

DISEÑO Y DESARROLLO DE SERVICIOS WEB - PROYECTO

Presentado por

JEFERSON GABRIEL FIAGA DIAZ

Instructor

ISRAEL ARBONA

Programa

DESARROLLO DE SOFTWARE

Institución

SENA CENTRO DE GESTIÓN AGROEMPRESARIAL DEL ORIENTE

Lugar Y Fecha

CANDELARIA, VALLE DEL CAUCA

02-01-2024

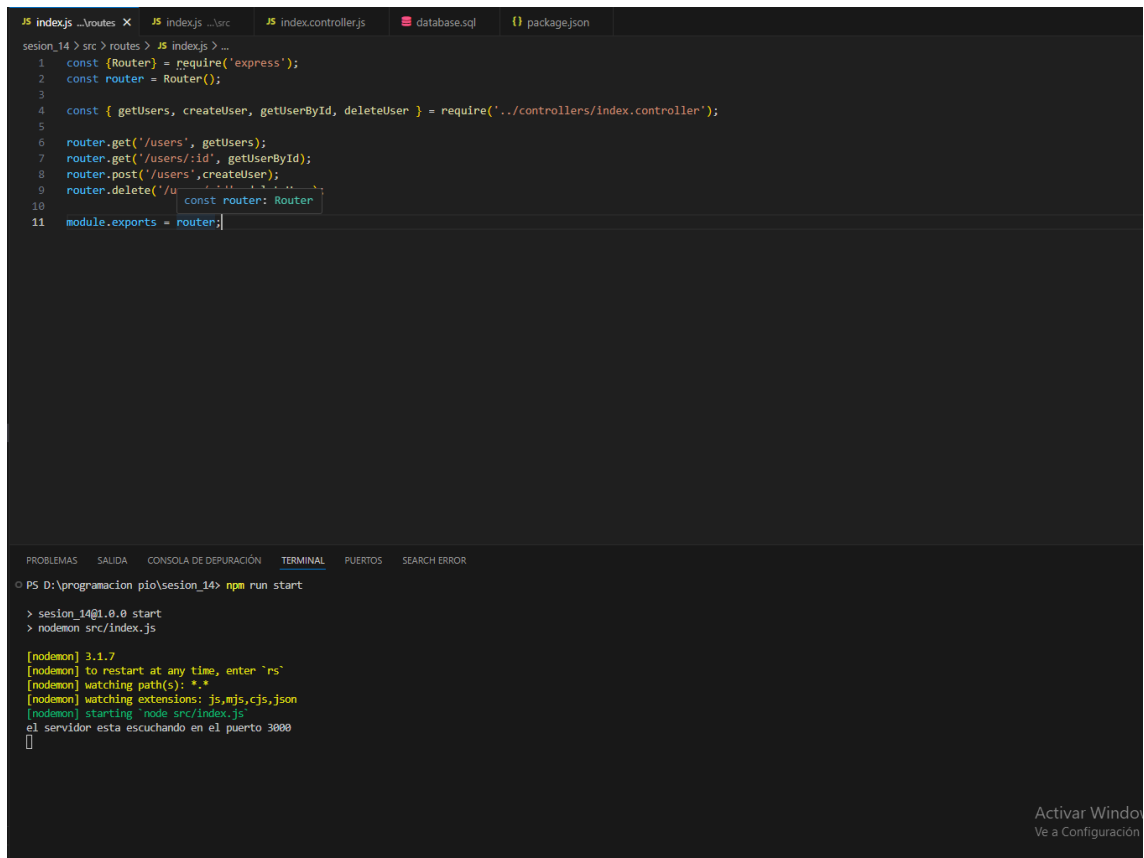
INTRODUCCION

Para esta evidencia aremos un trabajo escrito mostrando las evidencias de una API para agregar usuarios mostraremos los códigos con imágenes y mostraremos su funcionalidad de manera objetiva.

DESARROLLO

Se realiza un código de una API que nos ayude a ingresar eliminar actualizar usuarios con métodos GET POST DELETE y con ese mismo servicio poder ingresar también productos, pedidos manejando solo el _id que se le dé a un usuario de nuestros servicios, dando una respuesta un JSON nos mostrara en el puerto que estemos escuchando el servidor también tenemos en cuenta la base de datos en la cual vamos a trabajar sea postgresQL, MYSQLwordbench, o aplicaciones de testing como sea postman o insomnia

Aquí mostramos las rutas que vamos a utilizar para hacer los ingresos actualizaciones o eliminar usuarios

