

Exercices Node.JS

Exercice 1

- Écrire un programme Serveur NodeJS nommé ServeurExo1 qui gère les produits en utilisant le serveur Express.JS. Les services Web suivants doivent être créés :
 - createProduct
 - deleteProduct
 - getProduct
 - getProducts
 - updateProduct
- Le programme doit utiliser une base de données MySQL qui comporte la table Produit avec les données
- Le système fournit une interface Web client permet d'avoir les services suivantes via AJAX de JQuery

Exercice 1 (suite)

Nouveau Produit

Modification de Produit

Suppression de Produit

Liste Produit

Nouveau produit

Description

Image

Prix

Details

Enregistrer

Nouveau Produit

Modification de Produit

Suppression de Produit

Liste Produit

Modification de produit

ID du produit Rechercher

Modification souhaitée

Description

Image

Prix

Details

Modifier

Exercice 1 (suite)

Nouveau Produit Modification de Produit Suppression de Produit Liste Produit

Suppression de produit

ID du produit

Produit à supprimer

Description

Image






Prix

Details

Supprimer

Nouveau Produit Modification de Produit Suppression de Produit Liste Produit

Liste de produit

Id		Description	Image	Prix	Details
1		HP portatif		350.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
2		Portatif Macbook Pro MF839LA		1549.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
3		Portatif Macbook Air MJVE2C		1990.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
4		MSI Portatif de jeu Leopard		1099.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
5		HP portatif 001		350.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir

Étape 1 – Script de la création de la base de données et d'insertion de données dans la table produit

```
drop database if exists gestionProduit;  
create database gestionProduit;  
use gestionProduit;
```

```
drop table if exists Produit;  
create table Produit(  
    id int(10) primary key auto_increment,  
    description varchar(50),  
    image varchar(30),  
    prix decimal(10,2),  
    details varchar(200)  
);
```

```
insert into Produit values  
(null, "HP portatif", "portable1.jpg", 350.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Pro MF839LA", "portable2.jpg", 1549.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Air MJVE2C", "portable3.jpg", 1990.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "MSI Portatif de jeu Leopard", "portable4.jpg", 1099.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "HP portatif 001", "portable1.jpg", 350.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Pro MF839LA 001", "portable2.jpg", 1549.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Air MJVE2C 001", "portable3.jpg", 1990.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "MSI Portatif de jeu Leopard 001", "portable4.jpg", 1099.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "HP portatif 002", "portable1.jpg", 350.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Pro MF839LA 002", "portable2.jpg", 1549.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Air MJVE2C 002", "portable3.jpg", 1990.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "MSI Portatif de jeu Leopard 002", "portable4.jpg", 1099.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "HP portatif 003", "portable1.jpg", 350.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Pro MF839LA 003", "portable2.jpg", 1549.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "Portatif Macbook Air MJVE2C 003", "portable3.jpg", 1990.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"),  
(null, "MSI Portatif de jeu Leopard 003", "portable4.jpg", 1099.95, "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir");
```

Le script de création de BD, de tables
et d'insertion de données est fourni.

