Coding Bootcamp

Sprint 2

Temario

Temario

- MySQL y Node.
- Escapar valores.
- Funcionalidades útiles.

MySQL y Node

MySQL y Node (1/3)

En la clase anterior hablamos sobre Bases de Datos, particularmente sobre bases de datos relacionales o SQL. Y además vimos un motor de base de datos llamado MySQL.

A continuación veremos cómo interactuar con MySQL desde una aplicación Node.

Lo primero que vamos a hacer es instalar un módulo (paquete) llamado `mysql` (Docs: https://github.com/mysqljs/mysql).

```
npm install mysql
```

MySQL y Node (2/3)

Luego hay que importar el módulo y crear una conexión a la BD:

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

const connection = mysql.createConnection({
  host: "localhost",
  user: "root",
  password: "root",
  database: "hack_academy_db",
});

connection.connect(function (err) {
  if (err) throw err;
  console.log("¡Nos conectamos a la BD!");
});
```

Recordar que para poder conectarnos a la BD no sólo debe estar instalado MySQL sino que también debe estar corriendo.

MySQL y Node (3/3)

Luego de conectarnos a la BD, podemos hacerle consultas:

```
connection.query("SELECT * FROM clients", function (err, clients) {
  if (err) throw err;
  console.log(JSON.stringify(clients));
});


connection.end();
```

Se pueden hacer llamadas adicionales a la BD usando una misma conexión. Lo importante es acordarse de cerrar la conexión luego usarla.

De hecho, se suele recomendar:

* Abrir una conexión lo más tarde posible.
* Cerrar una conexión lo más temprano posible.

Las conexiones suelen ser limitadas y costosas.

Escapar valores

Escapar valores (1/3)

⚠️ Hay que tener mucho cuidado a la hora de hacer consultas a una base de datos usando valores que vinieron, por ejemplo, de un formulario HTML.

Veamos el siguiente ejemplo.

Supongamos que recibimos por una URL el `id` de un usuario, por ejemplo, el número 7, para luego realizar esta consulta:

```
const userId = req.query.id; // Dato que vino de “afuera”.

const sql = "SELECT * FROM users WHERE user_id = " + userId;
```

A priori, parece un código inofensivo. La idea es obtener los datos del usuario número 7 y luego mostrarlos.

Escapar valores (2/3)

Supongamos ahora que en lugar de recibir el número 7, se recibe el string “38 OR 1=1”. Es decir:

```
const userId = req.query.id; // userId = "38 OR 1=1"
```

Con este simple cambio, ahora la consulta pasa a ser:

```
"SELECT * FROM users WHERE user_id = 38 OR 1=1"
```

“1=1” siempre es true, por lo cual ahora la consulta retorna el listado completo de usuarios. 😱

Esto es lo que se conoce como una **SQL Injection**.

Escapar valores (3/3)

Moraleja: nunca podemos confiarnos de los datos que nos llegan al Back-End.

Siempre hay que realizar validaciones y en el caso de tener que hacer una consulta SQL utilizando valores que llegaron “desde afuera” es necesario “escaparlos” previamente con el fin de evitar Inyecciones SQL.

Para eso, el módulo `mysql` trae un método llamado `escape`:

```
const userId = req.query.id; // Dato que vino de “afuera”.

const sql = "SELECT * FROM users WHERE user_id = " + connection.escape(userId);
```

Para escapar el nombre de una columna o base de datos, se utiliza el método `escapeId` en lugar de `escape`.

Funcionalidades útiles

Obtener el `id` de un registro insertado

```
const sql = `
  INSERT INTO users (firstname, lastname)
  VALUES ("María", "López")
`;

connection.query(sql, function (error, results) {
  if (error) throw error;
  console.log(results.insertId);
});
```

Para que esto funcione, la tabla `users` debe tener un campo auto-incremental, generalmente la columna `id`. El valor automático que se generó para dicho campo queda disponible en el atributo `insertId`.

Obtener cantidad de filas afectadas

```
const sql = `DELETE FROM users WHERE lastname = "Pérez"`;



connection.query(sql, function (error, results) {

  if (error) throw error;

  console.log(`Se borraron ${results.affectedRows} filas`);

});
```

`affectedRows` retorna el número de filas afectadas en un INSERT, UPDATE o DELETE.

Obtener cantidad de filas cambiadas

```
const sql = `
  UPDATE users
  SET firstname = "Pablo"
  WHERE lastname = "Gómez";
`;

connection.query(sql, function (error, results) {
  if (error) throw error;
  console.log(`Se actualizaron ${results.changedRows} filas`);
});
```

`changedRows` es distinto de `affectedRows` porque sólo cuenta las filas que efectivamente cambiaron. Por ejemplo, si ya había una fila con el usuario “Pablo Gómez”, esa fila no se cambió.