esprit Se former autrement	Année Universitaire : 2020-2021 **Application côté client**

Workshop n°5 :

HttpClient avec Json-Server

Objectifs :

- Installer json-server
- Manipuler les méthodes de HttpClient : GET, POST, PUT, DELETE

Travail demandé :

Partie A : Installation de json server

1- Installez json-server en tapant la commande : npm i --global json-server

2- Au niveau du terminal du projet courant, tapez la commande : `json-server --watch db.json`

3- Un fichier db.json est généré automatiquement sous votre projet. Ce fichier contient déjà des données par défaut. Mettez à jour ce fichier en ajoutant le contenu suivant : (la liste des produits qu'on a utilisé dans les anciens workshops)

```
{
"products": [
{"id": 1, "title": "Tshirt 1", "price": 18, "quantity": 0, "like": 0},
{"id": 2, "title": "Tshirt 2", "price": 21, "quantity": 10, "like": 0},
{"id": 3, "title": "Tshirt 3", "price": 16, "quantity": 8, "like": 0}
],
  "posts": [….],
…..
}
```

4- Testez maintenant l'url **localhost:3000/products**, vous devez avoir la liste des produits contenu dans le fichier db.json

Partie B : Récupération des données

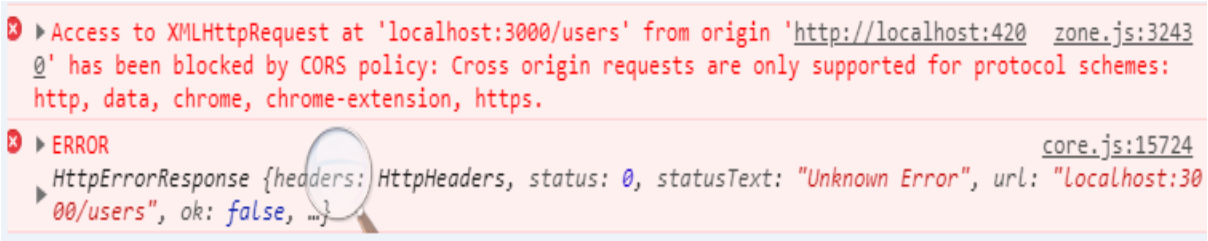
5- Créez le service **ProductService** si vous ne l'avez pas déjà créé et **injectez** dedans le service HttpClient

6- Définissez dans ce service une propriété appelé productsUrl comme suit :

```
productsUrl:string='localhost:3000/products'
```

7- Créez la méthode qui permet de retourner la liste des produits à partir de l'url mentionnée dans productsUrl. Il faut utiliser la méthode « get() »
8- Injectez le service « ProductService » au niveau du composant ProductsComponent que vous avez déjà manipulé dans les anciens workshops et récupérez le contenu de la propriété listProducts depuis ce service. L'affichage est normalement fait grâce aux anciens workshops. Sinon faites le nécessaire pour afficher la liste des produits.

RQ : si l'erreur suivante survient, veuillez voir la fin de ce workshop « Configuration de proxy » pour résoudre ce problème.



C- Suppression d'un produit.

9- Créez la méthode deleteProduct() au niveau du service ProductsService en utilisant la méthode delete() du service HttpClient comme suit :

```
deleteProduct (product: Product | number): Observable<Product> {
 const id = typeof product === 'number' ? product : product.id;
 const url=this.productsUrl+'/'+id;
 return this.http.delete<Product>(url);
 }
```

10- Devant chaque produit affiché, ajoutez un bouton « supprimer ». En cliquant dessus le produit correspondant est supprimé. Ajoutez dans le composant ProductsComponent la méthode suivante à exécuter en cliquant sur le bouton supprimer.

```
deleteProduct(id:number){
 this.productservice.deleteProduct(id).subscribe(); }
```

Sachant que :

```
constructor(private productservice:ProductService) { }
```

11- Consultez le fichier db.json

D- Ajout d'un produit

12- Au niveau du service ProductService, importez HttpHeaders from @angular/common/http.

13- Créez la méthode addProduct() dans le service ProductService en utilisant la méthode post du service HttpClient.

```
addProduct (product: Product): Observable<Product> {
 return this.http.post<Product>(this.productsUrl, product, this.httpOpti
ons);}
```

Sachant que:

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

14- Utilisez l'un des formulaires que vous avez déjà créé dans les workshops de formulaire (que ce soit template-driven form ou bien Reactive-form). Sinon vous créez un nouveau composant dans lequel vous définissez le formulaire d'ajout d'un produit.

15- En submittant le formulaire, la méthode save() est appelée et qui appelle à son tour la méthode addProduct() du service ProductService. Une redirection se fait vers la liste des produits une fois l'ajout est fait. (path = home)

```
save(){
this.product.like=0;
this.ps.addProduct(this.product).subscribe (res => {
      console.log('Product created!');
      this.router.navigateByUrl('/home');})
;
}
```

Sachant que :

```
constructor(private ps:ProductService, private router: Router) { }
```

E- Modification d'un produit

16- Ajouter un bouton « Modifier » devant chaque produit. En cliquant sur ce bouton un composant contenant un formulaire affichant les informations du produit sélectionné est affiché. La valeur de l'id est envoyée dans l'url et le champ id doit être inactif dans le formulaire.

17- Au niveau du composant contenant le formulaire à créer si vous ne l'avez pas, récupérez la valeur de l'id envoyé dans l'url et récupérez le produit correspondant

grâce à la méthode getProductById() que vous devez la créez dans le service ProductService.

- Au niveau du service : ProductService

```
getProductById(id: number): Observable<Product> {
return this.http.get<Product>(this.productsUrl +'/'+ id);  }
```

- Au niveau du composant contenant le formulaire de modification

```
ngOnInit (){
this.ac.paramMap.subscribe(next=>this.ps.getProductById(Number(next.get(
'id'))).subscribe(res=>{this.product=res}), error=>console.log(error));
}
```

18- Ajoutez la méthode updateProduct() au niveau du service ProductService

```
updateProduct (id: number, product: Product): Observable<Product> {
return this.http.put<Product>(this.productsUrl+'/'+ id, product, this.ht
tpOptions);
 }
```

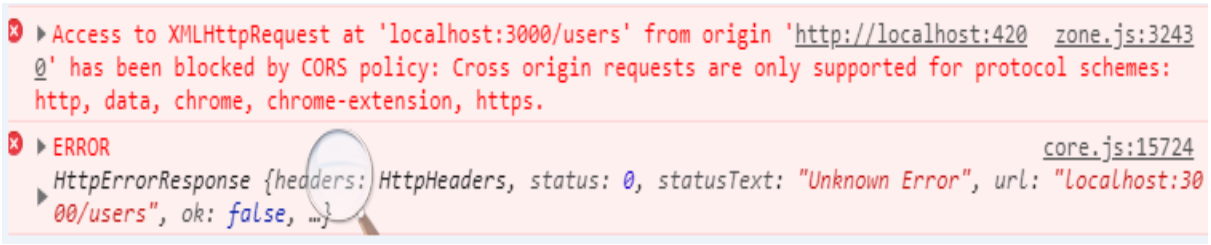
19- En cliquant sur le bouton « save » du formulaire, les informations du produit en question sont mises à jour sauf l'id bien sûr grâce à la méthode updateProduct()

```
update(){
this.ps.updateProduct(this.product.id,this.product).subscribe();}
```

20- Vous pouvez utilisez la méthode getProductById() du service ProductService() pour récupérer les détails d'un produit et les affichez dans le composant DetailsProduct du workshop n°4

Configuration de proxy

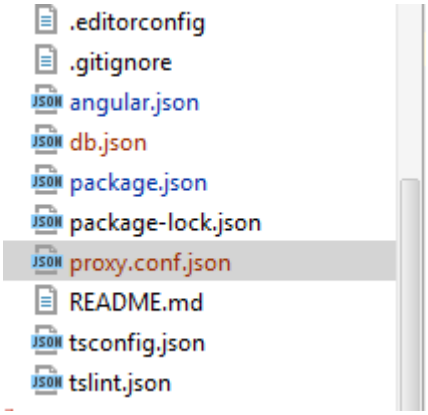
Problème :



Solution :

Cette solution est pour résoudre le problème de CORS. Bien sûr les url et les exemples mentionnés changent en fonction de votre besoin et le serveur que vous utilisez.

1- Créez le fichier **proxy.conf.json** sous votre projet en dehors du src.



2- Mettez dedans la configuration suivante :

```
{
  "/api/*": {
    "target": "http://localhost:3000",
    "secure": false,
    "logLevel": "debug",
    "changeOrigin": true,
    "pathRewrite": {"^/api" : ""}
  }
}
```

NB: la configuration ci dessus est pour la résolution de l'erreur rencontrée lors de l'utilisation du serveur "json-server". Cette configuration change en fonction de l'adresse de votre api.

3- Au niveau du fichier angular.json, ajoutez la ligne suivante :

```
"serve": {
  "builder": "@angular-devkit/build-angular:dev-server",
  "options": {
    "browserTarget": "app-twin1:build",
    "proxyConfig":"proxy.conf.json"
  },
  "configurations": {
    "production": {
      "browserTarget": "app-twin1:build:production"
    }
  }
},
"extract-i18n": {
  "builder": "@angular-devkit/build-angular:extract-i18n",
```

4- Au niveau de votre service, l'url ne doit plus pointer directement sur l'url (exemple : "localhost :3000/products"), mais elle doit pointer sur le préfixe choisi au niveau du fichier proxy.conf.json, ici c'est **/api** suivi par le path souhaité.
Exemple: L'url devient: **productsUrl:string="/api/products"**;

```
productsUrl : string = "/api/products";
getProducts(): Observable<Product[]>{
    return this.http.get<Product[]>(this.productsUrl);
  }
```

5- Redémarrez votre application.