

Formation Angular

TP N°10: Consommation des services web Rest fournis par la couche Backend sécurisée avec JWT et traitement des réponses au niveau des composants avec Angular

SOMMAIRE

1.	<u>Pré-requis :</u>	<u>3</u>
2.	<u>Objectifs</u>	<u>3</u>
3.	<u>Description des services web fournis par la couche Backend</u>	<u>3</u>
4.	<u>Développement de la couche front.....</u>	<u>4</u>
a.	Création de l'application.....	4
b.	Installation de bootstrap	4
c.	Installation de jquery	4
d.	Intégrer les feuilles de style et les scripts au niveau du fichier angular.json	4
e.	Démarrer ensuite l'application :.....	5
f.	Ajouter les modules : HttpClientModule, FormsModule et ReactiveFormsModule	5
g.	Créer les modèles (DTO) : User, Token, Emp	5
h.	Créer le service Auth	6
i.	Créer le service token-storage.....	6
j.	Créer le service emp.....	8
k.	Créer le composant navbar.....	9
l.	Créer le composant welcome	10
m.	Créer le composant emp-list.....	10
n.	Créer le composant emp-detail	12
o.	Créer le composant emp-create	12
p.	Créer le composant auth	14
q.	Créer les routes	16
r.	Créer l'intercepteur auth	17
s.	Modifier le module app.module.ts	18
t.	Copier l'image	19
5.	<u>Test.....</u>	<u>19</u>

1. Pré-requis :

- Réaliser les TPs 1-9 en premier.

2. Objectifs

- ✓ Communiquer avec une couche backend via Rest en utilisant le module HttpClientModule.
- ✓ Maîtriser comment traiter le token JWT côté Front.

3. Description des services web fournis par la couche Backend

- Les services web Rest que nous allons consommer ont été développés avec Spring Boot, Spring Security, JWT et H2.
- Le tableau ci-dessous décrit la liste des SW fournis par la couche Backend :

Service	Le lien	La requête	La réponse
Authentification par le compte admin1	http://localhost:9090/auth/signin Méthode :POST	{ "username":"admin1" , "password":"admin1" }	{ "username": "admin1", "jwttoken": "LE TOKEN ICI", "roles": ["ADMIN"] }
Authentification par le compte client1	http://localhost:9090/auth/signin Méthode :POST	{ "username":"client1", "password":"client1" }	{ "username": "admin1", "jwttoken": "LE TOKEN ICI", "roles": ["CLIENT"] }
Consulter les employés	http://localhost:9090/employees Méthode :GET	Authorization Bearer Token LE TOKEN ICI	Liste des employés : [{ "id": 452, "name": "emp1", "salary": 10000.0, "fonction": "Fonction1" }]
Consulter un employé par son id	http://localhost:9090/employees/7 Méthode :GET	Authorization Bearer Token LE TOKEN ICI	{ "id": 7, "name": "emp1", "salary": 10000.0, "fonction": "Fonction1" }
Ajouter un employé par le profile Admin	http://localhost:9090/admin/create Méthode :POST	Authorization Bearer Token LE TOKEN ICI { "name": "nouveau", "salary": 15000.0, "fonction": "ING"	employee is created successfully

		}	
Modifier un employé par le profile Admin	http://localhost:9090/admin/update/7 Méthode :PUT	Authorization Bearer Token LE TOKEN ICI { "name": "nouveau", "salary": 15000.0, "fonction": "ING" }	Employee is updated successsfully
Supprimer un employé par le profile Admin	http://localhost:9090/admin/delete/7 Méthode : DELETE	Authorization Bearer Token LE TOKEN ICI	Employee is deleted successsfully
Trier les employés par le salaire	http://localhost:9090/employees/sort/salary Méthode : GET	Authorization Bearer Token LE TOKEN ICI	La liste des employés triés par salaire.
Trier les employés par le nom	http://localhost:9090/employees/sort/name Méthode : GET	Authorization Bearer Token LE TOKEN ICI	La liste des employés triés par nom.
Trier les employés par la fonction	http://localhost:9090/employees/sort/function Méthode : GET	Authorization Bearer Token LE TOKEN ICI	La liste des employés triés par fonction.
Consulter les employés page par page (service de pagination)	http://localhost:9090/employees/pagination/0/2 Méthode : GET	Authorization Bearer Token LE TOKEN ICI	La liste des employés par page.

