



DATOS DEL ESTUDIANTE

Apellidos y Nombres:	Yataco Saravia Luis Alonso	ID:	1552780
Dirección Zonal/CFP:	Ica – Ayacucho		
Carrera:	Ingeniería de Software con Inteligencia Artificial	Semestre:	V
Curso/ Mód. Formativo:	Fullstack Developer Software		
Tema de Trabajo Final:	Tarea 08		

1. INFORMACIÓN**▪ Identifica la problemática del caso práctico propuesto.**

El caso práctico exige la construcción de un Backend robusto, diseñado específicamente para cumplir el objetivo principal de gestionar los registros de mascotas de manera persistente y confiable.

La problemática se centra en establecer una plataforma programática que permita la manipulación completa de una entidad simple, como lo es la mascota (solo nombre y edad).

▪ Identifica propuesta de solución y evidencias.

El caso práctico se enfoca en resolver la problemática de establecer un sistema persistente y programático para la gestión de registros de mascotas (solo nombre y edad). Se utiliza MySQL como base de datos para asegurar la persistencia, con una tabla simple (mascotas).

Esta backend requiere lo necesario para crear, leer, actualizar y eliminar los registros de mascotas.

2. PLANIFICACIÓN DEL TRABAJO

▪ **Cronograma de actividades:**

N°	ACTIVIDADES	CRONOGRAMA					
	Tarea 08	16/10					
			17/10				
						18/10	
							19/10

▪ **Lista de recursos necesarios:**

1. MÁQUINAS Y EQUIPOS	
Descripción	Cantidad
Computadora	1

2. HERRAMIENTAS E INSTRUMENTOS	
Descripción	Cantidad
XAMPP	1
Visual Studio Code	1
Postman	1
MySQL	1

3. MATERIALES E INSUMOS	
Descripción	Cantidad
USB	1
Escritorio	1

3. DECIDIR PROPUESTA

- Describe la propuesta determinada para la solución del caso práctico

PROPUESTA DE SOLUCIÓN

La propuesta de solución al caso práctico se basa en la implementación de un Backend modular. El objetivo es gestionar los registros de mascotas (nombre y edad) de manera confiable, resolviendo la problemática de la volatilidad de datos.

Persistencia: La solución usa MySQL para el almacenamiento, donde la tabla mascotas guarda los datos esenciales con un id único.

Funcionalidad backend. El Controlador (`mascotasController.js`) implementa todas las operaciones necesarias: crear (POST), leer (GET), actualizar (PUT) y eliminar (DELETE) registros.

Modularidad: El proyecto sigue una arquitectura Modelo-Controlador-Rutas, separando la lógica de negocio (`mascotasController.js`) de la interacción con la base de datos (`mascotasModel.js`).

Sostenibilidad: Se utilizan variables de entorno (`.env`) para la configuración segura de las credenciales de MySQL.

4. EJECUTAR

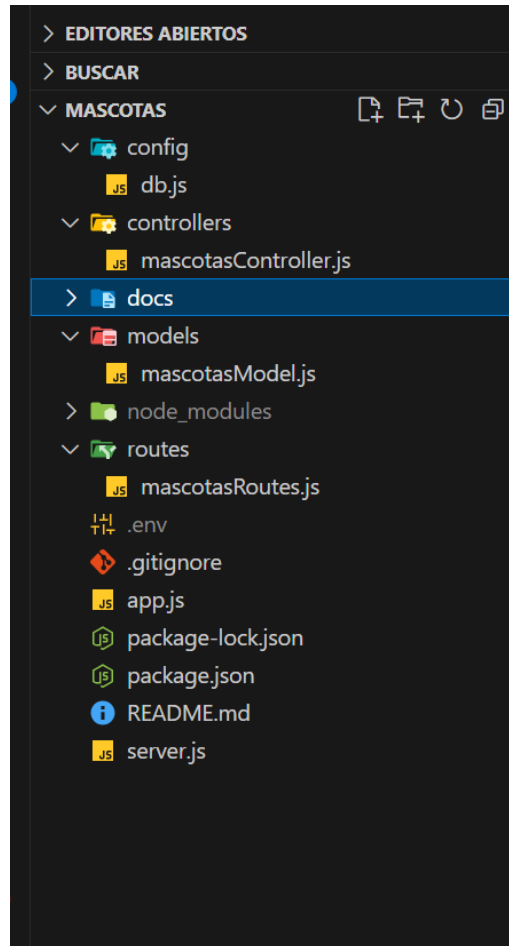
- ### Contenidos curriculares.

las ideas. Tomar en cuenta los aspectos de calidad, medio ambiente y SRI.

