TP 03 : Authentification (JWT) et gestion des Roles –(Partie 1 Angular)

1-Modifier le modèle **Userr1** en ajoutant la classe **Role** :

export class role { constructor(public id:number, public libelle:string){} }

puis modifier l'attribut « **role : number « par « roles:role[] » et modifier tous les fichier nécessaire.**

2- Générer les **composants**, les **services** et la **guard** suivants :

ng generate component login

ng generate component home

ng generate component board-admin

ng generate component board-user

ng generate service services/auth

ng generate service services/TokenStorage

ng generate interceptor helpers/auth

ng g guard services/auth

3-Compléter le fichier **: helpers**/*auth.interceptor.ts* avec le code suivant**:**

```
import { Injectable } from '@angular/core';
import {HttpRequest, HttpHandler,   HttpEvent, HttpInterceptor, HTTP_INTERCEPTORS
} from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>
{
    req = req.clone({
      withCredentials: true,
    });

    return next.handle(req);
  }
}
export const httpInterceptorProviders = [
  { provide: HTTP_INTERCEPTORS, useClass: AuthInterceptor, multi: true },
];
```

4-Ouvrez le fichier : **app.module.ts**, puis importez : ***httpInterceptorProviders, ensuite*** ajouter **authInterceptorProviders** dans les fournisseurs (**Providers**).

5-Compléter le fichier **:** services/***auth.service.ts*** avec le code suivant**:**

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Observable } from 'rxjs';

const AUTH_API = 'http://localhost:8080/api/auth/';

const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};

@Injectable({
  providedIn: 'root',
})
export class AuthService {
  constructor(private http: HttpClient) {}

  login(username: string, password: string): Observable<any> {
    return this.http.post(
      AUTH_API + 'signin',
      {
        username,
        password,
      },
      httpOptions
    );
  }

  logout(): Observable<any> {
    return this.http.post(AUTH_API + 'signout', { }, httpOptions);
  }
}
```

6-Compléter le fichier **: services/token-*storage.service.ts*** avec le code suivant:

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

const USER_KEY = 'auth-user';

@Injectable({
  providedIn: 'root'
})
export class TokenStorageService {
  constructor() {}

  clean(): void {
    window.sessionStorage.clear();
  }

  public saveUser(user: any): void {
    window.sessionStorage.removeItem(USER_KEY);
    window.sessionStorage.setItem(USER_KEY, JSON.stringify(user));
  }

  public getUser(): any {
    const user = window.sessionStorage.getItem(USER_KEY);
    if (user) {
      return JSON.parse(user);
    }

    return {};
  }

  public isLoggedIn(): boolean {
    const user = window.sessionStorage.getItem(USER_KEY);
    if (user) {
      return true;
    }

    return false;
  }
}
```

Explication :

TokenStorageService gère les informations utilisateur (*nom d'utilisateur, e-mail, rôles*) dans le stockage de session du navigateur. Pour la déconnexion, nous effacerons ce stockage de session.

7-Compléter le fichier **: services/http-*client.service.ts*** avec le code suivant:

```
.....
private  API_URL = 'http://localhost:8080/api/test/';

getPublicContent(): Observable<any> {
  return this.httpClient.get(this.API_URL + 'all', { responseType: 'text' });
}

getUserBoard(): Observable<any> {
  return this.httpClient.get(this.API_URL + 'user', { responseType: 'text' });
}

getAdminBoard(): Observable<any> {
  return this.httpClient.get(this.API_URL + 'admin', { responseType: 'text' });
}
```

8-Installer le module ***Bootstrap*** avec la commande : ***`npm install bootstrap@4.6.2`***
ensuite ajouter le code ci-dessous dans le fichier : **src/*style.css***:

```
@import "~bootstrap/dist/css/bootstrap.css"
```

9-Compléter le fichier **: login.*component.ts*** avec le code ci-desous:

Composant Login : Le composant de connexion utilise également ***AuthService*** pour travailler avec l'objet **Observable**. En plus de cela, il appelle les méthodes ***StorageService*** pour vérifier l'état de connexion et enregistrer les informations de l'utilisateur dans le stockage de session.

