

Taller BackEnd

PRESENTADO POR:

Jorge Luis Rojas Muñoz

PRESENTADO A:

Vicente Aux Revelo

UNIVERSIDAD DE NARIÑO

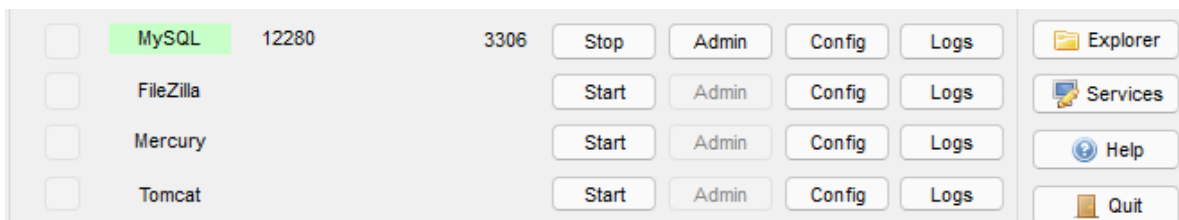
INGENIERIA DE SISTEMAS

DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL
DESARROLLO DE SOFTWARE

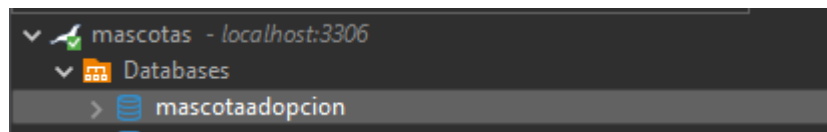
2024

1. Creación de la Base de Datos para la Empresa de Adopción de Mascotas

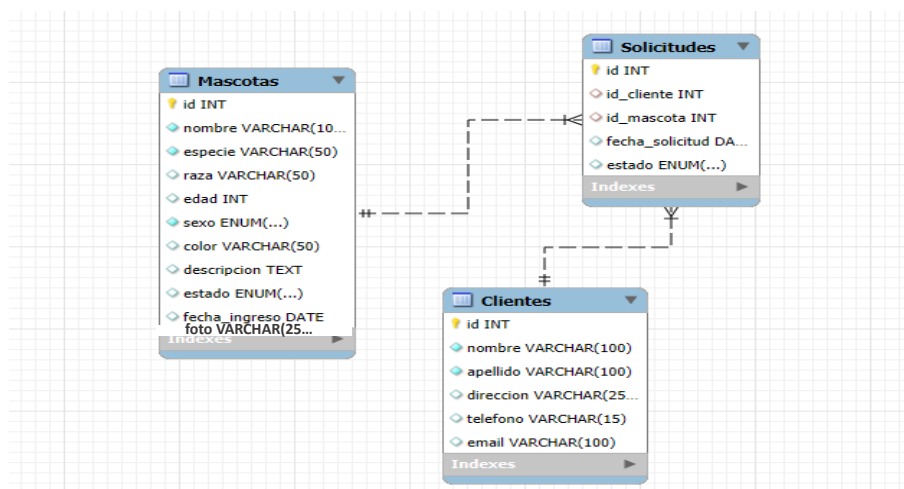
Primeramente en la conexión mascota ya creada anteriormente se hizo creación de una base de datos en MariaDB, destinada a la gestión de una empresa de adopción de mascotas la cual se le puso el nombre de **“mascotaadopcion”**, la base de datos permitirá administrar el registro de las mascotas disponibles para adopción, así como las solicitudes de adopción realizadas por posibles adoptantes, el entorno utilizado para su desarrollo fue DBeaver, en las imágenes siguientes se puede evidenciar que la base de datos esta creada además de un esquema relacional de las tablas que se van a crear después dentro del proyecto backend haciendo la respectiva programación no sin antes inicializar el servicio con xampp.



inicializar xampp para el servicio



creación base de datos



esquema relacional

2. Inicialización del proyecto Node.js:

Se utilizó el comando **npm init -y** para inicializar rápidamente un nuevo proyecto de Node.js en el directorio **TallerbackEnd**, este comando genera automáticamente el archivo **package.json**, que contiene la configuración inicial del proyecto (como el nombre, versión, descripción, scripts, etc.)

Posteriormente, se ejecutó el comando **npm install express mysql2 sequelize**, que instaló tres dependencias, **Express**, **mysql2**, **Sequelize**, en la imagen siguiente se evidencia lo dicho anteriormente.

```
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEnd> npm init -y
Wrote to C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEnd\package.json:

{
  "name": "tallerbackend",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEnd> npm install express mysql2 sequelize

added 98 packages, and audited 99 packages in 11s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

inicialización Backend

Luego, se actualizo el nombre del directorio del proyecto a **TallerbackEndPawPal**, reflejando el nombre de la empresa, además, se instaló la dependencia **nodemon** con el comando **npm install nodemon -D** nodemon es una herramienta de desarrollo que permite reiniciar automáticamente el servidor cada vez que se detectan cambios en los archivos, lo que facilita el proceso de desarrollo sin necesidad de reiniciar manualmente la aplicación,

```
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> npm install nodemon -D

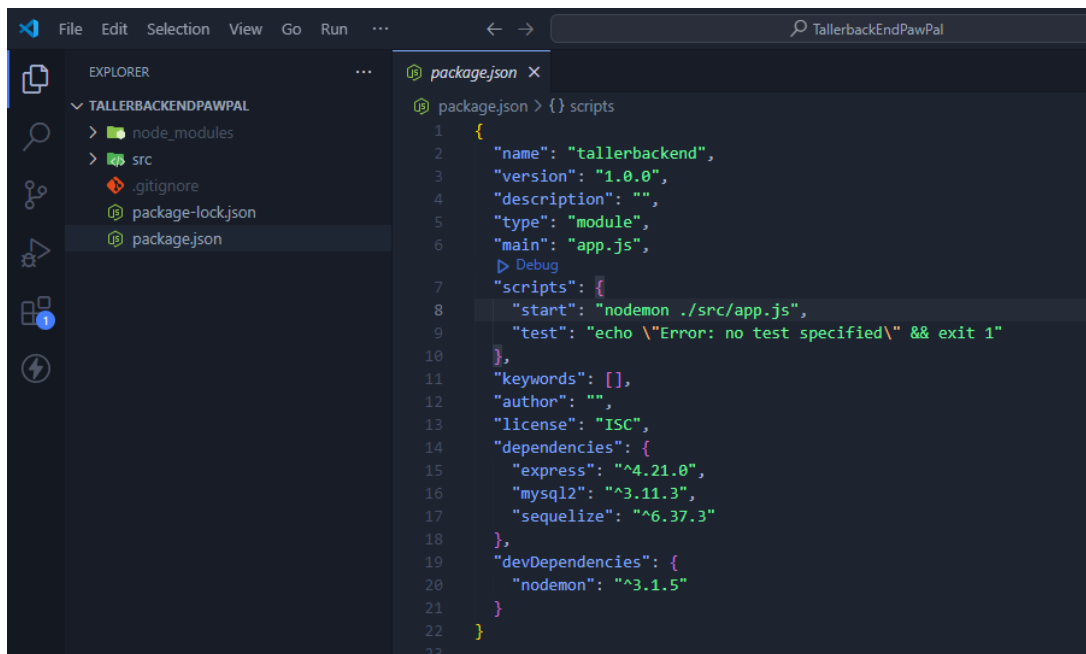
added 28 packages, and audited 127 packages in 4s