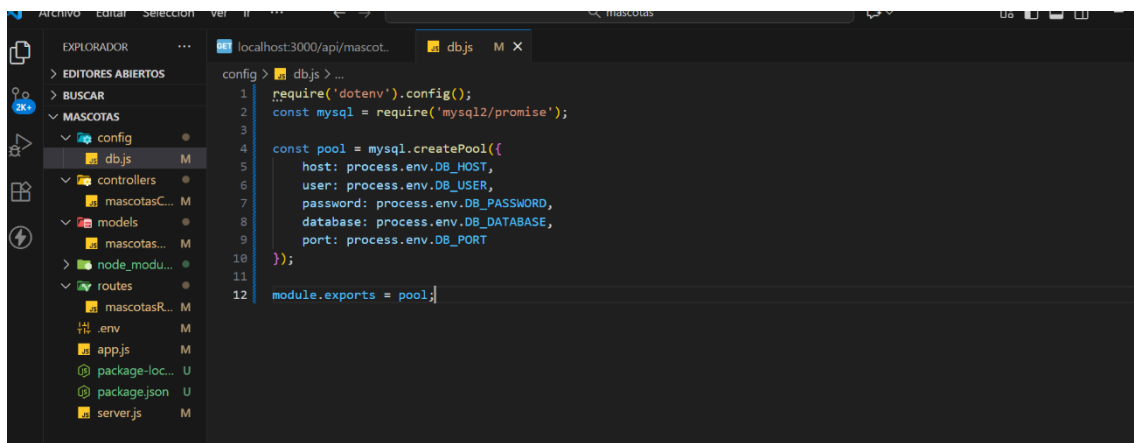
[illegible]

DIBUJO / ESQUEMA / DIAGRAMA DE PROPUESTA

(Adicionar las páginas que sean necesarias)



