Guía de ejercicios - Trabajo práctico (I)

¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada: Consultando con parámetros

Desarrollar otra inserción para la tabla **ropa**. En esta ocasión agregaremos unos calcetines verdes talla M, pero a diferencia del ingreso que hemos hecho en otros ejercicios, esta inserción la haremos a través de una consulta parametrizada.

- ***Paso 1:*** Crea una función asíncrona con el nombre *ingresar*.
- ***Paso 2:*** Realizar una consulta con parámetros para agregar unos calcetines verdes de talla M.
- ***Paso 3:*** Imprimir por consola la inserción.
- ***Paso 4:*** Ejecuta la función *ingresar*.

```
// Paso 1
const ingresar = async () => {
    // Paso 2
    const res = await pool.query(
        "insert into ropa (nombre, color, talla) values ($1, $2, $3)
RETURNING *;",
        ["calcetines", "verde", "M"]
    );
    // Paso 3
    console.log(res.rows[0]);
}

// Paso 4
ingresar();
```

Ahora ejecuta tu aplicación y deberás ver algo como en la siguiente imagen:

```
$ node index.js
{ id: 2, nombre: 'calcetines', talla: 'M', color: 'verde' }
```

Fuente: Desafío Latam

Ahí lo tienes, tu primera consulta con texto parametrizado. Cabe destacar que esto ayuda al proceso para evitar una inyección SQL. No obstante, podrías notar que aún es posible realizar un ataque a la base de datos, puesto que cada valor de parámetro pudiese contener una sentencia SQL. Para evitar esto, utilizamos los objetos JSON como argumento de una consulta.

¡Manos a la obra! - 1

Construir un string de conexión con los siguientes datos:

- **Usuario:** usuario
- **Password:** 123456
- **Server:** 127.0.0.1
- **Port:** 5432
- **Database:** usuarios

¡Manos a la obra! - 2

Construir un objeto de configuración con los siguientes datos:

- **Usuario:** usuario
- **Password:** 123456
- **Server:** 127.0.0.1
- **Port:** 5432
- **Database:** usuarios

¡Manos a la obra! - 3

Abrir la terminal psql y crear una tabla **usuarios** con el siguiente código:

```
CREATE TABLE usuarios(
  id SERIAL PRIMARY KEY,
  nombre varchar(50) NOT NULL,
  telefono varchar(10) NOT NULL
);
```

Desarrollar una aplicación con Node que al ser ejecutada llame a una función asíncrona para ejecutar una consulta a PostgreSQL que ingrese 1 registro en la tabla **usuarios.**

¡Manos a la obra! - 4

Continuando con el ejercicio propuesto 3 donde creamos una tabla de **usuarios** con los campos: id, nombre y teléfono, desarrollar una aplicación con Node que al ser ejecutada realice en secuencia lo siguiente:

- Ingrese otro registro en la tabla **usuarios.**
- Consulte todos los registros de la tabla **usuarios.**
- Consulte solo el registro de id 1.

Considera manejar las funciones asíncronas como promesas ejecutando una tras otra con el método then().

¡Manos a la obra! - 5

Ahora se solicita desarrollar una aplicación con Node que al ser ejecutada realice una actualización a la tabla **usuarios** cambiando el número de teléfono al primer registro por: 914215468. Obtén por consola el registro modificado y la cantidad de registros afectados.

¡Manos a la obra! - 6

Desarrollar una aplicación con Node que al ser ejecutada realice una consulta SQL para eliminar todos los registros de la tabla **usuarios** e imprime por consola la cantidad de registros afectados.

¡Manos a la obra! - 7

Realizar una nueva inserción a la tabla **usuarios** con los siguientes datos:

- Id: 5
- Nombre: Jonathan
- Teléfono: 989786545

Realizar la consulta usando texto plano parametrizado y async/await.

¡Manos a la obra! - 8

Realizar una consulta a la tabla **usuarios** para obtener todos los registros, pero en esta ocasión, pasa un JSON como argumento a la consulta y define el Row Mode con el valor "array". Finalmente, imprime por consola los registros obtenidos.

¡Manos a la obra! - 9

Haz una consulta a la tabla **usuarios** que intente realizar una inserción con un registro que contenga un id existente en otro registro de la tabla.

Realizar la consulta con un JSON como argumento y capturar el error de la consulta con try catch e imprimiendo su código por consola.

Preguntas de proceso

Reflexiona:

- ¿Qué he aprendido hasta ahora?
- ¿Hay algo que me está dificultando mucho?

¡Continúa aprendiendo y practicando!

Solucionario

1. Construir un string de conexión con los siguientes datos:

    - **Usuario:** usuario
    - **Password:** 123456.
    - **Server:** 127.0.0.1
    - **Port:** 5432
    - **Database:** usuarios

```
'postgresql://usuario:123456@127.0.0.1:5432/usuarios
```

2. Construir un objeto de configuración con los siguientes datos:

    - **Usuario:** usuario
    - **Password:** 123456
    - **Server:** 127.0.0.1
    - **Port:** 5432
    - **Database:** usuarios

```
const config = {
   user: 'usuario',
   host: '127.0.0.1',
   database: 'usuarios',
   password: '123456',
   port: 5432,
}
```

3. Abrir la terminal psql y crear una tabla **usuarios** con el siguiente código:

```
CREATE TABLE usuarios(
  id SERIAL PRIMARY KEY,
  nombre varchar(50) NOT NULL,
  telefono varchar(10) NOT NULL
);
```