Le fichier est : scriptsql.sql

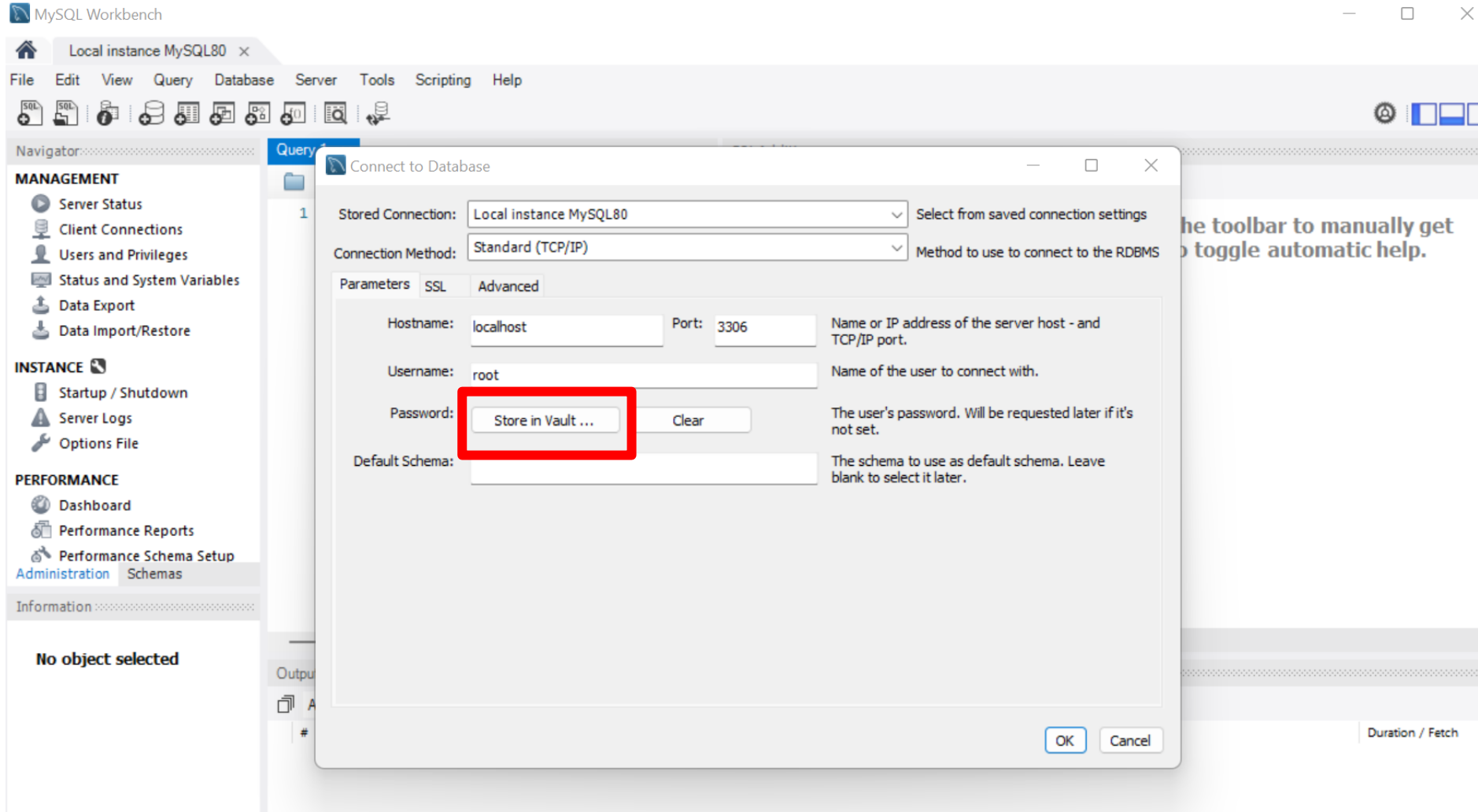
Lancer le script à l'aide d'un client

MySQL comme Workbench.

Étape 1.1 Exécution du script dans MySQL Workbench

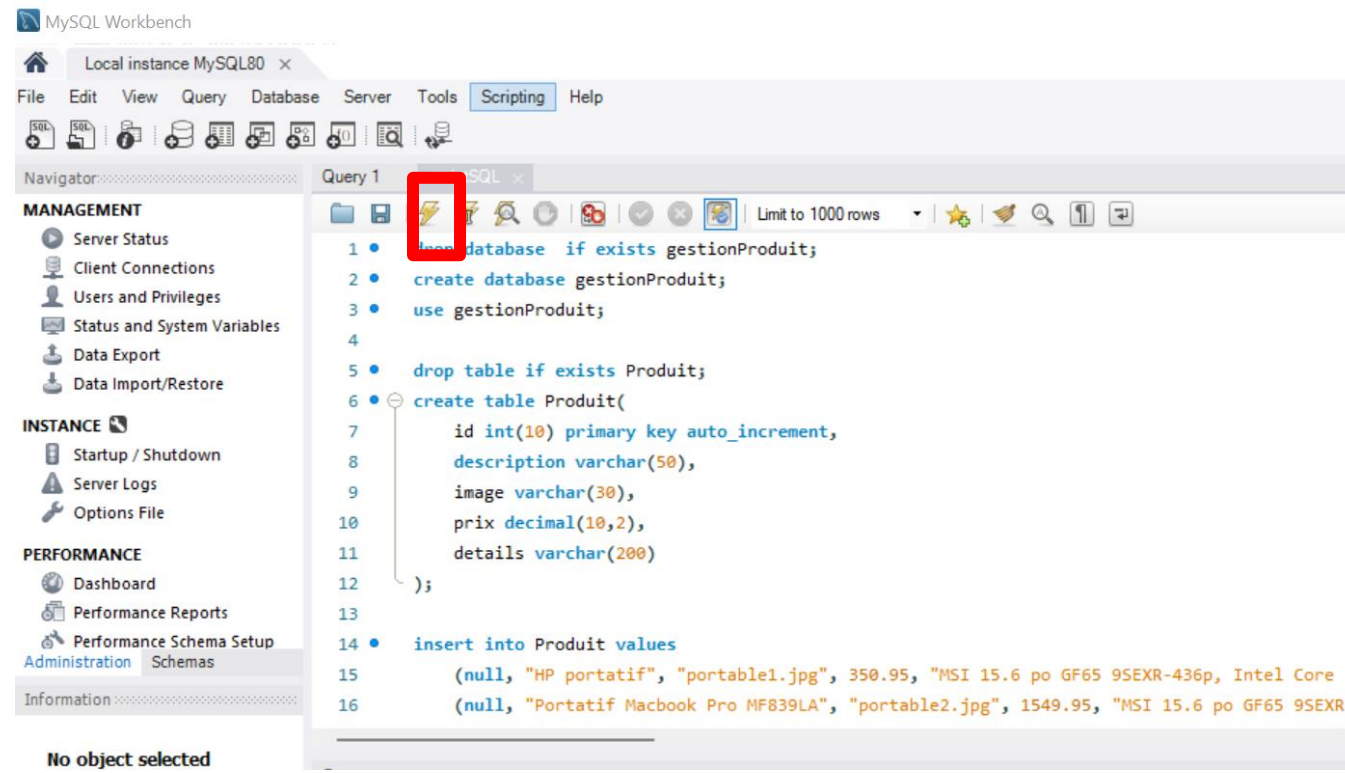
- Lancer MySQL Workbench
- Pour se connecter au serveur, cliquer Database/Connect to Database
- Cliquer sur le bouton Store in vault pour saisir le mot de passe
- Cliquer sur Ok pour sauvegarder le mot de passe
- Cliquer sur OK pour se connecter au serveur

Étape 1.1 Exécution du script dans MySQL Workbench (Suite)



