TP : Création de l'API (Backend)

Objectif : Créer un serveur capable de recevoir des requêtes HTTP et de renvoyer des données JSON.

Stack : Node.js, Express, TypeScript.

Étape 1 : Initialisation du Serveur

Contrairement au client, nous partons de zéro pour le serveur.

1. À la racine de votre projet (à côté du dossier client), créez un dossier server.
2. Ouvrez un terminal **dans ce dossier** (cd server).
3. Initialisez le projet Node :
   npm init -y
4. Installez les dépendances vitales :
   npm install express cors
5. Installez les outils de développement (TypeScript) :
   npm install -D typescript ts-node nodemon @types/node @types/express @types/cors

Étape 2 : Configuration TypeScript

Pour que Node comprenne le TypeScript, il faut un fichier de config.

1. Dans server, créez un fichier tsconfig.json.
2. Collez cette configuration :

```
{
  "compilerOptions": {
    "target": "es2016",
    "module": "commonjs",
    "outDir": "./dist",
    "rootDir": "./src",
    "strict": true,
    "esModuleInterop": true
  }
}
```

3. Ouvrez package.json et modifiez la partie "scripts" pour lancer le serveur facilement :

```
"scripts": {
  "start": "ts-node src/server.ts",
  "dev": "nodemon src/server.ts"
}
```

Étape 3 : Votre premier serveur

1. Créez un dossier src dans server.
2. Créez un fichier src/server.ts.
3. À vous de jouer ! Complétez les trous pour faire fonctionner le serveur.

```
import express from 'express';
import cors from 'cors';

const app = express();
const PORT = 3001; // Port 3001 pour ne pas gêner React (5173)

// --- MIDDLEWARES ---
// TODO : Ajoutez les deux middlewares essentiels :
// 1. Celui pour autoriser les requêtes externes (CORS)
// 2. Celui pour lire le JSON dans le body des requêtes
___
___

// --- DONNÉES ---
// Simulation de base de données en mémoire
let books = [
  { id: 1, title: 'Express pour les nuls', author: 'Node JS' }
];

// --- ROUTES ---

// Route de test
app.get('/', (req, res) => {
  res.send('API Library v1.0 is running...');
});

// TODO 1 : Route GET pour /api/books
// Cette route doit renvoyer le tableau 'books' au format JSON
app.get('/api/books', (req, res) => {

  ___
});

// TODO 2 : Route POST pour /api/books
// Cette route doit :
// 1. Récupérer le titre et l'auteur dans le body de la requête
```

```
// 2. Créer un nouvel objet livre avec un ID unique (ex: Date.now())
// 3. Ajouter ce livre au tableau 'books'
// 4. Renvoyer le livre créé avec le code HTTP 201
app.post('/api/books', (req, res) => {
    // A vous de coder la logique ici...

});

// --- DÉMARRAGE ---
// TODO : Lancez le serveur sur le port défini
```

Étape 4 : Tester sans React

Avant de brancher le frontend, il faut vérifier que le moteur tourne.

1. Lancez le serveur : npm run dev.
2. Ouvrez votre navigateur sur http://localhost:3001/api/books.
    - Voyez-vous le JSON ?
3. (Optionnel) Utilisez **Postman** ou Thunder Client (extension VS Code) pour tester le POST.
    - URL : http://localhost:3001/api/books
    - Method : POST
    - Body (JSON) : {"title": "Mon Livre", "author": "Moi"}

Challenge (Bonus)

Ajoutez une route pour supprimer un livre par son ID :

DELETE /api/books/:id

Indice : Vous aurez besoin de req.params pour récupérer l'ID dans l'URL, et probablement de la méthode .filter() pour mettre à jour le tableau.