TP Angular Création d'une application de gestion de tâches

Objectif

- Initialiser un projet Angular
- Comprendre la structure d'un projet Angular
- Créer des composants
- Utiliser le data binding
- Manipuler des listes avec `*ngFor` et `*ngIf`
- Gérer des formulaires
- Mettre en place un service et l'injection de dépendances
- Simuler une API avec un service

Prérequis

- Bases de **TypeScript**
- Bases de **HTML / CSS**
- Node.js (≥ 20)
- Angular CLI installé :

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

Contexte du TP

On développe une application **Task Manager** permettant de :

- Ajouter une tâche
- Afficher la liste des tâches
- Marquer une tâche comme terminée
- Supprimer une tâche

Étape 1 – Création du projet Angular

```
ng new task-manager
```

Options conseillées :

- Routing : **Yes**
- Styles : **CSS**

Puis :

```
cd task-manager
ng serve
```

➞ Vérifier que l'application fonctionne sur `http://localhost:4200`

Étape 2 – Comprendre la structure du projet

Faire analyser aux étudiants :

- `src/app/app.component.ts`
- `app.component.html`
- `app.module.ts`
- `angular.json`

Question pédagogique :

À quoi sert `AppModule` ? Pourquoi Angular est modulaire ?

Étape 3 – Création d'un composant `TaskList`

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

➞ Vérifier :

- `task-list.component.ts`
- `task-list.component.html`

Inclure le composant dans `app.component.html` :

```
<app-task-list></app-task-list>
```

Étape 4 – Création du modèle `Task`

Créer le fichier :

```
src/app/models/task.model.ts
```

```
export interface Task {
  id: number;
  title: string;
  completed: boolean;
}
```

💡 **Objectif** : typage fort avec TypeScript

Étape 5 – Affichage statique d'une liste de tâches

Dans `task-list.component.ts` :

```
tasks: Task[] = [
  { id: 1, title: 'Apprendre Angular', completed: false },
  { id: 2, title: 'Faire le TP', completed: true }
];
```

Dans le HTML :

```
<ul>
  <li *ngFor="let task of tasks">
    {{ task.title }} - {{ task.completed ? '✓' : '✗' }}
  </li>
</ul>
```

Étape 6 – Interaction : marquer une tâche comme terminée

HTML :

```
<li *ngFor="let task of tasks">
  <span [ngClass]="{ completed: task.completed }">
    {{ task.title }}
  </span>
  <button (click)="toggleTask(task)">Toggle</button>
</li>
```

TS :

```
toggleTask(task: Task) {
  task.completed = !task.completed;
}
```

🎯 Notions :

- Event binding `(click)`
- Property binding
- `ngClass`

Étape 7 – Ajout d'un formulaire

Importer `FormsModule` dans `app.module.ts`

```
import { FormsModule } from '@angular/forms';
```

HTML :

```
<input [(ngModel)]="newTaskTitle" placeholder="Nouvelle tâche">
<button (click)="addTask()">Ajouter</button>
```

TS :

```
newTaskTitle = '';

addTask() {
  this.tasks.push({
    id: Date.now(),
    title: this.newTaskTitle,
    completed: false
  });
  this.newTaskTitle = '';
}
```

🎯 Notions :

- Two-way binding `[(ngModel)]`

Étape 8 – Création d'un service `TaskService`

```
ng generate service services/task
@Injectable({ providedIn: 'root' })
export class TaskService {
  private tasks: Task[] = [];

  getTasks(): Task[] {
    return this.tasks;
  }

  addTask(task: Task) {
    this.tasks.push(task);
  }
}
```

➡ Injecter le service dans le composant

🎯 Notions :

- Service
- Injection de dépendances
- Séparation logique métier / UI