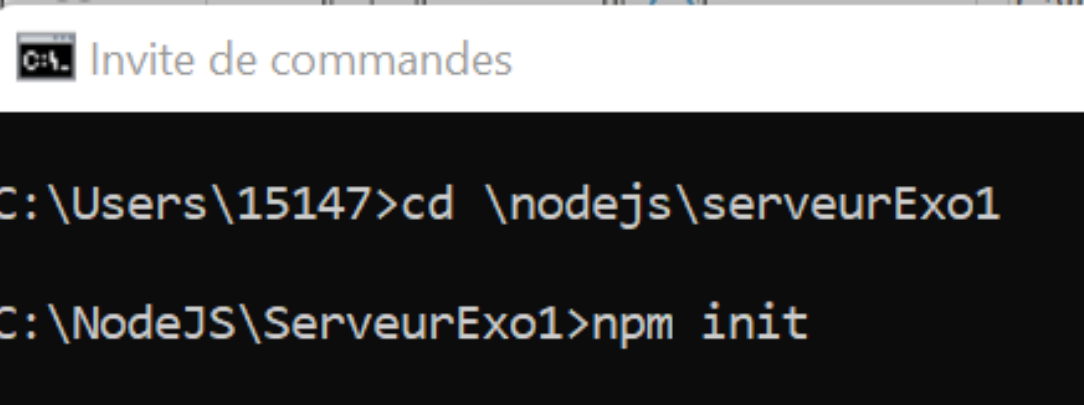
Étape 1.1 Exécution du script dans MySQL Workbench (Suite)

- Charger le script dans Workbench
 - File / Open SQL Script ou Ctrl + Maj + O
- Charger le fichier scriptSQL.sql disponible pour le lab
- Cliquer sur le bouton Exécuter pour lancer le script



Étape 2 – Créer votre dossier de projet NodeJS

- Créer le dossier ServeurExo1 qui va jouer le rôle de dossier du projet NodeJS de Gestion de produits
- À partir du menu Démarrer de Windows, taper CMD
- Changer de répertoire pour aller dans ServeurExo1
 - **Cd \NodeJS\ServeurExo1**
- Puis exécutez la commande ci-dessous pour que NPM crée un projet pour vous
 - **Npm init**



```
C:\Users\15147>cd \nodejs\serveurExo1  
C:\NodeJS\ServeurExo1>npm init
```

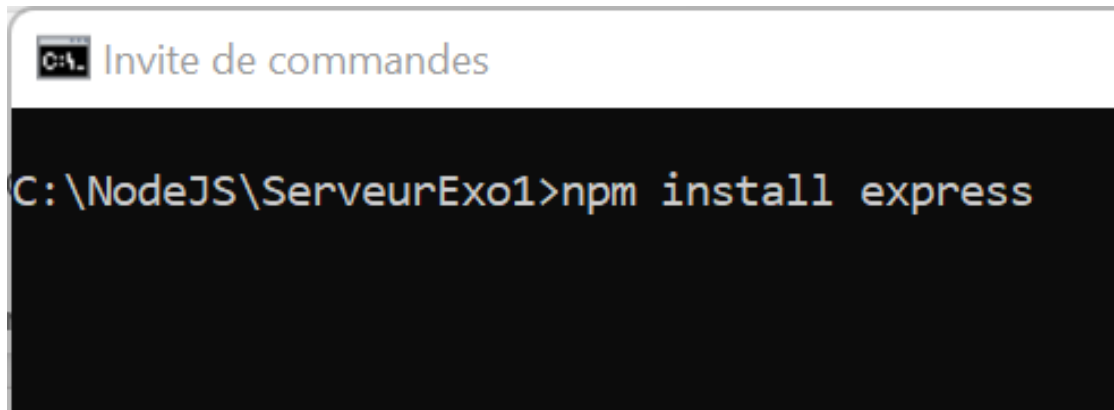
Étape 2.1 Le fichier package.json créé

- Noter la création du fichier package.json dans le projet

```
{  
  "name": "serveurex01",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

Étape 3 - Installation du serveur Express

- Le serveur Express va nous permettre de développer des API REST pour le projet pour nous permettre de retourner des données JSON au lieu des pages Web HTML.
- Taper à la ligne de commande
 - **Npm install express**



```
C:\> Invite de commandes

C:\NodeJS\ServeurExo1>npm install express
```

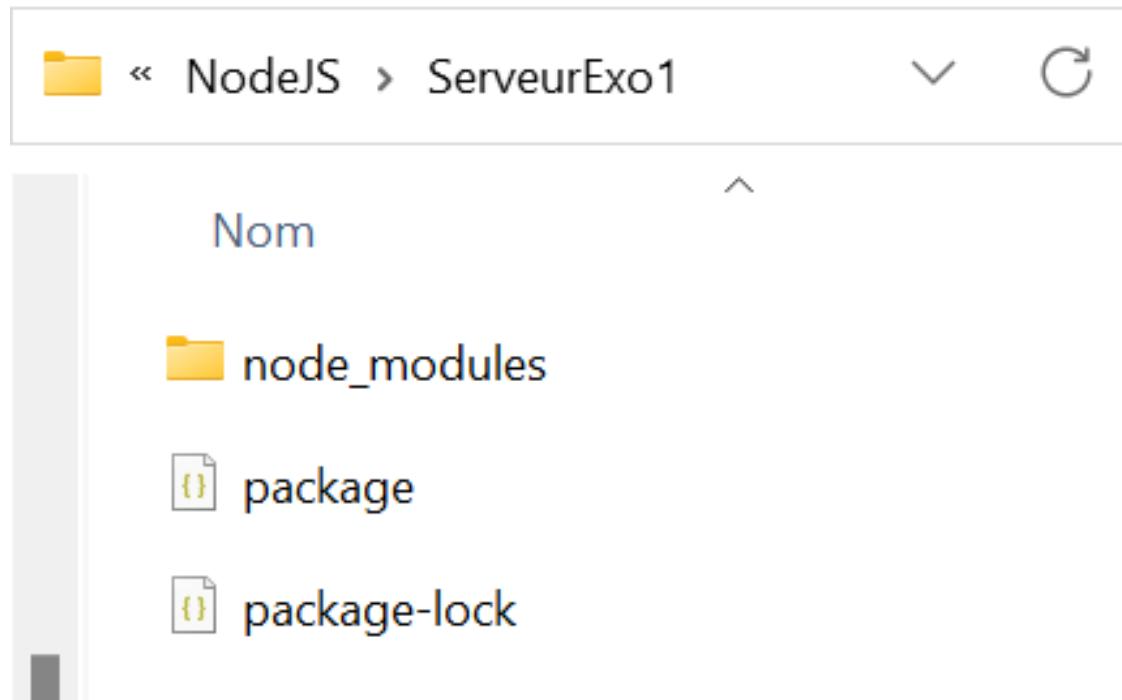
Étape 3.1 – Installation du serveur Express

- Noter le changement dans le fichier package.json