4. Développement de la couche front

a. Création de l'application

- Créer l'application tp10 avec la commande : **ng new tp10**

b. Installation de bootstrap

- Lancer la commande : **npm install bootstrap**

c. Installation de jquery

- Lancer la commande : **npm install jquery**

d. Intégrer les feuilles de style et les scripts au niveau du fichier angular.json

Au niveau du fichier angular.json, modifier les valeurs des clés « styles » et « scripts » comme suit :

```
"styles": [
  "src/styles.css",
  "../node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": [
```

```

        "./node_modules/jquery/dist/jquery.min.js",
        "./node_modules/bootstrap/dist/js/bootstrap.bundle.min.js",
        "./node_modules/bootstrap/dist/js/bootstrap.min.js"
    ]

```

e. Démarrer ensuite l'application :

```

➤ cd tp10
➤ ng serve.

```

f. Ajouter les modules : HttpClientModule, FormsModule et ReactiveFormsModule

- Ajouter les modules HttpClientModule, FormsModule et ReactiveFormsModule au niveau de la classe AppModule :

```

import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    ReactiveFormsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }

```

g. Créer les modèles (DTO) : User, Token, Emp

- Créer le modèle User : `ng g class model/User`

```

export class User {
    constructor(public username:string,
                public password:string) {}
}

```

- Créer le modèle Token : `ng g class model/Token`

```
export class Token {
  constructor(public username:string,
               public jwttoken:string,
               public roles:string[]) {}
}
```

- Créer le modèle Emp : `ng g class model/Emp`

```
export class Emp {
  constructor(public id:number,
               public name:string,
               public salary:string,
               public fonction:string) {}
}
```

h. Créer le service Auth

- Lancer la commande `ng g s services/auth`

```
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
const AUTH_API = 'http://localhost:9090/auth/signin';

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 , { username, password }, httpOptions);
  }
}
```

i. Créer le service token-storage

- Lancer la commande `ng g s services/token-storage`

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

```

import { Token } from '../model/token';

const TOKEN_KEY = 'auth-token';

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

  constructor() { }

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

  public saveToken(token: Token): void {
    window.sessionStorage.removeItem(TOKEN_KEY);
    window.sessionStorage.setItem(TOKEN_KEY, JSON.stringify(token));
  }

  public getToken(): Token | null {
    const token = window.sessionStorage.getItem(TOKEN_KEY);
    if (token) {
      return JSON.parse(token);
    }
    return null;
  }

  public getTokenValue(): string | null {
    const token=this.getToken();
    if (token) {
      return token.jwttoken;
    }
    return null;
  }

  public getRoles(): string[] | null {
    const token=this.getToken();
    if (token) {
      return token.roles;
    }
    return null;
  }

  public getUsername(): string | null {
    const token=this.getToken();
    if (token) {
      return token.username;
    }
  }
}

```

```

    }
    return null;
  }

  public hasRole(role:string): boolean | null {
    const token=this.getToken();
    if (token) {
      return token.roles.includes(role);
    }
    return null;
  }
}