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

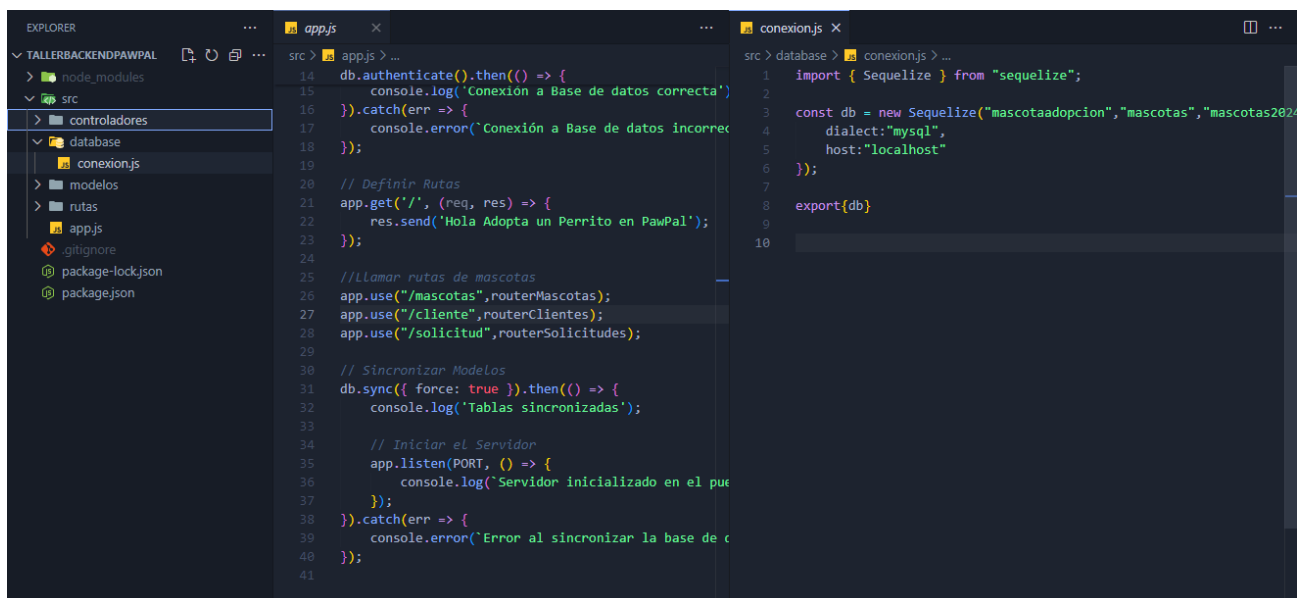
esta instalación se realizó como dependencia de desarrollo (-D), ya que nodemon solo se utiliza durante el desarrollo y no en producción según lo aprendido en clases .

Una vez hecho el paso anterior ya podemos mirar en visual studio el modelo que se creó a partir de los comandos, en **package.json** se realizó los cambios que son **"type": "module"** y para que pueda empezar a correr nodemon se puso la línea 8 **"start": "nodemon ./src/app.js"**. El archivo **app.js** se creó posteriormente para comprobar la conexión, además se creó un archivo **.gitignore** para que al guardar en git no se suba la carpeta de **node_modules** al repositorio de git con el fin de que el proyecto quede más limpio y liviano luego solo se clonaría el proyecto y en la terminal se volvería a ejecutar **npm install** y se agregara nuevamente la carpeta **node_modules**.



```
1 {
2   "name": "tallerbackend",
3   "version": "1.0.0",
4   "description": "",
5   "type": "module",
6   "main": "app.js",
7   "scripts": {
8     "start": "nodemon ./src/app.js",
9     "test": "echo \\\"Error: no test specified\\\" && exit 1"
10  },
11  "keywords": [],
12  "author": "",
13  "license": "ISC",
14  "dependencies": {
15    "express": "^4.21.0",
16    "mysql2": "^3.11.3",
17    "sequelize": "^6.37.3"
18  },
19  "devDependencies": {
20    "nodemon": "^3.1.5"
21  }
22 }
```

Se creó el archivo **app.js** donde se hizo la programación para verificar la conexión a la base de datos y dentro de **src** se creó la carpeta **database** donde en ella se creó el archivo **conexion.js** el cual nos permitió realizar la conexión a la base de datos, y también se creó las carpetas como **controladores**, **modelos** y **rutas** que serán necesarias para ordenar los archivos necesarios para el desarrollo del proyecto de adopción de mascotas.



```
14 db.authenticate().then(() => {
15   console.log('Conexión a Base de datos correcta');
16 }).catch(err => {
17   console.error('Conexión a Base de datos incorrecta');
18 });
19
20 // Definir Rutas
21 app.get('/', (req, res) => {
22   res.send('Hola Adopta un Perrito en PawPal');
23 });
24
25 // Llamar rutas de mascotas
26 app.use('/mascotas', routerMascotas);
27 app.use('/cliente', routerClientes);
28 app.use('/solicitud', routerSolicitudes);
29
30 // Sincronizar Modelos
31 db.sync({ force: true }).then(() => {
32   console.log('Tablas sincronizadas');
33 });
34
35 // Iniciar el Servidor
36 app.listen(PORT, () => {
37   console.log('Servidor inicializado en el puerto ' + PORT);
38 });
39
40 // Manejar errores
41 app.use((err, req, res, next) => {
42   console.error(err.stack);
43   res.status(500).send('Error al sincronizar la base de datos');
44 });
```

