

Workshop n°9 :

Le service HttpClient

Objectifs :

- Créer un service
- Injecter un service dans un composant
- Manipuler le service HttpClient
- Consommer des API REST

Travail demandé :

- 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 un nouveau attribut «**users**» comme suit:

```
"users":[

  {
    "idCustomer": 1,
    "firstName": "Mila",
    "lastName": "Kunis",
    "birthDate": "1999-06-30",
    "accountCategory": "Admin",
    "email": "mila@kunis.com",
    "password": "test",
    "picture":
    "https://bootdey.com/img/Content/avatar/avatar3.png",
    "profession": "Software Engineer"
  },
  {
    "idCustomer": 2,
    "firstName": "George",
    "lastName": "Clooney",
    "birthDate": "1999-06-30",
    "accountCategory": "Customer",
```

```

        "email": "marlon@brando.com",
        "password": "test",
        "picture":
"https://bootdey.com/img/Content/avatar/avatar2.png",
        "profession": "Software Engineer"
    },
    {
        "idCustomer": 3,
        "firstName": "George",
        "lastName": "Clooney",
        "birthDate": "1999-06-30",
        "accountCategory": "Customer",
        "email": "marlon@brando.com",
        "password": "test",
        "picture":
"https://bootdey.com/img/Content/avatar/avatar1.png",
        "profession": "Software Engineer"
    },
    {
        "idCustomer": 4,
        "firstName": "Ryan",
        "lastName": "Gossling",
        "birthDate": "1999-06-30",
        "accountCategory": "Golden",
        "email": "Ryan@nicholson.com",
        "password": "test",
        "picture":
"https://bootdey.com/img/Content/avatar/avatar4.png",
        "profession": "Software Engineer"
    },
    {
        "idCustomer": 5,
        "firstName": "Robert",
        "lastName": "Downey",
        "birthDate": "1999-06-30",
        "accountCategory": "Blocked Account",
        "email": "robert@nicholson.com",
        "password": "test",
        "picture":
"https://bootdey.com/img/Content/avatar/avatar5.png",
        "profession": "Software Engineer"
    }
]

```

4- Créer un service appelé **UserService**.

Récupération de la liste des utilisateurs

- 5- Dans ce service, définir une méthode `getAllUsers()` qui retourne la liste des utilisateurs contenu dans le fichier `db.json`.
- 6- Au niveau du composant `ListUserComponent`, injecter le service `UserService` et alimenter la propriété qui contient la liste des utilisateurs via la méthode `getAllUsers()`

Suppression d'un utilisateur

- 7- Créez la méthode `deleteUser()` au niveau du service `UserService` en utilisant la méthode `delete()` du service `HttpClient` comme suit :

```
deleteUser (user: User): Observable<User> {  
  const url=this.userUrl+'/' + user.id;  
  return this.http.delete<User>(url);  
}
```

- 8- Devant chaque utilisateur affiché, ajoutez un bouton « supprimer ». En cliquant dessus l'utilisateur correspondant est supprimé. Ajoutez dans le composant `listUserComponent` la méthode suivante à exécuter en cliquant sur le bouton supprimer.

```
deleteUser(u:User){  
  this.userservice.deleteUser(u).subscribe(); }  

```

- 9- Consultez le fichier `db.json`

D- Ajout d'un user

- 10- Au niveau du service `UserService`, importez `HttpHeaders` from `@angular/common/http`.
- 11- Créez la méthode `addUser()` dans le service `UserService` en utilisant la méthode `post` du service `HttpClient`.

```
addUser(user: User): Observable<User> {  
  return this.http.post<User>(this.userUrl,user, this.httpOptions);}
```

Sachant que:

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

- 12- 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 utilisateur.
- 13- En submittant le formulaire, la méthode `save()` est appelée et qui appelle à son tour la méthode `addUser()` du service `UserService`. Une redirection se fait vers la liste des utilisateurs une fois l'ajout est fait.

E- Modification d'un produit

- 14- 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.
- 15- 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 `getUserById()` que vous devez la créer dans le service `UserService`.
- Au niveau du service `UserService`

```
getUserById(id: number): Observable<User> {  
  return this.http.get<User>(this.userUrl + '/' + id); }
```

- Au niveau du composant contenant le formulaire de modification

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

- 16- Ajoutez la méthode `updateUser()` au niveau du service `UserService`

```
updateUser(id: number, user:User): Observable<User> {  
  return this.http.put<User>(this.userUrl + '/' + id, user, this.httpOptions);  
}
```

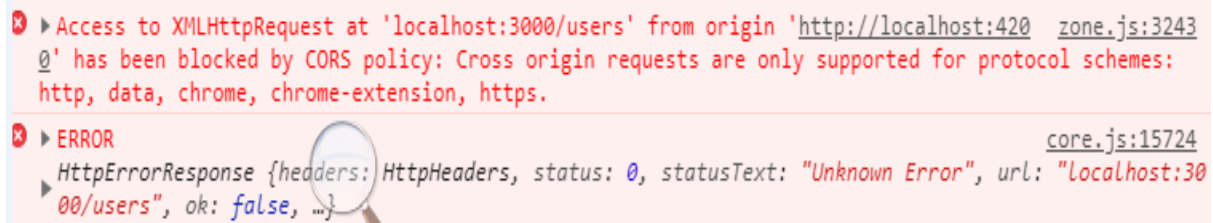
- 17- En cliquant sur le bouton « save » du formulaire, les informations de l'utilisateur en question sont mises à jour sauf l'id bien sûr grâce à la méthode `updateUser()`

```
update(){  
  this.us.updateUser(this.user.id, this.user).subscribe();}
```

- 18- Vous pouvez utiliser la méthode `getUserById()` du service `UserService()` pour récupérer les détails d'un utilisateur et les affichez dans le composant `DetailsUserComponent`.

Configuration de proxy

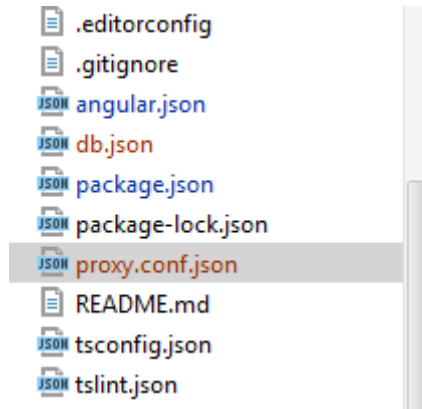
Problème :



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": {
    "proxyConfig": "proxy.conf.json"
  },
  "configurations": {
    "production": {
      "browserTarget": "app4TWIN2:build:production"
    },
    "development": {
      "browserTarget": "app4TWIN2:build:development"
    }
  },
  "defaultConfiguration": "development"
},
"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/users"), 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: **userUrl:string="/api/users"**;

```
userUrl : string = "/api/users";
getUsers(): Observable<User[]>{
    return this.http.get<User[]>(this.userUrl);
}
```

5- Redémarrez votre application.