esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire: 2024-2025 ANGULAR

Workshop d'authentification :

Guard dans un projet angular

Objectifs :

- Sécuriser l'accès à l'application en utilisant json web token (jwt)
- Créer un formulaire d'authentification
- Appliquer les différentes notions déjà vue dans le cours

Travail demandé :

1. Dans votre projet angular, installez-le package suivant pour gérer l'authentification avec json web token (jwt)

   npm install -g json-server json-server-auth

2. Créez un fichier db.json sous le projet
3. Lancer la commande **json-server-auth db.json**
4. Soit un modèle de données « **user** » ayant les attributs suivants : email, pasword, role, name, surname
5. Dans le fichier db.sjon ajoutez une source appelée « **user » de type** avec tableau de User.
6. Ouvrez l'application **Postman**, saisissez l'URL « **http://localhost:3000/register** » dans la barre d'adresse, et ajoutez les données suivantes dans le corps de la requête comme indiqué dans la **Figure 1**

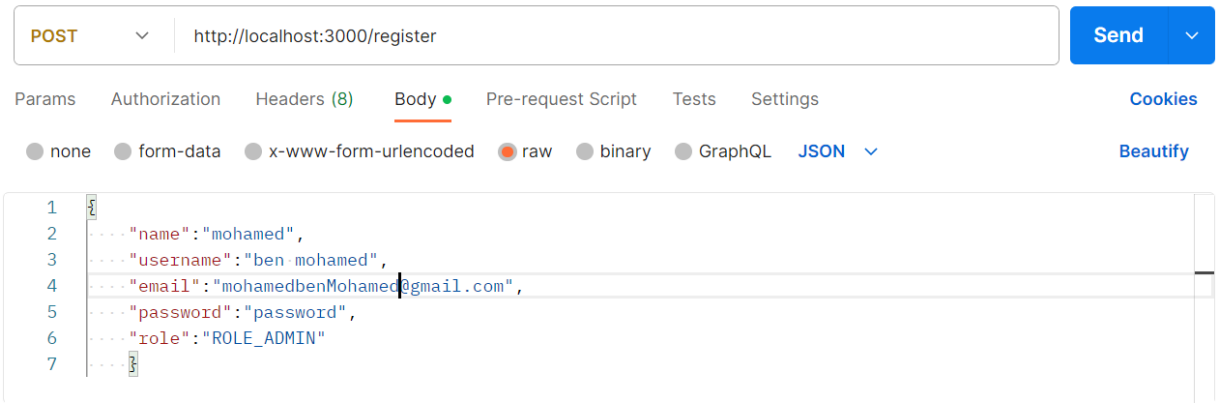
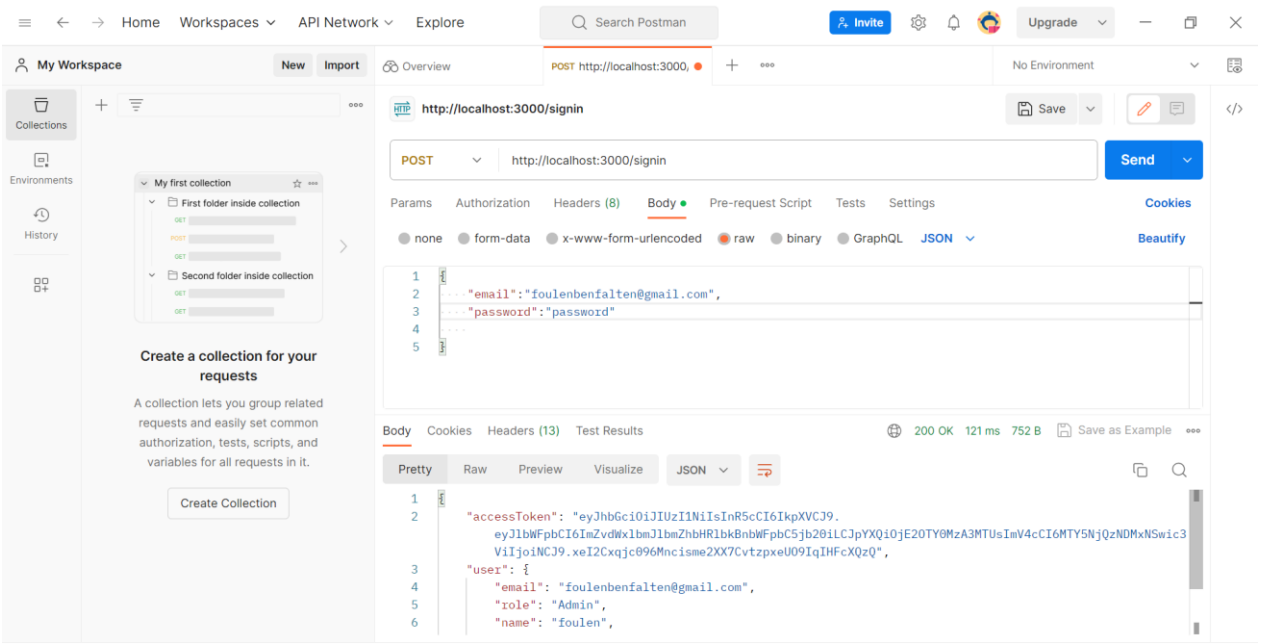
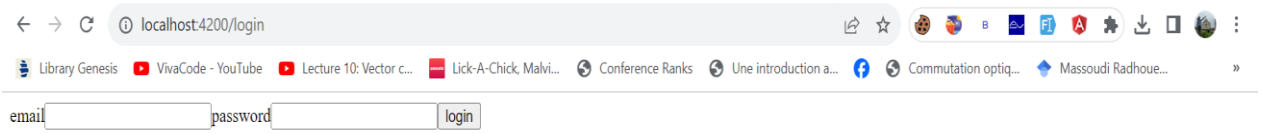