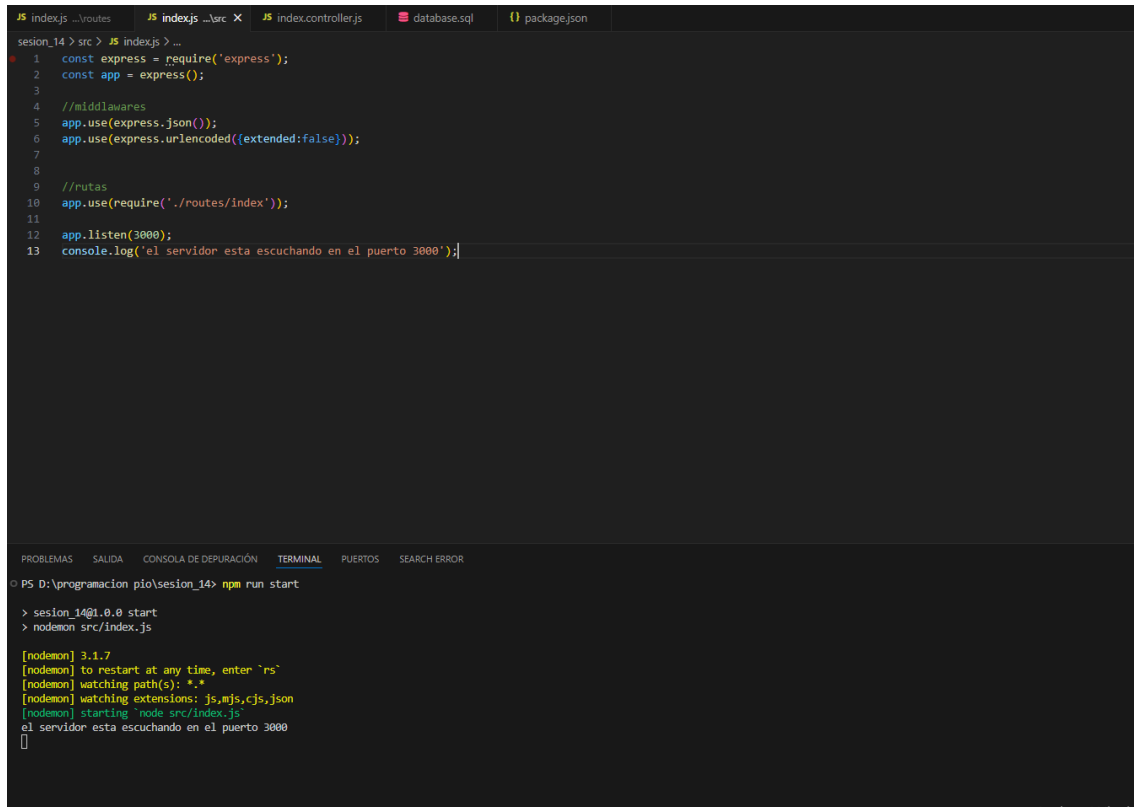


```
index.js  routes  index.js  index.controller.js  database.sql  package.json
sesion_14 > src > routes > index.js > ...
1  const {Router} = require('express');
2  const router = Router();
3
4  const { getUsers, createUser, getUserById, deleteUser } = require('../controllers/index.controller');
5
6  router.get('/users', getUsers);
7  router.get('/users/:id', getUserById);
8  router.post('/users', createUser);
9  router.delete('/u
10
11  module.exports = router;
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS  SEARCH ERROR
PS D:\programacion pio\sesion_14> npm run start
> sesion_14@1.0.0 start
> nodemon src/index.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting node src/index.js
el servidor esta escuchando en el puerto 3000
[]
```

Los requerimientos y las librerías utilizadas y un mensaje en consola de donde se esta esta escuchando el servidor puerto 3000 a si cuando este encendido se puede ir a localhost:3000 y ver si queremos poner mensajes



The screenshot shows a VS Code editor with a file explorer on the left containing 'index.js', 'routes', 'index.controller.js', 'database.sql', and 'package.json'. The main editor displays the content of 'index.js' with the following code:

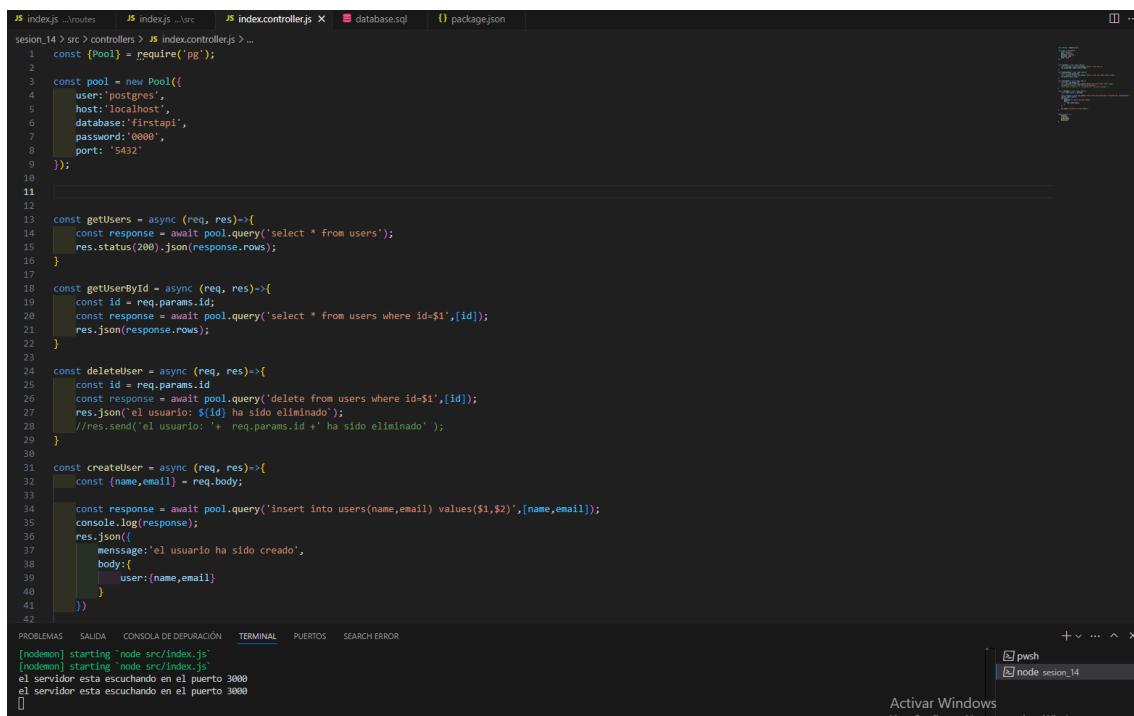
```
1 const express = require('express');
2 const app = express();
3
4 //middlewares
5 app.use(express.json());
6 app.use(express.urlencoded({extended:false}));
7
8 //rutas
9 app.use(require('./routes/index'));
10
11 app.listen(3000);
12 console.log('el servidor esta escuchando en el puerto 3000');
```

Below the editor, the 'TERMINAL' tab is active, showing the command 'npm run start' and its output:

```
> session_14@1.0.0 start
> nodemon src/index.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node src/index.js'
el servidor esta escuchando en el puerto 3000
```

Los códigos que controlan el como se va a ingresar actualizar o eliminar a los usuarios ya existentes o por crear y la conexión a la base de datos en este caso la de postgresSQL.



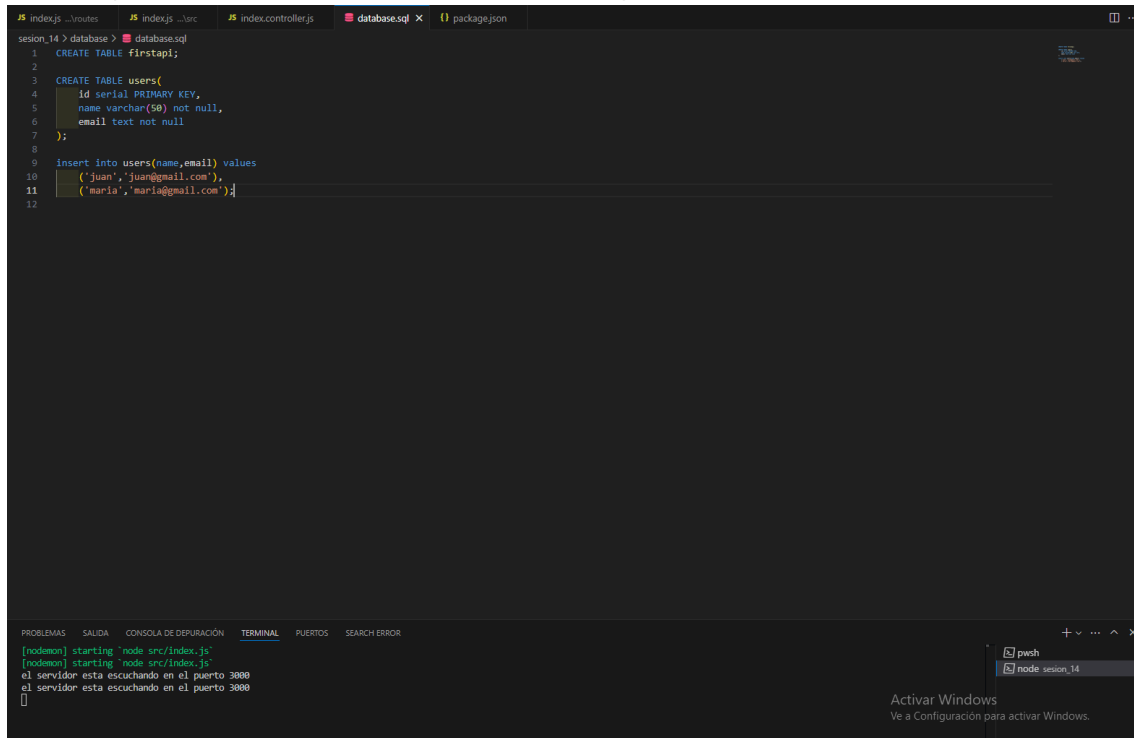
The screenshot shows a VS Code editor with a file explorer on the left containing 'index.js', 'routes', 'index.controller.js', 'database.sql', and 'package.json'. The main editor displays the content of 'index.controller.js' with the following code:

```
1 const {Pool} = require('pg');
2
3 const pool = new Pool({
4   user: 'postgres',
5   host: 'localhost',
6   database: 'firstapi',
7   password: '0000',
8   port: '5432'
9 });
10
11
12
13 const getUsers = async (req, res) => {
14   const response = await pool.query('select * from users');
15   res.status(200).json(response.rows);
16 }
17
18 const getUserById = async (req, res) => {
19   const id = req.params.id;
20   const response = await pool.query('select * from users where id=$1',[id]);
21   res.json(response.rows);
22 }
23
24 const deleteUser = async (req, res) => {
25   const id = req.params.id;
26   const response = await pool.query('delete from users where id=$1',[id]);
27   res.json('el usuario: ${id} ha sido eliminado');
28   //res.send('el usuario: '+ req.params.id +' ha sido eliminado' );
29 }
30
31 const createUser = async (req, res) => {
32   const {name,email} = req.body;
33
34   const response = await pool.query('insert into users(name,email) values($1,$2)',[name,email]);
35   console.log(response);
36   res.json({
37     message: 'el usuario ha sido creado',
38     body: {
39       user: {name,email}
40     }
41   });
42 }
```

Below the editor, the 'TERMINAL' tab is active, showing the command 'npm run start' and its output:

```
[nodemon] starting 'node src/index.js'
[nodemon] starting 'node src/index.js'
el servidor esta escuchando en el puerto 3000
el servidor esta escuchando en el puerto 3000
```

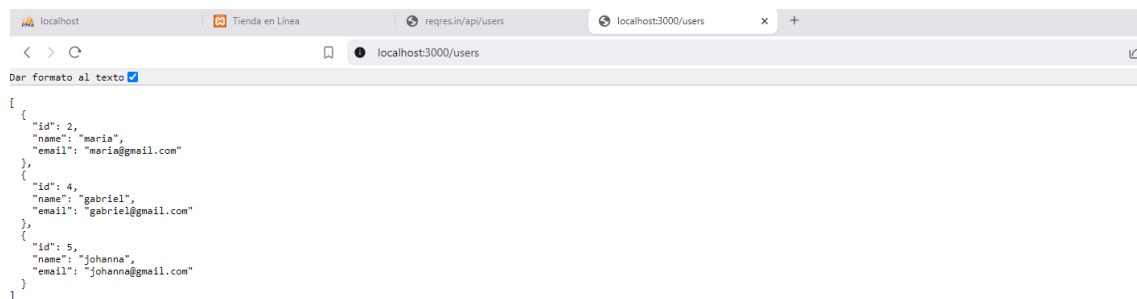
Al final puse los elementos que vamos a agregar a la base de datos en este caso solo el nombre y el correo electrónico con dos usuarios ya creados



```
1 CREATE TABLE firstapi;
2
3 CREATE TABLE users(
4   id serial PRIMARY KEY,
5   name varchar(50) not null,
6   email text not null
7 );
8
9 insert into users(name,email) values
10   ('juan','juan@gmail.com'),
11   ('maria','maria@gmail.com');
12
```

```
[nodemon] starting 'node src/index.js'
[nodemon] starting 'node src/index.js'
el servidor esta escuchando en el puerto 3000
el servidor esta escuchando en el puerto 3000
```

Y una vista de como se ve en el navegados (ya he hecho varias pruebas desde la base de datos postgres y con postman que veremos en la siguiente evidencia)



```
[
  {
    "id": 2,
    "name": "maria",
    "email": "maria@gmail.com"
  },
  {
    "id": 4,
    "name": "gabriel",
    "email": "gabriel@gmail.com"
  },
  {
    "id": 5,
    "name": "johanna",
    "email": "johanna@gmail.com"
  }
]
```