```
{
  "name": "serveurex01",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.1"
  }
}
```

Étape 3.2 Installation de Express

- Noter la création du dossier NodeModule dans le dossier du projet



Étape 4 - Ajouter le fichier app.js dans le projet

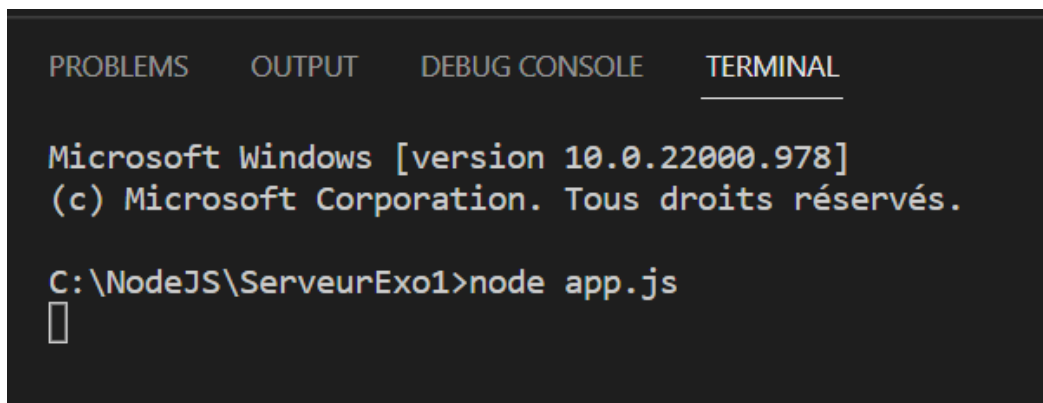
JS app.js X

ServeurExo1 > JS app.js > app.get("/") callback

```
1  const express = require("express");
2
3  const app = express();
4  const port = 3000;
5
6  app.get("/", function(request, response){
7    response.send("Bonjour tout le monde<br/>Utilisation du serveur Express")
8  });
9  app.listen(3000);
```

Étape 4.1 Lancer le serveur

- Dans le terminal, taper
 - **Node app.js**
- Dans le navigateur, taper
 - <http://localhost:3000>
 - Ce qui affiche la page suivante :



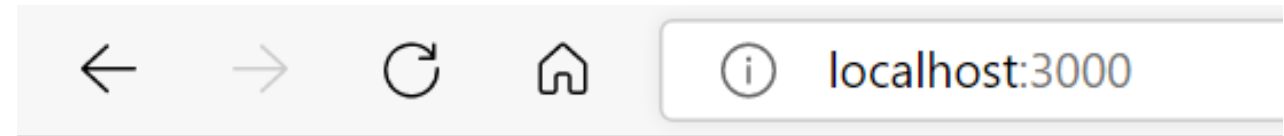
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

Microsoft Windows [version 10.0.22000.978]
(c) Microsoft Corporation. Tous droits réservés.

C:\NodeJS\ServeurExo1>node app.js
█
```

Lancer le command Line dans Visual Studio code

1. Press **Ctrl + Shift + P** to open the command palette.
2. Type '**profile**' in the searcher.
3. Select '**Terminal: Select Default Profile**'.
4. Select '**Command Prompt**'
5. The next time you try to open the terminal you should see '**CMD**' instead of '**PowerShell**'.



Bonjour tout le monde
Utilisation du serveur Express

Étape 4.2 Changer le fichier package.json pour démarrer le serveur avec start npm

```
ServeurExo1 > {} package.json > {} dependencies  
1  {  
2    "name": "serveurexo1",  
3    "version": "1.0.0",  
4    "description": "",  
5    "main": "app.js",  
6    "scripts": {  
7      "start" : "node app.js",  
8      "test": "echo \"Error: no test specified\" && exit 1"  
9    },  
10   "author": "",  
11   "license": "ISC",  
12   "dependencies": {  
13     "express": "^4.18.1"  
14   }  
15 }
```

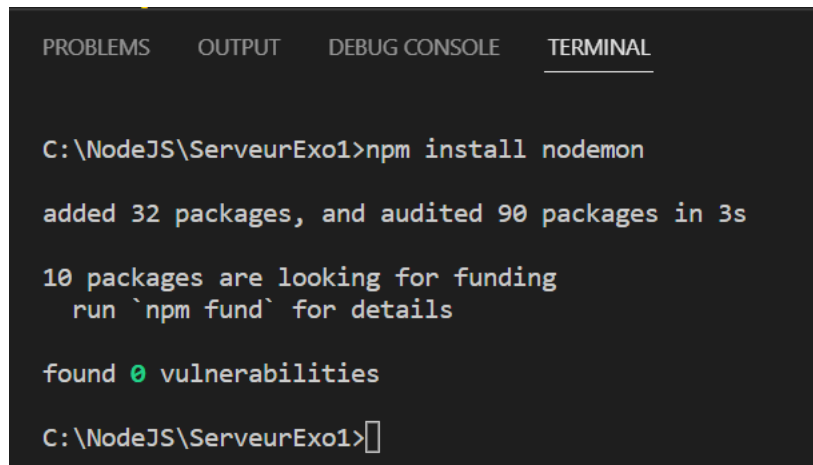
Démarrer le serveur avec la commande :

Npm Start

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  
  
Microsoft Windows [version 10.0.22000.978]  
(c) Microsoft Corporation. Tous droits réservés.  
  
