# TP CRUD JAVASCRIPT

1 Création de la VM Multipass 2 Installation de Node.js et MySQL dans la VM 3 Création de la base de données et de la table 4 Mise en place d'un projet Node.js + Express 5 Comprendre et coder les routes, middlewares et modèles 6 Tester avec Insomnia 7 Rédaction d'une documentation complète

**Etape 1: Création de la VM**

`multipass launch --name projet-api`

**Installation de Node.js et MySQL dans la VM**

Mettre à jour la vm: `sudo apt update && sudo apt upgrade -y sudo apt install -y nodejs` Verifier que tout est bien installé: `node -v npm -v` Install mysql: `sudo apt install mysql-server -y` Start: `sudo systemctl status mysql`

**Création de la base de données et de la table**

`sudo mysql`

```
CREATE DATABASE mini_api;
```

Création utilisateur pour la db:

```
CREATE USER 'apiuser'@'localhost' IDENTIFIED BY 'password123';
GRANT ALL PRIVILEGES ON mini_api.* TO 'apiuser'@'localhost';
FLUSH PRIVILEGES;
EXIT;
```

Pour se connecter à db: `sudo mysql -u apiuser -p` mdp = password123

Création table:

```
CREATE TABLE tasks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    completed BOOLEAN DEFAULT false
);
```

**Mise en place d'un projet Node.js + Express**

Qu'est-ce qu'Express ? Permet de:

- gérer des routes (GET, POST, PUT, DELETE),

- recevoir des requêtes HTTP,
- envoyer des réponses HTTP,
- et utiliser des middleware pour traiter les données (on verra ça juste après).

IP pour me connecter à ma vm: `172.28.163.171`

**Mise en place d'un projet Node.js + Express**

Installation `npm init -y` `npm install express mysql2` express = pour gérer ton serveur HTTP. mysql2 = pour connecter ton serveur Node.js à ta base MySQL dans la VM.

Création server.js Connexion à la db: `npm install mysql2`

mysql.createConnection({}) crée une connexion à ta base MySQL en lui donnant :

IP de la VM,

identifiants (user/password),

base de données.

connection.connect() tente de se connecter. Si la connexion réussit → affiche : message

**Comprendre et coder les routes, middlewares et modèles**

!! Middleware pour lire JSON `app.use(express.json());` Ajoutz avant toute les routes !

Structure d'une route:

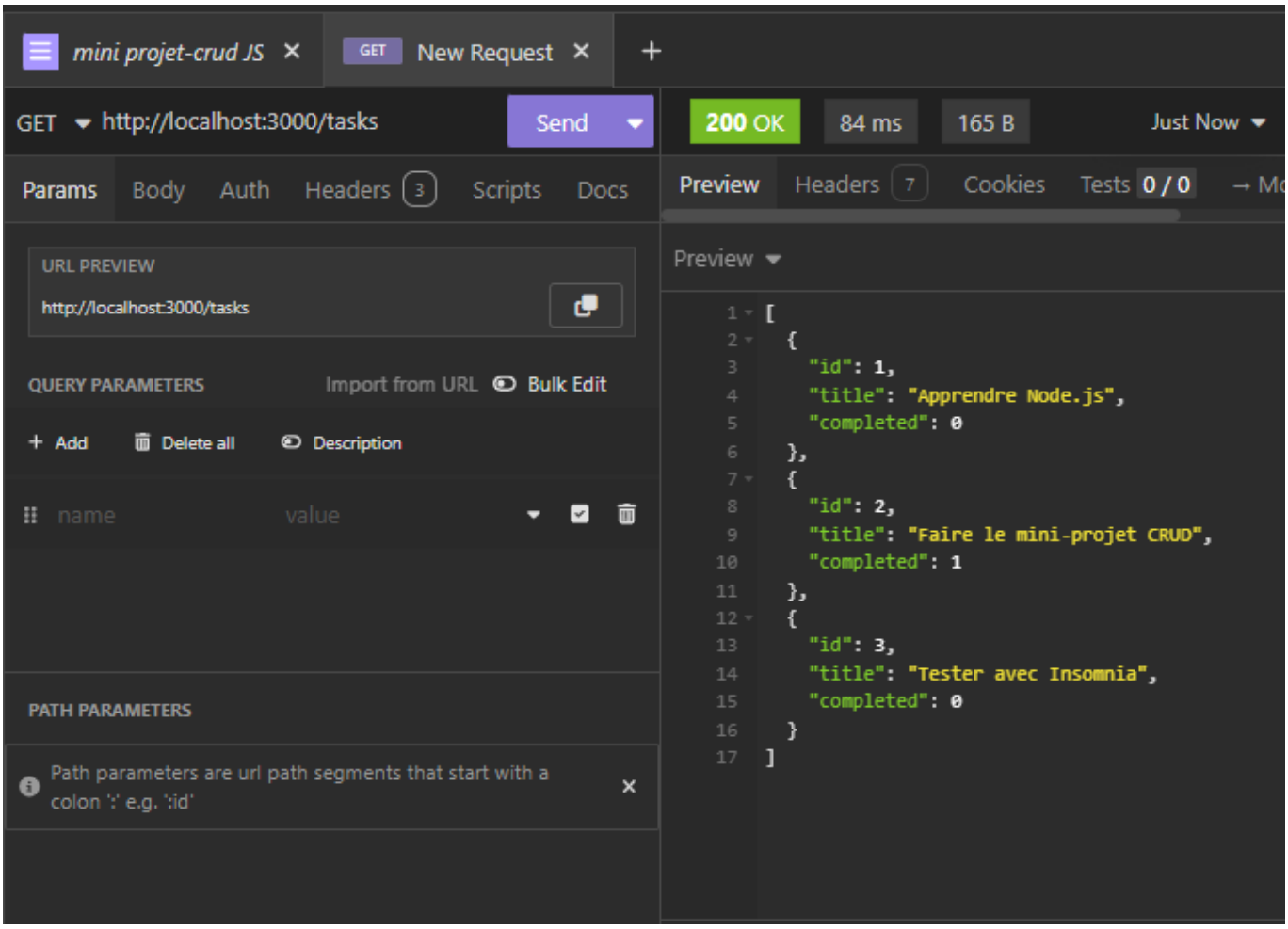
```
app.[METHODE]('/chemin', (req, res) => {
    // logique ici
    res.send('Réponse');
});
```

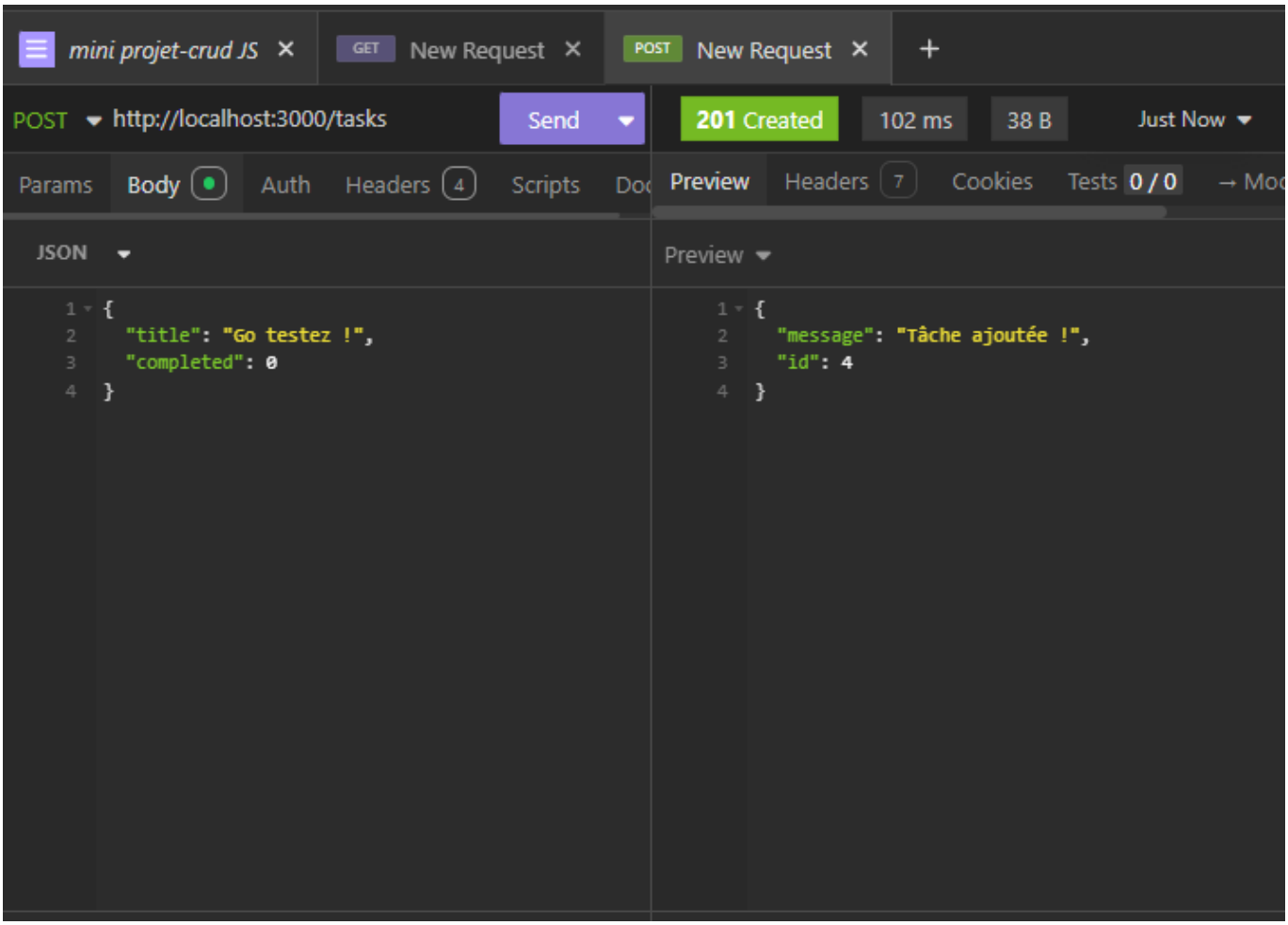
Passe l'id en param de la fonction `Tu récupères l'id avec req.params.id` `req.body` = récupère les données envoyées via Insomnia.

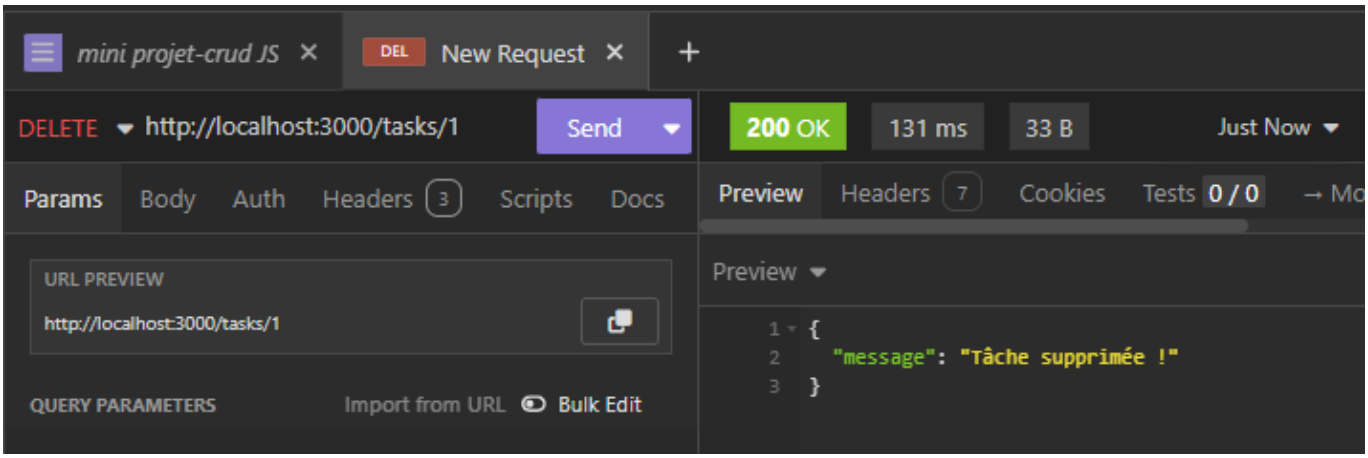
Élément Explication

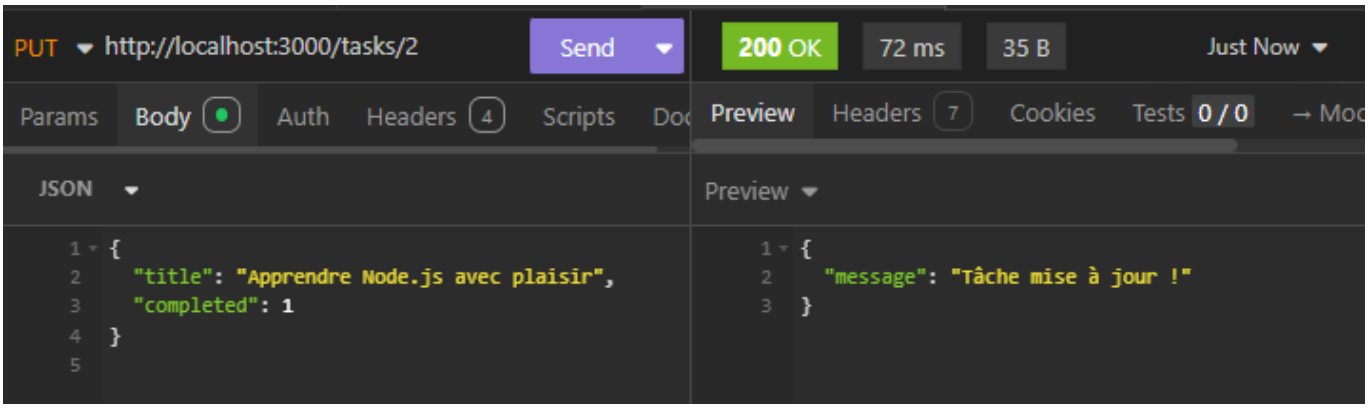
- `app` ton serveur Express.
- `[METHODE]` GET, POST, PUT, DELETE, etc.
- `'/chemin'` l'URL que tu souhaites écouter, ex: /tasks
- `(req, res)` req = ce que le client envoie, res = la réponse

Resultat:


mini projet-crud JS
GET New Request
GET http://localhost:3000/tasks
Send
200 OK
84 ms
165 B
Just Now
Params Body Auth Headers 3 Scripts Docs
Preview Headers 7 Cookies Tests 0 / 0
URL PREVIEW
http://localhost:3000/tasks
QUERY PARAMETERS
Import from URL
Bulk Edit
Add
Delete all
Description
name
value
PATH PARAMETERS
Path parameters are url path segments that start with a colon ':' e.g. ':id'
Preview
[
{
"id": 1,
"title": "Apprendre Node.js",
"completed": 0
},
{
"id": 2,
"title": "Faire le mini-projet CRUD",
"completed": 1
},
{
"id": 3,
"title": "Tester avec Insomnia",
"completed": 0
}
]


mini projet-crud JS
GET New Request
POST New Request
POST http://localhost:3000/tasks
Send
201 Created
102 ms
38 B
Just Now
Params Body Auth Headers 4 Scripts
Preview Headers 7 Cookies Tests 0 / 0
JSON
{
"title": "Go testez !",
"completed": 0
}
Preview
{
"message": "Tâche ajoutée !",
"id": 4
}


mini projet-crud JS
DEL
New Request
DELETE
http://localhost:3000/tasks/1
Send
200 OK
131 ms
33 B
Just Now
Params
Body
Auth
Headers 3
Scripts
Docs
Preview
Headers 7
Cookies
Tests 0 / 0
URL PREVIEW
http://localhost:3000/tasks/1
QUERY PARAMETERS
Import from URL
Bulk Edit
Preview
{
"message": "Tâche supprimée !"
}


PUT
http://localhost:3000/tasks/2
Send
200 OK
72 ms
35 B
Just Now
Params
Body
Auth
Headers 4
Scripts
Preview
Headers 7
Cookies
Tests 0 / 0
JSON
{
"title": "Apprendre Node.js avec plaisir",
"completed": 1
}
Preview
{
"message": "Tâche mise à jour !"
}

FIN