Figure 1

7. Pour s'authentifier il faut taper dans l'url **http://localhost:3000/signin**



8. Créez un composant **loginComponent** affichant le formulaire de la **figure XX** :



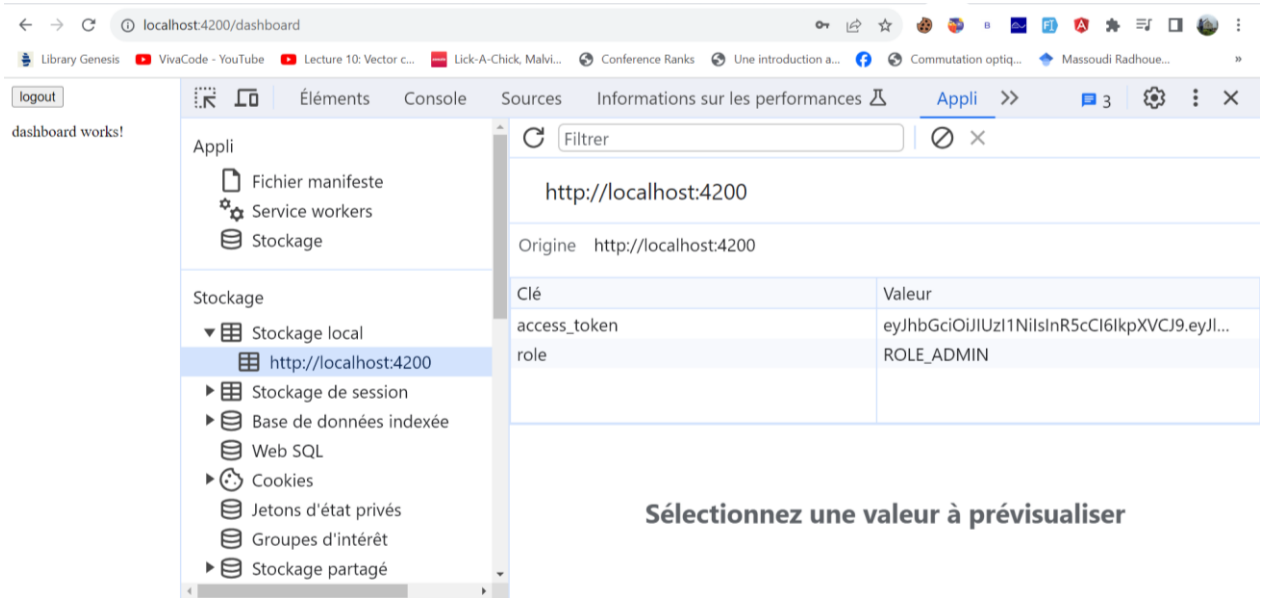
9. Créez un service « **AuthentificationService »** et ajoutez le code source suivant :

```
 constructor(private http:HttpClient) { }

  singin(data:any):Observable<User>{
    return this.http.post<User>('http://localhost:3000/signin',data);
  }
```

10. Ajoutez le code source suivant dans la classe de composant **loginComponent**

```
constructor(private authService: AuthentificationService){

  }


  login(d:any): void {

    this.authService.singin(d).subscribe(
      (response) => {
        // Stockez le jeton JWT dans le stockage local

       localStorage.setItem('access_token', response.accessToken);
       localStorage.setItem('role', response.role);




      }
    );
  }
```

11. Appelez la méthode login() depuis le template du composant **loginComponent**
12. Enter le login et password dans l'interface puis un tokon sera stocké dans le local Storage de navigateur comme indique la figure XX :



13. Dans le service **AuthentificationService,** implémentez la méthode **_is_logged()** qui permet de vérifier si l'utilisateur est connecté ou pas.

```
_is_logged(): boolean {
  return !!localStorage.getItem('access_token');
  }
```

14. Nous allons ajouter l'obligation de l'authentification, pour cela lancez la commande suivante pour générer un **guard « ng generate guard authentification »** et choisissez la méthode **canActivate()**

15. Appelez la méthode **is_logged()** dans la méthode **canActivate()** dans le fichier **authentification.guard.ts**

```
constructor(private authService: AuthentificationService) {
  }
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean
| UrlTree> | boolean | UrlTree {
    if (this.authService._is_logged() ) {
      return  true
        }
    return false;
  }
```

16. Ajouter le **canActivate** aux routes nécessitantes une authentification comme montre la figure suivante.

```
{    path: 'dashboard', component: DashboardComponent, canActivate:
[AuthentificationGuard]}
```

17. Nous allons attribuer un rôle pour accéder à la page "**dashboard**" et mettre en place la méthode "**getRole(role: String)**" dans le service **AuthentificationService** pour vérifier le rôle de l'utilisateur.

```
getRole(role:String) {

    this.roleAs = localStorage.getItem('role');
        if(this.roleAs==role)
      return true;
    return false
  }
```

18. Créez un garde en utilisant la commande '**ng generate guard roleGuard**' et sélectionnez la méthode 'canActivate'.
19. Ajouter le code source suivant dans le fichier **role.guard.ts**

```
 constructor(private authService:AuthentificationService ) {
  }
  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean
| UrlTree> | boolean | UrlTree {
    if (this.authService.getRole('ROLE_ADMIN') ) {
      return true; // L'utilisateur est authentifié, autorisez l'accès à la route
(rôle admin)
    }
    this.authService.logOut();
    return false;
  }
```

20. Ajoutez le **guard** rôle dans le fichier de routing

```
  {

    path: 'dashboard', component: DashboardComponent, canActivate:
[AuthentificationGuard,RoleGuard]
  },
```

21. Testez le code