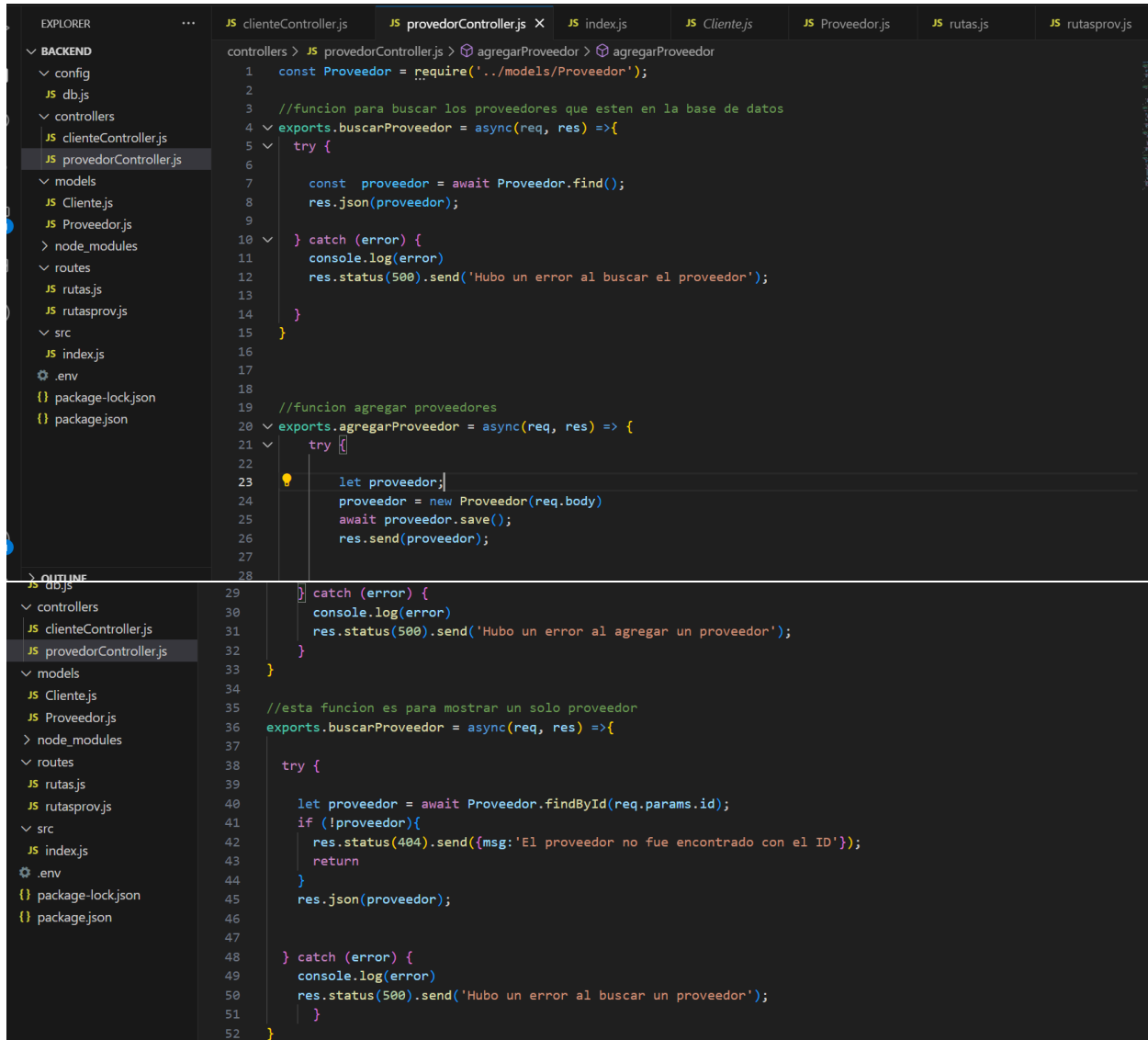


Creacion modulo para Base de datos no Relacional Por Oscar Javier Jaime Amaya

Se crea un controlador para proveedores



The screenshot shows a VS Code editor with a project structure on the left and a code editor on the right. The project structure includes a 'BACKEND' folder with subfolders for 'config', 'controllers', 'models', 'node_modules', 'routes', 'src', and 'OUTLINE'. The 'controllers' folder contains 'clienteController.js', 'proveedorController.js', and 'index.js'. The 'models' folder contains 'Cliente.js', 'Proveedor.js', and 'rutas.js'. The 'src' folder contains 'index.js'. The 'OUTLINE' folder contains 'db.js', 'clienteController.js', 'proveedorController.js', 'Cliente.js', 'Proveedor.js', 'node_modules', 'routes', 'rutas.js', 'rutasprov.js', 'index.js', '.env', 'package-lock.json', and 'package.json'.

The code editor shows the implementation of the 'proveedorController.js' file. The code is as follows:

```
1  const Proveedor = require('../models/Proveedor');
2
3  //funcion para buscar los proveedores que esten en la base de datos
4  exports.buscarProveedor = async(req, res) =>{
5    try {
6
7      const proveedor = await Proveedor.find();
8      res.json(proveedor);
9
10   } catch (error) {
11     console.log(error)
12     res.status(500).send('Hubo un error al buscar el proveedor');
13   }
14 }
15
16 //funcion agregar proveedores
17 exports.agregarProveedor = async(req, res) => {
18   try {
19
20     let proveedor;
21     proveedor = new Proveedor(req.body)
22     await proveedor.save();
23     res.send(proveedor);
24
25   } catch (error) {
26     console.log(error)
27     res.status(500).send('Hubo un error al agregar un proveedor');
28   }
29 }
30
31 //esta funcion es para mostrar un solo proveedor
32 exports.buscarProveedor = async(req, res) =>{
33   try {
34
35     let proveedor = await Proveedor.findById(req.params.id);
36     if (!proveedor){
37       res.status(404).send({msg:'El proveedor no fue encontrado con el ID'});
38       return
39     }
40     res.json(proveedor);
41
42   } catch (error) {
43     console.log(error)
44     res.status(500).send('Hubo un error al buscar un proveedor');
45   }
46 }
47
48
49
50
51
52 }
```

▼ controllers

JS clienteController.js

JS proveedorController.js

▼ models

JS Cliente.js

JS Proveedor.js

> node_modules

▼ routes

JS rutas.js

JS rutasprov.js

▼ src

JS index.js

🔗 .env

{ } package-lock.json

{ } package.json

> OUTLINE

> TIMELINE

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

//esta funcion nos sirve para eliminar un proveedor

exports.eliminarProveedor = async(req, res)=>{

try {

let proveedor = await Proveedor.findById(req.params.id);

if(!proveedor){

res.status(404).json({msg:'El proveedor no existe'});

return

}

await Proveedor.findOneAndDelete({_id: req.params.id});

res.json({msg:'El proveedor se eliminó correctamente'});

return

}

catch (error) {

console.log(error);

res.status(500).send('Hubo un error al eliminar el proveedor');

}

}

//esta funcion nos sirve para actualizar un proveedor

(property) actualizarProveedor: (req: any, res: any) => Promise<void>

exports.actualizarProveedor = async(req, res) => {

try {

const {empresa, nit, correo, telefono, direccion} = req.body

let proveedor = await Proveedor.findById(req.params.id);

if(!proveedor){

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

//esta funcion nos sirve para actualizar un proveedor

exports.actualizarProveedor = async(req, res) => {

try {

const {empresa, nit, correo, telefono, direccion} = req.body

let proveedor = await Proveedor.findById(req.params.id);

if(!proveedor){

res.status(404).json({msg:'El proveedor no existe'});

return

}else{

proveedor.empresa = empresa;

proveedor.nit = nit;

proveedor.correo = correo;

proveedor.telefono = telefono;

proveedor.direccion = direccion;

proveedor = await Proveedor.findOneAndUpdate({_id: req.params.id}, proveedor,{new:true});

res.json(proveedor);

}

catch (error) {

console.log(error);

res.status(500).send('Hubo un error al actualizar el proveedor');

return

}

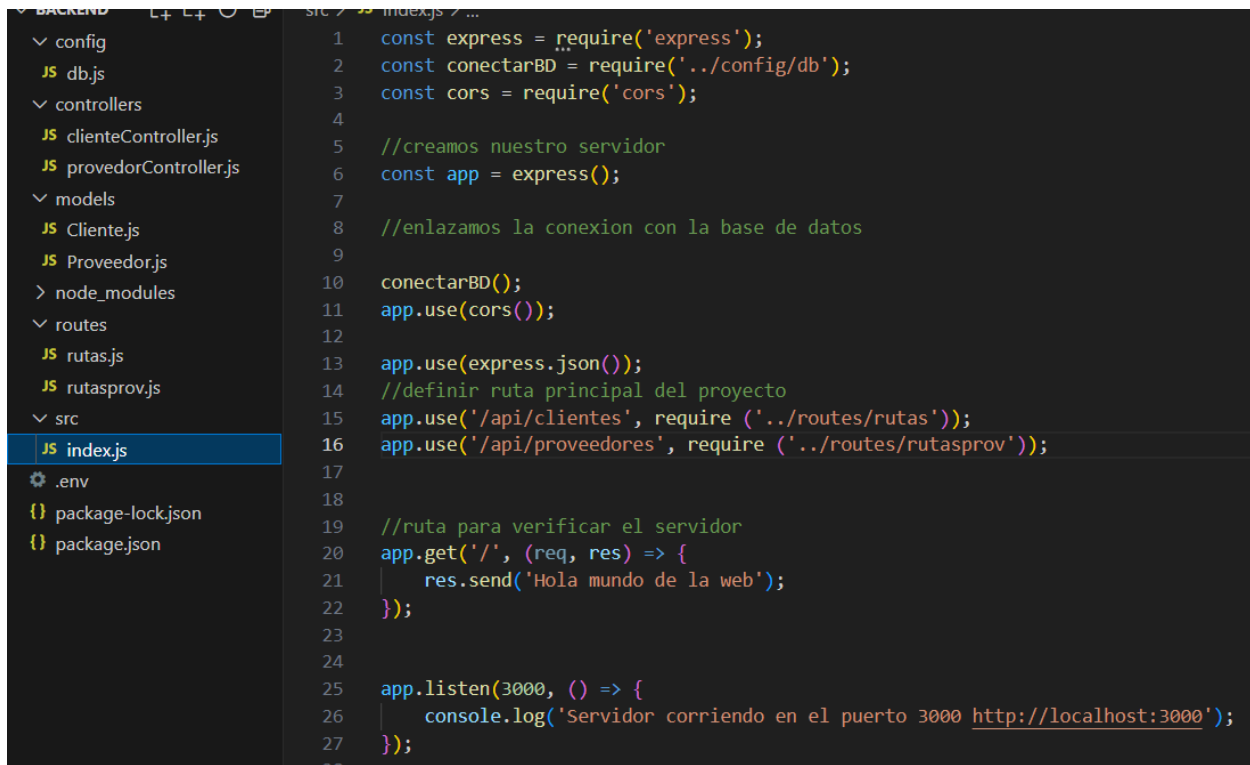
Creamos un modelo proveedor

```
models > JS Proveedor.js > proveedorSchema > telefono > required
1 const mongoose = require('mongoose');
2
3 //el modelo que se cree aca debe ser igual a la base de datos
4 const proveedorSchema = mongoose.Schema({
5
6   empresa: {
7     type: String,
8     required: true
9   },
10
11   nit: {
12     type: Number,
13     required: true
14   },
15   correo: {
16     type: String,
17     required: true
18   },
19   telefono: {
20     type: Number,
21     required: true
22   },
23   direccion: {
24     type: String,
25     required: true
26   }
27 }, {versionkey:false});
28
29 module.exports = mongoose.model('proveedor', proveedorSchema);
```

Creamos una ruta proveedor

```
routes > JS rutasprov.js > ...
1 const express = require('express');
2 const router = express.Router();
3 const proveedorController = require('../controllers/proveedorController');
4
5
6 //estas son las rutas de nuestro crud
7
8 //Proveedor
9 router.post('/', proveedorController.agregarProveedor);
10 router.get('/', proveedorController.buscarProveedor);
11 router.get('/:id', proveedorController.buscarProveedor);
12 router.delete('/:id', proveedorController.eliminarProveedor);
13 router.put('/:id', proveedorController.actualizarProveedor);
14
15
16 module.exports = router;
```

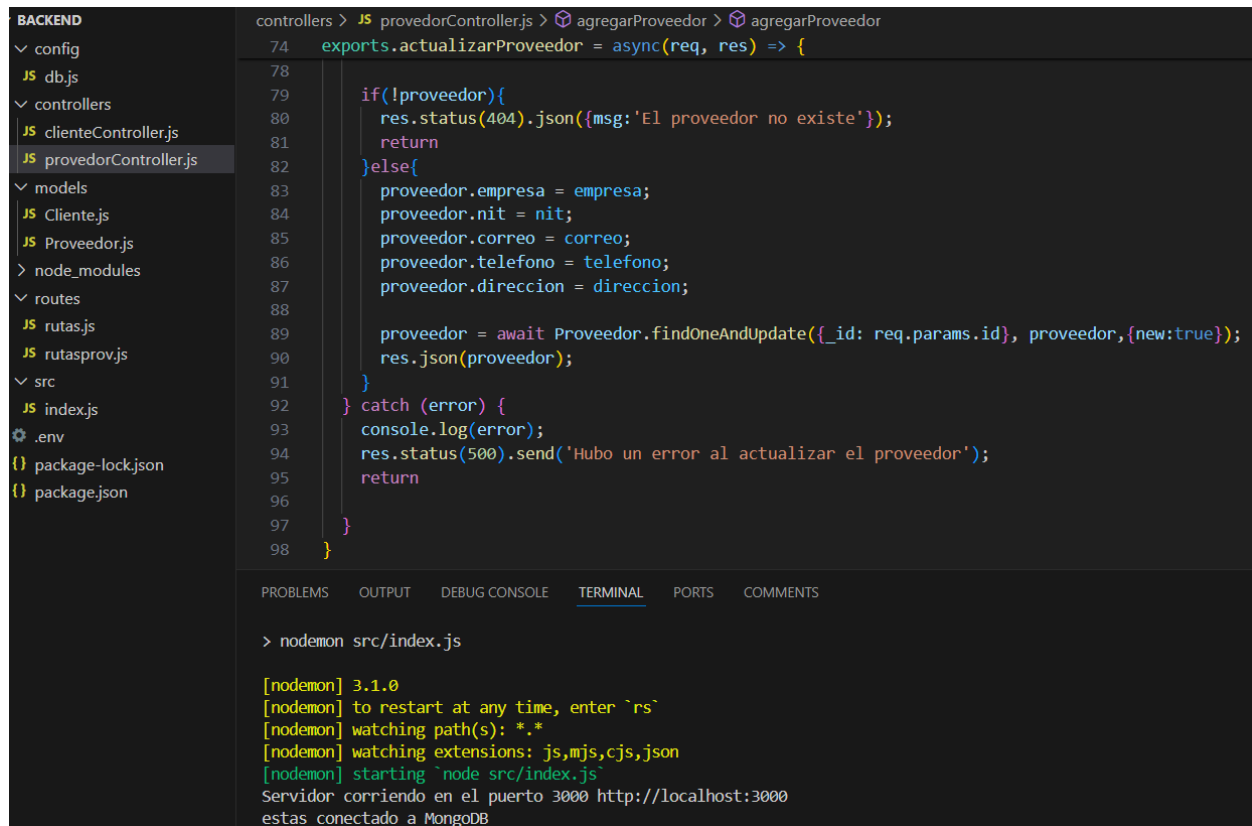
Agregamos la ruta en el index



The image shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders like config, controllers, models, routes, and src. The src folder is expanded, showing index.js selected. The code editor displays the content of index.js, which is a Node.js Express application. The code includes imports for express, db, and cors, initializes the app, connects to a database, and defines routes for clients and providers. It also has a root route for testing the server and a listen method to start the server on port 3000.

```
1  const express = require('express');
2  const conectarBD = require('../config/db');
3  const cors = require('cors');
4
5  //creamos nuestro servidor
6  const app = express();
7
8  //enlazamos la conexion con la base de datos
9
10 conectarBD();
11 app.use(cors());
12
13 app.use(express.json());
14 //definir ruta principal del proyecto
15 app.use('/api/clientes', require ('../routes/rutas'));
16 app.use('/api/proveedores', require ('../routes/rutasprov'));
17
18
19 //ruta para verificar el servidor
20 app.get('/', (req, res) => {
21   res.send('Hola mundo de la web');
22 });
23
24
25 app.listen(3000, () => {
26   console.log('Servidor corriendo en el puerto 3000 http://localhost:3000');
27 });
28
```

Iniciamos la base de datos en el terminal con NPM START



The image shows a code editor with a sidebar on the left displaying a file tree for a 'BACKEND' project. The tree includes folders for 'config', 'controllers', 'models', 'node_modules', 'routes', 'src', and files like 'db.js', 'clienteController.js', 'proveedorController.js', 'Cliente.js', 'Proveedor.js', 'index.js', '.env', 'package-lock.json', and 'package.json'. The 'proveedorController.js' file is selected and open in the main editor. The code in the editor is as follows:

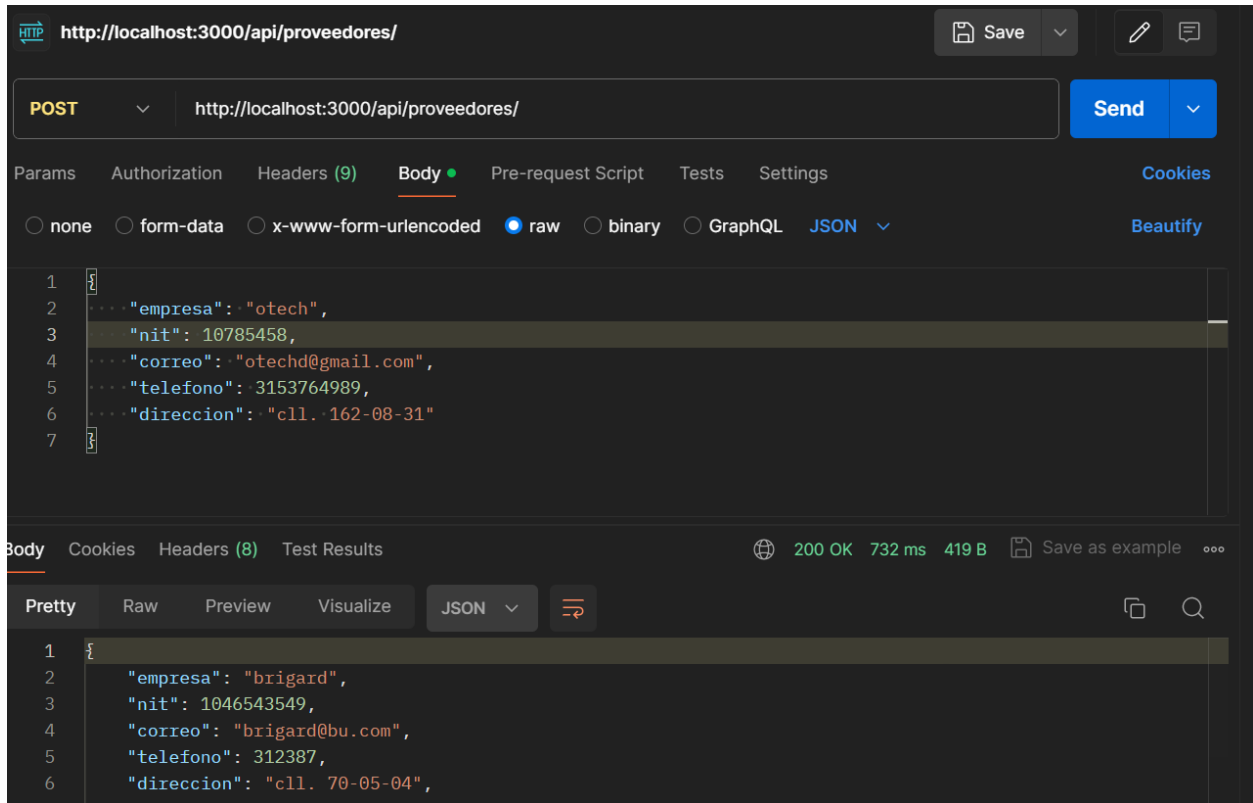
```
controllers > JS proveedorController.js > agregarProveedor > agregarProveedor
74 exports.actualizarProveedor = async(req, res) => {
78
79
80   if(!proveedor){
81     res.status(404).json({msg: 'El proveedor no existe'});
82     return
83   }else{
84     proveedor.empresa = empresa;
85     proveedor.nit = nit;
86     proveedor.correo = correo;
87     proveedor.telefono = telefono;
88     proveedor.direccion = direccion;
89
90     proveedor = await Proveedor.findOneAndUpdate({_id: req.params.id}, proveedor, {new:true});
91     res.json(proveedor);
92   }
93 } catch (error) {
94   console.log(error);
95   res.status(500).send('Hubo un error al actualizar el proveedor');
96   return
97 }
98 }
```

Below the code editor, there is a terminal window with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'COMMENTS'. The 'TERMINAL' tab is active, showing the command to start the application and the output from nodemon:

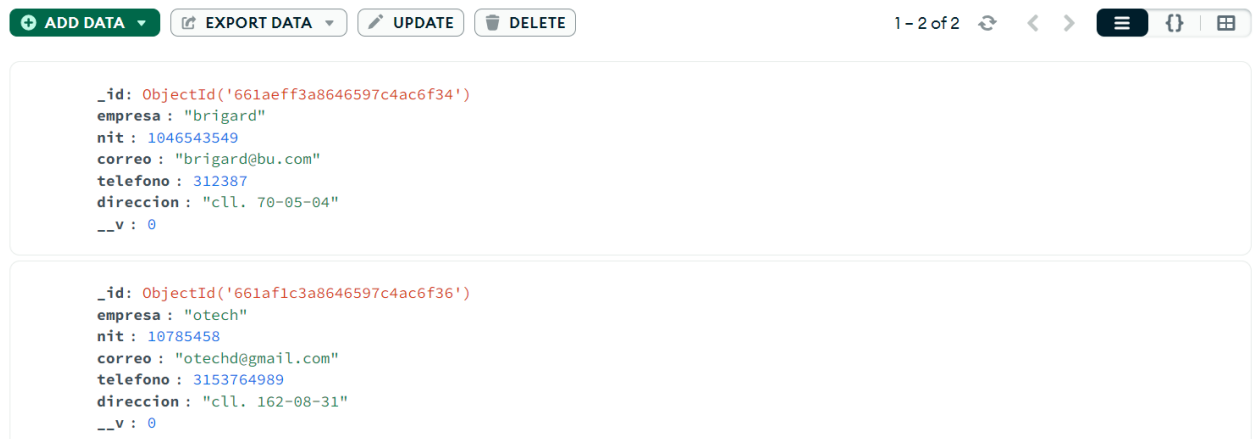
```
> nodemon src/index.js

[nodemon] 3.1.0
[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`
Servidor corriendo en el puerto 3000 http://localhost:3000
estas conectado a MongoDB
```

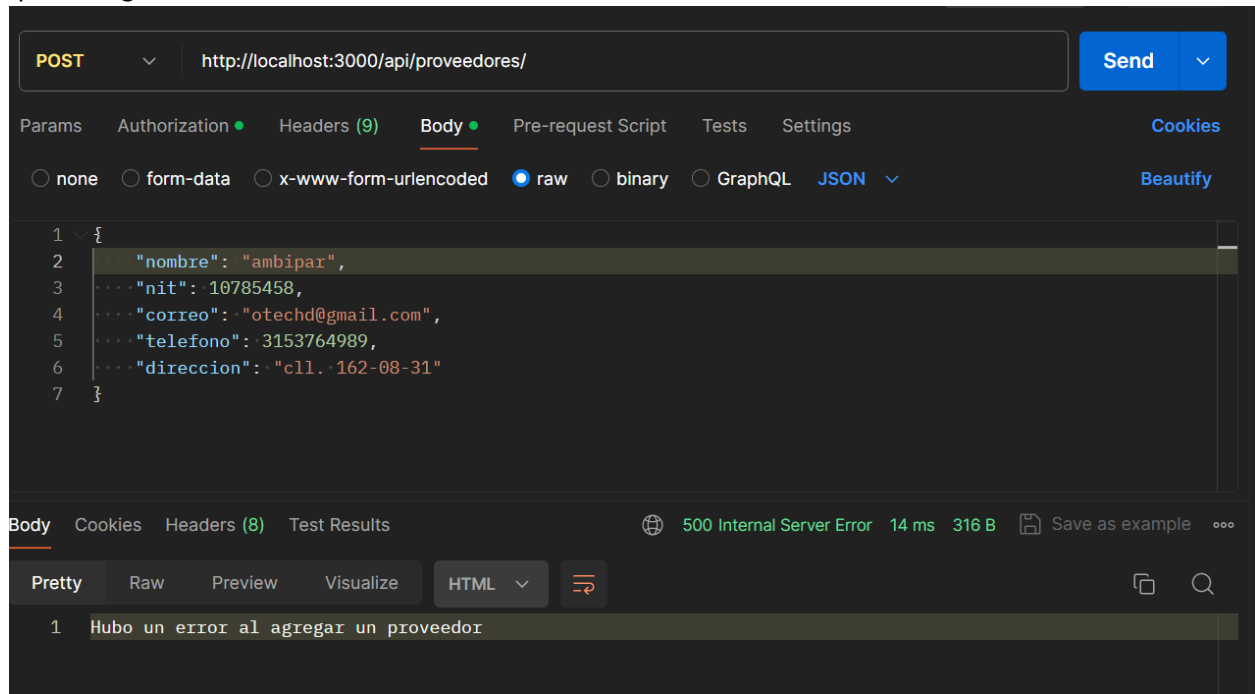
Se prueba en postman buscamos la ruta(URL) y procedemos a insertar datos a la base de datos (POST)



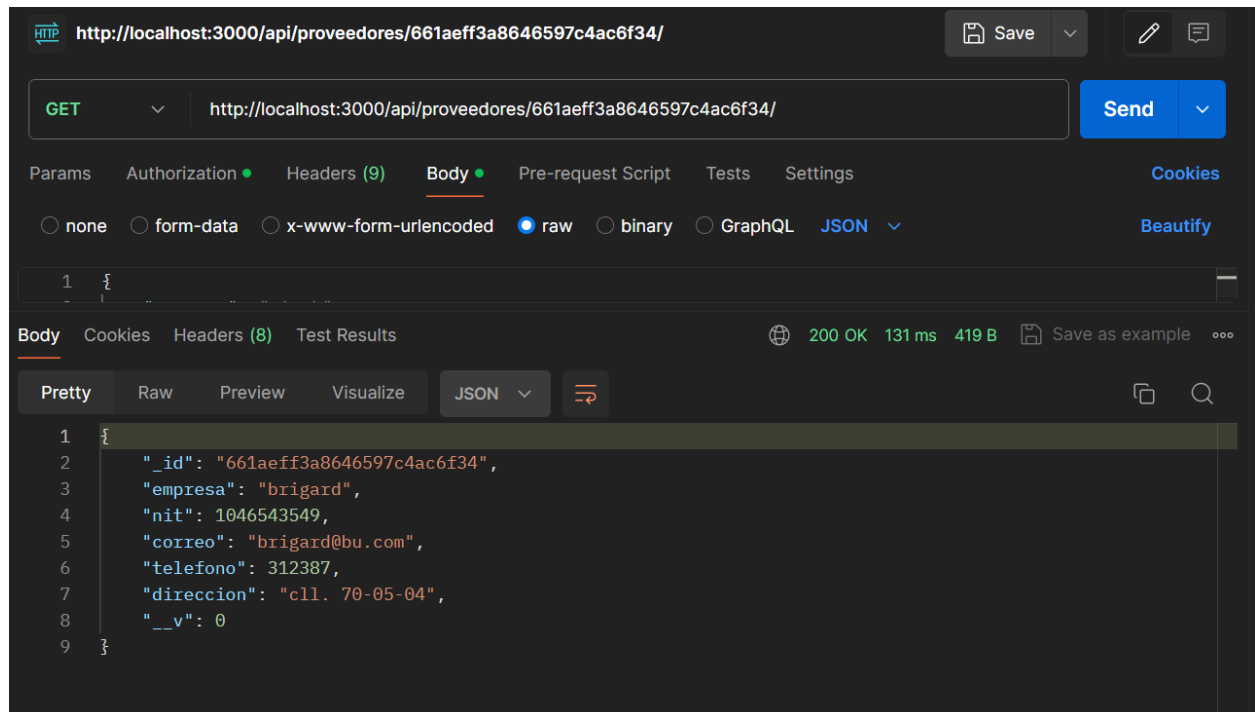
En MongoDB validamos que la inserción de datos se cargue



Validamos la inserción de datos cambiando el parámetro “empresa” por “nombre”, esto para validar que no ingrese datos erróneos.



Buscamos datos en la base de datos (GET)



Validamos con un ID falso para corroborar la busqueda

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** http://localhost:3000/api/proveedores/661aeff3a8646597c4ac6f38/
- Send Button:** A blue button labeled "Send".
- Tabs:** Params, Authorization, Headers (9), Body, Pre-request Script, Tests, Settings, Cookies, Beautify.
- Body Tab:** Selected, showing a JSON response with a "msg" field.
- Status Bar:** 404 Not Found, 103 ms, 324 B, Save as example.
- Response Body (JSON):**

```
1 {  
2   "msg": "El proveedor no fue encontrado con el ID"  
3 }
```


Probamos a modificar la información del proveedor (PUT)

```
_id: ObjectId('661af5fda8646597c4ac6f3b')
empresa: "ambipar"
nit: 10785458
correo: "ambipar@hotmail.com"
telefono: 3646564989
direccion: "c11. 85-08-116"
__v: 0
```

The screenshot shows a REST client interface with a PUT request to `http://localhost:3000/api/proveedores/661af5fda8646597c4ac6f3b`. The request body is a JSON object with the following fields: `empresa` (polar), `nit` (10785458), `correo` (ambipar.polar@hotmail.com), `telefono` (3646564989), `direccion` (c11. 85-08-116), and `__v` (0). The response is a 200 OK status with a response time of 302 ms and a size of 431 B. The response body is a JSON object with the same fields as the request, but the `__v` field is 0.

PUT `http://localhost:3000/api/proveedores/661af5fda8646597c4ac6f3b`

Params Authorization Headers (9) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "empresa": "polar",
3   "nit": 10785458,
4   "correo": "ambipar.polar@hotmail.com",
5   "telefono": 3646564989,
6   "direccion": "c11. 85-08-116"
7 }
```

Body Cookies Headers (8) Test Results 200 OK 302 ms 431 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "_id": "661af5fda8646597c4ac6f3b",
3   "empresa": "polar",
4   "nit": 10785458,
5   "correo": "ambipar.polar@hotmail.com",
6   "telefono": 3646564989,
7   "direccion": "c11. 85-08-116",
8   "__v": 0
}
```

Corroboramos el cambio de la base de datos



```
_id: ObjectId('661af5fda8646597c4ac6f3b')
empresa : "polar"
nit : 10785458
correo : "ambipar.polar@hotmail.com"
telefono : 3646564989
direccion : "cll. 85-08-116"
__v : 0
```

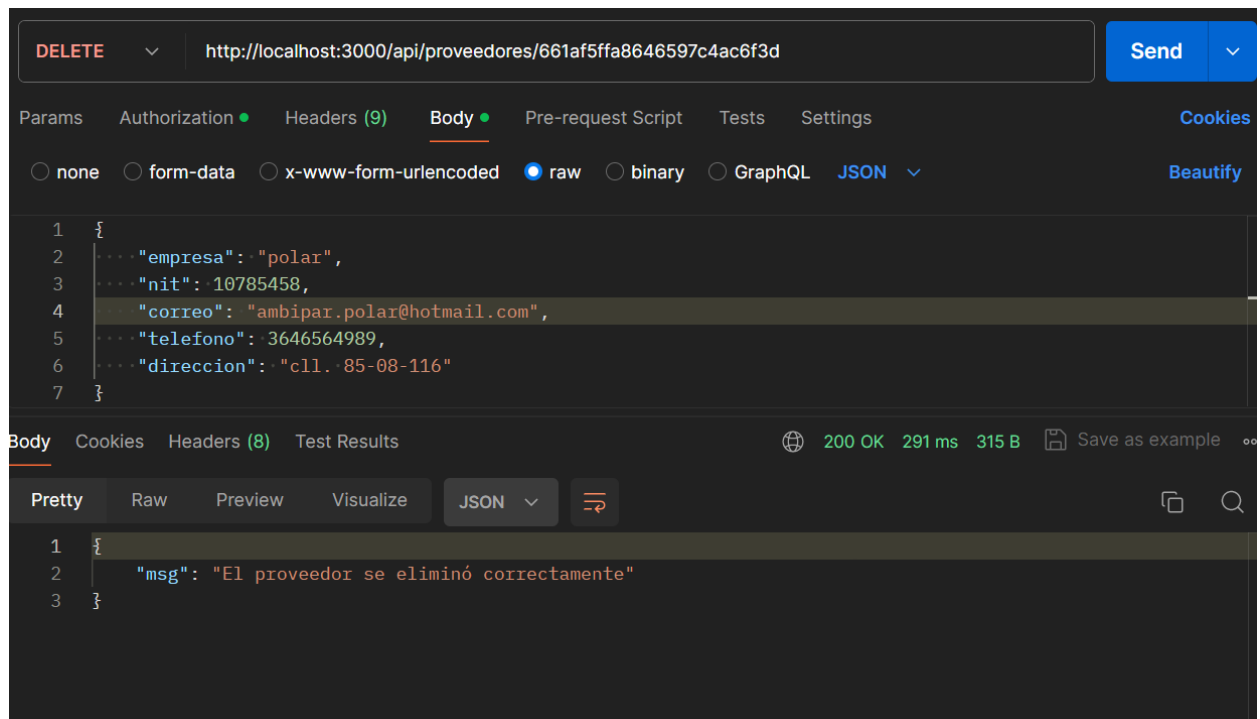
Eliminamos un registro de la base de datos (DELETE) en este caso el registro de ambipar

```
_id: ObjectId('661af5fda8646597c4ac6f3b')
empresa : "polar"
nit : 10785458
correo : "ambipar.polar@hotmail.com"
telefono : 3646564989
direccion : "cll. 85-08-116"
__v : 0
```



```
_id: ObjectId('661af5ffa8646597c4ac6f3d')
empresa : "ambipar"
nit : 10785458
correo : "ambipar@hotmail.com"
telefono : 3646564989
direccion : "cll. 85-08-116"
__v : 0
```





Verificamos en la base de datos que ya no aparezca el registro

