



**SERVICIO NACIONAL
DE APRENDIZAJE**

Evidencia de producto GA7-220501096-AA5-EV02 API

ARROYAVE JIMENEZ JOHN FREDDY

ANALISIS Y DESARROLLO DE SOFTWARE

CENTRO DE COMERCIOS Y SERVICIOS
(ATLANTICO)

FICHA

2834926

MEDELLIN

2024

Introducción

En el desarrollo de aplicaciones modernas, la autenticación de usuarios es un componente fundamental para garantizar la seguridad y personalización de los servicios. Este proyecto aborda la creación de un sistema de login implementado con Express.js, es una de las herramientas mas utilizadas en el entorno de desarrollo de backend con Node.js

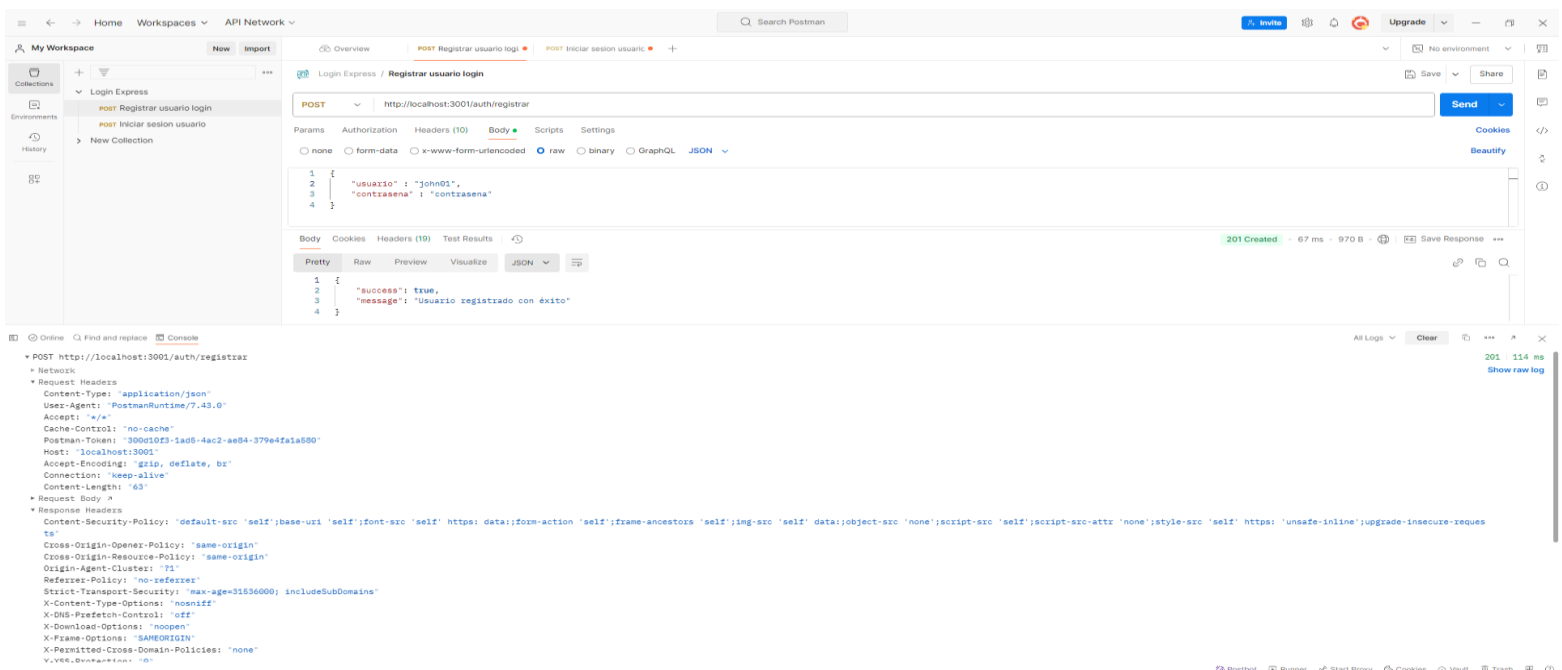
El objetivo principal es desarrollar un servicio web que permita validar las credenciales de los usuarios almacenadas en la base de datos MySQL, garantizando un manejo seguro de los datos. La funcionalidad del servicio se prueba utilizando Postman, una herramienta ampliamente reconocida para la realización de pruebas de API, permitiendo verificar el correcto funcionamiento de las rutas de registro e inicio de sesión.

