APUNTES NODE.JS - Abel Rios - 20/05/2022
Colección de Postman: Node_SQL

Hoy vamos a conectar Node con MySQL. En la primera parte crearemos una base de datos, haremos un ejercicio medio simple para comenzar. En la segunda parte haremos un segundo ejercicio algo más complejo sobre lo mismo.

Antes de nada, creamos un nuevo servidor (carpeta ServerSQL), y una vez creado e instalado las librerías que usaremos vamos a **instalar el conector de MySQL**:

npm i mysql

Al inicio del index añadimos el require de mysql:

```
const express = require("express");
const app = express();
app.use(express.json());
const mysql = require("mysql");
```

** No olvidemos al final del archivo hacer la conexión con Express:

```
const PORT = 3001; //esto es el puerto del Express
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Ahora crearemos la conexión con:

- host: localhost / 127.0.0.1
- user: root / nombre de usuario
- password: contraseña que quieras
- database: base de datos con la que vamos a trabajar

```
// crear conexion a la base de datos
const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "Pentone12",
  database: "notas",
});
```

Y definimos las funciones conectar y desconectar:

```
function conectar() {
    // conectarse a la base de datos
    connection.connect(function (err) {
      if (err) {
        return console.error("error: " + err.message);
      }
      console.log("Conectado!");
    });
  }

  function desconectar() {
    // Cerrar conexion
    connection.end(function (err) {
      if (err) {
        return console.error("error: " + err.message);
      }
      console.log("Desconectado!");
    });
  }
```

Ahora vamos a crear una base de datos de prueba en SQL que se llame Notas. Con una tabla que se llame Cartas, que tendrá como datos: id, texto, idPersona. Y otra segunda tabla que se llame Personas con id y nombre. Aunque por ahora no las vamos a relacionar.

Hemos creado la base de datos 3 registros en la tabla Personas: María (id 1), Ramón (id 2) y Paco (id 3); y 6 registros en la tabla Cartas aleatorios:

id	texto	idPersona
1	Texto 1	1
2	Texto 2	2
3	Texto 3	3
4	Texto 4	1
6	Texto 5	2
7	Texto 6	1

Realizar un GET, un POST, un PUT y un DELETE con la base de datos Notas. Da igual si usamos params, querystring. A nuestra preferencia.

Para realizar las queries de SQL dentro de los endpoints, primero conectamos a la base de datos, luego realizamos la query y luego desconectamos de la base de datos.

```
// Esqueleto del endpoint con una query de SQL dentro
app.get("/", (request, response) => {
    conectar();
    connection.query(
        "sentencia",
        (err, rows, fields) => {
          if (err) throw err;
          response.json(rows);
        }
      );
    desconectar();
});
```

Donde “sentencia” es nuestra query de SQL.

**** OJO **** La variable “rows” que nos llega desde la base de datos viene como un array, así que si queremos quedarnos sólo con un dato, o con el primero o algo habría que hacer rows[0] o por el estilo.

- **GET de las cartas donde la idPersona es 1:**

```
// Haciendo un GET de la BBDD Notas, tabla Cartas

app.get("/cartasidpersona/:id", (request, response) => {
    conectar();
    connection.query(
        `SELECT * FROM cartas WHERE idPersona = ${request.params.id}`,
        (err, rows, fields) => {
          if (err) throw err;
          response.json(rows);
        }
      );
    desconectar();
});
```

y entramos a la url: http://localhost:3001/cartasidpersona/1

- **POST**

***OJO* Tiene que estar en la misma línea la query (imagen):**

```
49
50  app.post("/nuevacarta", (request, response) => {
51      conectar();
52      connection.query(
53          `INSERT INTO cartas(texto,idPersona) VALUES ('${request.body.text}','${request.body.idPersona}')`,
54          (err, rows, fields) => {
55            if (err) throw err;
56            response.json(rows);
57          }
58        );
59      desconectar();
60  });
61
```

* **Postman** request: introducirCarta

- **PUT**

(usando params + body a la vez)

```
62  // Haciendo un PUT (modificando el valor texto de una carta)
63  app.put("/modificarcarta/:id", (request, response) => {
64      conectar();
65      connection.query(
66          `UPDATE cartas SET texto = '${request.body.text}' WHERE id = ${request.params.id}`,
67          (err, rows, fields) => {
68            if (err) throw err;
69            response.json(rows);
70          }
71        );
72      desconectar();
73  });
```

*__Postman__ request: modificarCarta

- **DELETE**

```
76  // Haciendo el DELETE para eliminar una carta
77
78  app.delete("/borrarcarta/:id", (request, response) => {
79      conectar();
80      connection.query(
81          `DELETE FROM cartas WHERE id = ${request.params.id}`,
82          (err, rows, fields) => {
83            if (err) throw err;
84            response.json(rows);
85          }
86        );
87      desconectar();
88  });
```

EJERCICIO 2

Ahora vamos a hacer el mismo ejercicio con los datos de “Alumnos” que hicimos la semana pasada. Crear una nueva tabla en la base de datos (seguiré usando la base de datos “notas”) e insertar los alumnos que teníamos antes como un array en la tabla nueva.

1 - Sacar todos los alumnos que tienen 23 años

http://localhost:3001/api/getalumnos/23

2 - Sacar aquellas personas que tienen el sexo femenino y no tienen 23 años

http://localhost:3001/api/getsexodistintoedad/Femenino/23

3 - Introducir un alumno nuevo, si su DNI ya existe, entonces debe de sacar un mensaje de error y no crear el alumno