Documentation du projet
Angular CRUD

(Adem Gattoussi)

1. Introduction :

- Titre du projet : **CRUD App**
- Description générale :
  - Ce projet est une application Angular CRUD permettant de gérer différentes entités, notamment les salles, classes, sessions, étudiants, sujets, et enseignants. Chaque entité peut être créée, consultée, modifiée, et supprimée via une interface utilisateur intuitive.
- Objectif :
  - Apprentissage des concepts Angular, comme les composants, services, et routage.
  - Mise en œuvre des appels à une API REST pour la manipulation des données.
  - Création d'une interface utilisateur avec Angular Material pour offrir une expérience utilisateur cohérente et esthétique.

2. Installation et configuration :

- Prérequis :
  - **Node.js** : Version 18 ou supérieure.
  - **npm** : Version 9 ou supérieure.
  - **Angular CLI** : Installé via `npm install -g @angular/cli`.
  - **Serveur JSON** : Utilisé pour simuler une base de données, installé avec `npm install -g json-server`.
- Étapes d'installation :
  1. `git clone https://github.com/ademgattoussi/angular-CRUD.git`
  2. `cd angular-CRUD`

3. `npm install`
4. `ng serve`
5. `json-server --watch timetable.json`
6. Accéder à l'application dans un navigateur: “http://localhost:4200”

3. Structure des fichiers du projet

- Arborescence principale :
  - src/
  - ├── app/
  - │ ├── components/
  - │ ├── services/
  - │ ├── styles/
  - │ ├── app.component.css
  - │ ├── app.component.html
  - │ ├── app.component.ts
  - │ ├── app.config.ts
  - │ ├── app.routes.ts
  - timetable.json
- Description des fichiers :
  - **`components/`** : Contient tous les composants Angular pour les différentes fonctionnalités (CRUD pour les entités comme `rooms`, `classes`, etc.).
  - **`services/`** : Regroupe les services utilisés pour effectuer les appels API REST (par exemple, `api.service.ts`).
  - **`styles/`** : Fichiers CSS globaux pour styliser l'application.
  - **`app.component.css`** : Styles spécifiques au composant principal.
  - **`app.component.html`** : Modèle HTML pour le composant principal.
  - **`app.component.ts`** : Fichier TypeScript du composant principal contenant la logique associée.

    - `app.config.ts` : Configuration générale de l'application (par exemple, URL de l'API).
    - `app.routes.ts` : Définition des routes Angular pour naviguer entre les différentes pages de l'application.

4. Composants Angular :

- **Localisation des composants :**
    - Tous les composants sont situés dans le dossier `src/app/components/`.
- **Exemple :**
    - `rooms/` : Composants pour gérer les salles (`rooms-create`, `rooms-show`, etc.).
    - `classes/` : Composants pour gérer les classes.

Chaque composant suit une structure classique Angular :

- **HTML** : Modèle pour afficher les données.
- **CSS** : Styles spécifiques.
- **TS** : Logique et gestion des événements.

5. Services:

- **Localisation des services :** Tous les services sont situés dans le dossier `src/app/services/`.
- **api.service.ts :** Service central pour les appels à l'API REST.
- **Objectif :**
    - Le principal objectif du service `ApiService` est de centraliser toutes les requêtes HTTP afin que le reste de l'application puisse y accéder facilement et de manière uniforme. Ce service permet de récupérer des données, d'envoyer des nouvelles données, de modifier ou de supprimer des entités dans la base de données via des appels API REST.
- **Structure du service :**
    - Le service utilise `HttpClient` d'Angular pour effectuer des requêtes HTTP vers un serveur ou une API. L'API peut être une API REST qui fournit des points de terminaison pour gérer les données.

6. Routage :

- **Fichier :** `src/app/app.routes.ts`
- **Code :**

```
export const routes: Routes = [
  { path: '', redirectTo: '/rooms', pathMatch: 'full' },
  { path: 'rooms', component: RoomsShowComponent },
  { path: 'rooms-create', component: RoomsCreateComponent },
  { path: 'rooms-update/:id', component: RoomsUpdateComponent },
  { path: 'classes', component: ClassesShowComponent },
  { path: 'classes-create', component: ClassesCreateComponent },
  { path: 'classes-update/:id', component: ClassesUpdateComponent },
  { path: 'sessions', component: SessionsShowComponent },
  { path: 'sessions-create', component: SessionsCreateComponent },
  { path: 'sessions-update/:id', component: SessionsUpdateComponent },
  { path: 'students', component: StudentsShowComponent },
  { path: 'students-create', component: StudentsCreateComponent },
  { path: 'students-update/:id', component: StudentsUpdateComponent },
  { path: 'subjects', component: SubjectsShowComponent },
  { path: 'subjects-create', component: SubjectsCreateComponent },
  { path: 'subjects-update/:id', component: SubjectsUpdateComponent },
  { path: 'teachers', component: TeachersShowComponent },
  { path: 'teachers-create', component: TeachersCreateComponent },
  { path: 'teachers-update/:id', component: TeachersUpdateComponent },
];
```

7. Exécution :

- **Étapes :**
  1. **Lancer le projet Angular :** `ng serve`
  2. **Lancer le backend (JSON Server)**

     ```
     json-server --watch db.json
     ```
  3. **Accéder à l'application :** http://localhost:4200

8. Conclusion :

Ce projet Angular CRUD est conçu pour gérer les entités avec une interface utilisateur moderne et des fonctionnalités complètes. Grâce à Angular Material et JSON Server, il offre une base solide pour des projets similaires.