TP : Stack MERN Complète (Connexion Front-Back)

Objectif

Finaliser l'API (Update/Delete), autoriser les connexions extérieures (CORS), et transformer l'application React du Jour 2 pour qu'elle affiche les données réelles venant de votre base de données MongoDB.

Fichiers concernés

- **Backend :** src/controllers/movie.controller.ts, src/routes/movie.routes.ts, src/server.ts
- **Frontend :** Votre projet React du Jour 2 (App.tsx ou le composant principal).

Partie 1 : Finalisation du Backend (API)

Avant de connecter le front, terminons la logique du back.

1. **Contrôleur (src/controllers/movie.controller.ts)**
   Ajoutez les fonctions manquantes :
   - **updateMovie** : Utilise findByIdAndUpdate avec l'option { new: true }.
   - **deleteMovie** : Utilise findByIdAndDelete.
2. **Routes (src/routes/movie.routes.ts)**
   Créez le fichier et définissez les 5 routes :
   - POST / (create)
   - GET / (getAll)
   - GET /:id (getOne)
   - PUT /:id (update)
   - DELETE /:id (delete)
3. **Serveur (src/server.ts)**
   Importez et utilisez les routes avec le préfixe /api/movies.

Partie 2 : Débloquer la connexion (CORS)

Par sécurité, le navigateur bloque par défaut les requêtes entre deux ports différents (React : 5173 vers Node : 3000). Il faut lever cette barrière.

1. **Installation**
   Dans le terminal de votre **Backend** (dossier cinema-api), installez le module CORS et ses types :
   npm install cors
   npm install @types/cors --save-dev

2. **Configuration**
   Dans src/server.ts, importez et activez CORS **avant** vos routes :

```
import cors from 'cors';
// ...
const app = express();

// Autoriser le Frontend à communiquer avec l'API
app.use(cors());
app.use(express.json());
// ...
```

Partie 3 : La Métamorphose du Frontend

Reprenez votre projet **React du Jour 2** (qui était une "To-Do List" ou gestionnaire de tâches). Nous allons le transformer en "Gestionnaire de Films".

Instructions dans le code React :

1. **Mise à jour de l'Interface TypeScript**
   Dans votre fichier (ex: App.tsx), remplacez l'interface Task par l'interface Movie pour qu'elle corresponde à votre Backend :
   ```
   interface Movie {
    _id: string;     // MongoDB génère _id, pas id
    title: string;
    director: string;
    year: number;
   }
   ```

2. **Mise à jour du State**
   Changez le nom et le type de votre état :
   ```
   // Avant : const [tasks, setTasks] = useState<Task[]>([]);
   const [movies, setMovies] = useState<Movie[]>([]);
   ```

3. **Récupération des Données (Fetch)**
   Utilisez useEffect pour appeler votre API au chargement de la page :
   ```
   useEffect(() => {
    fetch('http://localhost:3000/api/movies')
      .then(response => response.json())
      .then(data => setMovies(data))
      .catch(error => console.error("Erreur API:", error));
   }, []);
   ```

4. **Mise à jour de l'Affichage (JSX)**
   Modifiez votre méthode .map() pour afficher les films.
   - Remplacez task.name par movie.title.

- Affichez aussi le réalisateur et l'année (ex: movie.director).
- **Attention :** La clé unique dans la boucle doit être key={movie._id}.

Résultat attendu :

En lançant votre backend (npm run dev) ET votre frontend (npm run dev), vous devriez voir s'afficher dans votre page React la liste des films que vous avez créés précédemment dans MongoDB via Postman.