

coursNodeJs

04-API

Installations

PS C:\Workspaces\coursNodeJs\04-API> **npm init -y**

PS C:\Workspaces\coursNodeJs\04-API> npm install nodemon express cors pg

création: **.gitignore**

PS C:\Workspaces\coursNodeJs\04-API> **npm start**

Lancer l'extension Thunder NewRequest



```
1  const express = require('express');
2  const app = express();
3  const port = 3000;
4
5  app.get('/', (req, res) => {
6    res.send('Hello World!')
7  })
8
9  app.listen(port, () => {
10    console.log(`Example app listening on port ${port}`)
11  })
```

PS C:\Workspaces\coursNodeJs\04-API> **psql -U postgres -h localhost**

postgres=# \l

```
1  const {Client} = require('pg')
2
3  const client = new Client({
4    host: "localhost",
5    user: "postgres",
6    port: 5432,
7    password: "postgres",
8    database: "my_first_api"
9  })
10
11 module.exports = client
```

postgres=# **CREATE DATABASE my_first_api;**

postgres=# \c **my_first_api;** Vous êtes maintenant connecté à la base de données « my_first_api » en tant qu'utilisateur « postgres ».

Creation TABLE jeux

my_first_api=# CREATE TABLE jeux (my_first_api(# id SERIAL PRIMARY KEY, my_first_api(# name VARCHAR (50) NOT NULL, my_first_api(# release_date DATE, my_first_api(# console_id INT DEFAULT 1 my_first_api(#); CREATE TABLE

Creation du fichier de données : addGame.sql

```
1  INSERT INTO jeux ('name', 'release_date') VALUES
2  (`Tetris`, '1989-06-14'),
3  (`The Legend of Zelda: Link's Awakening`, '1993-06-06'),
4  (`Super Mario Land`, '1989-04-21'),
5  (`Pokémon Red Version`, '1996-09-28'),
```

Modification du fichier app pour le lier à body-parser

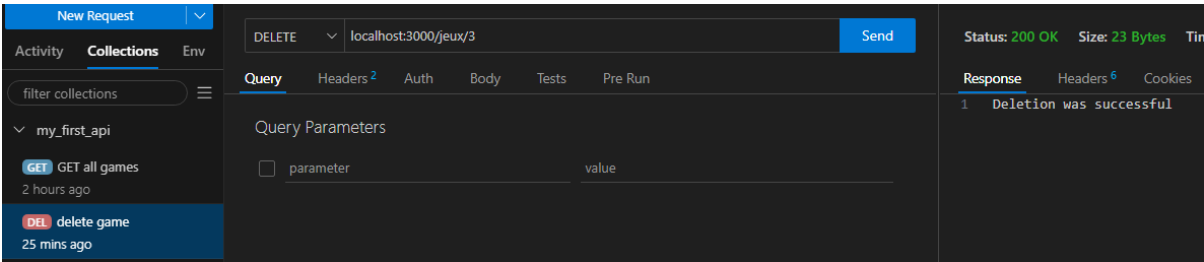
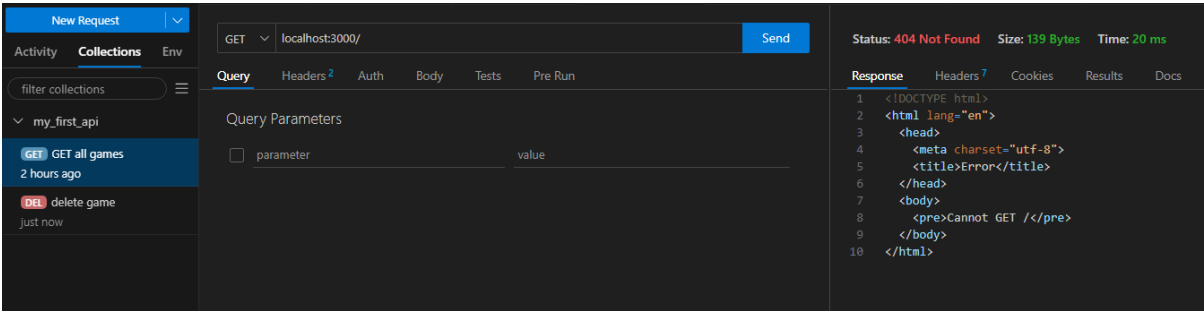
A screenshot of a code editor with a dark background and light-colored text. The code is written in JavaScript and uses ES6 syntax (const, let, arrow functions). It includes comments in French. The code is numbered from 1 to 21. The code defines a client, a body parser, an Express app, and a port. It sets up a GET route for '/jeux' that queries a database for all rows in the 'jeux' table. The app listens on port 3000 at localhost.

```
1  const client = require('./connection.js');
2  const bodyParser = require('body-parser');
3  const express = require('express');
4  const app = express();
5  const port = 3000;
6
7  app.use(bodyParser.json());
8
9  app.get('/jeux', (req, res) => {
10     client.query(`SELECT * FROM jeux`, (err, result)=>{
11         if(!err){
12             res.send(result.rows);
13         }
14     })
15     client.end;
16 })
17 client.connect();
18
19 app.listen(port, 'localhost', () => {
20     console.log(`Example app listening on port ${port}`)
21 })
```

Installation de body-parser pour afficher les données dans le fichier json

PS C:\Workspaces\coursNodeJs\04-API> **npm install body-parser;**

Gestion de THUNDER REQUEST pour afficher la table, supprimer une ligne...



Pour gérer les erreurs si le jeu n'existe pas :

```
1 // afficher un jeu
2 app.get('/jeux/:id', (req, res) => {
3   client.query('SELECT * FROM jeux WHERE id=${req.params.id}',
4     // conditions avec une TURNER (au lieu de if...)
5     (err, data) => !err ? !data.rows ? res.send('Game was not found') : res.send(data.rows[0]) : res.send(err.message));
6   // version longue avec if-else
7   // {
8   //   if (!err) {
9   //     if (data.rows.length === 0) {
10      //       res.send('Game was not found')
11      //     } else {
12      //       res.send(data.rows[0])
13      //     }
14      //   } else {
15      //     res.send(err.message)
16      //   }
17      // }
18   client.end;
19 })
```

```
1 // effacer un jeu
2 app.delete('/jeux/:id', (req, res) => {
3   let insertQuery = `delete from jeux where id=${req.params.id}`;
4
5   client.query(insertQuery, (err, result) => {
6     if (!err) {
7       if (result.rowCount === 0) {
8         res.send('Deletion was not found')
9       } else {
10        res.send('Deletion was successful')
11      }
12    }
13    else { console.log(err.message) }
14  })
15  client.end;
16 })
```

Insertion d'un nouveau jeu

```
1 //ajouter un jeu
2 app.post('/jeux', (req, res) => {
3   const jeu = req.body;
4   let insertQuery = `insert into jeux(name, release_date) values('${jeu.name}', '${jeu.release_date}')`;
5
6   client.query(insertQuery, (err, result) => {
7     if (!err) {
8       res.send('Insertion was successful')
9     }
10    else { console.log(err.message) }
11  })
12  client.end;
13 })
```

POST	localhost:3000/jeux	Send	Status: 200 OK	Size: 24 Bytes	Time: 111 ms
Query	Headers ²	Auth	Body ¹	Tests	Pre Run
JSON				XML	Text
Form				Form-encode	GraphQL
Binary					
JSON Content				Format	
<pre>1 { 2 "name": "toto", 3 "release_date": "2024-03-21" 4 }</pre>					
Response				Headers ⁶	Results
1				Insertion was successful	