Création d'une Application CRUD avec MEAN

Étape 3 : Mise en place du Frontend (Angular 17 ou supérieur)

Installer Angular Cli

```
npm install -g @angular/cli
```

3.1 Initialiser Angular

Dans un dossier parent, crée le projet Angular :

```
ng new frontend
cd frontend
```

3.2 Créer le service Angular pour appeler l'API

Dans Angular, crée un service pour gérer les requêtes HTTP :

```
ng generate service services/article
```

Dans src/app/services/article.service.ts :

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class ArticleService {
  private apiUrl = 'http://localhost:5000/articles'; //vous pouvez
modifier le port
```

```
  constructor(private http: HttpClient) {}

  getArticles(): Observable<any> {
    return this.http.get(this.apiUrl);
  }

  addArticle(article: any): Observable<any> {
    return this.http.post(this.apiUrl, article);
  }

  updateArticle(id: string, article: any): Observable<any> {
    return this.http.put(`${this.apiUrl}/${id}`, article);
  }

  deleteArticle(id: string): Observable<any> {
    return this.http.delete(`${this.apiUrl}/${id}`);
  }
}
```

Dans app.config.ts :

Rajouter

```
import { provideHttpClient } from '@angular/common/http';
```

Et modifier le contenu du provider en rajoutant provideHttpClient() à la fin.

```
export const appConfig: ApplicationConfig = {
  providers: [provideZoneChangeDetection({ eventCoalescing: true }),
provideRouter(routes),provideHttpClient()]
};
```

3.3 Créer un composant pour afficher les articles

```
ng generate component components/article-list
```

Dans src/app/components/article-list/article-list.component.ts :

Modifier les importations

```
import { Component, OnInit } from '@angular/core';
import { ArticleService } from '../../services/article.service';
```

Modifier le contenu dans export class

```
export class ArticleListComponent implements OnInit {
  articles: any[] = [];

  constructor(private articleService: ArticleService) {}

  ngOnInit(): void {
    this.loadArticles();
  }

  loadArticles(): void {
    this.articleService.getArticles().subscribe(data => this.articles =
data);
  }

  deleteArticle(id: string): void {
    this.articleService.deleteArticle(id).subscribe(() =>
this.loadArticles());
  }
}
```

Dans src/app/components/article-list/article-list.component.html :

```
<div>
  @for (article of articles; track article._id) {
    <div>
      <h2>{{ article.title }}</h2>
      <p>{{ article.content }}</p>
      <button (click)="deleteArticle(article._id)">Supprimer</button>
    </div>
  }
</div>
```

Ajoute dans app.component.html.

3.4 Modifier le routage

Dans le fichier app.route.ts :

```
import { Routes } from '@angular/router';
import { ArticleListComponent } from './components/article-list/article-
list.component';

export const routes: Routes = [
  { path: 'articles', component: ArticleListComponent },  // Route pour
article-list
  { path: '', redirectTo: 'articles', pathMatch: 'full' }  // Redirection
```

```
par défaut
];
```

3.5 Nettoyer le fichier app.component.html

Supprimer le contenu du fichier **app.component.html** et laisser seulement ce code

```
<router-outlet />
```

Angular utilise **router-outlet** pour afficher le composant à afficher. Ceci a été défini dans les routes.

```
  { path: 'articles', component: ArticleListComponent },  // Route pour
article-list
```

Dans notre cas, le composant **ArticleListComponent** va s'afficher lorsqu'on va afficher l'url **/articles**

3.6 Lancer l'application Angular

```
ng serve
```

Vous devez voir le contenu de la base mongodb. Rajoutez via postman si besoin.

3.7 Utilisation des fichiers d'environnements.

Pour centraliser et rendre plus maintenable la configuration de votre URL d'API (http://localhost:5000), il est recommandé de la placer dans un fichier d'environnement dédié. Cela vous permet de modifier facilement les URLs pour différents environnements (développement, production, test) sans toucher au code source de vos services.

1. Utiliser les fichiers d'environnement d'Angular

Angular fournit des fichiers d'environnement par défaut dans le dossier src/environments :

- environment.ts : Pour l'environnement de développement.
- environment.prod.ts : Pour l'environnement de production.

📝 Dans src/environments/environment.ts

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:5000'  // URL pour le développement
};
```

📝 Dans src/environments/environment.prod.ts

```
export const environment = {
  production: true,
  apiUrl: 'https://api.monsite.com'  // URL pour la production
};
```

Modifier le fichier **article.service.ts** pour utiliser le fichier de config environment.
Rajouter l'import

```
import { environment } from '../../environments/environment';  // Import de l'environnement
```

Et modifier le code suivant

```
  private apiUrl = `${environment.apiUrl}/articles`;  // Utilisation de la variable d'environnement
```

🔔 Bonus : Changer d'environnement au build

Quand vous compilez votre projet :

- Pour le développement : ng serve → utilisera environment.ts.
- Pour la production : ng build --prod → utilisera automatiquement environment.prod.ts.