Config/db.js



mascotasController

```

EXPLORADOR ... localhost:3000/api/mascot... dbjs M mascotasControllerjs M X
> EDITORES ABIERTOS
> BUSCAR
MASCOTAS
  config
  dbjs M
  controllers
  mascotasC... M
  models
  node_modu...
  routes
  mascotasR... M
  .env M
  appjs M
  package-loc... U
  packagejson U
  serverjs M

controllers > mascotasControllerjs > crearMascota > crearMascota
1 const MascotasModel = require("../models/mascotasModel");
2
3 exports.crearMascota = async (req, res) => {
4   const { nombre, edad } = req.body;
5
6   if (!nombre || !edad) {
7     return res.status(400).json({ mensaje: "Falta completar los campos obligatorios" });
8   }
9
10  try {
11    const data = { nombre, edad };
12    const id = await MascotasModel.create(data);
13    res.status(201).json({ id: id, mensaje: "Mascota registrada correctamente" });
14  } catch (e) {
15    console.error(e);
16    res.status(500).json({ mensaje: "Error interno del servidor" });
17  }
18 };
19
20 exports.obtenerMascotas = async (req, res) => {
21
22   try {
23     const mascotas = await MascotasModel.getAll();
24     res.status(200).json(mascotas);
25   } catch (e) {
26     console.error(e);
27     res.status(500).json({ mensaje: "Error interno del servidor" });
28   }
29 };
30
31 exports.obtenerMascotaPorId = async (req, res) => {
32   const { id } = req.params;
33   try {
34     const mascota = await MascotasModel.getById(id);
35     if (!mascota) {
36       return res.status(404).json({ mensaje: "Mascota no encontrada" });
37     }
38     res.status(200).json(mascota);
39   } catch (e) {
40     console.error(e);
41     res.status(500).json({ mensaje: "Error interno del servidor" });
42   }
43 };
44
45 exports.actualizarMascota = async (req, res) => {
46   const { id } = req.params;
47   const updateData = req.body;
48
49   if (Object.keys(updateData).length === 0) {
50     return res.status(400).json({ mensaje: "No hay datos por actualizar" });
51   }
52
53   try {
54     const affectedRows = await MascotasModel.update(id, updateData);
55
56     if (affectedRows === 0) {
57       return res.status(404).json({ mensaje: "Mascota no encontrada para actualizar" });
58     }
59     res.status(200).json({ mensaje: "Mascota actualizada correctamente" });
60   } catch (e) {

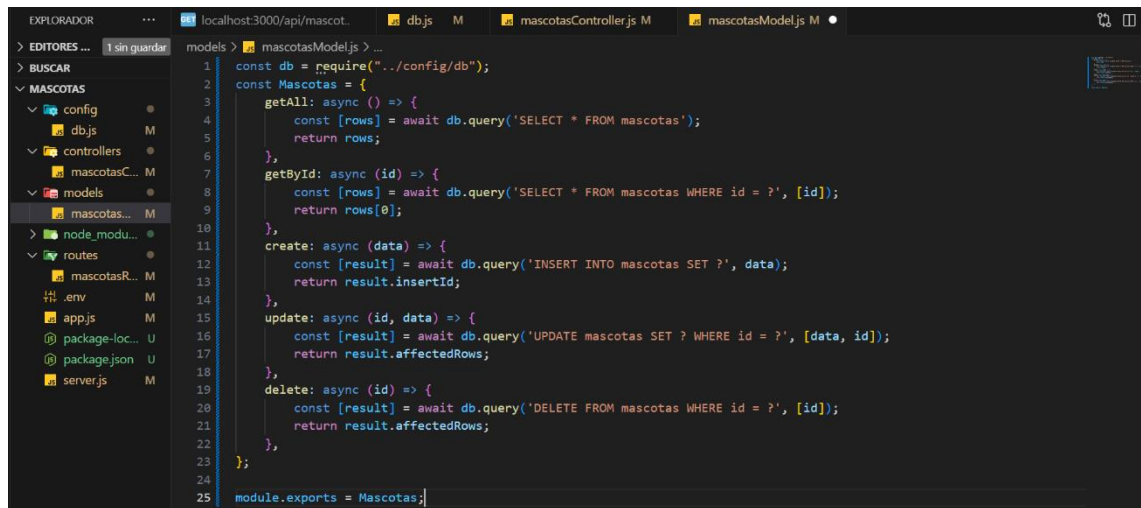
```

```

61     console.error(e);
62     res.status(500).json({ mensaje: "Error interno del servidor" });
63   }
64 };
65
66 exports.eliminarMascota = async (req, res) => {
67   const { id } = req.params;
68   try {
69     const affectedRows = await MascotasModel.delete(id);
70
71     if (affectedRows === 0) {
72       return res.status(404).json({ mensaje: "Mascota no encontrada para eliminar" });
73     }
74
75     res.status(200).json({ mensaje: "Mascota eliminada correctamente" });
76   } catch (e) {
77     console.error(e);
78     res.status(500).json({ mensaje: "Error interno del servidor" });
79   }
80 };

