■■■ Cours & TP : ReactJS + ExpressJS + MongoDB

Contenu : Rappel ReactJS, Théorie Express/Mongo/Postman, Guide TP Complet + Connexion Front/Back

■■ Rappel – ReactJS (Frontend)

React est une bibliothèque JavaScript qui permet de construire des interfaces utilisateur modernes, dynamiques et modulaires. Elle utilise des composants réutilisables, du JSX et des hooks pour gérer les interactions.

■ *Concepts clés :*

- JSX : HTML dans du JavaScript (syntaxe transformée par Babel)
- Composants : fonctions qui retournent du JSX
- Props : paramètres passés d'un composant parent à enfant
- State : variable interne à un composant
- useState() : hook pour gérer un état local
- useEffect() : hook pour les effets secondaires (API, timers…)

■ Théorie – ExpressJS, MongoDB, Postman

■ *ExpressJS (Backend)*

Express est un framework minimaliste basé sur Node.js permettant de créer des API, serveurs web ou microservices rapidement. Il utilise une architecture REST et permet de définir des routes (GET, POST, PUT, DELETE).

■ *MongoDB (NoSQL DB)*

MongoDB est une base de données NoSQL orientée document. Les données sont stockées au format BSON (JSON étendu). Elle permet une structure souple, idéale pour les projets web modernes. On l'utilise ici via Mongoose (ORM JS).

■ *Postman (Client de test API)*

Postman est un outil pour tester les APIs REST (Express). Il permet :
- d'envoyer des requêtes HTTP (GET/POST/PUT/DELETE),
- de lire les réponses JSON,
- de simuler des formulaires ou fichiers.

■■ TP – API REST Express + MongoDB + Tests avec Postman

■ *Initialisation*

```
mkdir api-express-mongo cd api-express-mongo npm init -y npm install express mongoose nodemon cors
```

■ *index.js – serveur principal*

```
const express = require('express'); const mongoose = require('mongoose'); const cors = require('cors'); const app = express(); app.use(cors()); app.use(express.json()); mongoose.connect('mongodb://localhost:27017/etudiants'); app.listen(3000, () => console.log('Serveur sur http://localhost:3000'));
```

■ *models/Etudiant.js – modèle Mongoose*

```
const mongoose = require('mongoose'); const EtudiantSchema = new mongoose.Schema({ nom: String, age: Number, email: String }); module.exports = mongoose.model('Etudiant', EtudiantSchema);
```

■ *routes/etudiants.js – routes API*

```
const express = require('express'); const router = express.Router(); const Etudiant = require('../models/Etudiant'); router.post('/', async (req, res) => { const e = new Etudiant(req.body); const saved = await e.save(); res.json(saved); }); router.get('/', async (req, res) => { const list = await Etudiant.find(); res.json(list); }); module.exports = router;
```

■ BONUS – Connexion ReactJS (Vite) au Backend ExpressJS

■ *Exemple dans React (frontend) :*

```
useEffect(() => { fetch('http://localhost:3000/api/etudiants') .then(res =>
res.json()) .then(data => setEtudiants(data)); }, []);
```

■ *Test API avec Postman :*

1. POST http://localhost:3000/api/etudiants
2. Body : raw JSON : { "nom": "Ali", "age": 23, "email": "ali@example.com" } 3. Vérifiez dans MongoDB Compass ou en GET que l'élément est ajouté.

■ Résultat : une application complète front React ↔ back Express + Mongo.