Desarrollar una aplicación con Node que al ser ejecutada llame a una función asíncrona para ejecutar una consulta a PostgreSQL que ingrese 1 registro en la tabla **usuarios.**

```
const ingresar = async () => {
```

```
  const res = await pool.query(
    "insert into usuarios (nombre, telefono) values ('Astrid',
'912345678') RETURNING *;"
  );
  console.log('Registro agregado', res.rows[0]);;
}

ingresar();
```

4. Continuando con el ejercicio propuesto 3 donde se creó una tabla de usuarios con los campos: id, nombre y teléfono. Desarrollar una aplicación con Node que al ser ejecutada realice en secuencia lo siguiente:

    - Ingrese otro registro en la tabla **usuarios**
    - Consulte todos los registros de la tabla **usuarios**
    - Consulte solo el registro de id 1

    Considera manejar las funciones asíncronas como promesas ejecutando una tras otra con el método then().

```
const ingresar = async () => {
  const res = await pool.query(
    "insert into usuarios (nombre, telefono) values ('Jocelyn',
'987654321') RETURNING *"
  );
  console.log("Registro agregado", res.rows[0]);
}

const consulta = async () => {
  const res = await pool.query("select * from usuarios");
  console.log("Registro: ", res.rows[0]);
}

const consultaId = async (id) => {
  const res = await pool.query(`select * from usuarios where id = ${id}
`);
  console.log(`Registro con el id: ${id}`, res.rows[0]);
}

ingresar()
  .then(() => consulta())
  .then(() => consultaId(1))
```

5. Ahora se solicita desarrollar una aplicación con Node que al ser ejecutada realice una actualización a la tabla **usuarios** cambiando el número de teléfono al primer registro por: 914215468. Obtén por consola el registro modificado y la cantidad de registros afectados.

```
const editar = async () => {
  const res = await pool.query(
    "UPDATE usuarios SET telefono = '914215468' WHERE id = 1 RETURNING
*;"
  );
  console.log('Registro modificado', res.rows[0]);
  console.log('Cantidad de registros afectados',res.rowCount);
}

editar();
```

6. Desarrollar una aplicación con Node que al ser ejecutada realice una consulta SQL para eliminar todos los registros de la tabla **usuarios** e imprime por consola la cantidad de registros afectados.

```
const eliminar = async () => {
  const res = await pool.query(
    "DELETE FROM usuarios"
  );
  console.log('Cantidad de registros afectados',res.rowCount);
}

eliminar();
```

7. Utilizar la tabla **usuario** que creaste anteriormente e ingresa por medio de la clase Pool un nuevo registro imprimiendo por consola la inserción de la consulta.

```
const ingresar = async () => {

  const res = await pool.query(
    "insert into usuarios (id, nombre, telefono) values (4, 'Brian',
'12345678') RETURNING *;"
  );

  console.log("Ultimo registro agregado: ", resul.rows[0]);
```

```
})
```

8. Realizar una nueva inserción a la tabla usuarios con los siguientes datos:
    - Id: 5
    - Nombre: Jonathan
    - Teléfono: 989786545

    Realizar la consulta usando texto plano parametrizado y async/await.

```
const ingresar = async () => {

  const res = await pool.query(
    "insert into usuarios (id, nombre, telefono) values ($1, $2, $3) RETURNING *;",
    [5, "Jonathan", "87654321"]
  );

  console.log("Ultimo registro agregado: ", res.rows[0]);

}
```

9. Realizar una consulta a la tabla **usuarios** para obtener todos los registros, pero en esta ocasión pasa un JSON como argumento a la consulta y define el Row Mode con el valor “array”. Imprimir por consola los registros obtenidos.

```
const consultar = async () => {
  const SQLQuery = {
    rowMode: "array",
    text:
      "SELECT * FROM usuarios",
  };

  const res = await pool.query(SQLQuery);

  console.log("Ultimo registro agregado: ", res.rows);

}
consultar()
```

10. Hacer una consulta a la tabla **usuarios** que intente realizar una inserción con un registro que contenga un id existente en otro registro de la tabla. Debes realizar la consulta con un JSON como argumento y capturar el error de la consulta con try catch e imprimiendo su código por consola.

```
const ingresar = async () => {
  const SQLQuery = {
    rowMode: "array",
    text:
      "insert into usuarios (id, nombre, telefono) values ($1, $2, $3)
RETURNING *;",
    values: [5, "Jonathan", "87654321"],
  };

  try {
    const res = await pool.query(SQLQuery);
    console.log("Ultimo registro agregado: ", res.rows);
  } catch (error) {
    console.log(error.code);
  }
}
ingresar()
```

Preguntas de cierre

- ¿Cómo manejarías la eliminación de todos los registros de la tabla usuarios en PostgreSQL y cuál sería la salida esperada en la consola?

- Después de realizar la actualización en la tabla usuarios cambiando el número de teléfono al primer registro, ¿cómo imprimirías por consola el registro modificado y la cantidad de registros afectados?

- Para la consulta a la tabla usuarios con JSON como argumento y definición de Row Mode como "array", ¿cómo imprimirías por consola los registros obtenidos? Además, en el caso de la consulta que intenta realizar una inserción con un id existente, ¿cómo capturarías el error y qué código imprimirías por consola?