

---

# Laboratoire 5

## Développement framework Côté serveur

Procédure d'intégration Node.js et MySQL

**420-FCS-TT**  
**420-4FS-TT**

2021-09-08 11:56

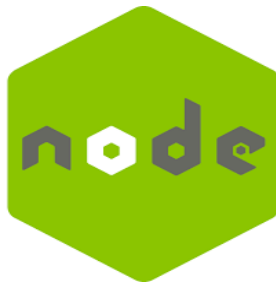
## ***Objectifs du laboratoire***

- Interconnecter **Node.js** et **MySQL**
- Envoyer une requête **SQL** et afficher les données avec EJS

## ***Outils nécessaires :***

Logiciel d'édition *Visual Studio Code*  
PHPMyAdmin (Xampp)

---



## **Objectif: être capable d'intégrer ses compétences en programmation.**

*Maintenant que vous avez réussi le cours d'introduction aux bases de données, vous devrez intégrer vos compétences en mettant en relation MySQL et Node.js.*

### **Étape 1**

#### **Création de la base de données**

*Si vous ne l'avez pas déjà dans PHPMYAdmin (voir le cours 420-CBD-BDA-TT), vous devrez créer votre base de données.*

1. Utilisez la ligne de commande ou PHPMYAdmin et respectez les consignes suivantes :
  - Nom de la base : **ecole**.
  - Nom de la table : **etudiant**.
  - Champs et contenu de départ :

<b>code</b>	<b>Nom</b>	<b>prenom</b>	<b>photo</b>
GuaLu	Guay	Luc	GuaLu.jpg
PauFr	Paul	Francis	PauFr.jpg

## Étape 2

### Préparation du pseudo code

Toute l'information est ici :

[https://www.w3schools.com/nodejs/nodejs\\_mysql.asp](https://www.w3schools.com/nodejs/nodejs_mysql.asp)

Préparez-vous un pseudocode des étapes à réaliser pour se connecter à MySQL

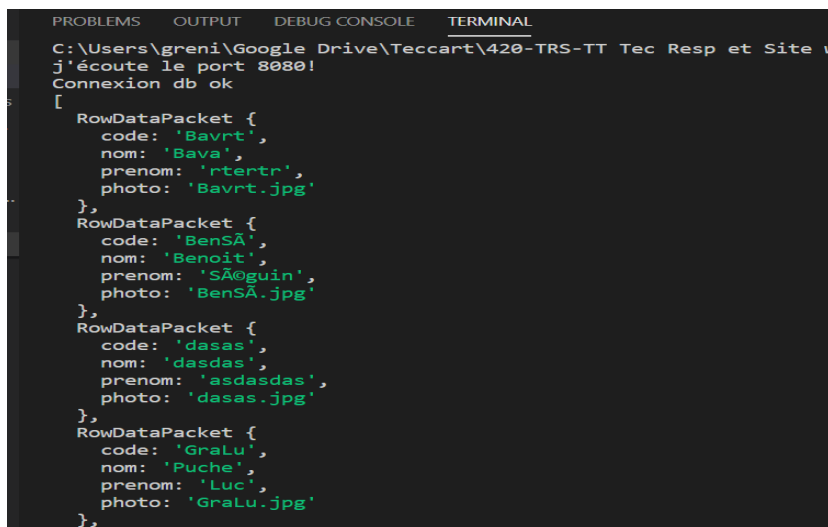
## Étape 3

### Connexion à la base de données

Nous devons nous connecter à la base de données avant d'envoyer les requêtes.

1. Créez un nouveau projet **Node.js** fonctionnel avec comme fichier principal, **labo5.js**.
2. Installez le module **mysql** avec les commandes adéquates.
3. Adaptez le fichier **labo5.js** pour utiliser le module.
4. Assurez-vous de bien vous connecter à la base de données et d'envoyer **tous les contenus** des champs dans la console (directement dans le code principal, pas dans une route, pour l'instant).
5. Vérifiez le résultat dans la console lors du chargement de la page.

Voici un exemple :



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
C:\Users\greni\Google Drive\Teccart\420-TRS-TT Tec Resp et Site w
j'écoute le port 8080!
Connexion db ok
[
  RowDataPacket {
    code: 'Bavrt',
    nom: 'Bava',
    prenom: 'rtertr',
    photo: 'Bavrt.jpg'
  },
  RowDataPacket {
    code: 'BenSA',
    nom: 'Benoit',
    prenom: 'SÃ@guin',
    photo: 'BenSA.jpg'
  },
  RowDataPacket {
    code: 'dasas',
    nom: 'dasdas',
    prenom: 'asdasdas',
    photo: 'dasas.jpg'
  },
  RowDataPacket {
    code: 'GraLu',
    nom: 'Puche',
    prenom: 'Luc',
    photo: 'GraLu.jpg'
  },
  RowDataPacket {
    code: 'GraLu',
    nom: 'Puche',
    prenom: 'Luc',
    photo: 'GraLu.jpg'
  }
]
```

## Étape 4

### Modification de la connexion à la base de données

1. Veuillez maintenant mettre la requête **SELECT** en commentaire et ne garder que la connexion à la base de données avec une confirmation console, côté serveur :

#### Example

Select all records from the "customers" table, and display the result object:

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

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

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

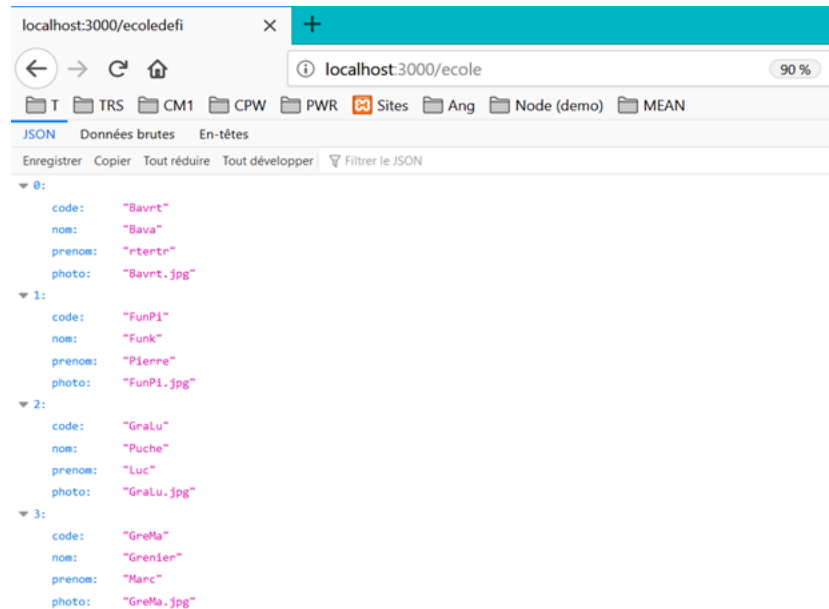
```
4  var mysql = require('mysql');
5  var con = mysql.createConnection({
6    host: "localhost",
7    user: "root",
8    password: "",
9    database: "ecole"
10  });
11  con.connect(function(err) {
12    if (err) throw err;
13    /* con.query("SELECT * FROM etudiant", function (err, result, fields) {
14      if (err) throw err;
15      console.log(result);
16    }); */
17
18    console.log("Connexion db ok");
19  });
20
21
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

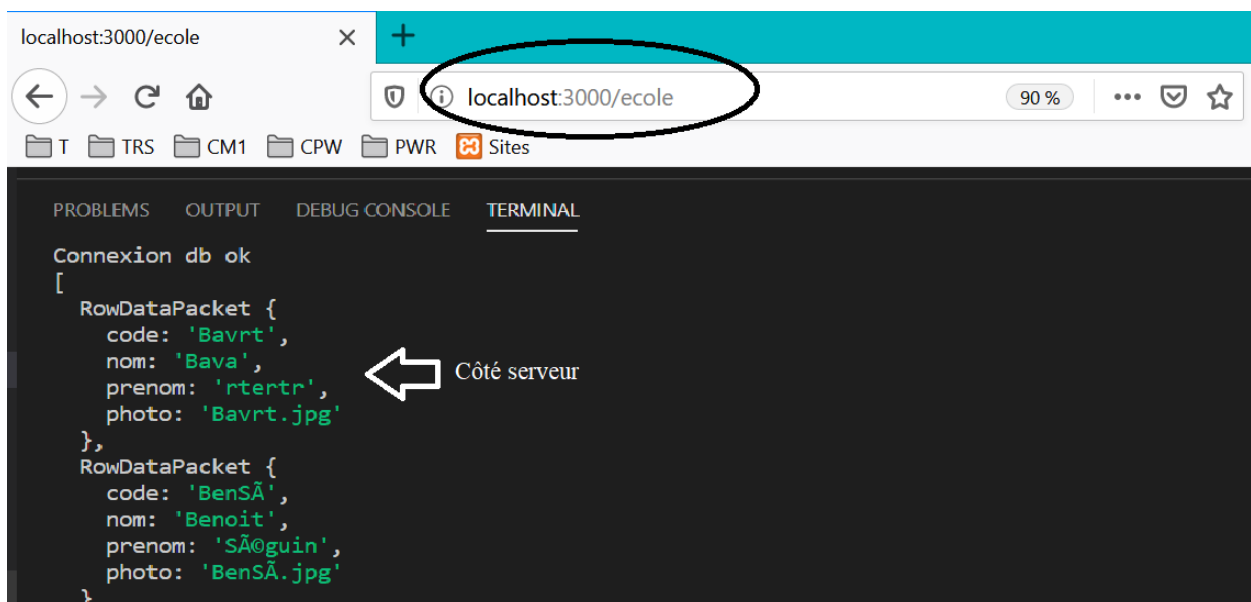
^C  
C:\Users\greni\Google Drive\Teccart\420-TRS-TT Tec Resp et Site web\Contenu cours\Labo  
j'écoute le port 3000!  
Connexion db ok  
]

2. Récupérez les lignes mises en commentaires et ajoutez-les directement dans la route `/ecole` (retirez les commentaires évidemment).
3. Veuillez maintenant intégrer la requête **SELECT** dans la route `/ecole`.
4. Vous devez envoyer le contenu de la requête **SELECT** au client avec un **res.send** dans cette route (voir labo1, « bonjour le monde », au besoin).

**Voici le résultat côté client :**



2. Lors du chargement de la page `/ecole`, vous devriez toujours voir apparaître le contenu de la BD dans la console, côté serveur :



## Étape 5

### Passons les données à EJS du côté serveur !

Lors du labo précédent, EJS nous a permis d'afficher les données depuis un objet **JSON**. Voyons comment récupérer les données dans MySQL pour les transférer à EJS et que ce dernier procède à l'injection de ces dernières.

1. Ajoutez la route suivante dans monApp.js :

```
31  var obj = {};  
32  app.get('/ecole', function(req, res){  
33  
34      con.query('SELECT * FROM etudiant', function(err, result) {  
35  
36          if(err){  
37              throw err;  
38          } else {  
39              obj = {print: result};  
40              console.log(obj)  
41              res.render('ViewSQL', obj);  
42          }  
43      });  
44  });  
45  
46  });  
47
```

Voyons à quelles lignes il y a du nouveau :

**31** : Déclaration d'un objet JSON en vue d'y mettre les données.

**39** : Nous envoyons les données récupérées à la ligne 106 (**result**) dans l'objet nommé, **obj**.

**41** : Lors du **res.render**, nous passons **obj** à EJS pour qu'il puisse fabriquer la page **ViewSQL.ejs** en utilisant les données.

2. Nous devons donc maintenant créer la page **ViewSQL.ejs** et y prévoir les injections comme au laboratoire 4.

```
1 <!doctype html>
2 <body>
3   <table id="table" >
4     <thead>
5       <tr>
6         <th>Nom</th>
7         <th>prenom</th>
8       </tr>
9     </thead>
10    <tbody>
11      <% print.forEach(function (user) { %>
12        <tr>
13          <td><%= user.nom %></td>
14          <td><%= user.prenom %></td>
15        </tr>
16      <% }) %>
17    </tbody>
18  </table>
19 </body>
20
21 </html>
```

Remarquez qu'à la ligne 11, le **forEach** demande la propriété **print**, car celle-ci avait été ajoutée à la ligne 111 de **monApp.js** :

```
111 obj = {print: result};
```

Ajoutez un `console.log` juste après cette ligne

```
111 obj = {print: result};
112 console.log(obj)
```

Redémarrez le serveur et redemandez la page pour bien comprendre comment l'objet est bâti :

```
{
  print: [
    RowDataPacket {
      code: '12312',
      nom: '123',
      prenom: '123',
      photo: '12312.jpg'
    },
    RowDataPacket {
      code: 'AAAAA',
      nom: 'AAAAAAA',
      prenom: 'AAAAAAA',
      photo: 'AAAAA.jpg'
    },
    RowDataPacket {
      code: 'AAAbb',
      etc.
```

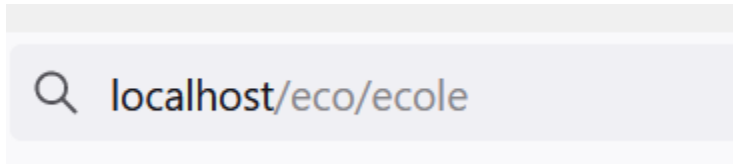


## Étape 6

### **Formatif : déplacer la route dans un fichier de routes**

*Pendant la réalisation de ce laboratoire, nous avons fait la route à même le fichier du serveur, **labo5.js**. Vous devrez maintenant modifier le laboratoire pour travailler les routes comme dans le laboratoire 4.*

1. Votre but est d'utiliser la route suivante pour accéder à la page école :



Voyez le : **/eco**

2. Copiez votre répertoire **labo4** et renommez-le **labo5Route**.
3. Ouvrez votre nouveau **labo5Route**, dans Visual Studio Code
4. Renommez **labo4.js** en **labo5Route.js**.
5. Ajoutez la page **ecole.ejs** le répertoire adéquat de ce projet.
6. Ajoutez au répertoire **Routes**, un fichier **ecoRoute.js**.
7. Faites les modifications nécessaires pour que la route fonctionne à partir de **ecoRoute.js**.