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from '../services/auth.service';
import { TokenStorageService } from '../services/token-storage.service';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
```

```
export class LoginComponent implements OnInit {
  form: any = {
    username: null,
    password: null
  };
  isLoggedIn = false;
  isLoginFailed = false;
  errorMessage = '';
  roles: string[] = [];

  constructor(private authService: AuthService, private storageService:
TokenStorageService) { }

  ngOnInit(): void {
    if (this.storageService.isLoggedIn()) {
      this.isLoggedIn = true;
      this.roles = this.storageService.getUser().roles;
    }
  }

  onSubmit(): void {
    const { username, password } = this.form;

    this.authService.login(username, password).subscribe({
      next: data => {
        this.storageService.saveUser(data);

        this.isLoginFailed = false;
        this.isLoggedIn = true;
        this.roles = this.storageService.getUser().roles;
        this.reloadPage();
      },
      error: err => {
        this.errorMessage = err.error.message;
        this.isLoginFailed = true;
      }
    });
  }

  reloadPage(): void {
    window.location.reload();
  }
}
```

10-Compléter le fichier : login.*component.html* **avec le code suivant:**

```
<div class="col-md-12">
    <div class="card card-container">
      <img id="profile-img"  src="//ssl.gstatic.com/accounts/ui/avatar_2x.png"
        class="profile-img-card"       />
      <form
        *ngIf="!isLoggedIn"
        name="form"
        (ngSubmit)="f.form.valid && onSubmit()"
        #f="ngForm"
        novalidate>
        <div class="form-group">
          <label for="username">Username</label>
          <input
            type="text"
            class="form-control"
            name="username"
            [(ngModel)]="form.username"
            required
            #username="ngModel"
            [ngClass]="{ 'is-invalid': f.submitted && username.errors }"
          />
          <div *ngIf="username.errors && f.submitted" class="invalid-feedback">
            Username is required!
          </div>
        </div>
        <div class="form-group">
          <label for="password">Password</label>
          <input
            type="password"
            class="form-control"
            name="password"
            [(ngModel)]="form.password"
            required
            minlength="6"
            #password="ngModel"
            [ngClass]="{ 'is-invalid': f.submitted && password.errors }"
          />
          <div *ngIf="password.errors && f.submitted" class="invalid-feedback">
            <div *ngIf="password.errors['required']">Password is required</div>
            <div *ngIf="password.errors['minlength']">
              Password must be at least 6 characters
            </div>
          </div>
```

```
        </div>
        <div class="form-group">
          <button class="btn btn-primary btn-block">            Login
          </button>
        </div>
        <div class="form-group">
          <div *ngIf="f.submitted && isLoginFailed" class="alert alert-danger"
role="alert">
            Login failed: {{ errorMessage }}
          </div>
        </div>
      </form>
      <div class="alert alert-success" *ngIf="isLoggedIn">
        Logged in as {{ roles }}.
      </div> </div> </div>
```

11-Compléter le fichier : home.*component.ts* **avec le code suivant:**

```
import { Component, OnInit } from '@angular/core';
import { HttpClientService } from '../services/http-client.service';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent implements OnInit {
  content?: string;

  constructor(private userService: HttpClientService) { }

  ngOnInit(): void {
    this.userService.getPublicContent().subscribe({
      next: data => {
        this.content = data;
      },
      error: err => {console.log(err)
        if (err.error) {
          this.content = JSON.parse(err.error).message;
        } else {
          this.content = "Error with status: " + err.status;
        }       }    });  } }
```

12-Compléter le fichier : home.*component.html* **avec le code suivant:**

```
<div class="container">
    <header class="jumbotron"> <p>{{ content }}</p>     </header> </div>
```

13-Compléter le fichier : *board-admin.component.ts* **avec le code suivant:**

