TP no 3: MongoDB / Node.js

November 28, 2024

1 Express pour créer une API

Nous avons vu dans le TD6 comment créer une application Web avec **Node.js** et le framework **Express**.

https://expressjs.com/en/starter/hello-world.html

Vous allez utiliser ce framework pour créer une web API. Une web API est une façon simple d'exposer des données via des **URL**.Par exemple l'url :

https://api.northwind.com/product/17

Permet d'obtenir les détails du produit d'id 17

```
{
    _id: ObjectId('673cc26ecae692def51ac534'),
    productID: 17,
    productName: 'Alice Mutton',
    supplierID: 7,
    categoryID: 6,
    quantityPerUnit: '20 - 1 kg tins',
    unitPrice: 39,
    unitsInStock: 0,
    unitsOnOrder: 0,
    reorderLevel: 0,
    discontinued: 1
  }
```

On peut donc utiliser cette URL pour développer une application (Web, Mobile, Client lourd, ...) qui n'a besoin d'aucune données relative au système de gestion de données sous-jacent.

1.1 Créer une route

Une **route** est une url particulière que l'on paramètre pour exécuter une requête spécifique. Par exemple dans l'url précédente :

https://api.northwind.com/ product / 17

product va permettre d'exécuter des requêtes sur la collection **product** et **17** permet de paramétrer **productID**.

Pour créer une route:

```
app.get('/product/:productID', async (req, res) => {
    const productID = parseInt(req.params.productID, 10);
    if (isNaN(productID)) {
        return res.status(400).json({ error: 'Invalid productID. Must be an
   integer.' });
    }

    let client;
    try {
        client = new MongoClient(mongoUrl);
        await client.connect();
        console.log('Connected to MongoDB');

        const db = client.db(dbName);
        const collection = db.collection('products');

        // Recherche dans MongoDB
        const product = await collection.findOne({ productID });
        if (product) {
            res.json(product);
        } else {
            res.status(404).json({ error: 'Product not found' });
        }
    } catch (err) {
        console.error('Error:', err);
        res.status(500).json({ error: 'Internal Server Error' });
    } finally {
        if (client) {
            await client.close();
        }
    }
});
```

Tester le fichier **api.js** vérifiez si vous obtenez bien les documents voulus en tapant :

https://localhost:3000/product/11

1.2 L'API

Créez les routes suivantes :

- localhost:3000/supplier/:supplierID pour obtenir le fournisseur par son ID
- localhost:3000/customer/:customerID pour obtenir le fournisseur par son ID
- localhost:3000/order/:orderID pour obtenir la commande par son ID

Exécuter des requêtes complexes comme des **Agregations** par exemple :

- localhost:3000/customer_order/:customerID La liste trié des commandes pour un utilisateur par date.
- localhost:3000/supplier_product/:supplierID La liste liste trié par nom des produits par fournisseur.
- ...

La documentation exhaustive du **connecteur mongodb** se trouve à l'adresse suivante:

https://www.mongodb.com/docs/drivers/node/current/

1.3 Serveur Web

Nous allons simuler une application web externe qui n'utiliserai que des appels API.

Pour cela :

- Renommer votre fichier **api.js** en **server.js**
- Créer un répertoire WWW (votre serveur web)
- Ajouter une route statique qui sera la racine de votre serveur web

Pour ajouter une route statique :

```
// WWW:

app.use(express.static(path.join(__dirname, 'WWW')));

// Route par defaut pour renvoyer index.html
app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'WWW', 'index.html'));
});
```

Vous avez maintenant un serveur web disponible à l'adresse : https://localhost:3000

Vous pouvez maintenant ajouter **index.html** à WWW et le tester !

Ce fichier exécute un appels de votre API (Il simule une application tiers)

Vous pouvez débuter un développement web classique de **frontend**, ajouter des bibliothèques externes Vue.js etc ...

1.4 Frontend pour l'API

Vous devez développer quelque chose comme ça (en plus jolie) :

http://excel.dataweb.com/Categories/default.view

1.5 Dashboard

Vous allez développer comme deuxième outil un tableau d'aide à la décision comme on peut le voir ici :

https://github.com/asyaparfenova/dashboard_NorthWind_Database?tab=readme-ov-file

Vous pouvez récupérer des données supplémentaires Shippers, Employees etc

https://github.com/pawlodkowski/northwind_data_clean/blob/master/data/README.md

Pour le réaliser il existe de nombreuse bibliothèques de Chart Js comme **AnyChart**

https://docs.anychart.com/Quick_Start/Quick_Start