CONSUMO DE BASE DE DATOS

Actualmente tenemos el proyecto contactos en react native, que se une a un API de servicios REST creados en node.js. Este API siempre retorna la misma información y en el caso de guardar o actualizar solo responde exitoso.

Se va a complementar este proyecto, conectándonos a la base de datos desde Node.js.

Para este propósito vamos a realizar los siguientes pasos en el proyecto de nodejs.

1) Cambiar la ip, por su ip local (ipconfig). Levantar el proyecto y probar los métodos desde postman
2) Conectarse a la base de datos de postgres y en alguna de las bases de datos disponibles, crear la tabla de contactos, para almacenar esta información

```
{id:1,nombre:"Santiago",apellido:"Mosquera",celular:"0992920306"}
```

Usar los mismos nombres para las columnas.
El id es autogenerado.
Crear unos inserts para registrar 2 contactos.

3) Para conectarnos a postgres desde nodejs:
    1. Instalar en el proyecto de node el módulo pg: npm install pg

    2. En el app.js, importar la función Client del módulo pg

```
const { Client } = require("pg")
```

    3. Instanciar un objeto Client, pasándole como parámetro un objeto con las credenciales de la bdd a la que se va a conectar. Poner sus datos!!!

```
const client = new Client({
  user: "postgres",
  host: "192.168.68.106",
  database: "pruebas",
  password: "postgres",
  port: 5432,
});
```

    4. RECUPERAR CONTACTOS
    Modificar el método que permite consultar todos los contactos desde node.

    Método actual

```
app.get("/contactos",(request,response)=>{
    const contactos=[
```

```
{id:1,nombre:"Santiago",apellido:"Mosquera",celular:"0992920
306"},

{id:2,nombre:"Richard",apellido:"Muñoz",celular:"0992745903"
},

{id:3,nombre:"Pedro",apellido:"Muñoz",celular:"0992645906"},
    ];
    console.log("ingresa a get");

    response.send(contactos);
});
```

Dentro del método, antes de responder:

1) Conectarse a la bdd, usando la variable client.
   client.connect();
2) Ejecutar la consulta a la bdd, con el método query, este método es asíncrono, por lo tanto, tiene un then y un catch client.query("select * from contactos").then().catch() En el then, pasamos una función que se va a ejecutar si la respuesta es exitosa. A esta función se le pasa como parámetros la respuesta obtenida. client.query("select * from contactos"). then(responseQuery=>{}).
   catch()
   Imprimimos en consola las filas de la respuesta obtenida y cerramos la conexión
   then(responseQuery=>{
   console.log(responseQuery.rows); client.end();
   }).
   catch()
   En el catch, le pasamos una función que se va a ejecutar cuando ocurra un error. Esta función recibe como parámetro el error y por ahora lo imprimimos y cerramos la conexión then(responseQuery=>{
   console.log(responseQuery.rows); client.end();
   }).
   catch(err=>{
   console.log(err);
   client.end();

   })

Al final seguimos respondiendo la lista de contactos fija del arreglo

```
response.send(contactos);
```

Probar desde postman que responda la lista de contactos fija y en la consola ver que imprima los datos recuperados de la bdd.

Es probable que obtenga el siguiente error:

----------no hay una l nea en pg_hba.conf para 192.168.68.106 , usuario postgres , base de datos test , sin cifrado

Indicando que se requiere una configuración en postgres, para poder acceder desde la ip específica. Para corregir este error, debe modificar el archivo pg_hba.conf que se encuentra en la ruta: C:\Program Files\PostgreSQL\16\data

En el archivo agregar la línea

host all all 192.168.68.106/24 md5

Usando el valor de su IP.

Volver a ejecutar el proyecto Node.js. Probar desde postman y validar que imprima en consola los datos que trae de la base de datos.

Finalmente, modificar el código para que cuando sea exitoso la invocación a la bdd, responda al rest web service, luego de imprimir las filas.

response.send(responseQuery.rows);

Como ya responde en este punto, eliminar el response.send(contactos), del final del archivo.

Probar desde postman y si ya funciona, eliminar la lista de contactos fija que se tenía dentro de la función.

PARTE 2

Probar desde el proyecto de contactos en React Native, que se pinten los elementos guardados en la base de datos.

INSERTAR

Para insertar, basarse en este ejemplo:

```
conexion.query('insert into productos
    (nombre, precio)
    values
    ($1, $2)', [nombre, precio]);
```

Donde los valores de las variables $1 y $2 se reemplazan por los elementos del arreglo nombre y precio, estas variables deberían llenarse con los atributos del objeto que llega a la función insertar

Al igual que la consulta, en el then vamos a tener un arreglo de filas, imprimir para validar. Si es el caso. Tomar la primera fila para poder armar el response con el elemento insertado

ELIMINAR
Ejemplo:

```
conexion.query('delete from productos
      where id = $1', [id]);
```

ACTUALIZAR
Ejemplo:

```
conexion.query(‘update productos
      set nombre = $1, precio = $2
      where id = $3’, [nombre, precio, id]);
```

PROBAR TODO DESDE POSTMAN y LUEGO DESDE EL CELULAR