```
import { Component, OnInit } from '@angular/core';
import { HttpClientService } from '../services/http-client.service';

@Component({
  selector: 'app-board-admin',
  templateUrl: './board-admin.component.html',
  styleUrls: ['./board-admin.component.css']
})
export class BoardAdminComponent implements OnInit {
  content?: string;

  constructor(private userService: HttpClientService) { }

  ngOnInit(): void {
    this.userService.getAdminBoard().subscribe({
      next: data => {
        this.content = data;
      },
      error: err => {console.log(err)
        if (err.error) {
          this.content = JSON.parse(err.error).message;
        } else {
          this.content = "Error with status: " + err.status;
        }
      }}); }}
```

14-Compléter le fichier : *board-admin.component.html* **avec le code suivant:**

```
<div class="container">
    <header class="jumbotron">
      <p>{{ content }}</p>
    </header>
  </div>
```

15-Configurer le routage pour notre application Angular dans **app-routing.module.ts** :

```
const routes: Routes = [
  { path: 'home', component: HomeComponent },
  { path: 'login', component: LoginComponent },
  { path: 'user', component: BoardUserComponent },
  { path: 'admin', component: BoardAdminComponent },
  { path: '', redirectTo: 'home', pathMatch: 'full' }];
```

16-Compléter le fichier : *app.component.ts* **avec le code suivant:**

```
import { Component } from '@angular/core';
import { TokenStorageService } from './services/token-storage.service';
import { AuthService } from './services/auth.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  private roles: string[] = [];
  isLoggedIn = false;
  showAdminBoard = false;
  showModeratorBoard = false;
  username?: string;

  constructor(private storageService: TokenStorageService, private authService:
AuthService) { }

  ngOnInit(): void {
    this.isLoggedIn = this.storageService.isLoggedIn();

    if (this.isLoggedIn) {
      const user = this.storageService.getUser();
      this.roles = user.roles;

      this.showAdminBoard = this.roles.includes('ROLE_ADMIN');

      this.username = user.username;
    }
  }

  logout(): void {
    this.authService.logout().subscribe({
      next: res => {
        console.log(res);
        this.storageService.clean();

        window.location.reload();
      },
      error: err => {
        console.log(err);
      }     });    } }
```

Explication : Tout d'abord, nous vérifions le statut **isLoggedIn** à l'aide de **StorageService**, s'il est vrai, nous obtenons les rôles de l'utilisateur et définissons la valeur pour le drapeau **showAdminBoard** & . Ils contrôleront la façon dont la barre de navigation du modèle affiche ses éléments.

Le modèle de composant **APP** comporte également un lien de bouton de **déconnexion** qui appelle la méthode **logout()** et recharge la fenêtre.

17-Compléter le fichier : *app.component.html* **avec le code suivant:**

```
<div id="app">
  <nav class="navbar navbar-expand navbar-dark bg-dark">
    <a href="#" class="navbar-brand">TP03: Auth & Roles</a>
    <ul class="navbar-nav mr-auto" routerLinkActive="active">
      <li class="nav-item">
        <a href="/home" class="nav-link" routerLink="home">Home </a>
      </li>
      <li class="nav-item" *ngIf="showAdminBoard">
        <a href="/admin" class="nav-link" routerLink="admin">Admin Board</a>
      </li>
      <li class="nav-item">
        <a href="/user" class="nav-link" *ngIf="isLoggedIn"
routerLink="user">User</a>
      </li>
    </ul>

    <ul class="navbar-nav ml-auto" *ngIf="!isLoggedIn">
      <li class="nav-item">
        <a href="/login" class="nav-link" routerLink="login">Login</a>
      </li>
    </ul>

    <ul class="navbar-nav ml-auto" *ngIf="isLoggedIn">
      <li class="nav-item">
        <a href class="nav-link" (click)="logout()">LogOut</a>
      </li>
    </ul>
  </nav>

  <div class="container">
    <router-outlet></router-outlet>
  </div>
</div>
```