

Guía de ejercicios - Implementación y gestión de una base de datos



¡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

¡Manos a la obra! - Ejercicio propuesto 1	2
¡Manos a la obra! - Ejercicio propuesto 2	2
¡Manos a la obra! - Ejercicio propuesto 3	2
¡Manos a la obra! - Ejercicio propuesto 4	3
¡Manos a la obra! - Ejercicio propuesto 5	3
¡Manos a la obra! - Ejercicio propuesto 6	3
¡Manos a la obra! - Ejercicio propuesto 7	4
¡Manos a la obra! - Ejercicio propuesto 8	4
¡Manos a la obra! - Ejercicio propuesto 9	4
Preguntas de proceso	5
Solucionario	5



¡Comencemos!



¡Manos a la obra! - Ejercicio propuesto 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! - Ejercicio propuesto 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! - Ejercicio propuesto 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! - Ejercicio propuesto 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! - Ejercicio propuesto 5

Continuando con el ejercicio propuesto 4 (de la lectura anterior), en el que empezamos a realizar consultas a la tabla **usuarios**. 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. Obten por consola el registro modificado y la cantidad de registros afectados.



¡Manos a la obra! - Ejercicio propuesto 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! - Ejercicio propuesto 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! - Ejercicio propuesto 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! - Ejercicio propuesto 9

Realizar 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.



¡Continúa aprendiendo y practicando!

Preguntas de proceso

Reflexiona:

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



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. Continuando con el ejercicio propuesto 4, en el que empezamos a realizar consultas a la tabla **usuarios**. 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. Obten 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 en la lectura anterior 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: ", res.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()
```