C:\NodeJS\ServeurExo1>node app.js  
^C  
C:\NodeJS\ServeurExo1>npm start  
  
> serveurexo1@1.0.0 start  
> node app.js
```


Étape 4.3 Installation de nodemon pour lancer le serveur

- On est obligé à chaque fois Ctrl + C pour arrêter le serveur puis le redémarrer avec npm start pour chaque changement apporté avant de rafraîchir la page.
- Le module nodemon qui est utilisé uniquement dans le développement permet de relancer le serveur à chaque changement enregistré.
- La commande est :
 - Npm install nodemon



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

C:\NodeJS\ServeurExo1>npm install nodemon

added 32 packages, and audited 90 packages in 3s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\NodeJS\ServeurExo1>
```

Étape 5 Install MySQL DB

- Ajouter le driver de MySQL au projet
 - Npm install mysql
- À utiliser le module de MySQL dans le projet
 - Var mysql = require(« mysql »)
- On teste avec la commande
 - Npm start

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

C:\NodeJS\ServeurExo1>npm install mysql

added 12 packages, and audited 102 packages in 2s

10 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\NodeJS\ServeurExo1>
```

```
C:\NodeJS\ServeurExo1\node_modules\mysql\lib\protocol\Parser.js:437
    throw err; // Rethrow non-MySQL errors
    ^

Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server; consider upgrading MySQL client
    at Handshake.Sequence._packetToError (C:\NodeJS\ServeurExo1\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
    at Handshake.ErrorPacket (C:\NodeJS\ServeurExo1\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
    at Protocol._parsePacket (C:\NodeJS\ServeurExo1\node_modules\mysql\lib\protocol\Protocol.js:291:23)
    at Parser._parsePacket (C:\NodeJS\ServeurExo1\node_modules\mysql\lib\protocol\Parser.js:433:10)
```

Étape 5.1 Tester La connection de MySQL

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

var con = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "Toscane2000**",
  database: "gestionProduit"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM Produit", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

Créer ce code dans le fichier app.js pour tester

5.1 Tester la connection - Erreur

La cause de l'erreur ci-dessus est que vous utilisez la base de données trop nouvelle **MySQL** (version >= 8.x). Les anciennes versions de **MySQL** (5.x) utilisent le plugin d'authentification tel que **SHA256_PASSWORD**. La version **MySQL 8.x** utilise le plugin d'authentification tel que **SHA2_PASSWORD**. La bibliothèque **NodeJS MySQL** n'a pas encore changé. Il utilise le plugin d'authentification tel que **SHA256_PASSWORD** et n'a pas supporté **SHA2_PASSWORD**.

Pour résoudre l'erreur ci-dessus, ouvrez la fenêtre "**MySQL Command Line Client**" et exécutez la commande suivante :

```
# Syntax:
```

```
ALTER USER 'my_username'@'my_host' IDENTIFIED WITH 'mysql_native_password' BY 'my_password';
```

```
# Example:
```

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH 'mysql_native_password' BY '12345';
```

5.1 Tester la connection - Erreur

MySQL 8.0 Command Line Client - Unicode

```
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 19
Server version: 8.0.30 MySQL Community Server - GPL

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> ALTER USER "root"@"localhost" IDENTIFIED WITH "mysql_native_password" BY "Toscane2000**";
Query OK, 0 rows affected, 1 warning (0.06 sec)

mysql>
```

5.2 Lancer à nouveau la commande

- Npm start
- Retourne un tableau d'objets JavaScript

```
[
  RowDataPacket {
    id: 1,
    description: 'HP portatif',
    image: 'portable1.jpg',
    prix: 350.95,
    details: 'MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir'
  },
  RowDataPacket {
    id: 2,
    description: 'Portatif Macbook Pro MF839LA',
    image: 'portable2.jpg',
    prix: 1549.95,
    details: 'MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir'
  },
  RowDataPacket {
    id: 3,
    description: 'Portatif Macbook Air MJVE2C',
    image: 'portable3.jpg',
    prix: 1990.95,
    details: 'MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir'
  },
  RowDataPacket {
    id: 4,
    description: 'MSI Portatif de jeu Leopard',
    image: 'portable4.jpg',
    prix: 1099.95,
    details: 'MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir'
  },
  RowDataPacket {
    id: 5,
    description: 'HP portatif 001',
    image: 'portable1.jpg',
    prix: 350.95,
    details: 'MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir'
  },
  RowDataPacket {
    id: 6,
```

Étape 6 – Écrire la fonction permettant de récupérer tous les produits dans app.js

```
app.get("/", function(request, response){
  response.send("Bonjour tout le monde<br/>Utilisation du serveur Express")
});
app.listen(3000);

app.get("/getAllProducts", function(request, response){
  const con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Toscane2000**",
    database: "gestionProduit"
  });

  con.connect(function(err) {
    if (err) throw err;
    con.query("SELECT * FROM Produit", function (err, result, fields) {
      if (err) throw err;
      console.log(JSON.stringify(result));
      response.status(200).json(result);
    });
  });
});
```

Tester dans le navigateur avec
<http://localhost:3000/getAllProducts>

Étape 6 – Écrire la fonction permettant de récupérer tous les produits dans app.js

```
app.get("/", function(request, response){
  response.send("Bonjour tout le monde<br/>Utilisation du serveur Express")
});
app.listen(3000);
```

```
app.get("/getAllProducts", function(request, response){
  const con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Toscane2000**",
    database: "gestionProduit"
  });

  con.connect(function(err) {
    if (err) throw err;
    con.query("SELECT * FROM Produit", function (err, result, fields) {
      if (err) throw err;
      console.log(JSON.stringify(result));
      response.status(200).json(result);
    });
  });
});
```

Tester dans le navigateur avec
<http://localhost:3000/getAllProducts>

Étape 6.1 Résultat – Test Api PostMan

Overview

GET http://localhost:3000/getAllProducts

GET http://localhost:3000/getAllProducts

POST http://localhost:3000/getAllProducts

PUT http://localhost:3000/getAllProducts

DEL http://localhost:3000/getAllProducts

+ ...

