Javascript /node.js

- Node.js :Moteur JavaScript côté serveur pour exécuter le code.
- Express.js :Framework Node.js utilisé pour créer les routes de l'API.
- SQLite : Base de données légère pour stocker les données de produit, avec le module sqlite3 pour interagir avec elle en JavaScript.
- Promesses (Promises) : Utilisées pour gérer les requêtes asynchrones vers la base de données.
- async/await : Syntaxe moderne pour gérer les opérations asynchrones en JavaScript, rendant le code plus lisible.

- **Initialiser le projet**

Créer un nouveau dossier pour le projet , puis ouvrez-le dans votre terminal :

```
mkdir product-manager
cd product-manager
```

2. Initialiser le projet avec npm :

```
npm init -y
```

3. Installer les dépendances :

```
npm install express sqlite3
```

4. Créer le fichier app.js

Dans le dossier de votre projet, créez un fichier app.js et collez le code suivant :

```
// Import des modules
const express = require('express');
const sqlite3 = require('sqlite3').verbose();
const app = express();
const PORT = 3000;

// Configurer le middleware pour le JSON
app.use(express.json());

// Connexion à la base de données SQLite (mémoire ou fichier)
const db = new sqlite3.Database(':memory:');

// Création de la table produits
db.serialize(() => {
  db.run(` CREATE TABLE IF NOT EXISTS products (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    price REAL NOT NULL
  )`);
});
// Fonction pour ajouter un produit (CREATE)
function addProduct(name, price) {
  return new Promise((resolve, reject) => {
    const query = ` INSERT INTO products (name, price) VALUES (?, ?)` ;
    db.run(query, [name, price], function (err) {
      if (err) {
        return reject(err.message);
      }
```

```
        resolve({ id: this.lastID, name, price });
      });
    });
}

// Fonction pour récupérer tous les produits (READ)
function getAllProducts() {
    return new Promise((resolve, reject) => {
      db.all("SELECT * FROM products", [], (err, rows) => {
        if (err) {
          return reject(err.message);
        }
        resolve(rows);
      });
    });
}

// Fonction pour mettre à jour un produit (UPDATE)
function updateProduct(id, name, price) {
    return new Promise((resolve, reject) => {
      const query = `UPDATE products SET name = ?, price = ? WHERE id = ?`;
      db.run(query, [name, price, id], function (err) {
        if (err) {
          return reject(err.message);
        }
        resolve({ id, name, price });
      });
    });
```

```
}

// Fonction pour supprimer un produit (DELETE)
function deleteProduct(id) {
  return new Promise((resolve, reject) => {
    const query = `DELETE FROM products WHERE id = ?`;
    db.run(query, id, function (err) {
      if (err) {
        return reject(err.message);
      }
      resolve(`Product with ID ${id} has been deleted.`);
    });
  });
}

// Route pour ajouter un produit
app.post('/products', async (req, res) => {
  const { name, price } = req.body;
  try {
    const product = await addProduct(name, price);
    res.status(201).json(product);
  } catch (err) {
    res.status(400).json({ error: err });
  }
});

// Route pour récupérer tous les produits
app.get('/products', async (req, res) => {
```

```
  try {
    const products = await getAllProducts();
    res.status(200).json(products);
  } catch (err) {
    res.status(500).json({ error: err });
  }
});

// Route pour mettre à jour un produit
app.put('/products/:id', async (req, res) => {
  const { id } = req.params;
  const { name, price } = req.body;
  try {
    const product = await updateProduct(id, name, price);
    res.status(200).json(product);
  } catch (err) {
    res.status(400).json({ error: err });
  }
});

// Route pour supprimer un produit
app.delete('/products/:id', async (req, res) => {
  const { id } = req.params;
  try {
    const message = await deleteProduct(id);
    res.status(200).json({ message });
  } catch (err) {
    res.status(400).json({ error: err });
```

```
  }
});
```

// Lancement du serveur

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

5.Lancer le serveur

1. Assurez-vous d'être dans le dossier de votre projet dans le terminal.
2. Lancez le serveur avec la commande suivante :

```
node app.js
```

3. Vous verrez dans le terminal :

```
Server is running on http://localhost:3000
```

6. Tester les Routes API

Pour tester les différentes opérations (CRUD), vous pouvez utiliser [Postman](https://www.postman.com/)