Login

Nous allons utiliser un package pour Firebase afin d'utiliser une méthode d'authentifcation classique.

Installez le package suivant :

```
npm install firebase --save
```

Puis dans le module **AppModule** importez firebase et configurez la connexion à la base de données. Vous trouverez ces informations dans les paramètres de Firebase dans "ajoutez Firebae à votre application Web" :

```
import * as firebase from 'firebase';

// complétez avec vos propres identifiants
const firebaseConfig = {
  apiKey: "KEY_API",
  authDomain: "NAME_BD",
  databaseURL: "URL",
  projectId: "PROJECT_ID",
  storageBucket: "STORAGE",
  messagingSenderId: "MESSAGING_ID"
};

// initialisez Firebase
firebase.initializeApp(firebaseConfig);
```

Vous devez également configurer Firebase. Allez dans l'onglet "Authentification", choisissez le mode d'authentification suivant dans l'onglet "Mode de connexion" : Adress e-mail/Mot de passe et activez cette option. Enfin, ajoutez un utilisateur pour l'application, créez un email et mot de passe.

Nous allons maintenant utiliser une méthode de Firebase pour s'authentifier.

Exercice 3

Créez un service AuthService dans l'application et implémentez la méthode auth dans ce service. Ici l'approche change dans le code car Firebase utilise des promesses pour la gestion de l'authentification :

```
import { Injectable } from '@angular/core';

// Importez les modules nécessaires pour l'authentification
import * as firebase from 'firebase/app';
import 'firebase/auth';

@Injectable({
```

```
  providedIn: 'root'
})
export class AuthService {

  constructor() { }

// méthode d'authentification
  auth(email: string, password: string): Promise<any> {

    return firebase.auth().signInWithEmailAndPassword(email, password);
  }
}
```

Créez un component DashboardComponent, pour l'instant ne faite rien dans ce dernier.

En utilisant la méthode auth que nous venons de voir terminez l'exercice en utilisant cette méthode dans la partie login de l'application. Vous utiliserez le service router d'Angular pour faire une redirection vers la page du dashboard une fois authentifié.

Gérez un message d'erreur si l'utilisateur se trompe dans le choix de l'email et du password.

Pensez à créer la route dashboard.

Exercice 4

Nous allons maintenant mettre en place la protection des routes pour accéder au dashboard.

Créez tout d'abord le service GuardService et importez dans ce service les services router et AuthService :

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, Router } from '@angular/r
import { AuthService } from './auth.service';

@Injectable({
  providedIn: 'root'
})
export class GuardService implements CanActivate {

  constructor(private aS: AuthService, private router: Router) { }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): any | boolean {
```

```
    // TODO ...

    return false;
  }
}
```

Protégez les routes dans l'AppModule de la manière suivante :

```
const albumsRoutes: Routes = [
    // ...
    {
        path: 'dashboard', canActivate: [GuardService],
        component: DashboardComponent
    },
];
```

Dans le service AuthService vous allez mettre en place la logique d'authentification avec Firebase. C'est dans ce service que nous allons vérifier que l'utilisateur est bien connecté.

Définissez une propriété authState boolean que nous utiliserons pour tester l'authentification. Voyez le code qui suit :

```
export class AuthService {

  // état de la connexion
  private authState: boolean = false;

  constructor(private router: Router) {
    // Observable il teste si l'utilisateur est connecté
    firebase.auth().onAuthStateChanged( (user) => {
      if (user) {
        this.authState = true;
      } else {
        this.authState = null;
      }
    });
  }
  // ...
}
```

Dans la classe GuardService il faudra alors mettre la logique suivante pour vérifier le statut de la connexion :

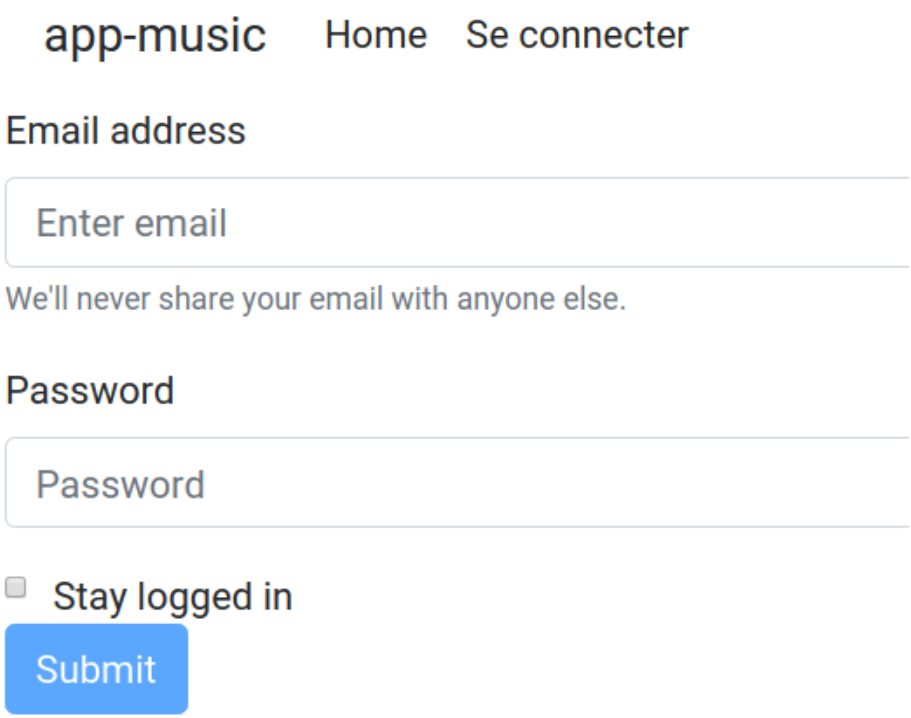
```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, RouterStateSnapshot, Router } from '@angular/r
import { AuthService } from './auth.service';
```

```
@Injectable({
  providedIn: 'root'
})
export class GuardService implements CanActivate {

  constructor(private aS: AuthService, private router: Router) {}

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): any | boolean {

    if (this.aS.authenticated()) return true;

    return this.aS.currentUserObservable().onAuthStateChanged(
      user => {
        if (user === null) {
          this.router.navigate(['/login'], {
            queryParams: { messageError: 'Error authentification' }
          });
        }
      }
    )
  }

}
```

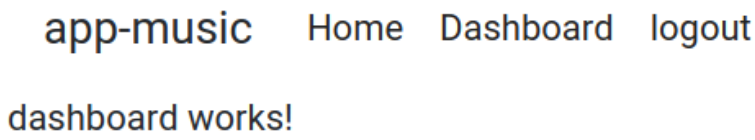
Terminez l'exercice en créant la méthode logout pour se déconnecter et faites en sorte que lorsqu'on est connecté on est dans le menu principal les options logout et dashboard qui apparaissent à la place de login, voyez l'image qui suit.

Indications : vous devrez utiliser la propriété authState du service AuthService dans le menu pour vérifier ce point.

Non connecté :



Et une fois connecté :



dashboard works!