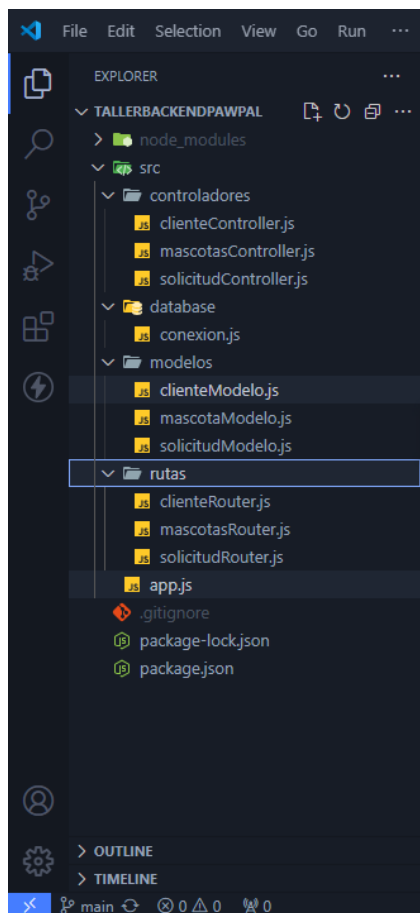
en la imagen siguiente se observa como al ejecutar en la terminal el comando **npm start** se inicia **nodemon** configurado en el archivo **app.js** y la base de datos satisfactoriamente se inicializa en el puerto 4000 y se hace una conexión exitosa con la base de datos.

```
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> npm start

> tallerbackend@1.0.0 start
> nodemon ./src/app.js

[nodemon] 3.1.5
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] restarting due to changes...
[nodemon] starting `node ./src/app.js`
Servidor Inicializado en el puerto 4000
Executing (default): SELECT 1+1 AS result
Conexion a Base de datos correcta
```

Como parte final del proyecto completo esta la estructura del backend con sus respectivas rutas, controladores y modelos para que implemente los cambios en la base de datos de la siguiente manera como se evidencia en la siguiente imagen:



En el repositorio remoto de github se podra ver de manera mas detallada el codigo respectivo de cada archivo, centrandome en la estructura quedaron en,

rutas: clienteRouter.js, mascotasRouter.js y solicitudrouter.js (son responsables de mapear las peticiones HTTP (GET, POST, PUT, DELETE, etc.))

modelos: clienteModelo.js, mascotaModelo.js, solicitudModelo.js (representan la estructura de los datos y cómo estos se almacenan en la base de datos.)

controladores: clienteController.js, mascotasController y solictudController.js (son los responsables de manejar la lógica de la aplicación, como procesar las solicitudes, manipular los datos usando los modelos y enviar la respuesta adecuada al cliente.)

Por último, se fue haciendo uso de git y del repositorio github para ir guardando parte por parte del desarrollo del proyecto:

Primero se hizo la inicialización del proyecto luego se agregó los cambios y se generó el commit con la conexión de la base de datos.

Posteriormente, se estableció la conexión con github para que se guarde en el repositorio remoto.

```
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git init
Initialized empty Git repository in C:/Users/7Jrmo7/Documents/Diplomado2024B/TallerbackEndPawPal/.git/
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git commit -m "commit conexión"
[main (root-commit) ffe048e] commit conexión
3 files changed, 1405 insertions(+)
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 src/app.js
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git remote add origin https://github.com/JorgeMJ-14/TallerbackEndPawPal.git
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git push -u origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 14.51 KiB | 1.81 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JorgeMJ-14/TallerbackEndPawPal.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal>
```

luego se hacen los commits correspondientes a mascota, cliente y solicitud con la implementación de sus 3 archivos mencionados en la parte final del proyecto anteriormente además en la imagen se observa la descripción de los cambios realizados y actualizados en el repositorio remoto.

```
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git add .
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git commit -m "commit base de datos mascotaadopcion"
[main d2a87e8] commit base de datos mascotaadopcion
2 files changed, 16 insertions(+)
create mode 100644 src/database/conexion.js
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git add .
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git commit -m "commit actualización mascota (modelo,router,controller)"
[main 1f2efb8] commit actualización mascota (modelo,router,controller)
4 files changed, 179 insertions(+), 18 deletions(-)
create mode 100644 src/controladores/mascotasController.js
create mode 100644 src/modelos/mascotaModelo.js
create mode 100644 src/rutas/mascotasRouter.js
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git add .
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git commit -m "commit actualización cliente (modelo,router,controller)"
[main c2f773e] commit actualización cliente (modelo,router,controller)
5 files changed, 125 insertions(+), 5 deletions(-)
create mode 100644 src/controladores/clienteController.js
create mode 100644 src/modelos/clienteModelo.js
create mode 100644 src/rutas/clienteRouter.js
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git add .
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git commit -m "commit actualización solicitud (modelo,router,controller)"
[main 063ff90] commit actualización solicitud (modelo,router,controller)
4 files changed, 121 insertions(+), 1 deletion(-)
create mode 100644 src/controladores/solicitudController.js
create mode 100644 src/modelos/solicitudModelo.js
create mode 100644 src/rutas/solicitudRouter.js
PS C:\Users\7Jrmo7\Documents\Diplomado2024B\TallerbackEndPawPal> git push
Enumerating objects: 40, done.
Counting objects: 100% (40/40), done.
Delta compression using up to 8 threads
Compressing objects: 100% (33/33), done.
Writing objects: 100% (37/37), 6.06 KiB | 563.00 KiB/s, done.
Total 37 (delta 12), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (12/12), completed with 1 local object.
To https://github.com/JorgeMJ-14/TallerbackEndPawPal.git
ffe048e..063ff90  main -> main
```

3. Verificación de las diferentes operaciones a través de Thunder Client

Verificación operaciones mascotas:

Verificación crear Mascota → “/crearM”

Thunder Client interface showing a successful POST request to `http://127.0.0.1:4000/mascotas/crearM`. The status is **200 OK**, size is **281 Bytes**, and time is **132 ms**. The response is a JSON object:

```
{
  "estado": "Disponible",
  "id": 1,
  "nombre": "Max",
  "especie": "Perro",
  "raza": "Labrador",
  "edad": 3,
  "sexo": "Macho",
  "color": "Marrón",
  "descripcion": "Un perro muy enérgico y amigable.",
  "foto": "url_ejemplofoto.jpg",
  "updatedAt": "2024-09-20T23:51:53.414Z",
  "createdAt": "2024-09-20T23:51:53.414Z"
}
```

Thunder Client interface showing a successful POST request to `http://127.0.0.1:4000/mascotas/crearM`. The status is **200 OK**, size is **284 Bytes**, and time is **31 ms**. The response is a JSON object:

```
{
  "estado": "Disponible",
  "id": 2,
  "nombre": "Orus",
  "especie": "Perro",
  "raza": "Golden Retriever",
  "edad": 3,
  "sexo": "Macho",
  "color": "negro",
  "descripcion": "Un perro muy obediente y fiel.",
  "foto": "url_ejemplo2foto.jpg",
  "updatedAt": "2024-09-20T23:53:32.502Z",
  "createdAt": "2024-09-20T23:53:32.502Z"
}
```

Verificación base de datos en dbeaver para evidenciar que los datos se estén creando correctamente:

DBeaver interface showing the `mascotas` table. The table contains the following data:

	id	nombre	especie	raza	edad	sexo	color	descripcion	estado
1	1	Max	Perro	Labrador	3	Macho	Marrón	Un perro muy enérgico y amigable.	Disponible
2	2	Orus	Perro	Golden Retriever	3	Macho	negro	Un perro muy obediente y fiel.	Disponible

Verificación buscar todas las Mascotas → “/buscarM”

mascoRouter.js TC 127.0.0.1:4000/mascotas/c... X

GET http://127.0.0.1:4000/mascotas/buscarM Send

Status: 200 OK Size: 610 Bytes Time: 33 ms

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nombre": "Orus",
3   "especie": "Perro",
4   "raza": "Golden Retriever",
5   "edad": 3,
6   "sexo": "Macho",
7   "color": "negro",
8   "descripcion": "Un perro muy obediente y fiel.",
9   "foto": "url_ejemplo2foto.jpg"
10 }
11
```

Response Headers 7 Cookies Results Docs {}

```
3   "id": 1,
4   "nombre": "Max",
5   "especie": "Perro",
6   "raza": "Labrador",
7   "edad": 3,
8   "sexo": "Macho",
9   "color": "Marrón",
10  "descripcion": "Un perro muy enérgico y amigable.",
11  "estado": "Disponible",
12  "fecha_ingreso": null,
13  "foto": "url_ejemplofoto.jpg",
14  "createdAt": "2024-09-20T23:51:53.000Z",
15  "updatedAt": "2024-09-20T23:51:53.000Z"
16 },
17 {
18   "id": 2,
19   "nombre": "Orus",
20   "especie": "Perro",
21   "raza": "Golden Retriever",
22   "edad": 3,
23   "sexo": "Macho",
24   "color": "negro",
25   "descripcion": "Un perro muy obediente y fiel.",
26   "estado": "Disponible",
27   "fecha_ingreso": null,
28   "foto": "url_ejemplo2foto.jpg",
29   "createdAt": "2024-09-20T23:53:32.000Z",
30   "updatedAt": "2024-09-20T23:53:32.000Z"
31 }
32 ]
```

Response Chart

mascoRouter.js TC 127.0.0.1:4000/mascotas/c... X

GET http://127.0.0.1:4000/mascotas/buscarM Send

Status: 200 OK Size: 610 Bytes Time: 33 ms

Query Headers 2 Auth Body 1 Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nombre": "Orus",
3   "especie": "Perro",
4   "raza": "Golden Retriever",
5   "edad": 3,
6   "sexo": "Macho",
7   "color": "negro",
8   "descripcion": "Un perro muy obediente y fiel.",
9   "foto": "url_ejemplo2foto.jpg"
10 }
11
```

Response Headers 7 Cookies Results Docs {}

```
12   "fecha_ingreso": null,
13   "foto": "url_ejemplofoto.jpg",
14   "createdAt": "2024-09-20T23:51:53.000Z",
15   "updatedAt": "2024-09-20T23:51:53.000Z"
16 },
17 {
18   "id": 2,
19   "nombre": "Orus",
20   "especie": "Perro",
21   "raza": "Golden Retriever",
22   "edad": 3,
23   "sexo": "Macho",
24   "color": "negro",
25   "descripcion": "Un perro muy obediente y fiel.",
26   "estado": "Disponible",
27   "fecha_ingreso": null,
28   "foto": "url_ejemplo2foto.jpg",
29   "createdAt": "2024-09-20T23:53:32.000Z",
30   "updatedAt": "2024-09-20T23:53:32.000Z"
31 }
32 ]
```

Response Chart

Verificación buscar Mascota por id → “/buscarIdM/:id” → consulta id=2

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

- Request:** GET `http://127.0.0.1:4000/mascotas/buscarIdM/2`
- Status:** 200 OK, Size: 305 Bytes, Time: 31 ms
- Response Body (JSON):**

```
1 {
2   "id": 2,
3   "nombre": "Orus",
4   "especie": "Perro",
5   "raza": "Golden Retriever",
6   "edad": 3,
7   "sexo": "Macho",
8   "color": "negro",
9   "descripcion": "Un perro muy obediente y fiel.",
10  "estado": "Disponible",
11  "fecha_ingreso": null,
12  "foto": "url_ejemplo2foto.jpg",
13  "createdAt": "2024-09-20T23:53:32.000Z",
14  "updatedAt": "2024-09-20T23:53:32.000Z"
15 }
```

Verificación buscar Mascota por id → “/buscarIdM/:id” → consulta id=1

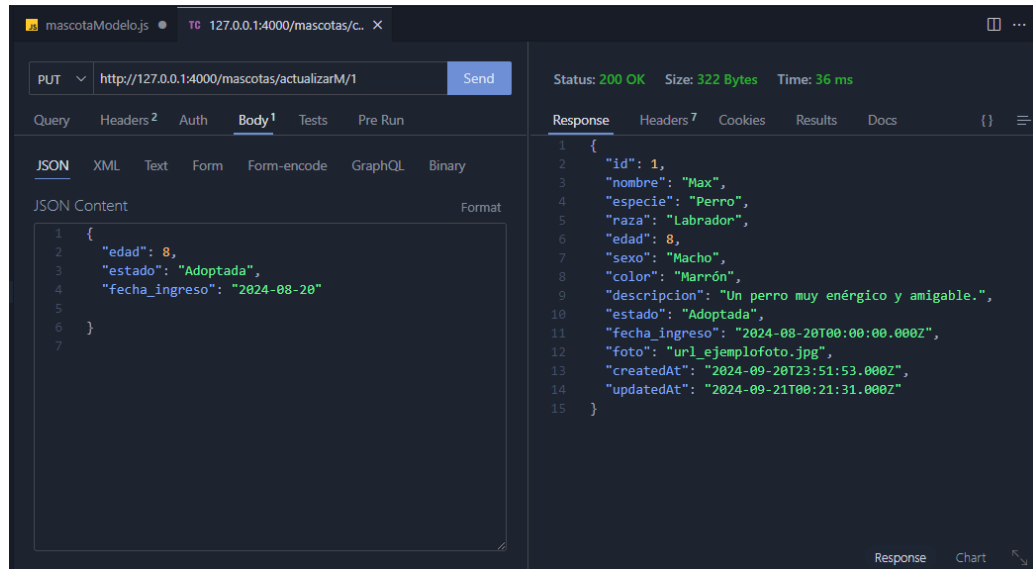
The screenshot shows a REST client interface with the following details:

- Request:** GET `http://127.0.0.1:4000/mascotas/buscarIdM/1`
- Status:** 200 OK, Size: 302 Bytes, Time: 17 ms
- Response Body (JSON):**

```
1 {
2   "id": 1,
3   "nombre": "Max",
4   "especie": "Perro",
5   "raza": "Labrador",
6   "edad": 3,
7   "sexo": "Macho",
8   "color": "Marrón",
9   "descripcion": "Un perro muy energético y amigable.",
10  "estado": "Disponible",
11  "fecha_ingreso": null,
12  "foto": "url_ejemplofoto.jpg",
13  "createdAt": "2024-09-20T23:51:53.000Z",
14  "updatedAt": "2024-09-20T23:51:53.000Z"
15 }
```

Verificación actualizar Mascota → '/actualizarM/:id' → actualización mascota id=1

Datos actualizados: edad, estado de adopción y fecha de ingreso

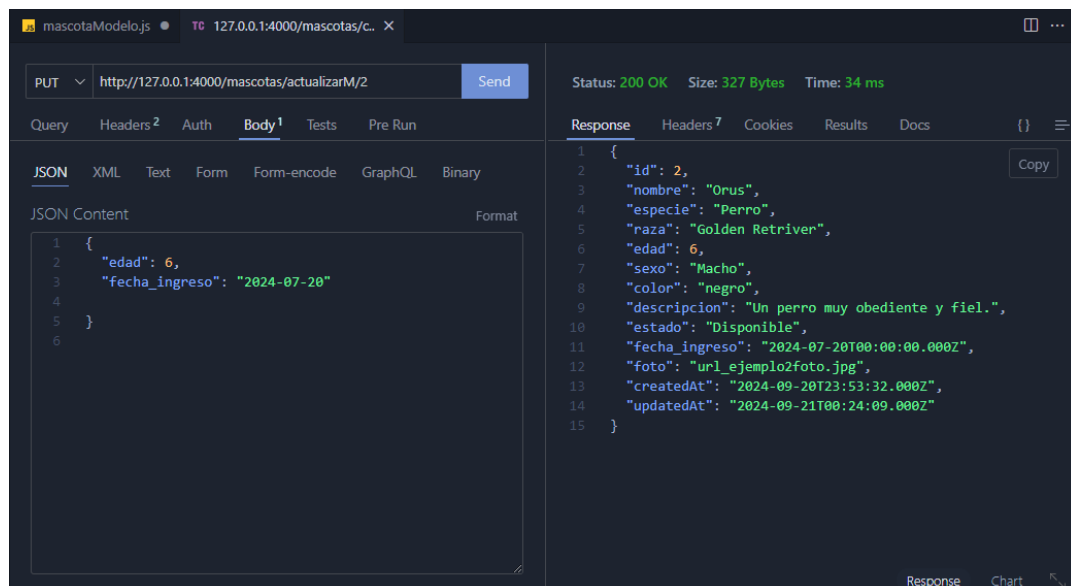


Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

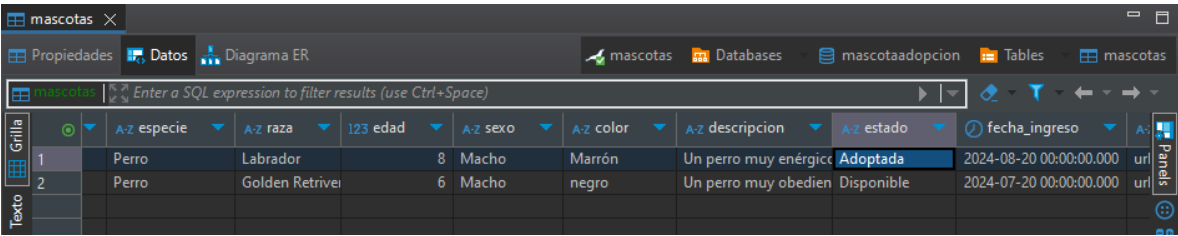
	A-Z especie	A-Z raza	edad	A-Z sexo	A-Z color	A-Z descripcion	A-Z estado	fecha_ingreso	A-Z url
1	Perro	Labrador	8	Macho	Marrón	Un perro muy energético	Adoptada	2024-08-20 00:00:00.000	url
2	Perro	Golden Retriever	6	Macho	negro	Un perro muy obediente	Disponible	2024-07-20 00:00:00.000	url

Verificación actualizar Mascota → '/actualizarM/:id' → actualización mascota id=1

Datos actualizados: edad y fecha de ingreso

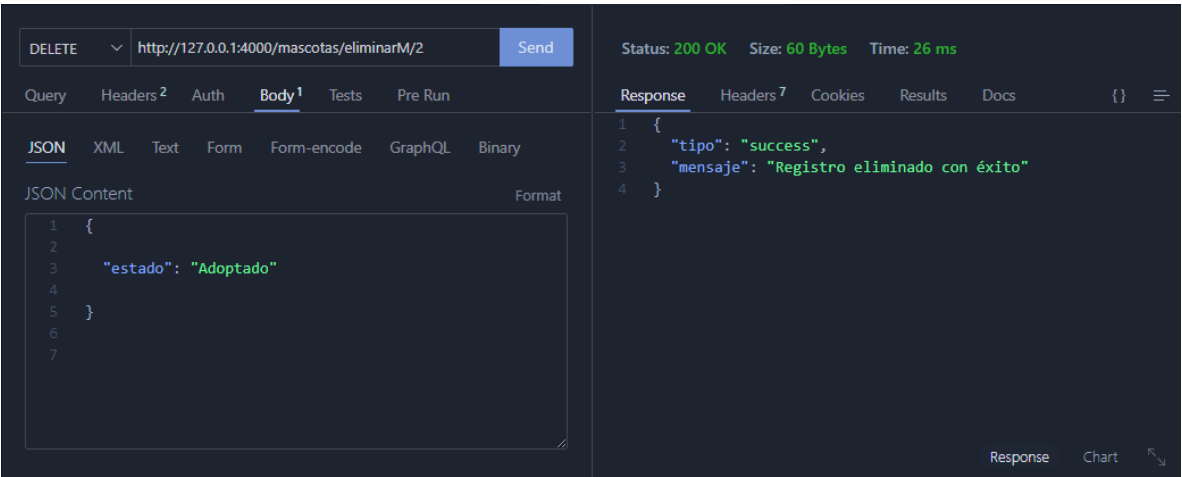


Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:



	especie	raza	edad	sexo	color	descripcion	estado	fecha_ingreso	url
1	Perro	Labrador	8	Macho	Marrón	Un perro muy energético	Adoptada	2024-08-20 00:00:00.000	url
2	Perro	Golden Retriever	6	Macho	negro	Un perro muy obediente	Disponible	2024-07-20 00:00:00.000	url

Verificación eliminar Mascota → “/eliminarM/:id” → eliminar id=2



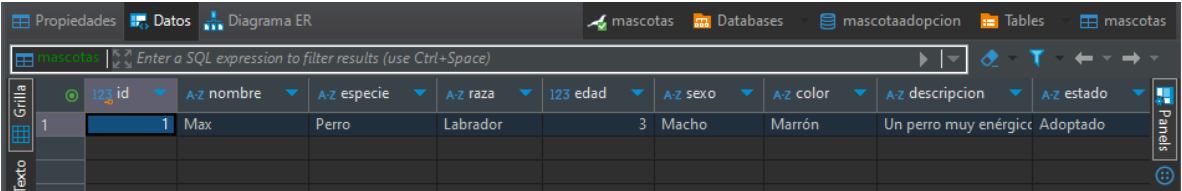
DELETE <http://127.0.0.1:4000/mascotas/eliminarM/2> Send

Status: 200 OK Size: 60 Bytes Time: 26 ms

Response

```
{
  "tipo": "success",
  "mensaje": "Registro eliminado con éxito"
}
```

Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:



	id	nombre	especie	raza	edad	sexo	color	descripcion	estado
1	1	Max	Perro	Labrador	3	Macho	Marrón	Un perro muy energético	Adoptado

Verificación operaciones clientes:

Verificación crear Cliente → “/crearCliente”

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:4000/cliente/crearCliente`. The request body is a JSON object representing a client. The response is a 200 OK status with a JSON body containing the created client's details.

Request:

```
POST http://127.0.0.1:4000/cliente/crearCliente
```

Request Body (JSON):

```
{  "nombre": "Juan Perez",  "email": "juan.perez@example.com",  "telefono": "555-1234",  "direccion": "Calle Falsa 123"}
```

Response: Status: 200 OK, Size: 193 Bytes, Time: 31 ms

Response Body (JSON):

```
{  "id": 1,  "nombre": "Juan Perez",  "email": "juan.perez@example.com",  "telefono": "555-1234",  "direccion": "Calle Falsa 123",  "updatedAt": "2024-09-21T00:53:37.642Z",  "createdAt": "2024-09-21T00:53:37.642Z"}
```

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:4000/cliente/crearCliente`. The request body is a JSON object representing a client. The response is a 200 OK status with a JSON body containing the created client's details.

Request:

```
POST http://127.0.0.1:4000/cliente/crearCliente
```

Request Body (JSON):

```
{  "nombre": "carlos Toro",  "email": "Toro@example.com",  "telefono": "888-666",  "direccion": "Calle nueva 123"}
```

Response: Status: 200 OK, Size: 187 Bytes, Time: 28 ms

Response Body (JSON):

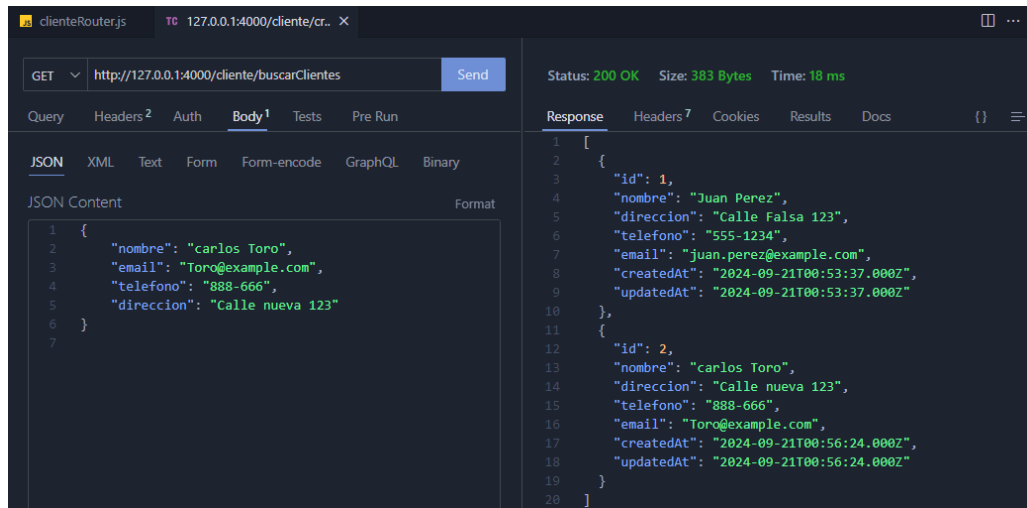
```
{  "id": 2,  "nombre": "carlos Toro",  "email": "Toro@example.com",  "telefono": "888-666",  "direccion": "Calle nueva 123",  "updatedAt": "2024-09-21T00:56:24.803Z",  "createdAt": "2024-09-21T00:56:24.803Z"}
```

Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

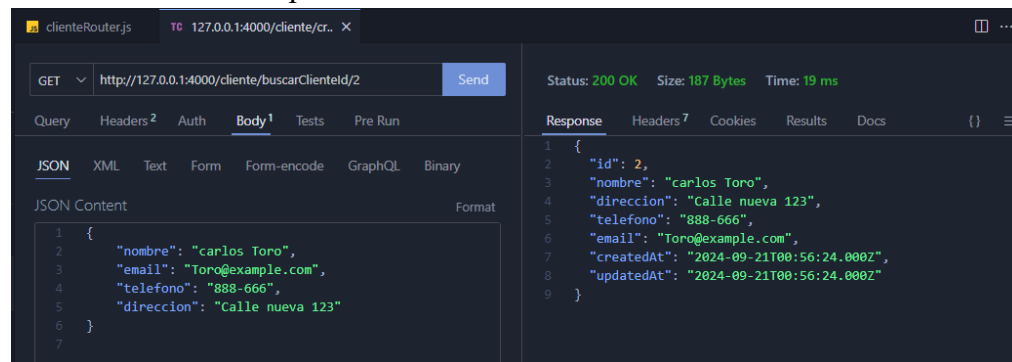
The screenshot shows a DBeaver interface displaying a table of clients. The table has columns for id, nombre, direccion, telefono, email, createdAt, and updatedAt. Two rows are visible, corresponding to the clients created in the previous screenshots.

	id	nombre	direccion	telefono	email	createdAt	updatedAt
1	1	Juan Perez	Calle Falsa 123	555-1234	juan.perez@example.com	24-09-21 00:53:37.000	24-09-21 00:53:37.000
2	2	carlos Toro	Calle nueva 123	888-666	Toro@example.com	24-09-21 00:56:24.000	24-09-21 00:56:24.000

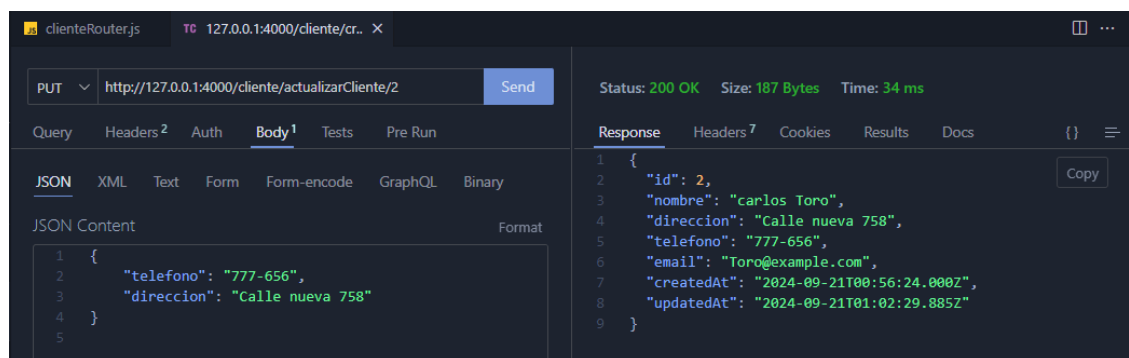
Verificación buscar todos los clientes → “/buscarClientes”



Verificación buscar Cliente por id → “/buscarClienteId/:id” → consulta id=2



Verificación actualizar Cliente → “/actualizarCliente/:id” → actualización cliente id=2

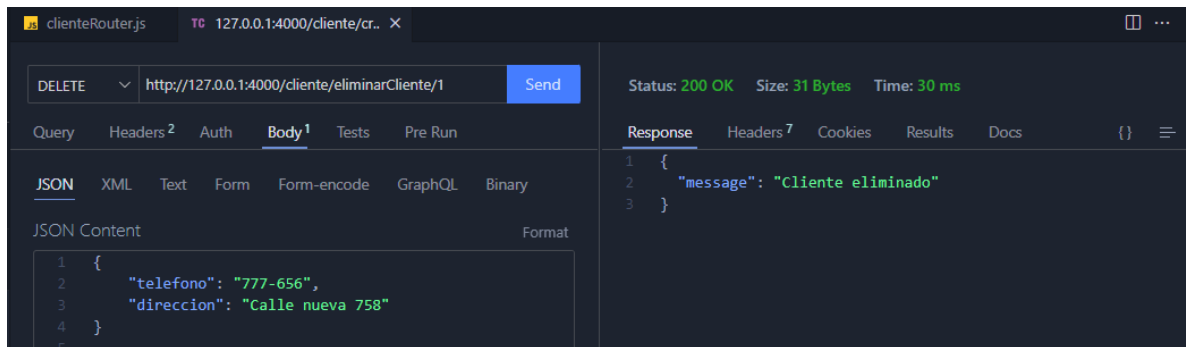


Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

The screenshot shows the DBeaver interface with a table of client data. The table has columns: id, nombre, direccion, telefono, email, createdAt, and updatedAt. The data is as follows:

	id	nombre	direccion	telefono	email	createdAt	updatedAt
1	1	Juan Perez	Calle Falsa 123	555-1234	juan.perez@exar	4-09-21 00:53:37.000	24-09-21 00:53:37.000
2	2	carlos Toro	Calle nueva 758	777-656	Toro@example.c	4-09-21 00:56:24.000	24-09-21 01:02:29.000

Verificación eliminar Cliente → “/eliminarM/:id” → eliminar id=1

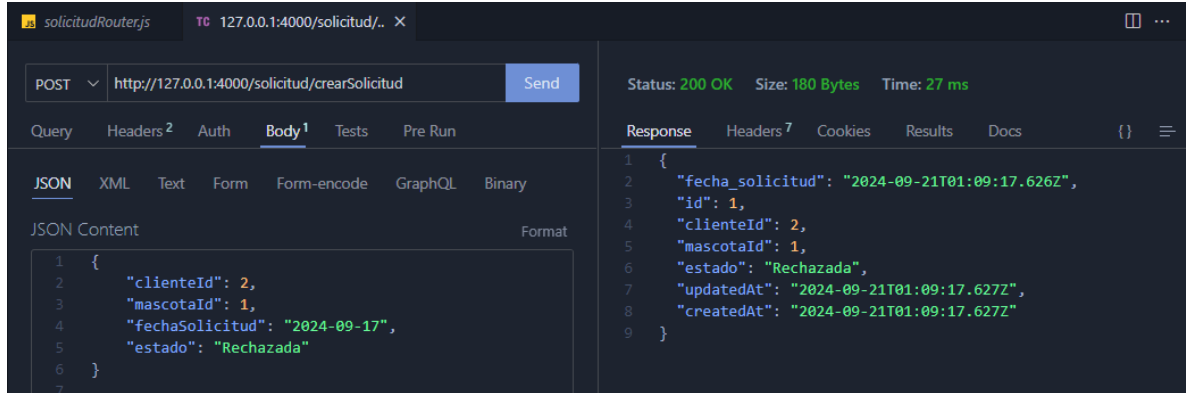


Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

id	nombre	direccion	telefono	email	createdAt	updatedAt
2	carlos Toro	Calle nueva 758	777-656	Toro@example.com	24-09-21 00:56:24.000	24-09-21 01:02:29.000

Verificación operaciones Solicitudes:

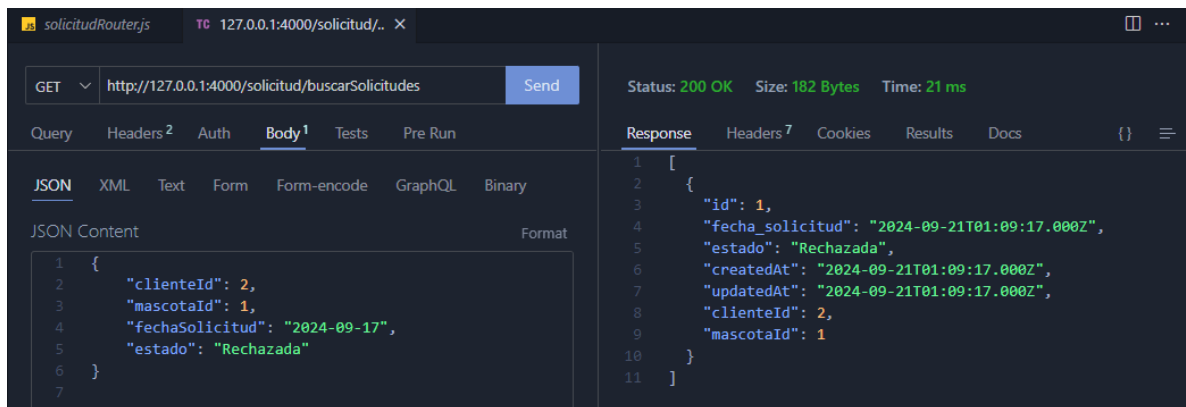
Verificación crear Solicitud → “/crearSolicitud”



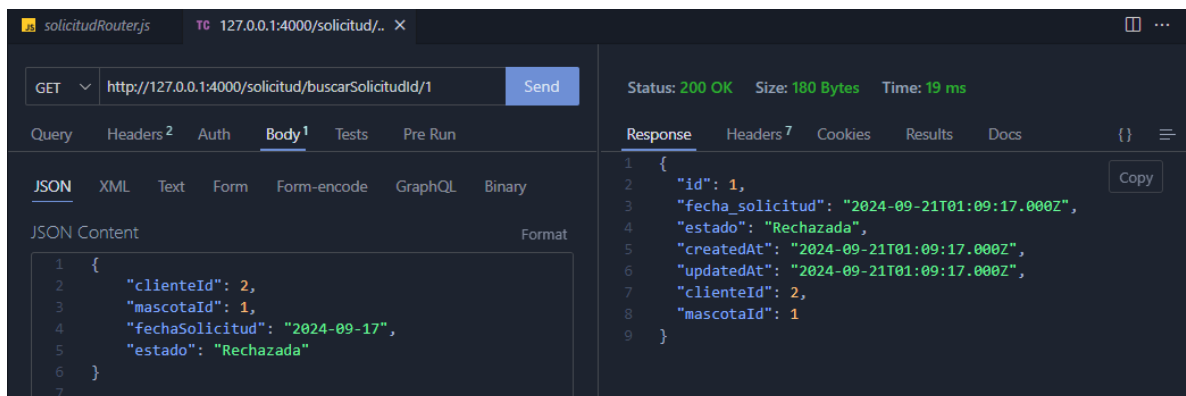
Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

id	fecha_solicitud	estado	createdAt	updatedAt	clienteId	mascotaId
1	2024-09-21 01:09:17.000	Rechazada	24-09-21 01:09:17.000	24-09-21 01:09:17.000	2	1

Verificación buscar todas solicitudes → “/buscarSolicitudes”

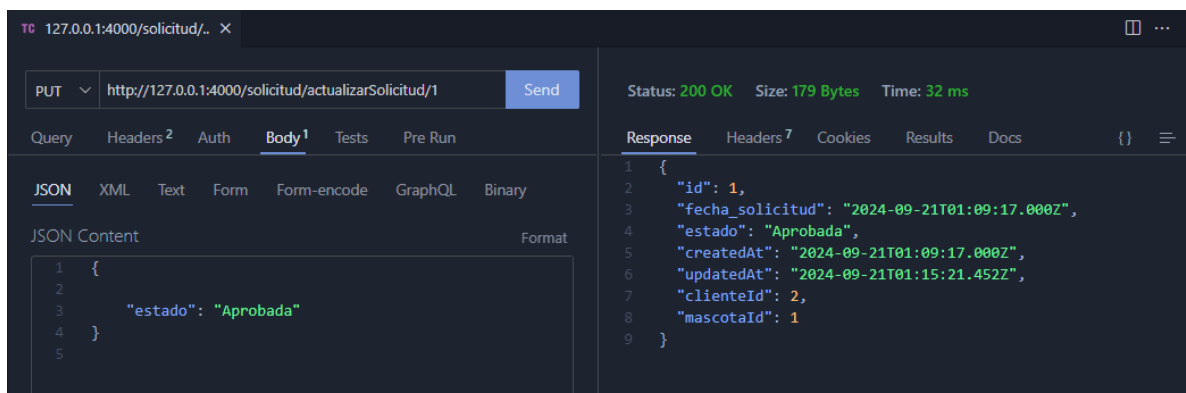


Verificación buscar Solicitud por id → “/buscarSolicitudId/:id” → consulta id=1



Verificación actualizar Solicitud → “/actualizarSolicitud/:id” → actualización id=1

Datos actualizados: estado

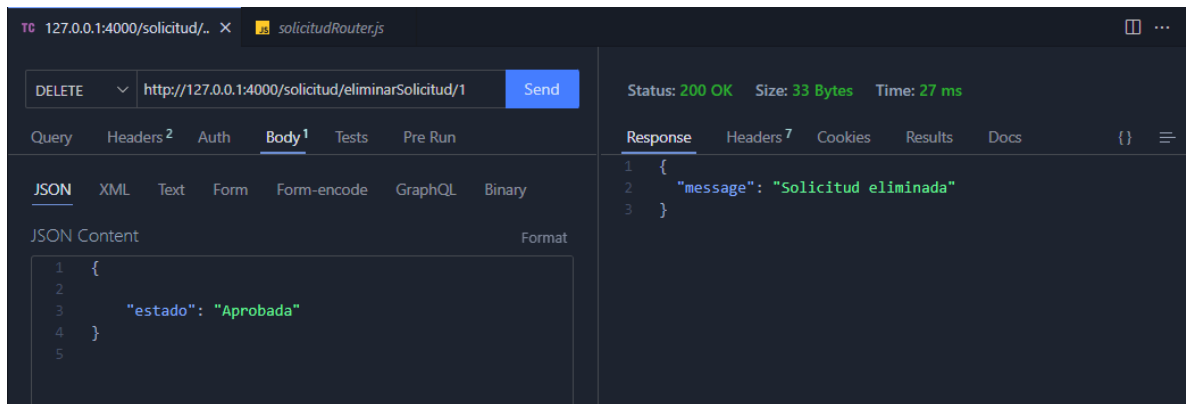


Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

The screenshot shows the DBeaver interface with the 'solicitud' table selected. The table has the following columns: id, fecha_solicitud, estado, createdAt, updatedAt, clienteId, and mascotaId. The table contains one row with the following data:

id	fecha_solicitud	estado	createdAt	updatedAt	clienteId	mascotaId
1	2024-09-21 01:09:17.000	Aprobada	2024-09-21 01:09:17.000	2024-09-21 01:15:21.000	2	1

Verificación eliminar Solicitud → “/eliminarSolicitud/:id” → eliminar id=1



Confirmación de dbeaver para evidenciar que los cambios si se hayan realizado:

