Fiche résumé : Construisez un backend avec Node.js et NestJS

1. Comprendre le rôle de Node.js dans une architecture web moderne
2. Explorer le fonctionnement asynchrone et la boucle d'événements
3. Créer un serveur HTTP simple avec le module natif Node.js
4. Gérer les dépendances avec npm et configurer package.json
5. Structurer une API avec NestJS en modules, contrôleurs et services

Code : Javascript

```
const http = require('http');
const server = http.createServer((req, res) => {
res.statusCode = 200;
res.setHeader('Content-Type', 'text/plain');
res.end('Bonjour depuis Node.js !');
});
server.listen(3000);
```

```
npm install express
npm install eslint --save-dev
```

```
@Get('projects')
getProjects() {
return this.appService.getProjects();
}
```

```
@Get(':id')
getOneProject(@Param('id', ParseIntPipe) id) {
return `ID: ${id}`;
}
```

Bonnes pratiques 👍

- ✅ Utiliser un seul langage JavaScript pour front-end et back-end
- ✅ Structurer le code avec des frameworks comme NestJS
- ✅ Séparer les responsabilités entre Controller et Service
- ✅ Centraliser les dépendances dans package.json
- ✅ Utiliser npm pour gérer automatiquement les paquets
- ✅ Tester les routes avec un client HTTP comme Postman
- ✅ Valider les données avec des Pipes avant traitement
- ✅ Protéger les routes avec des Guards pour sécuriser l'accès

Erreurs classiques 👎

- ❌ Mélanger logique métier et gestion HTTP dans le Controller
- ❌ Coder un serveur complexe sans framework structurant
- ❌ Versionner le dossier node_modules dans Git
- ❌ Modifier manuellement le fichier package-lock.json
- ❌ Dupliquer les vérifications de sécurité dans chaque route
- ❌ Ignorer la séparation entre dependencies et devDependencies
- ❌ Négliger la validation des données entrantes
- ❌ Laisser une API accessible sans contrôle d'accès

Définitions 🔍

Node.js
Environnement d'exécution JavaScript côté serveur basé sur V8

API
Interface permettant la communication entre front-end et back-end

Event loop
Mécanisme gérant les opérations asynchrones non bloquantes

npm
Gestionnaire de paquets pour installer et gérer des bibliothèques

Controller
Composant recevant les requêtes HTTP et renvoyant les réponses

Service
Composant contenant la logique métier de l'application

Module
Structure regroupant les éléments liés à une fonctionnalité