Implementación y gestión de una base de datos

Inserción, actualización y borrado de datos

{desafío} latam_

Implementar la conexión a una base de datos PostgreSQL en una aplicación Node utilizando buenas prácticas.

Programar instrucciones para la obtención y manipulación de información desde una base de datos utilizando buenas prácticas, acorde al entorno Node.js.

{desafío} latam_

- Unidad 1:
  Implementación y gestión de una base de datos
- Unidad 2:
  Transacciones y API REST
- Unidad 3:
  Trabajo práctico

Te encuentras aquí

¿Qué aprenderás en esta sesión?

- *Utiliza sentencias para la inserción, actualización y borrado de datos utilizando el entorno Node.js.*

¿Has tenido algún acercamiento previo a Base de datos?

/* Recordatorio de instrucciones básicas SQL */

{desafío}
latam_

Instrucciones básicas SQL

INSERT

La sentencia INSERT de SQL, permite crear nuevos registros en una tabla de la base de datos, podemos crear registros individuales utilizando la forma:

```
INSERT INTO NOMBRE_TABLA(campo1, campo2) VALUES (valor1, valor2)
```

o un conjunto de ellos utilizando la siguiente estructura:

```
INSERT INTO NOMBRE_TABLA (campo1, campo2)
VALUES
   (valor1, valor2),
   (valor1, valor2),
   (valor1, valor2),
   (valor1, valor2);
```

Instrucciones básicas SQL

SELECT

La sentencia SELECT de SQL, permite consultar los registros en una tabla de la base de datos, podemos obtener todos los registros utilizando la forma:

```
SELECT * FROM NOMBRE_TABLA
```

O si deseamos obtener puntualmente uno de los registros utilizando la siguiente estructura:

```
SELECT * FROM NOMBRE_TABLA WHERE campo = valor
```

Instrucciones básicas SQL
UPDATE

La sentencia UPDATE de SQL, permite modificar los registros de una tabla en la base de datos, podemos actualizar uno o más registros dependiendo de los registros que involucre la condición utilizada en la instrucción WHERE de la consulta, la forma de la sentencia es:

```
UPDATE NOMBRE_TABLA
   SET campo1=valor1
   AND campo2=valor2
   WHERE campo=valor;
```

Instrucciones básicas SQL

DELETE

La sentencia DELETE de SQL, permite eliminar registros de una tabla, esta acción se puede realizar en uno o más registros dependiendo de la condición generada en la instrucción WHERE de la consulta, la sentencia tiene la siguiente forma:

```
DELETE FROM NOMBRE_TABLA
    WHERE campo=valor;
```

/* Ingresando y consultando datos: INSERT, SELECT, WHERE */

Ingresando y consultando datos

INSERT

La librería pg a través de su función query nos permite utilizar la sentencia SQL INSERT para crear registros en una tabla de la base de datos PostgreSQL. Para poder crear un nuevo registro en una aplicación Node, debemos contar en nuestro script con una conexión activa.

La librería pg al ejecutar una sentencia SQL INSERT nos entrega como respuesta de la base de datos un objeto result

Ejercicio guiado: Insertando datos con la librería PG

{desafío} latam_

Insertando datos con la librería PG

A continuación, realizaremos un ejercicio donde insertaremos datos a una base de datos utilizando la librería PG. Esta librería nos provee una forma de insertar datos a través de valores parametrizados en una instrucción SQL.

Ejemplo:

```
"insert into ropa (tipo, talla) values ($1, $2)";
```

Nótese los values, entre paréntesis se pasa el símbolo $. Esta sentencia hace que los valores sean pasados como parámetros.

Veamos un ejemplo...

Insertando datos con la librería PG

- **Paso 1:** Creamos una base de datos llamada prendasVestir con una tabla llamada ropa.

```
create database prendasVestir;
```

```
create table ropa(id serial primary key, tipo varchar(255), talla varchar(50));
```

Insertando datos con la librería PG

- **Paso 2:** Realizamos el pool de conexión.