No Environment

http://localhost:3000/getAllProducts

Save

GET

http://localhost:3000/getAllProducts

Send

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

Cookies

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (8)

Test Results

200 OK

181 ms

3.92 KB

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  [  
2    {  
3      "id": 1,  
4      "description": "HP portatif",  
5      "image": "portable1.jpg",  
6      "prix": 350.95,  
7      "details": "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"  
8    },  
9    {  
10     "id": 2,  
11     "description": "Portatif Macbook Pro MF839LA",  
12     "image": "portable2.jpg",  
13     "prix": 1549.95,
```

File/New/http Request

Étape 7- Écrire la fonction permettant de récupérer un produit dont id est donné

```
app.get("/getProduct/:id", function(request, response){
  const id = request.params.id;
  const con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Toscane2000**",
    database: "gestionProduit"
  });
  con.connect(function(err) {
    if (err) throw err;
    con.query("SELECT * FROM Produit where id=" + id, function (err, result, fields) {
      if (err) throw err;
      if(result.length > 0){
        console.log(JSON.stringify(result[0]));
        response.status(200).json({
          message : "produit(s) trouvé(s)",
          data : result[0]
        });
      }
      else{
        console.log("Produit non trouvé");
        response.status(200).json({
          message : "Aucun produit trouvé",
          data : {}
        });
      }
    });
  });
});
```

Test avec:
<http://localhost:3000/getProduct/5>

Étape 7.1 – Test avec PostMan (produit existant)

http://localhost:3000/getProduct/8

Save

GET http://localhost:3000/getProduct/8 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Headers 6 hidden

KEY	VALUE	DESCRIPTION	Bulk Edit	Presets
-----	-------	-------------	-----------	---------

Body Cookies Headers (8) Test Results

200 OK 15 ms 501 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "produit(s) trouvé(s)",
3   "data": {
4     "id": 8,
5     "description": "MSI Portatif de jeu Leopard 001",
6     "image": "portable4.jpg",
7     "prix": 1099.95,
8     "details": "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"
9   }
10 }
```

Étape 7.1 – Test avec PostMan (produit non existant)

The screenshot shows the Postman interface for a GET request to `http://localhost:3000/getProduct/50`. The request is saved and has 6 hidden headers. The response is a 200 OK status with a 36 ms response time and 312 B of data. The response body is displayed in JSON format, showing a message and an empty data object.

Request URL: `http://localhost:3000/getProduct/50`

Method: GET

Headers (6):

KEY	VALUE	DESCRIPTION

Response:

Status: 200 OK, 36 ms, 312 B

Body (JSON):

```
{
  "message": "Aucun produit trouvé",
  "data": {}
}
```

Étape 8 - Ajouter cors pour tester les requêtes de n'importe quelle source

- Npm install cors

```
CA: Invite de commandes

C:\NodeJS\ServeurExo1>npm install cors

added 2 packages, and audited 116 packages in 1s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\NodeJS\ServeurExo1>
```

```
JS app.js  JS DBManager.js  JS script1.js  <> exercice1.ejs

ServeurExo1 > JS app.js > ...

1  const express = require("express");
2  const mysql = require("mysql");
3  const cors = require('cors');
4
5  const app = express();
6  const port = 3000;
7  // enabling CORS for all requests
8  app.use(cors());
9
10 app.get("/", function(request, response){
11     response.send("Bonjour tout le monde<br/>Utilisation du serveur Express")
12 });
13 app.listen(3000);
14
15 const con = mysql.createConnection({
16     host: "localhost",
17     user: "root",
18     password: "Toscane2000**",
19     database: "gestionProduit"
20 });
21
22 app.get("/getAllProducts", function(request, response){
23     con.connect(function(err) {
24         if (err) throw err;
25         con.query("SELECT * FROM Produit", function (err, result, fields) {
26             if (err) throw err;
27             response.send(JSON.stringify(result));
28         });
29     });
30 });
```

Étape 9 - Ajouter body-parser les objets JSON

- Npm install body-parser

```
Invite de commandes

C:\NodeJS\ServeurExo1>npm install body-parser

up to date, audited 116 packages in 1s

12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\NodeJS\ServeurExo1>
```

```
app.js  exercise1.html  JS script1.js Exercice1  JS DBManager.js  JS script1.js ServeurExo1

ServeurExo1 > JS app.js > app.get("/getAllProducts") callback > con

1  const express = require("express");
2  const mysql = require("mysql");
3  const cors = require('cors');
4  const bodyParser = require('body-parser');
5
6  const app = express();
7  const port = 3000;
8  // enabling CORS for all requests
9  app.use(cors());
10 // using bodyParser to parse JSON bodies into JS objects
11 app.use(bodyParser.json());
12
13 app.get("/", function(request, response){
14   response.send("Bonjour tout le monde<br/>Utilisation du serveur Express")
15 });
16 app.listen(3000);
```

Étape 10 – Écrire une fonction permettant d'ajouter un produit

```
✓ app.post("/createProduct", function(request, response){  
  const produit = request.body;  
  console.log(produit.description + " " + produit.image + " " + produit.prix + " " + produit.details)  
  console.log(JSON.stringify(produit));  
  
  ✓ const con = mysql.createConnection({  
    host: "localhost",  
    user: "root",  
    password: "Toscane2000**",  
    database: "gestionProduit"  
  });  
  
  ✓ con.connect(function(err) {  
    if (err) throw err;  
    ✓ con.query("INSERT INTO Produit values(null, '" + produit.description + "', '" + produit.image +  
    ✓ ' ', " + produit.prix + ", '" + produit.details + "')";", function (err, result, fields) {  
      if (err) throw err;  
  
      response.status(200).send("Produit ajouté");  
    });  
  });  
});
```

Étape 10.1 – Tester avec PostMan

À vérifier aussi
dans MySQL

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/createProduct`. The request body is a JSON object with the following fields:

```
1 {  
2   "description": "Portatif Macbook Pro MF839LA 001",  
3   "image": "portable1.jpg",  
4   "prix": 1549.95,  
5   "details": "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"  
6 }
```

The response is a 200 OK status with a response time of 202 ms and a size of 274 B. The response body is `Produit ajouté`.

Étape 11 – Ajouter la méthode pour modifier un produit

```
app.put("/updateProduct/:id", function(request, response){
  const id = request.params.id;
  const produit = request.body;
  console.log(produit.description + " " + produit.image + " " + produit.prix + " " + produit.details)
  console.log(JSON.stringify(produit));
  const con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Toscane2000**",
    database: "gestionProduit"
  });

  con.connect(function(err) {
    if (err) throw err;
    con.query("update Produit set description = '" + produit.description + "', image ='" + produit.image +
      "', prix ='" + produit.prix + "', details='" + produit.details + "' where id=" + id, function (err, result, fields) {
        if (err) throw err;
        response.status(200).send("Produit modifié");
      });
  });
});
```

Étape 11.1 – Tester avec PostMan

À vérifier aussi
dans MySQL

The screenshot shows the Postman interface for a PUT request. The URL bar at the top displays `http://localhost:3000/updateProduct/6`. Below the URL bar, the request method is set to **PUT**. The request body is configured as **raw** with the **JSON** format selected. The body content is a JSON object with the following fields:

```
1 {  
2   "description": "Un changement",  
3   "image": "portable1.jpg",  
4   "prix": 1549.95,  
5   "details": "MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir"  
6 }
```

Below the request body, the response section shows a status of **200 OK** with a response time of **33 ms** and a size of **276 B**. The response body is displayed in the **Pretty** format, showing the text `1 Produit modifié`.

Étape 12 – Ajouter la méthode pour supprimer un produit

```
app.delete("/deleteProduct/:id", function(request, response){
  const id = request.params.id;
  const con = mysql.createConnection({
    host: "localhost",
    user: "root",
    password: "Toscane2000**",
    database: "gestionProduit"
  });

  con.connect(function(err) {
    if (err) throw err;
    con.query("DELETE FROM Produit where id=" + id, function (err, result, fields) {
      if (err) throw err;
      response.status(200).send("Produit supprimé");
    });
  });
});
```

Étape 12.1 – Tester avec PostMan

À vérifier aussi
dans MySQL

http://localhost:3000/deleteProduct/20

Save

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (8) Test Results

200 OK 183 ms 277 B Save Response

Pretty Raw Preview Visualize HTML

1 Produit supprimé

Tester avec un client html+CSS+JavaScript

- Soit le fichier de base pour ce laboratoire contenant deux exercices
- Dans le dossier Exercice1 se trouve le HTML et une partie Javascript pour construire les pages.
- Modifier les pages pour appeler les requêtes

Étape 13 - Ajout du code dans la fonction listeProduit() qui existe déjà






```
function listerProduit() {  
    $("#zoneModificationProduit").hide();  
    $("#zoneSuppressionProduit").hide();  
    $("#zoneNouveauProduit").hide();  
    $("#zoneListeProduit").show();  
  
    var noeudtbody = $("#idCorpsTableau");  
    $.ajax({  
        url: "http://localhost:3000/getAllProducts",  
        method: "get",  
        dataType: "json",  
        success: function (data) {  
            $.each(data, function (index, produit) {  
                var description = produit.description;  
                var image = produit.image;  
                var prix = produit.prix;  
                var details = produit.details;  
                var id = produit.id;  
  
                var noeudTr = $("<tr></tr>");  
                var noeudTd1 = $("<td></td>");  
                $(noeudTd1).text(id);  
                $(noeudTr).append(noeudTd1);  
  
                var noeudTd2 = $("<td></td>");  
                $(noeudTd2).text(description);  
                $(noeudTr).append(noeudTd2);
```

```
                var noeudTd3 = $("<td></td>");  
                var noeudTd31 = $("<img/>");  
                $(noeudTd31).attr({ src: image, width: "50%" });  
                $(noeudTd3).append(noeudTd31);  
                $(noeudTr).append(noeudTd3);  
  
                var noeudTd4 = $("<td></td>");  
                $(noeudTd4).text(prix);  
                $(noeudTr).append(noeudTd4);  
  
                var noeudTd5 = $("<td></td>");  
                $(noeudTd5).text(details);  
                $(noeudTr).append(noeudTd5);  
  
                $(noeudtbody).append(noeudTr);  
            });  
        },  
        error: function (error) {  
            alert("Erreur " + error);  
        }  
    });  
}
```

Étape 13.1 Test de liste de produit

Nouveau Produit Modification de Produit Suppression de Produit **Liste Produit**

Liste de produit

Id	Description	Image	Prix	Details
1	HP portatif		350.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
2	Portatif Macbook Pro MF839LA		1549.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
3	Portatif Macbook Air MJVE2C		1990.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
4	MSI Portatif de jeu Leopard		1099.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir
5	HP portatif 001		350.95	MSI 15.6 po GF65 9SEXR-436p, Intel Core i7-9750H, 16GB RAM, 512GB SSD, Win10, Noir

À tester la fonctionnalité

Étape 14 – Ajout du code dans le javascript pour rechercher des produits (Modification)

```
function rechercherModification() {  
    var idProduit = $("#txtIdM").val()  
    $.ajax({  
        url: "http://localhost:3000/getProduct/" + idProduit,  
        method: "get",  
        dataType: "json",  
        success: function (data) {  
            if(data.message != "produit(s) trouvé(s)"){  
                alert(data.message)  
            }  
            else{  
                var reponse = data.data;  
                $("#txtDescriptionM").val(reponse.description);  
                $("#txtImageM").val(reponse.image);  
                $("#txtPrixM").val(reponse.prix);  
                $("#txtDetailM").val(reponse.details);  
            }  
        },  
        error: function (error) {  
            alert("Erreur" + error);  
        }  
    });  
}
```


Étape 14.1 – Ajout du code dans le javascript pour rechercher des produits (Suppression)

```
function rechercher() {  
    var idProduit = $("#txtIdS").val();  
    $.ajax({  
        url: "http://localhost:3000/getProduct/" + idProduit,  
        method: "get",  
        dataType: "json",  
        success: function (data) {  
            if(data.message != "produit(s) trouvé(s)"){  
                alert(data.message)  
            }  
            else{  
                var reponse = data.data;  
                $("#txtDescriptionM").val(reponse.description);  
                $("#txtImageM").val(reponse.image);  
                $("#txtPrixM").val(reponse.prix);  
                $("#txtDetailM").val(reponse.details);  
            }  
        },  
        error: function (error) {  
            alert("Erreur" + error);  
        }  
    });  
}
```

14.2 Tester la modification pour la recherche de produit

[Nouveau Produit](#) [Modification de Produit](#) [Suppression de Produit](#) [Liste Produit](#)

Modification de produit

ID du produit [Rechercher](#)

Modification souhaitée

Description

Image

Prix

Details

[Modifier](#)

**A tester aussi la recherche
dans la suppression**

[Nouveau Produit](#) [Modification de Produit](#) [Suppression de Produit](#) [Liste Produit](#)

Modification de produit

ID du produit [Rechercher](#)

Modification souhaitée

Description

Image

Prix

Details

[Modifier](#)

Étape 15- À ajouter le code pour la création

```
function enregistrer() {  
    var descriptionProduit = $("#txtDescriptionN").val();  
    var imageProduit = $("#txtImageN").val();  
    var prixProduit = parseFloat($("#txtPrixN").val());  
    var detailProduit = $("#txtDetailN").val();  
    alert(descriptionProduit + " " + imageProduit + " " + prixProduit + " " + detailProduit)  
    $.ajax({  
        contentType: "application/json; charset=utf-8",  
        processData: false,  
        url: "http://localhost:3000/createProduct",  
        data: JSON.stringify( {  
            description: descriptionProduit,  
            image: imageProduit,  
            prix: prixProduit,  
            details: detailProduit  
        } ),  
        method: "post",  
        dataType: "json",  
        success: function (data) {  
            $("#txtDescriptionN").val("");  
            $("#txtImageN").val("");  
            $("#txtPrixN").val("");  
            $("#txtDetailN").val("");  
            alert("Produit enregistré");  
        },  
        error: function (error) {  
            alert("Erreur " + error);  
        }  
    });  
}
```

Étape 16- À ajouter le code pour la modification

```
function modifier() {  
    var idProduit = $("#txtIdM").val();  
    var descriptionProduit = $("#txtDescriptionM").val();  
    var imageProduit = $("#txtImageM").val();  
    var prixProduit = parseFloat($("#txtPrixM").val());  
    var detailProduit = $("#txtDetailM").val();  
    $.ajax({  
        url: "http://localhost:3000/updateProduct/" + idProduit,  
        contentType: "application/json; charset=utf-8",  
        processData: false,  
        data: JSON.stringify({  
            description: descriptionProduit,  
            image: imageProduit,  
            prix: prixProduit,  
            details: detailProduit  
        }  
    )  
    },  
    method: "put",  
    dataType: "json",  
    success: function (data) {  
        $("#txtIdM").val("");  
        $("#txtDescriptionM").val("");  
        $("#txtImageM").val("");  
        $("#txtPrixM").val("");  
        $("#txtDetailM").val("");  
        alert("Produit modifié");  
    },  
    error: function (error) {  
        alert("Erreur " + error);  
    }  
    }  
});  
}
```

Étape 17- À ajouter le code pour la suppression

```
function supprimer() {  
    var idProduit = $("#txtIdS").val();  
    if (confirm("Êtes-vous sûr de vouloir supprimer le produit de code " + idProduit)) {  
        $.ajax({  
            url: "http://localhost:3000/deleteProduct/" + idProduit,  
            method: "delete",  
            dataType: "json",  
            success: function (data) {  
                $("#txtDescriptionS").val("");  
                $("#txtImageS").val("");  
                $("#txtPrixS").val("");  
                $("#txtDetailS").val("");  
                alert("Confirmation : " + data);  
            },  
            error: function (error) {  
                alert("Erreur " + error);  
            }  
        });  
    } else {  
        alert("Suppression annulée");  
    }  
}
```

Exercice 2

- À modifier le code d'appel ajax pour faire fonctionner l'exercice 2 avec la fonction `getAllProducts` du service Web.

Exercice 3

- À refaire l'exercice 1 en utilisant les modules pour les connexions à la base de données et l'utilisation d'une classe Produit.