Este trabajo presenta la estructura del proyecto, los procesos de desarrollo, las validaciones realizadas y ñillo9s resultados obtenidos durante las pruebas con postman.

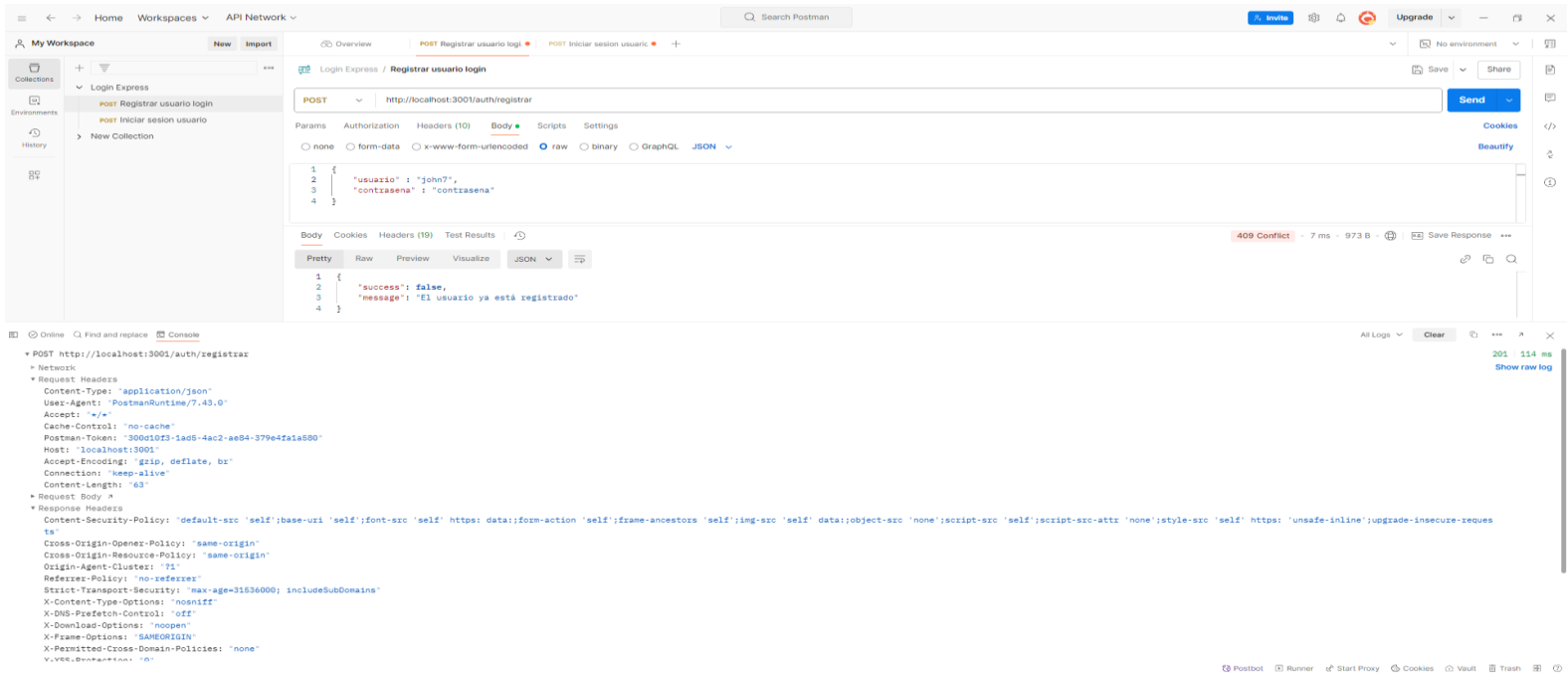
Test en postman

Para usar postman, creamos una colección la cual he llamado **login express** allí en el método post agregamos nuestro servidor con el puerto que creamos por defecto (3001) vamos al apartado body (cuerpo), raw y escogemos en este caso Json.