Recuerda instalar el paquete pg en el proyecto

`npm i pg`

```
const { Pool } = require("pg");

const config = {
  host: "localhost",
  port: 5432,
  database: "tu_base_de_datos",
  user: "tu_usuario",
  password: "tu_password",
};

const pool = new Pool(config);
```

Insertando datos con la librería PG

- **Paso 3:** Realizamos un insert a la tabla ropa con los datos parametriazados.

```
const insertRopa = async () => {
  const text = "insert into ropa (tipo, talla) values ($1, $2)";
  const values = ["Jean gris", "44"];

  const response = await pool.query(text, values);
  console.log(response);
};

insertRopa();
```

- Tenemos una constante values la cual contiene un arreglo.
- Este arreglo es el que pasará a los datos parametrizados dado que el hacer el pool.query estamos pasando ambos argumentos.

Insertando datos con la librería PG

- **Paso 4:** Al correr la aplicación, veremos que el comando 'INSERT' se ejecutó y podemos comprobar la inserción en la base de datos con `select * from ropa;`

```
Result {
  command: 'INSERT',
  rowCount: 1,
  oid: 0,
  rows: [],
```

Insertando datos con la librería PG

- **Paso 5:** También podemos hacer la operación **delete**. En este caso aplicaremos los valores de igual manera por parámetros.

```
const deleteRopa = async () => {
  const text = "delete from ropa where
tipo = $1";
  const values = ["Jean gris"];

  const response = await
pool.query(text, values);
  console.log(response);
};
deleteRopa();
```

Insertando datos con la librería PG

- **Paso 6:** Veamos cómo hacer ahora la actualización de un registro con valores como parámetros.

```
const upDateRopa = async () => {
  const text = "update ropa set tipo = $1 where tipo = $2";
  const values = ["Jean rojo", "Jean azul"];

  const response = await pool.query(text, values);
  console.log(response);
};
upDateRopa();
```

Actualizamos el registro donde su **tipo** sea “Jean azul”.

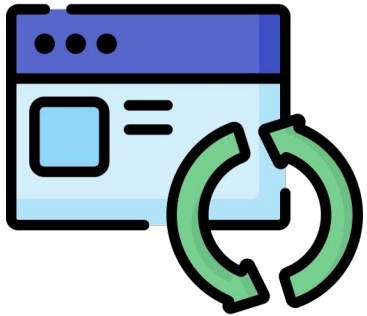
Comentemos:
¿En qué nos ayuda la instrucción where?

/* Actualizando y eliminando datos: UPDATE, DELETE */

Actualizando y eliminando datos

UPDATE

La actualización de registros es clave para un control completo de datos que queremos persistir en nuestras aplicaciones. El paquete pg por medio de su método query() nos permite enviar cualquier sentencia SQL a PostgreSQL, por lo que una actualización no es diferente a la inserción y selección de datos vistos en el capítulo anterior, simplemente se trata de cambiar la sentencia.



Ejercicio guiado:
Corrigiendo datos (UPDATE)

{desafío}
latam_

Ejercicio guiado

Corrigiendo datos (UPDATE)

Continuando con la temática de la tienda de ropa B&H, agregar una función a nuestra aplicación que al ser ejecutada modifique la talla de la polera agregada anteriormente por una talla L, suponiendo un hipotético caso en el que la tienda B&H ingresó erróneamente esta talla al momento de cargar la nueva mercancía.

Paso 1: Crear una función asíncrona con el nombre “editar”.

Paso 2: Guardar en una constante “res” una consulta SQL con la palabra reservada “await”, que actualice la talla de la polera registrada con el id 2 por la letra L. Considera incluir el RETURNING * para obtener el registro afectado.

Paso 3: Imprimir por consola la propiedad “rows”, en su índice 0 del objeto resultante de la consulta, para ver el registro actualizado.

Paso 4: Imprimir por consola la cantidad de registros afectados con la propiedad rowCount.

Ejercicio guiado: Eliminando registros (DELETE)

{desafío} latam_

Ejercicio guiado

Eliminando registros (DELETE)

Crear una función que elimine todos los registros de la tabla ropa y devuelva por consola la cantidad total de filas eliminadas.

Paso 1: Crear una función asíncrona con el nombre “eliminar”.

Paso 2: Guardar en una constante “res” una consulta SQL con la palabra reservada “await” que elimine todos los registros de la tabla ropa.

Paso 3: Imprimir por consola la propiedad “rowCount” para conocer la cantidad de registros eliminados.

¿Hay diferencias entre las instrucciones básicas de SQL y las que vimos al comienzo de la clase?

{desafío} latam_

Próxima sesión...

- *Desafío evaluado - Always Music*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales