Node.js & MongoDB

Node.js et MongoDB : deux technologies incontournables

Objectifs du module

Compétences

Maîtriser la mise en place d'un serveur Node.js et l'utilisation d'une base de données MongoDB.

Notation du module

Contrôle Continu

Devoir surveillé comprenant des questions à choix multiples, des questions ouvertes, et des mises en situation réelles.

Projet

Projet visant à mettre en place un serveur Node.js et une base de données MongoDB, afin de mettre en pratique les concepts théoriques.

Qu'est-ce que Node.js ?

Environnement JavaScript côté serveur.

Basé sur le moteur V8 de Google Chrome.

Avantages :

- Utilisation de JavaScript côté front & back
- Système asynchrone et non-bloquant, ce qui le rend très performant pour les applications temps réel et les serveurs à forte charge.

Installation de Node.js

- Téléchargez Node.js sur nodejs.org en choisissant la version LTS.
- Vérifiez l'installation :

```
node -v
npm -v
```

- Créez un projet Node.js :

```
mkdir monprojet
cd monprojet
npm init -y
```

Serveur Node.js

```
const http = require('http');

const server = http.createServer((req, res) => {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Hello World\n');
});

server.listen(3000, () => {
    console.log('Server running at http://127.0.0.1:3000/');
});
```

Modules dans Node.js

Un module est un fichier JavaScript qui est importer à l'aide de require().

```
// math.js
function add(a, b) {
    return a + b;
}
module.exports = { add };

// app.js
const math = require('./math');
console.log(math.add(2, 3));  // Affiche 5
```

Express.js

Express facilite la création d'application web.

Vous pouvez l'installer avec la commande suivante :

```
npm install express
```

Serveur Node.js avec Express

```
const express = require('express');
const app = express();

app.get('/', (req, res) => {
    res.send('Hello from Express');
});

app.listen(3000, () => {
    console.log('Server is running on http://localhost:3000');
});
```

API REST

Interface qui permet à des applications d'interagir entre elles via HTTP en utilisant des méthodes comme GET, POST, PUT et DELETE.

```
app.get('/users', (req, res) => {
    res.send('Liste des utilisateurs');
});

app.post('/users', (req, res) => {
    res.send('Ajouter un utilisateur');
});

app.put('/users/:id', (req, res) => {
    res.send(`Modifier l'utilisateur avec l'ID ${req.params.id}`);
});

app.delete('/users/:id', (req, res) => {
    res.send(`Supprimer l'utilisateur avec l'ID ${req.params.id}`);
});
```

Qu’est-ce que MongoDB

Base de données NoSQL stockant les données sous forme de documents JSON.

Flexible et adaptée aux applications modernes.

Aucune table relationnelle.

Structuration des données en collections et documents.

Utilisations de MongoDB

Il existe deux manières d'utiliser MongoDB :

- Installation locale
- Utilisation de MongoDB Atlas, version cloud

Installation de Mongoose

ODM (Object Data Modeling) qui simplifie l'interaction avec MongoDB.

```
npm install mongoose
```

Exemple de connexion avec Mongoose :

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/monappli', { useNewUrlParser: true, useUni
    .then(() => console.log('Connexion à MongoDB réussie'))
    .catch((err) => console.error('Erreur de connexion', err));
```

CRUD avec Node.js et MongoDB

Création d'un modèle avec Mongoose :

```
const userSchema = new mongoose.Schema({
    name: String,
    age: Number,
    email: String
});

const User = mongoose.model('User', userSchema);
```

Opérations CRUD

Create :

```
const newUser = new User({ name: 'John', age: 30, email: 'john@example.com' });
newUser.save().then(() => console.log('Utilisateur ajouté'));
```

Opérations CRUD

Read :

```
User.find().then(users => console.log(users));
```

Opérations CRUD

Update :

```
User.updateOne({ _id: 'ID' }, { name: 'John Updated' }).then(() => console.log('Utilis
```

Opérations CRUD

Delete :

```
User.deleteOne({ _id: 'ID' }).then(() => console.log('Utilisateur supprimé'));
```

JSON Web Tokens (JWT)

JWT est une méthode sécurisée pour transmettre des informations entre client et serveur.

Installez JWT :

```
npm install jsonwebtoken
```

Exemple d'authentification avec JWT :

```
const jwt = require('jsonwebtoken');

const user = { id: 1, username: 'testuser' };
const token = jwt.sign(user, 'secret_key', { expiresIn: '1h' });

console.log(token);
```

Protection d'une route avec JWT

```
const verifyToken = (req, res, next) => {
    const token = req.headers['authorization'];
    if (!token) return res.status(403).send('Token manquant.');

    jwt.verify(token, 'secret_key', (err, decoded) => {
        if (err) return res.status(500).send('Échec de l'authentification.');
        req.user = decoded;
        next();
    });
};

app.get('/private', verifyToken, (req, res) => {
    res.send('Bienvenue sur une route protégée');
});
```

PROJET

Concevoir une application Web avec Node.js et MongoDB