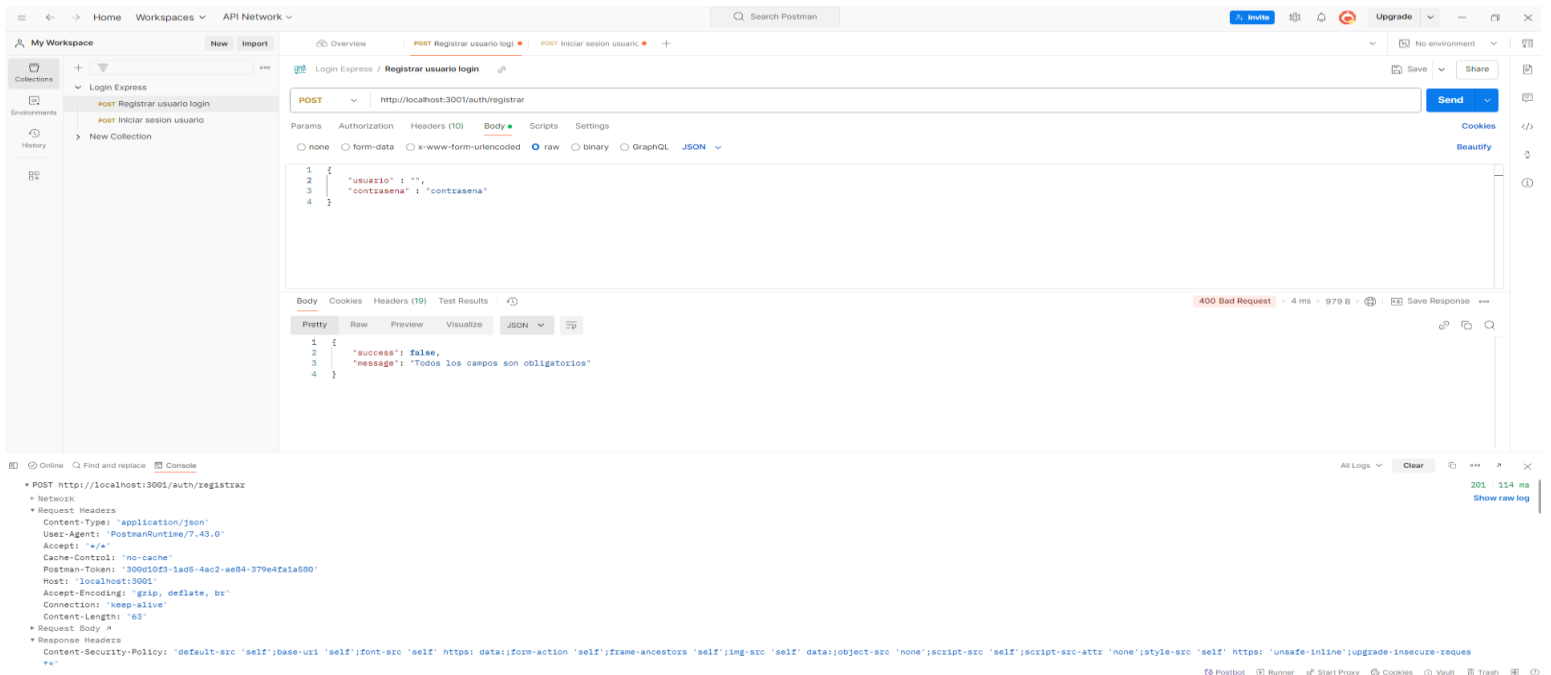
Esta prueba consiste en agregar a un usuario a nuestra base de datos con una contraseña encriptada. En la imagen siguiente vemos que todo salió bien ya que nos arroja un código 201 created.



