Practica 2

Nombre: Marcela Nadia Bonilla Mamani

Experimente con las 3 formas de conexión con la base de datos:

a) **Conexión básica**

Conexión básica: Se establece una conexión directa a la base de datos para cada operación.

Practica2 > mysql-basic-connection > JS app.js > ...

```
1   const mysql = require('mysql2');
2
3   const connection = mysql.createConnection({
4     host: 'localhost',
5     user: 'root',
6     password: 'mypassword',
7     database: 'testdb'
8   });
9   connection.connect((err) => {
10    if (err) throw err;
11    console.log('Connected to MySQL Database!');
12
13    connection.query('SELECT * FROM users', (err, results, fields) => {
14      if (err) throw err;
15      console.log(results);
16    });
17
18    connection.end();
19  });
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
]
PS D:\PROGRAMACION WEB 3\Practica2\mysql-basic-connection> node app.js
Connected to MySQL Database!
[
  { id: 1, name: 'John Doe', email: 'john.doe@example.com' },
  { id: 2, name: 'Jane Smith', email: 'jane.smith@example.com' },
  { id: 3, name: 'Alice Johnson', email: 'alice.johnson@example.com' },
  { id: 4, name: 'Bob Brown', email: 'bob.brown@example.com' },
  { id: 5, name: 'Charlie Davis', email: 'charlie.davis@example.com' },
  { id: 6, name: 'Eve White', email: 'eve.white@example.com' },
  { id: 7, name: 'Frank Black', email: 'frank.black@example.com' },
  { id: 8, name: 'Grace Green', email: 'grace.green@example.com' },
  { id: 9, name: 'Hank Blue', email: 'hank.blue@example.com' },
  { id: 10, name: 'Ivy Yellow', email: 'ivy.yellow@example.com' }
]
Conexión Básica: 43.563ms
```

TIEMPO DE EJECUCION: 43.563ms

b) **Conexión utilizando promesas.**

Conexión utilizando promesas: Se emplea el módulo `mysql2/promise` para manejar las operaciones de manera asincrónica

Practica2 > mysql-basic-connection > JS app1.js > main

```
1   const mysql = require('mysql2/promise');
2
3   async function main() {
4     try {
5       const connection = await mysql.createConnection({
6         host: 'localhost',
7         user: 'root',
8         password: 'mypassword',
9         database: 'testdb'
10      });
11
12      console.log('Connected to MySQL Database!');
13      // Execute a query using promise
14      const [rows, fields] = await connection.execute('SELECT * FROM users');
15      console.log('Query Result:', rows);
16
17      await connection.end();
18    } catch (err) {
19      console.error('Error:', err);
20    }
21  }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\PROGRAMACION WEB 3\Practica2\mysql-basic-connection> node app1.js
Connected to MySQL Database!
Query Result: [
  { id: 1, name: 'John Doe', email: 'john.doe@example.com' },
  { id: 2, name: 'Jane Smith', email: 'jane.smith@example.com' },
  { id: 3, name: 'Alice Johnson', email: 'alice.johnson@example.com' },
  { id: 4, name: 'Bob Brown', email: 'bob.brown@example.com' },
  { id: 5, name: 'Charlie Davis', email: 'charlie.davis@example.com' },
  { id: 6, name: 'Eve White', email: 'eve.white@example.com' },
  { id: 7, name: 'Frank Black', email: 'frank.black@example.com' },
  { id: 8, name: 'Grace Green', email: 'grace.green@example.com' },
  { id: 9, name: 'Hank Blue', email: 'hank.blue@example.com' },
  { id: 10, name: 'Ivy Yellow', email: 'ivy.yellow@example.com' }
]
Conexión con Promesas: 40.468ms
```

TIEMPO DE EJECUCION: 40.468ms

c) **Conexión utilizando Pooling.**

Conexión utilizando Pooling: Se utiliza un pool de conexiones que permite reutilizar conexiones existentes, mejorando la eficiencia en aplicaciones con múltiples solicitudes

Practica2 > mysql-basic-connection > JS app2.js > ...

```
1   const mysql = require('mysql2');
2
3   const pool = mysql.createPool({
4     host: 'localhost',
5     user: 'root',
6     password: 'mypassword',
7     database: 'testdb',
8     waitForConnections: true,
9     connectionLimit: 10,
10    queueLimit: 0
11  });
12
13  // Query the database using a pooled connection
14  pool.query('SELECT * FROM users', (err, results, fields) => {
15    if (err) throw err;
16    console.log(results);
17  });
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\PROGRAMACION WEB 3\Practica2\mysql-basic-connection> node app2.js
[
  { id: 1, name: 'John Doe', email: 'john.doe@example.com' },
  { id: 2, name: 'Jane Smith', email: 'jane.smith@example.com' },
  { id: 3, name: 'Alice Johnson', email: 'alice.johnson@example.com' },
  { id: 4, name: 'Bob Brown', email: 'bob.brown@example.com' },
  { id: 5, name: 'Charlie Davis', email: 'charlie.davis@example.com' },
  { id: 6, name: 'Eve White', email: 'eve.white@example.com' },
  { id: 7, name: 'Frank Black', email: 'frank.black@example.com' },
  { id: 8, name: 'Grace Green', email: 'grace.green@example.com' },
  { id: 9, name: 'Hank Blue', email: 'hank.blue@example.com' },
  { id: 10, name: 'Ivy Yellow', email: 'ivy.yellow@example.com' }
]
Conexión Pooling: 40.527ms
```

TIEMPO DE EJECUCION: 40.527ms

3. Proponga una aplicación utilizando las 3 formas de conexión, compare y comente los resultados obtenidos (mejor si es con cifras de tiempo de ejecución). Suba al GitHub esta aplicación incluyendo los resultados obtenidos.

Observaciones:

- La conexión básica establece y cierra una conexión por cada operación, lo que introduce una sobrecarga significativa en aplicaciones con alto volumen de solicitudes.
- La conexión utilizando promesas mejora la legibilidad y manejo de errores en el código, pero no presenta una mejora sustancial en el rendimiento en comparación con la conexión básica.
- La conexión utilizando Pooling demuestra ser la más eficiente, ya que reutiliza conexiones existentes, reduciendo el tiempo de establecimiento de nuevas conexiones y mejorando el rendimiento general de la aplicación.

Comparación de Resultados

Esta comparación se puede notar más al ingresar más datos

- Conexión Básica: Toma más tiempo debido a la apertura y cierre de conexiones en cada consulta.
- Conexión utilizando Promesas: Mejora la legibilidad y permite manejo asincrónico, pero el tiempo de ejecución no mejora sustancialmente respecto a la conexión básica.
- Conexión utilizando Pooling: Es la más eficiente ya que reutiliza conexiones, lo que reduce significativamente los tiempos de ejecución.