path));
39     return docRef;
40   }
41
42   // OBTENER UNA COLECCION
43   Codeium: Refactor | Explain | X
44   getCollectionData(path: string, collectionQuery?: QueryConstraint) {
45     const ref = collection(getFirestore(), path);
46     if (collectionQuery) {
47       return collectionData(query(ref, collectionQuery));
48     } else {
49       return collectionData(ref, { id: 'data' });
50     }
51   }
52
53   Codeium: Refactor | Explain | Generate JSDoc | X
54   setDocument(path: string, data: any) {
55     return setDoc(doc(getFirestore(), path), data);
56   }
57
58   // ACTUALIZAR UN DOCUMENTO
59   Codeium: Refactor | Explain | X
60   updateDocument(path: string, data: any) {
61     return updateDoc(doc(getFirestore(), path), data);
62   }
63 }
```

```
TS firebase.service.ts X
src > app > services > TS firebase.service.ts > ...
18 export class FirebaseService {
19
20   // OBTENER TODOS LOS GRUPOS
21   Codeium: Refactor | Explain | X
22   getAllGroups(): Observable<Grupo[]> {
23     const path = 'grupos';
24     return this.getCollectionData(path);
25   }
26
27   // OBTENER UN GRUPO POR ID
28   Codeium: Refactor | Explain | X
29   getGroupById(groupId: string) {
30     const path = `grupos/${groupId}`;
31     return this.getDocument(path);
32   }
33
34   // CREAR UN NUEVO GRUPO
35   Codeium: Refactor | Explain | X
36   async createGroup(group: Grupo, userId: string) {
37     try {
38       group.members = [userId];
39       group.createdAt = new Date();
40       const path = `grupos/${groupId}`;
41       const { id, ...groupWithoutId } = group;
42       return addDoc(collection(getFirestore(), path), groupWithoutId);
43     } catch (error) {
44       console.error('Error al crear grupo', error);
45       throw error;
46     }
47   }
48
49   // ACTUALIZAR UN GRUPO
50   Codeium: Refactor | Explain | X
51   updateGroup(groupId: string, updateData: Partial<Grupo>) {
52     const path = `grupos/${groupId}`;
53     return this.updateDocument(path, updateData);
54   }
55
56   // ELIMINAR UN GRUPO
57   Codeium: Refactor | Explain | X
58   deleteGroup(groupId: string) {
59     const path = `grupos/${groupId}`;
60     return this.deleteDocument(path);
61   }
62 }
```



## ESTRUCTURA Y CREACION DE RUTAS

```
ionic g c components/custom-input
ionic g c components/header
ionic g c components/nuevaPublicacion
components.module.ts

ionic g page pages/auth
ionic g page pages/auth/login
ionic g page pages/auth/register
ionic g page pages/auth/forgot-password
auth.module.ts

ionic g page pages/main
ionic g page pages/main/about
ionic g page pages/main/calendar
ionic g page pages/main/cooperatives
ionic g page pages/main/home
ionic g page pages/main/lista-usuarios
ionic g page pages/main/map
ionic g page pages/main/perfil-grupo
ionic g page pages/main/portada
ionic g page pages/main/profil
ionic g page pages/main/publicar
ionic g page pages/main/register-group

ionic g p pipes/filtrado

ionic g service services/firebase
ionic g service services/utills
```

## 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.

```

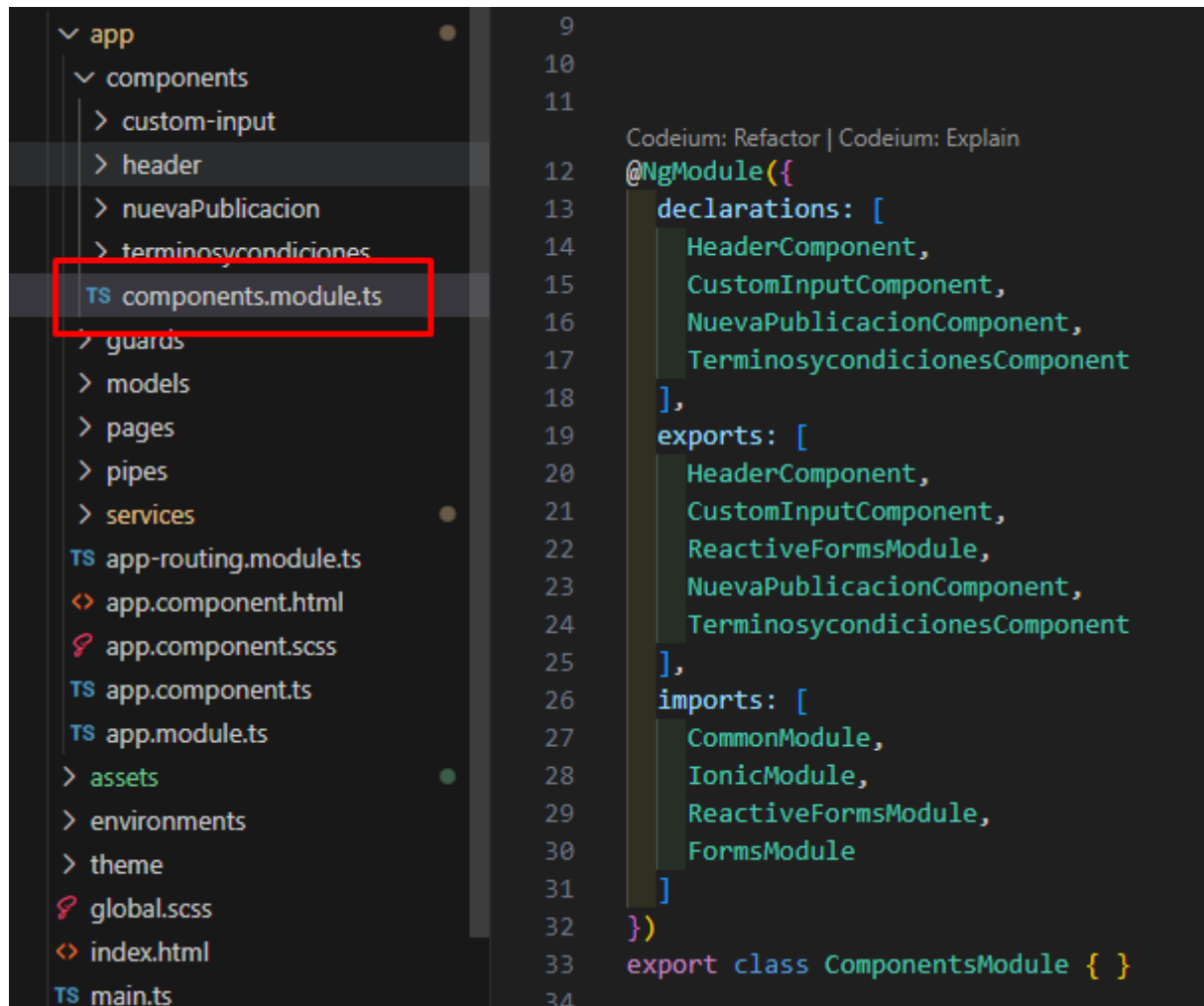
const routes: Routes = [
  {
    path: '',
    redirectTo: 'auth',
    pathMatch: 'full'
  },
  {
    path: 'auth',
    loadChildren: () => import('./pages/auth/auth.module').then( m => m.AuthPageModule), canActivate: [NoAuthGuard]
  },
  {
    path: 'main',
    loadChildren: () => import('./pages/main/main.module').then(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

```

> components	10	export class AuthGuard implements CanActivate {
▼ guards	11	
TS auth.guard.ts	12	firebaseSvc = inject(FirebaseService);
TS no-auth.guard.ts	13	utilsSvc = inject(UtilsService);
> models	14	
> pages	15	Codeium: Refactor   Explain   Generate JSDoc   X
> pipes	16	canActivate(
> services	17	route: ActivatedRouteSnapshot,
TS app-routing.module.ts	18	state: RouterStateSnapshot): Observable<boolean   UrlTree>   Prom
<> app.component.html	19	
app.component.scss	20	let user = localStorage.getItem('user');
TS app.component.ts	21	
TS app.module.ts	22	return new Promise((resolve) => {
> assets	23	this.firebaseSvc.getAuth().onAuthStateChanged((auth) => {
> environments	24	if(auth){
> theme	25	if(user){
global.scss	26	resolve(true);
<> index.html	27	}
TS main.ts	28	}else{
{ manifest.webmanifest U	29	// this.utilsSvc.routerLink('/auth');
TS polyfills.ts	30	this.firebaseSvc.signInOut();
TS test.ts	31	resolve(false);
TS zone-flags.ts	32	}
	33	});
	34	}
	35	

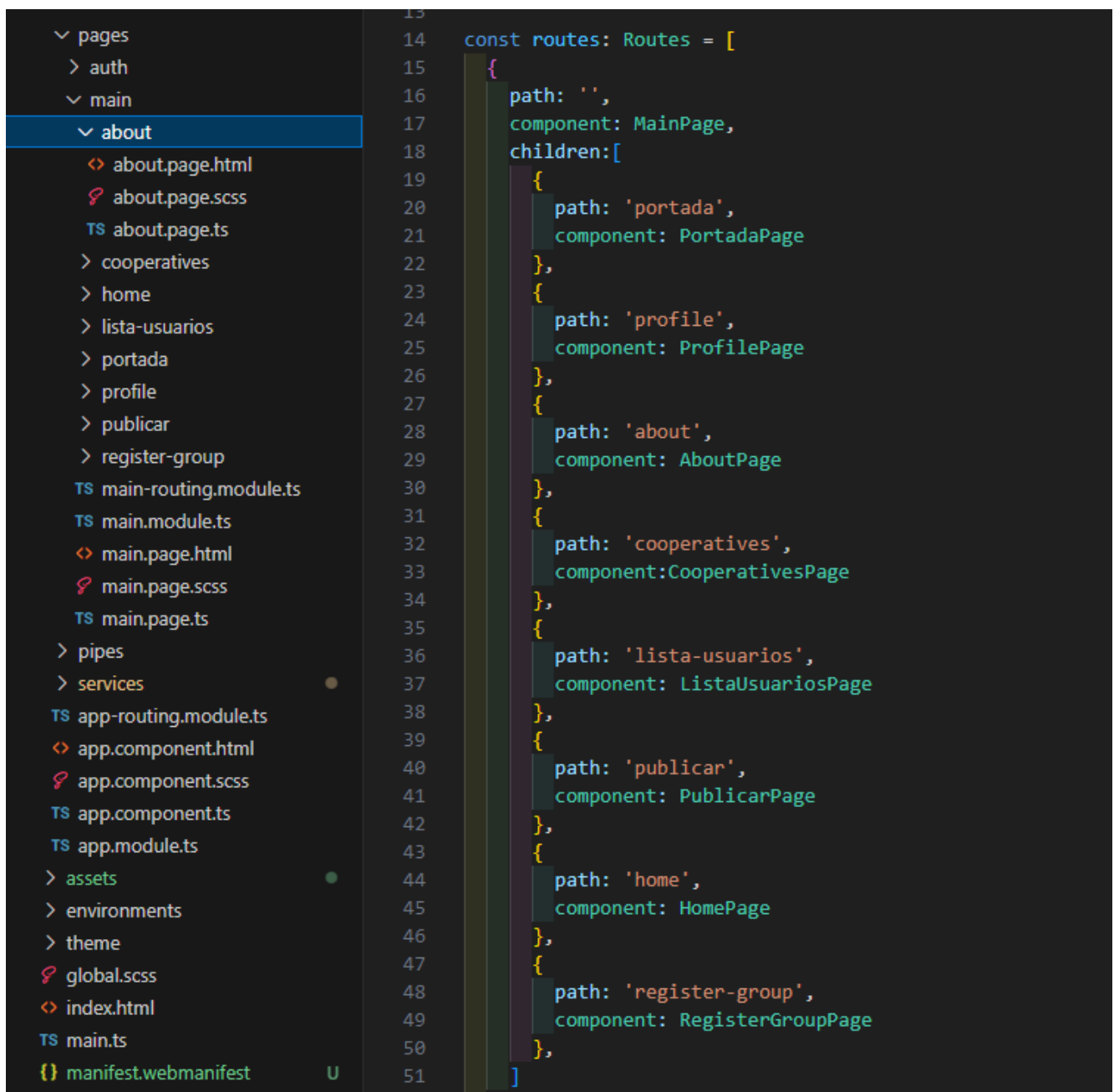
## Paginas y Componentes:

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



```
9
10
11
12 Codeium: Refactor | Codeium: Explain
13 @NgModule({
14   declarations: [
15     HeaderComponent,
16     CustomInputComponent,
17     NuevaPublicacionComponent,
18     TerminosycondicionesComponent
19   ],
20   exports: [
21     HeaderComponent,
22     CustomInputComponent,
23     ReactiveFormsModule,
24     NuevaPublicacionComponent,
25     TerminosycondicionesComponent
26   ],
27   imports: [
28     CommonModule,
29     IonicModule,
30     ReactiveFormsModule,
31     FormsModule
32   ]
33 })
34 export class ComponentsModule { }
```

- En cuanto a las paginas he modificado la estructura para tratarlos también como componentes siguiendo la arquitectura de angular en el cual no existen paginas como tal, lo trata todo como componentes independientemente de su función.



```
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>TS galeria.model.ts</div><div>TS grupo.model.ts</div><div>TS publicacion.model.ts</div><div>TS 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>TS forgot-password.page.ts</div><div>▼ login</div><div>◀ login.page.html</div><div>🔗 login.page.scss</div><div>TS login.page.ts</div><div>▼ register</div><div>◀ register.page.html</div><div>🔗 register.page.scss</div><div>TS register.page.ts</div><div>TS auth-routing.module.ts</div><div>TS auth.module.ts</div><div>&gt; main</div><div>▼ pipes</div><div>TS filtrado.pipe.ts</div><div>▼ services</div><div>TS firebase.service.ts</div><div>TS utils.service.ts</div></div>	<div>70</div> <div>71</div> <div>72</div> <div>73</div> <div>74</div> <div>75</div> <div>76</div> <div>77</div> <div>78</div> <div>79</div> <div>80</div> <div>81</div> <div>82</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 Codeium: Refactor   Explain   X getCurrentUser() {   const user = this.getFromLocalStorage('user');   return user ? user : null; }  // ACTUALIZAR USUARIO ACTUAL Codeium: Refactor   Explain   X updateCurrentUser(updatedUser: any) {   this.saveInLocalStorage('user', updatedUser); }  // MOSTRAR FOTO DEL USUARIO ACTUAL Codeium: Refactor   Explain   X async showCurrentUserPhoto() {   const currentUser = this.getCurrentUser();    if (currentUser &amp;&amp; currentUser.photoUrl) {     const alertOptions: AlertOptions = {       header: 'Foto de Perfil',       message: `&lt;img src="\${currentUser.photoUrl}" class="profile-photo" /&gt;`,       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.

## PRODUCCIÓN

ionic build

npx cap open android

## DESDE ANDROID STUDIO

Genero la apk:

