# TP à rendre : Programmation Front-End (Angular)

**Echéance** : **22 décembre 2023**

L'objectif de ce TP est de développer une application Angular afin de revoir les différentes notions vues en cours résumées succinctement par les points suivants:

- Les principes de développement en Angular
- Les interactions avec l'utilisateur au sein des composants
- Les interactions au sein d'une hiérarchie de composants
- Les directives *ngIf et *ngFor
- Les services, notamment *Http*
- Le routing

**Livrables attendus**: lien vers votre page mise en production sur le serveur IUT (public_html) ou sur le Gitlab de la forge de l'université (dernier recours sur github).
**Lien vers votre dépôt (repository) sur Gitlab**. Accès aux enseignants en rôle « Reporter »

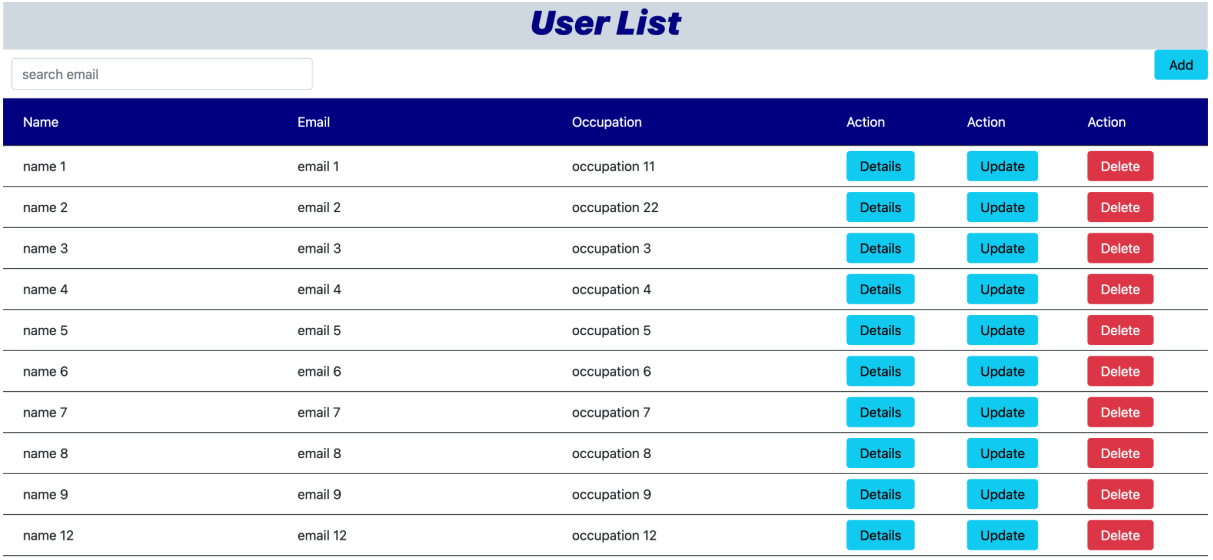
**Le projet doit être développé en monôme et il vous prépare au sujet du DS. Pénalité en cas de retard.**

**Sujet**. L'application permet de gérer et manipuler des comptes d'utilisateurs. Il est possible d'ajouter de nouveaux comptes utilisateurs, de mettre à jour ou supprimer un. Un composant angular sera créé pour chacune de ces fonctionnalités.

1. **Affichage de la liste des utilisateurs dans une table**

La page d'accueil de l'application permet à la fois de lister tous les comptes utilisateurs et de les filtrer par l'email. Vous reproduisez l'aspect de la capture d'écran ci-dessous pour la vue HTML du composant `user-list`. Vous utiliserez une `mat-table` (cf. l'URL https://v5.material.angular.io/components/table/overview) d'Angular Material. Rappelez vous que chaque fois que vous utilisez des composants d'Angular Material, il faut importer des modules dans `app.module.ts`. Pour déterminer ce

qu'il faut importer, cliquez, tout en haut de la page d'Angular Material sur l'onglet API. Vous verrez alors l'import à faire.



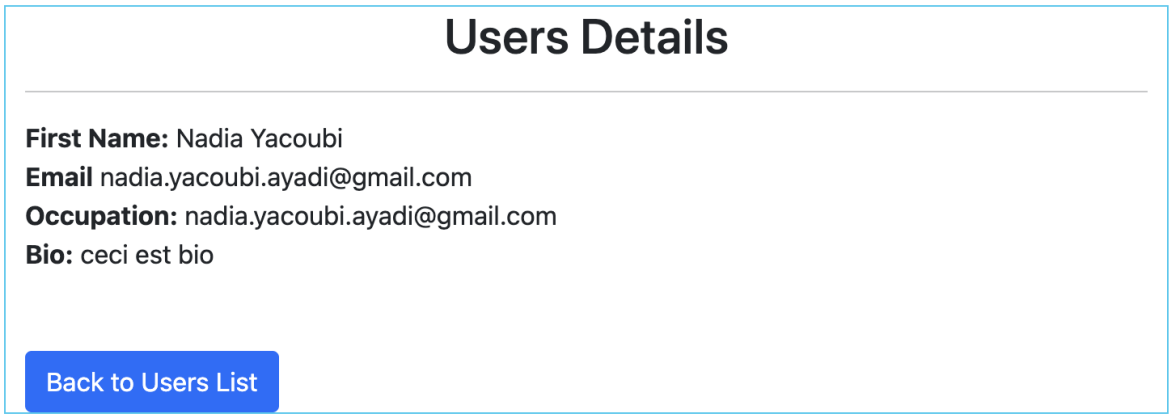

En première version, stocker en "*dur*" un tableau d'objets *user* dans le composant `user-list`. Les informations dont on dispose d'un utilisateur sont l'identifiant (id), le nom, l'email, l'occupation et une bio tel que décrit par la classe typescript suivante:

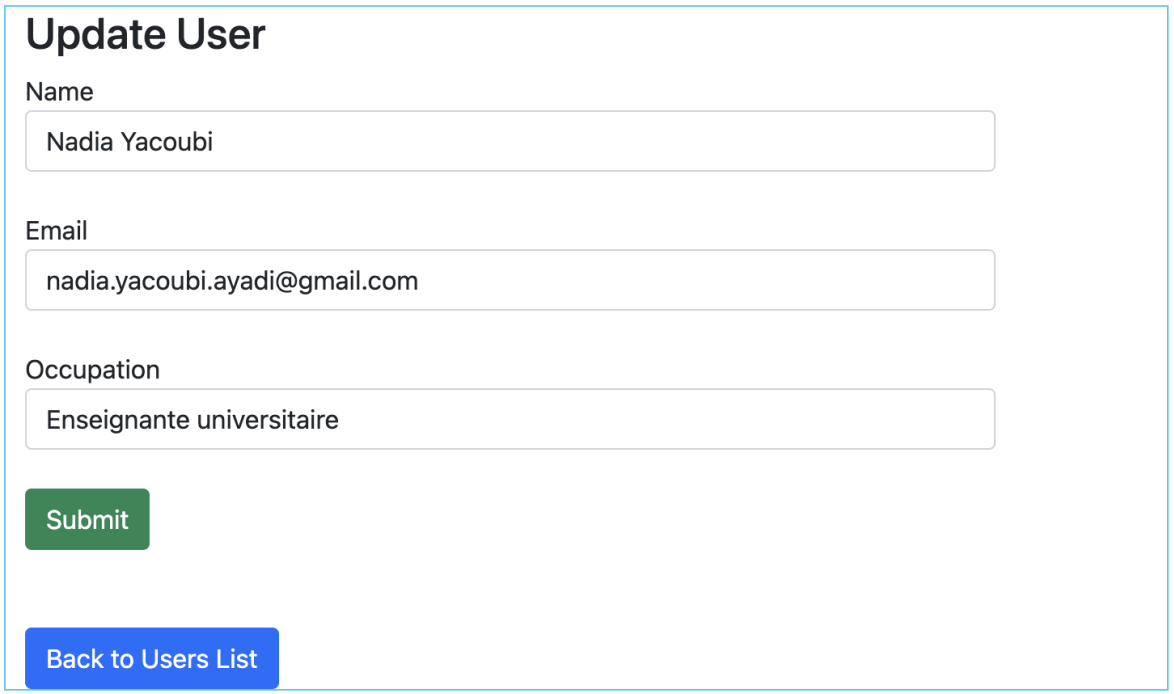
```
export class User {
   id!: number;
   name!: String;
   occupation!: String;
   email!: String;
   bio!: String;
}
```

Pour générer des données, vous pourrez utiliser par exemple le site https://www.generatedata.com/ ou mieux https://mockaroo.com/. Vous pouvez générer un tableau JSON d'utilisateurs que vous pouvez mettre dans un fichier `userData.js` dans votre projet.

2. **Implémentez le différents boutons de la table user-list**

Comme leur nom l'indique, les boutons "*Details*", "*update*" et "*Delete*" permettent respectivement d'afficher le détail d'un utilisateur, de mettre à jour les infos utilisateur et de le supprimer.




A ce niveau, définissez les routes de votre application en fonction des composants que vous allez créer de sorte qu'on puisse naviguer dynamiquement à travers les différentes vues. Vous vous baserez sur le tableau de routes suivant :

```
const routes: Routes = [
 { path: '', pathMatch: 'full', redirectTo: 'users' },
 { path: 'users', component: UserListComponent },
 { path: 'user/:id', component: UserDetailsComponent },
 { path: 'update/:id', component: UpdateUserComponent },
 { path: 'add', component: AddUserComponent },
];
```

### 3. Création d'un service

Créez un service `user.service.ts` qui sera utilisé pour abstraire la transmission des requêtes http à un backend. Pour cela, il faut commencer par passer à son constructeur une instance *http* de *HttpClient* par *dependency injection*, comme vu en cours. C'est cette instance qui enverra réellement les requêtes au backend. Pourquoi, dans ce cas, ne pas utiliser directement HttpClient dans toutes les pages de votre application pour transmettre les requêtes ? Pour le besoin de test, utiliser une API REST telle que MockAPI ou autre.

### 4. Implémentez le composant add-user

Pour finir, implémentez le composant `add-user` pour ajouter un nouvel utilisateur à la liste.

## Partie BONUS

Le projet se prête à beaucoup d'améliorations :

1. Faire une pagination avec un réglage du nombre d'utilisateurs par page, etc.
2. Filtrer selon plusieurs critères pas seulement par l'email.
3. Ajouter une gestion de login/password