```

j. Créer le service emp

- Lancer la commande **ng g service services/emp**
- Modifier la classe comme suit :

```

import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
const API_GET_EMPLOYEES = 'http://localhost:9090/employees/';
const API_ADMIN_EMPLOYEES = 'http://localhost:9090/admin/';
const httpOptions = {
  headers: new HttpHeaders({ 'Content-Type': 'application/json' })
};

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

  constructor(private http: HttpClient) { }

  getAll(): Observable<any> {
    return this.http.get(API_GET_EMPLOYEES);
  }

  getById(id:string): Observable<any> {
    return this.http.get(`${API_GET_EMPLOYEES}/${id}`);
  }

  create(data: any): Observable<any> {
    return this.http.post(API_ADMIN_EMPLOYEES+'create', data,{responseType:
'text' as 'json'});
  }

  update(id: number, data: any): Observable<string> {

```



```

        return this.http.put<string>(`${API_ADMIN_EMPLOYEES+'update'}/${id}`,
data,{responseType: 'text' as 'json'});
    }

    delete(id: number): Observable<string> {
        return this.http.delete<string>(`${API_ADMIN_EMPLOYEES+'delete'}/${id}`,
{responseType: 'text' as 'json'});
    }
}

```

k. Créer le composant navbar

- Lancer la commande **ng g c n avbar**
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { TokenStorageService } from '../services/token-storage.service';

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {

  isLoggedIn = false;
  isAdmin = false;
  isClient = false;
  username?: string;

  constructor(private tokenStorageService: TokenStorageService, private route:
ActivatedRoute, private router: Router) { }

  ngOnInit(): void {

    console.log("tokrn",this.tokenStorageService.getTokenValue());
    if (this.tokenStorageService.getTokenValue() != null)
      this.isLoggedIn=true;

    if (this.isLoggedIn) {
      const user = this.tokenStorageService.getUsername();
      this.isAdmin = <boolean>this.tokenStorageService.hasRole('ADMIN');
      this.isClient = <boolean>this.tokenStorageService.hasRole('CLIENT');
      this.username = <string>this.tokenStorageService.getUsername();
    }
  }

  logout(): void {

```

```

    this.tokenStorageService.signOut();
    this.router.navigate([{ outlets: { primary: 'login', contenu: null } }]);
  }
}

```

- Modifier le Template comme suit :

```

<nav class="navbar navbar-expand navbar-dark bg-dark">
  <a href="#" class="navbar-brand">CIGMA</a>
  <ul class="navbar-nav mr-auto" routerLinkActive="active">
    <li class="nav-item" *ngIf="isAdmin || isClient">
      <a class="nav-link" [routerLink]="['/',{ outlets: { primary:
['navbar'],contenu: ['employees'] } }]">Gestion des employés</a>
    </li>

    <li class="nav-item" *ngIf="isAdmin">
      <a class="nav-link" *ngIf="isLoggedIn" routerLink="user">Gestion des
utilisateurs</a>
    </li>
  </ul>

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

```

l. Créer le composant welcome

- Lancer la commande **ng g c welcome**

m. Créer le composant emp-list

- Lancer la commande **ng g c emp/emp-list**
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { Emp } from 'src/app/model/emp';
import { EmpService } from 'src/app/services/emp.service';
import { TokenStorageService } from 'src/app/services/token-storage.service';

@Component({
  selector: 'app-emp-list',
  templateUrl: './emp-list.component.html',

```

```

    styleUrls: ['./emp-list.component.css']
  })
  export class EmpListComponent implements OnInit {

    employees?: any;
    errorMessage?: string;

    currentEmployee: Emp = new Emp(0, '', 0, '');
    currentIndex = -1;
    name = '';
    isAdmin: boolean = false;

    constructor(private empService: EmpService, private tokenStorageService:
TokenStorageService) {

    }

    ngOnInit(): void {

      this.empService.getAll().subscribe(
        data => {
          this.employees = data;
          if (this.tokenStorageService.hasRole('ADMIN')) {
            this.isAdmin = true;
          }
        },
        err => {
          this.errorMessage = JSON.parse(err.error).message;
        }
      );
    }
  }
}

```

- Modifier le Template comme suit :

```

<br>
<div class="container">
  <table class="table">
    <thead class="thead-dark">
      <tr>
        <th scope="col">Id</th>
        <th scope="col">Nom</th>
        <th scope="col">Salaire</th>
        <th scope="col">Fonction</th>
      </tr>
    </thead>
    <tbody>
      <tr employee-detail *ngFor="let employee of employees"

```

```

[employee]='employee'>
  </tr>
  <tr>
    <a [routerLink]="['/',{ outlets: { primary: ['navbar'],contenu:
['create'] } }]" *ngIf="isAdmin">
      Nouveau employé
    </a>
  </tr>
</tbody>
</table>
</div>

```

n. Créer le composant emp-detail

- Lancer la commande **ng g c emp/emp-detail**
- Modifier la classe comme suit :

```

import { Component, Input, OnInit } from '@angular/core';

@Component({
  selector: '[employee-detail]',
  templateUrl: './emp-detail.component.html',
  styleUrls: ['./emp-detail.component.css']
})
export class EmpDetailComponent implements OnInit {
  @Input()
  employee:any;

  constructor() { }

  ngOnInit(): void {
  }
}

```

- Modifier le Template comme suit :

```

<th scope="row">{{employee.id}}</th>
<td>{{employee.name}}</td>
<td>{{employee.salary}}</td>
<td>{{employee.fonction}}</td>

```

o. Créer le composant emp-create

- Lancer la commande **ng g c emp/emp-create**
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';

```

```

import { ActivatedRoute, Router } from '@angular/router';
import { Emp } from 'src/app/model/emp';
import { EmpService } from 'src/app/services/emp.service';

@Component({
  selector: 'app-emp-create',
  templateUrl: './emp-create.component.html',
  styleUrls: ['./emp-create.component.css']
})
export class EmpCreateComponent implements OnInit {

  employee = new Emp(0, '', 0, '');
  submitted = false;
  message:string='';

  constructor(private empService: EmpService, private route: ActivatedRoute,
private router: Router) { }

  ngOnInit(): void { }

  createEmployee(): void {
    this.empService.create(this.employee)
      .subscribe(
        response => {
          this.submitted = true;
          this.router.navigate([{outlets: {primary: 'navbar' },contenu:
'employees'}}]);
        },
        error => {
          this.message=error.message;
          console.log(error);
        });
  }
}

```

- Modifier le Template comme suit :

```

<div *ngIf="submitted">
  <h3>Employé crée avec succès!</h3>
</div>
<div style="width: 500px; margin: auto;">
  <div class="submit-form">
    <div *ngIf="!submitted">

      <div class="form-group">
        <label for="name">Name</label>
        <input type="text" class="form-control" id="name" required
[(ngModel)]="employee.name" name="name"

```

```

        ngModel />
    </div>

    <div class="form-group">
        <label for="salary">Salaire</label>
        <input type="text" class="form-control" id="salary" required
[(ngModel)]="employee.salary" name="salary"
        ngModel />
    </div>

    <div class="form-group">
        <label for="fonction">Fonction</label>
        <input type="text" class="form-control" id="fonction" required
[(ngModel)]="employee.fonction"
        name="fonction" />
    </div>
    <button (click)="createEmployee()" class="btn btn-
success">Create</button>
</div>

    <div class="form-group">
        <div class="alert alert-danger" role="alert">
            Login failed: {{ message }}
        </div>
    </div>

</div>
</div>

```

p. Créer le composant auth

- Lancer la commande **ng g c auth**
- Modifier la classe comme suit :

```

import { Component, OnInit } from '@angular/core';
import { FormControl, FormGroup, Validators } from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { AuthService } from '../services/auth.service';
import { TokenStorageService } from '../services/token-storage.service';

@Component({
  selector: 'app-auth',
  templateUrl: './auth.component.html',
  styleUrls: ['./auth.component.css']
})
export class AuthComponent implements OnInit {

```

```

isLoggedIn = false;
isLoginFailed = false;
errorMessage = '';
userLoggedIn='';

constructor(private authService: AuthService, private tokenStorage:
TokenStorageService, private route: ActivatedRoute, private router: Router) { }

form = new FormGroup({
  username: new FormControl('', [Validators.required]),
  password: new FormControl('', [Validators.required])
});

get f() {
  return this.form.controls;
}

ngOnInit(): void {

}

submit() {
  if (this.form.status === 'VALID') {
    this.authService.login(this.form.controls['username'].value,
this.form.controls['password'].value).subscribe(
    data => {
      this.tokenStorage.saveToken(data);
      this.isLoginFailed = false;
      this.isLoggedIn = true;
      this.userLoggedIn=<string>this.tokenStorage.getUsername();
      this.router.navigate([{ outlets: { primary: 'navbar', contenu:
'welcome' } }]);
    },
    err => {
      this.errorMessage = err.error.message;
      this.isLoginFailed = true;
    }
  );
}
}
}

```

- Modifier le Template comme suit :

```

<div class="wrapper fadeInDown">
  <div id="formContent">

```

```

<!-- Tabs Titles -->

<!-- Icon -->
<div class="fadeIn first">
  
</div>

<div *ngIf="errorMessage" class="alert alert-danger" role="alert">
  {{errorMessage}}
</div>

<!-- Login Form -->
<form [formGroup]="form" (ngSubmit)="submit()">
  <input formControlName="username" id="username" type="text"
class="fadeIn second" placeholder="login">
  <div *ngIf="f['username'].touched && f['username'].invalid"
class="alert alert-danger">
    <div *ngIf="f['username'].errors != null &&
f['username'].errors['required']">Nom utilisateur obligatoire.
    </div>
  </div>
  <input formControlName="password" id="password"
type="text" class="fadeIn third" placeholder="password">
  <div *ngIf="f['password'].touched && f['password'].invalid"
class="alert alert-danger">
    <div *ngIf="f['password'].errors != null &&
f['password'].errors['required']">Mot de passe
    obligatoire.</div>
  </div>
  <input type="submit" class="fadeIn fourth" value="Log In">
</form>
</div>
</div>

```

q. Créer les routes

Modifier le fichier app-routing.module comme suit :

```

import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AuthComponent } from '../auth/auth.component';
import { EmpCreateComponent } from '../emp/emp-create/emp-create.component';
import { EmpListComponent } from '../emp/emp-list/emp-list.component';
import { NavbarComponent } from '../navbar/navbar.component';
import { WelcomeComponent } from '../welcome/welcome.component';

const routes: Routes = [
  { path: '', redirectTo: 'login', pathMatch: 'full' },

```



```

    { path: 'login', component: AuthComponent },
    { path: 'navbar', component: NavbarComponent },
    { path: 'employees', component: EmpListComponent, outlet: 'contenu' },
    { path: 'create', component: EmpCreateComponent, outlet: 'contenu' },
    { path: 'welcome', component: WelcomeComponent, outlet: 'contenu' },
    { path: 'logout', component: AuthComponent }
  ];

  @NgModule({
    imports: [RouterModule.forRoot(routes)],
    exports: [RouterModule]
  })
  export class AppRoutingModule { }

```

r. Créer l'intercepteur auth

- Lancer la commande suivante : **ng g interceptor interceptors/auth**
- Modifier la classe AuthInterceptor comme suit :

```

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

import { TokenStorageService } from '../services/token-storage.service';
import { Observable } from 'rxjs';

const TOKEN_HEADER_KEY = 'Authorization'; // for Spring Boot back-end

@Injectable()
export class AuthInterceptor implements HttpInterceptor {

  constructor(private token: TokenStorageService) {}

  intercept(req: HttpRequest<any>, next: HttpHandler):
  Observable<HttpEvent<any>> {
    let authReq = req;
    const token = this.token.getTokenValue();
    if (token != null) {
      authReq = req.clone({ headers: req.headers.set(TOKEN_HEADER_KEY,
      'Bearer ' + token) });
    }
    return next.handle(authReq);
  }
}

```

```
export const authInterceptorProviders = [
  {
    provide: HTTP_INTERCEPTORS,
    useClass: AuthInterceptor,
    multi: true
  }
];
```

s. Modifier le module app.module.ts

- Modifier la classe **AppModule** en ajoutant **authInterceptorProviders** au niveau du tableau **providers** comme suit :

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { FormsModule, ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AuthComponent } from './auth/auth.component';
import { NavbarComponent } from './navbar/navbar.component';
import { authInterceptorProviders } from './interceptors/auth.interceptor';
import { WelcomeComponent } from './welcome/welcome.component';
import { EmpListComponent } from './emp/emp-list/emp-list.component';
import { EmpDetailComponent } from './emp/emp-detail/emp-detail.component';
import { EmpCreateComponent } from './emp/emp-create/emp-create.component';

@NgModule({
  declarations: [
    AppComponent,
    AuthComponent,
    NavbarComponent,
    WelcomeComponent,
    EmpListComponent,
    EmpDetailComponent,
    EmpCreateComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    ReactiveFormsModule
  ],
  providers: [authInterceptorProviders],
```

```
bootstrap: [AppComponent]
})
export class AppModule { }
```

t. Copier l'image

- Créer le dossier images dans le dossier assets.
- Copier l'image login.jpg dans assets/images.

5. Test

- Lancer la couche backend : **java -jar backend.jar**
- Lancer la couche front : **ng serve**
- Accéder au lien : <http://localhost:4200>. Le formulaire suivant est affiché :



The image shows a login form with a white background and a light gray border. At the top, the word "LOGIN" is displayed in large, green, 3D block letters. Below this, there are two input fields: the first is labeled "login" and the second is labeled "password". Both fields have a light gray background and a thin border. Below the input fields is a blue button with the text "LOG IN" in white capital letters.

- Entrer le compte admin1/admin1 et cliquer sur le bouton LOGIN IN. La page suivante est affichée :

```
CIGMA  Gestion des employés  Gestion des utilisateurs  admin1  LogOut
welcome works!
```

- Pour consulter les employés, cliquer sur le menu « Gestion des employés », la page suivante est affichée :

Id	Nom	Salaire	Fonction
7	emp1	10000	Fonction1
8	emp2	20000	Fonction3
9	emp3	30000	Fonction4
10	emp4	40000	Fonction5
11	emp5	50000	Fonction6

[Nouveau employé](#)

- Pour ajouter un nouvel employé, cliquer sur « Nouveau employée ». La page suivante est affichée :

Name

Salaire

Fonction

Create

- Entrer les données du nouvel employé et cliquer ensuite sur le bouton Create :

Name

Salaire

Fonction

Create

La fenêtre suivante est affichée :

Id	Nom	Salaire	Fonction
7	emp1	10000	Fonction1
8	emp2	20000	Fonction3
9	emp3	30000	Fonction4
10	emp4	40000	Fonction5
11	emp5	50000	Fonction6
12	jjjj	12000	mmm
13	nouveau	12000	mmm
14	Foulane	11000	Ingénieur

[Nouveau employé](#)

- Entrer le compte client1/client1 et cliquer sur le bouton LOGIN IN. La page suivante est affichée :

CIGMA Gestion des employés client1 LogOut

welcome works!

- Cliquer sur le menu « Gestion des employés » pour consulter les employés.
- Cliquer sur le menu « LogOut » pour se déconnecter.

Questions :

- Implémenter les deux use cas suivantes : Suppression d'un employé et modification d'un employé (actions à faire uniquement par les administrateurs).
- Traiter les messages d'erreurs au cas où les opérations (ajout/suppression/modification) génèrent des erreurs.

Vous venez de développer une application communiquant avec une couche backend moyennant Rest en utilisant JWT !

Vous pouvez passer au TP n° 11.

Fin du TP 10.