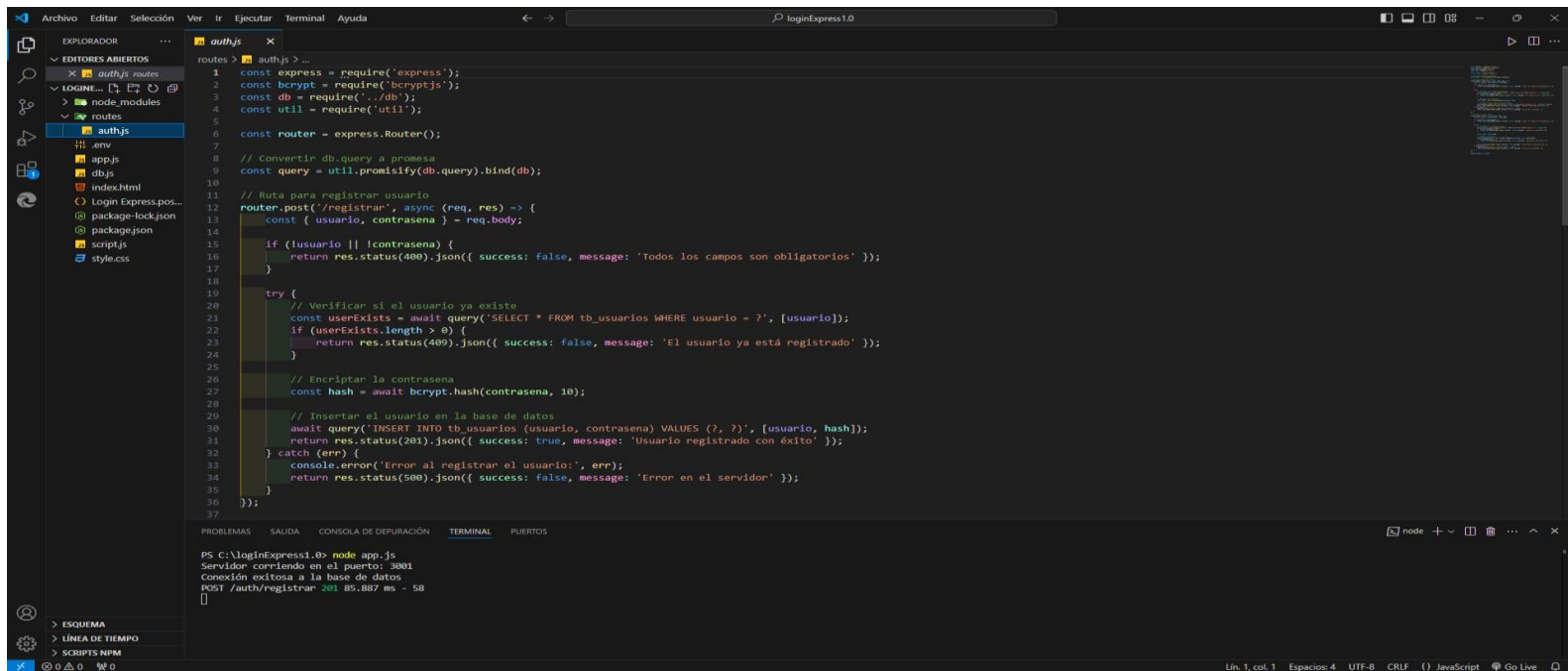
En la siguiente imagen vemos como al hacer la misma operación, postman nos dice que el usuario ya está registrado, este mensaje esta representado por el código 409 conflict. En este caso determinamos que, según el resultado, nuestra aplicación reconoce que ya existe un usuario con estas características y por ende no lo vuelve a ingresar.



En esta prueba hemos omitido la información del nombre del usuario a la cual postman responde que todos los campos son obligatorios, esto nos permite que la información que nosotros tenemos en nuestra base de datos sea la que se está pidiendo en el login.



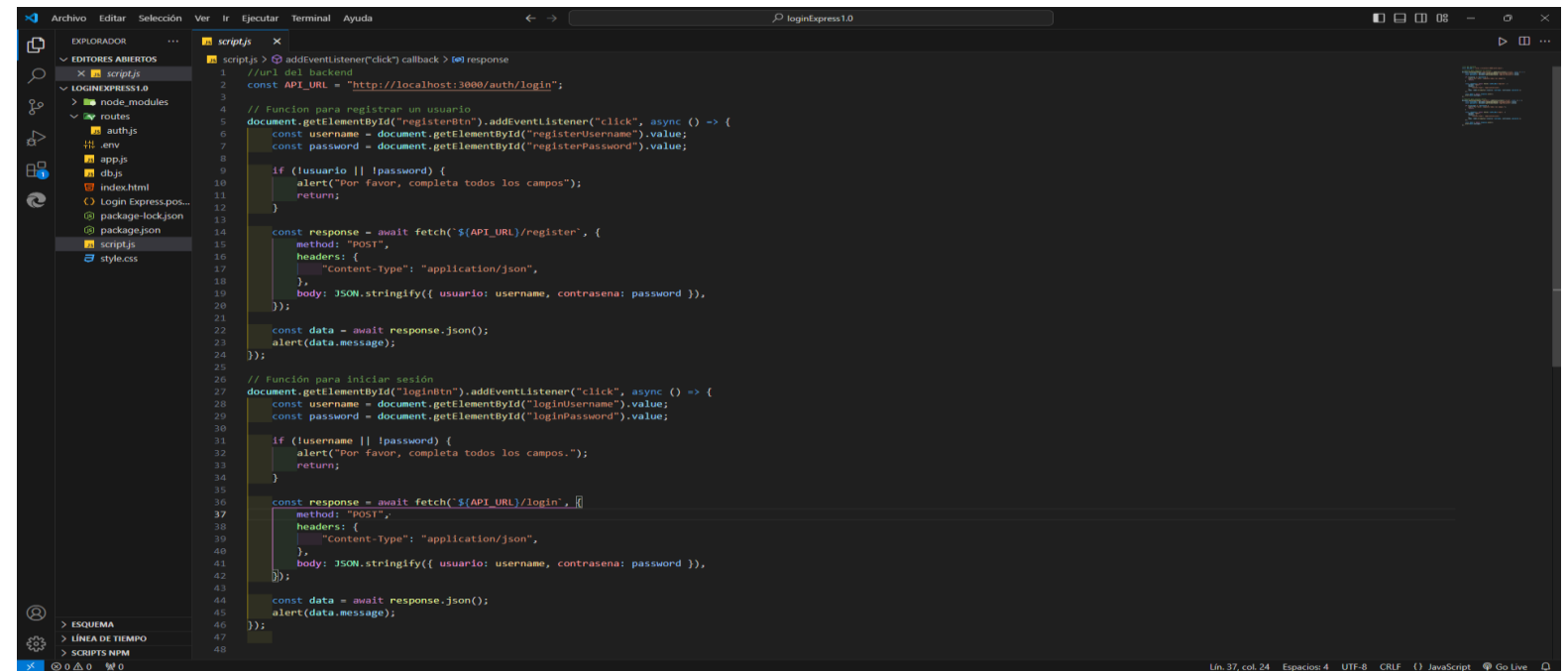
Este archivo que hemos llamado auth.js es el que contiene los comandos para manejar los mensajes de error y autenticación, también nos permite la conversión del query a promesa, la ruta para registrar el usuario y comparación de usuario y contraseña para evitar duplicaciones



```
routes > auth.js > -
1 const express = require('express');
2 const bcrypt = require('bcryptjs');
3 const db = require('../db');
4 const util = require('util');
5
6 const router = express.Router();
7
8 // Convertir db.query a promesa
9 const query = util.promisify(db.query).bind(db);
10
11 // Ruta para registrar usuario
12 router.post('/registrar', async (req, res) => {
13   const { usuario, contraseña } = req.body;
14
15   if (!usuario || !contraseña) {
16     return res.status(400).json({ success: false, message: 'Todos los campos son obligatorios' });
17   }
18
19   try {
20     // Verificar si el usuario ya existe
21     const userExists = await query('SELECT * FROM tb_usuarios WHERE usuario = ?', [usuario]);
22     if (userExists.length > 0) {
23       return res.status(400).json({ success: false, message: 'El usuario ya está registrado' });
24     }
25
26     // Encriptar la contraseña
27     const hash = await bcrypt.hash(contraseña, 10);
28
29     // Insertar el usuario en la base de datos
30     await query('INSERT INTO tb_usuarios (usuario, contraseña) VALUES (?, ?)', [usuario, hash]);
31     return res.status(201).json({ success: true, message: 'Usuario registrado con éxito' });
32   } catch (err) {
33     console.error('Error al registrar el usuario:', err);
34     return res.status(500).json({ success: false, message: 'Error en el servidor' });
35   }
36 });
37
```

PS C:\loginExpress1.0> node app.js
Servidor corriendo en el puerto: 3001
Conexión exitosa a la base de datos
POST /auth/registrar 201 85.887 ms - 58

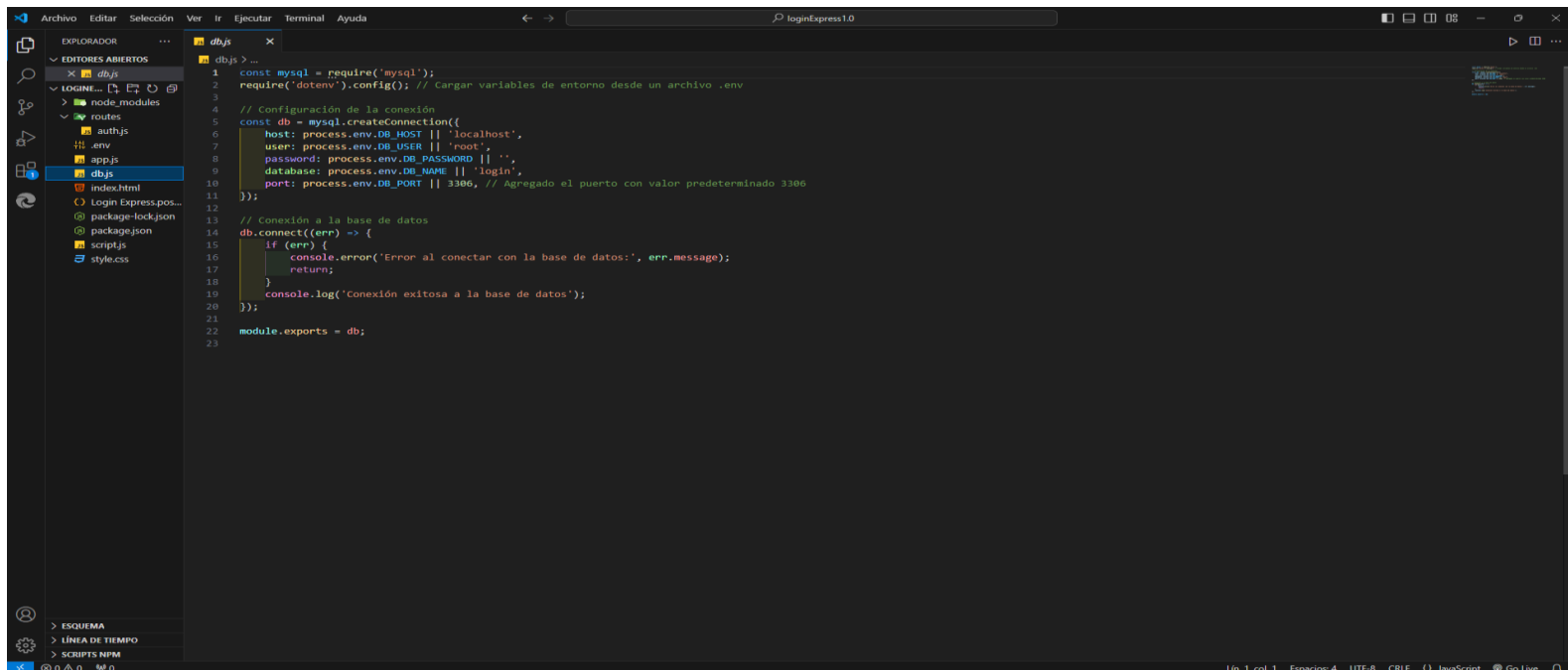
En la siguiente imagen podemos observar nuestro archivo script.js el cual maneja la dirección del backend, la función para registrar el usuario y la función para iniciar sesión.



```
script.js > addEventListener('click') callback > @ response
1 //url del backend
2 const API_URL = "http://localhost:3000/auth/login";
3
4 // Función para registrar un usuario
5 document.getElementById("registerBtn").addEventListener("click", async () => {
6   const username = document.getElementById("registerUsername").value;
7   const password = document.getElementById("registerPassword").value;
8
9   if (!username || !password) {
10     alert("Por favor, completa todos los campos.");
11     return;
12   }
13
14   const response = await fetch(`${API_URL}/register`, {
15     method: "POST",
16     headers: {
17       "Content-Type": "application/json",
18     },
19     body: JSON.stringify({ usuario: username, contraseña: password }),
20   });
21
22   const data = await response.json();
23   alert(data.message);
24 });
25
26 // Función para iniciar sesión
27 document.getElementById("loginBtn").addEventListener("click", async () => {
28   const username = document.getElementById("loginUsername").value;
29   const password = document.getElementById("loginPassword").value;
30
31   if (!username || !password) {
32     alert("Por favor, completa todos los campos.");
33     return;
34   }
35
36   const response = await fetch(`${API_URL}/login`, {
37     method: "POST",
38     headers: {
39       "Content-Type": "application/json",
40     },
41     body: JSON.stringify({ usuario: username, contraseña: password }),
42   });
43
44   const data = await response.json();
45   alert(data.message);
46 });
47
48
```

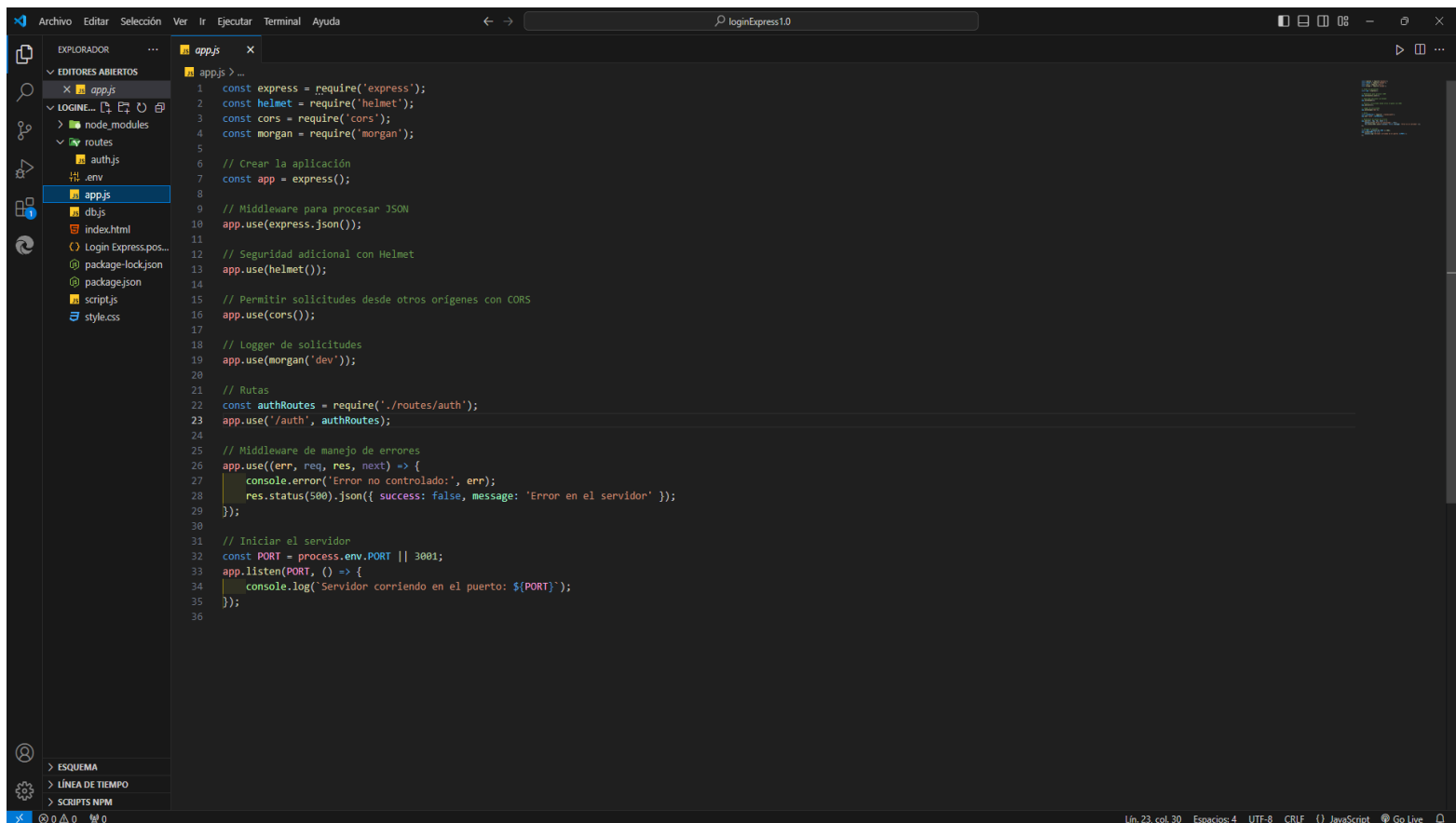
Ln. 37, col. 24 - Espacios: 4 - UTF-8 - CRLF - JavaScript - Go Live

db.js nos permite hacer configuración y la conexión a la base de datos.



```
1 const mysql = require('mysql');
2 require('dotenv').config(); // Cargar variables de entorno desde un archivo .env
3
4 // Configuración de la conexión
5 const db = mysql.createConnection({
6   host: process.env.DB_HOST || 'localhost',
7   user: process.env.DB_USER || 'root',
8   password: process.env.DB_PASSWORD || '',
9   database: process.env.DB_NAME || 'login',
10  port: process.env.DB_PORT || 3306, // Agregado el puerto con valor predeterminado 3306
11});
12
13 // Conexión a la base de datos
14 db.connect((err) => {
15   if (err) {
16     console.error('Error al conectar con la base de datos:', err.message);
17     return;
18   }
19   console.log('Conexión exitosa a la base de datos');
20 });
21
22 module.exports = db;
```

El archivo que dirige nuestro proyecto, en el cual hacemos el llamado a la conexión por medio de node app.js es llamado precisamente app.js. en este archivo creamos la aplicación, agregamos los middlewares para procesar Json, rutas y demás.



```
1 const express = require('express');
2 const helmet = require('helmet');
3 const cors = require('cors');
4 const morgan = require('morgan');
5
6 // Crear la aplicación
7 const app = express();
8
9 // Middleware para procesar JSON
10 app.use(express.json());
11
12 // Seguridad adicional con Helmet
13 app.use(helmet());
14
15 // Permitir solicitudes desde otros orígenes con CORS
16 app.use(cors());
17
18 // Logger de solicitudes
19 app.use(morgan('dev'));
20
21 // Rutas
22 const authRoutes = require('./routes/auth');
23 app.use('/auth', authRoutes);
24
25 // Middleware de manejo de errores
26 app.use((err, req, res, next) => {
27   console.error('Error no controlado:', err);
28   res.status(500).json({ success: false, message: 'Error en el servidor' });
29 });
30
31 // Iniciar el servidor
32 const PORT = process.env.PORT || 3001;
33 app.listen(PORT, () => {
34   console.log('Servidor corriendo en el puerto: ${PORT}');
35 });
36
```

Conclusión

En este breve documento se explica de una manera muy corta pero precisa sobre la funcionalidad y servicio que presta postman, igualmente podemos apreciar algunos archivos con los cuales se asegura el correcto funcionamiento de nuestra aplicación. El código completo se podrá apreciar en el archivo zip que precede este documento con una también breve explicación en video sobre el mismo.