```

mascotasModels

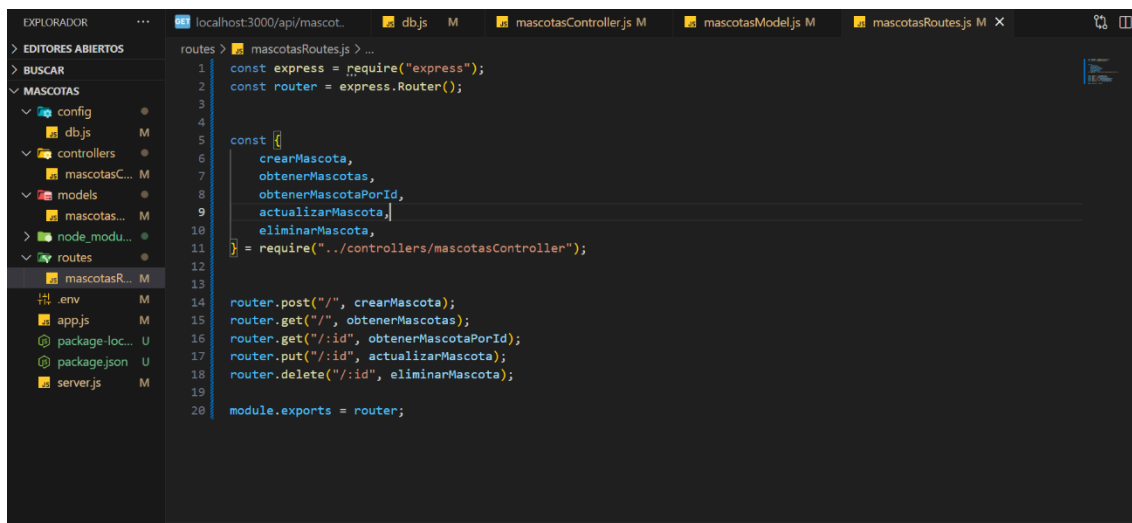


```

1  const db = require("../config/db");
2  const Mascotas = {
3    getAll: async () => {
4      const [rows] = await db.query('SELECT * FROM mascotas');
5      return rows;
6    },
7    getById: async (id) => {
8      const [rows] = await db.query('SELECT * FROM mascotas WHERE id = ?', [id]);
9      return rows[0];
10   },
11   create: async (data) => {
12     const [result] = await db.query('INSERT INTO mascotas SET ?', data);
13     return result.insertId;
14   },
15   update: async (id, data) => {
16     const [result] = await db.query('UPDATE mascotas SET ? WHERE id = ?', [data, id]);
17     return result.affectedRows;
18   },
19   delete: async (id) => {
20     const [result] = await db.query('DELETE FROM mascotas WHERE id = ?', [id]);
21     return result.affectedRows;
22   },
23 };
24
25 module.exports = Mascotas;

```

mascotasRoutes

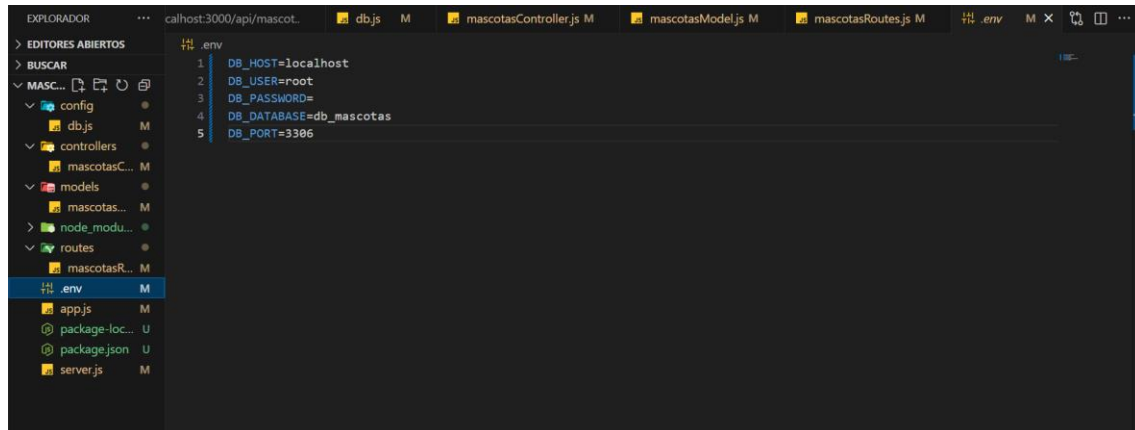


```

1  const express = require("express");
2  const router = express.Router();
3
4
5  const {
6    crearMascota,
7    obtenerMascotas,
8    obtenerMascotaPorId,
9    actualizarMascota,
10   eliminarMascota,
11 } = require("../controllers/mascotasController");
12
13
14 router.post("/", crearMascota);
15 router.get("/", obtenerMascotas);
16 router.get("/:id", obtenerMascotaPorId);
17 router.put("/:id", actualizarMascota);
18 router.delete("/:id", eliminarMascota);
19
20 module.exports = router;

```

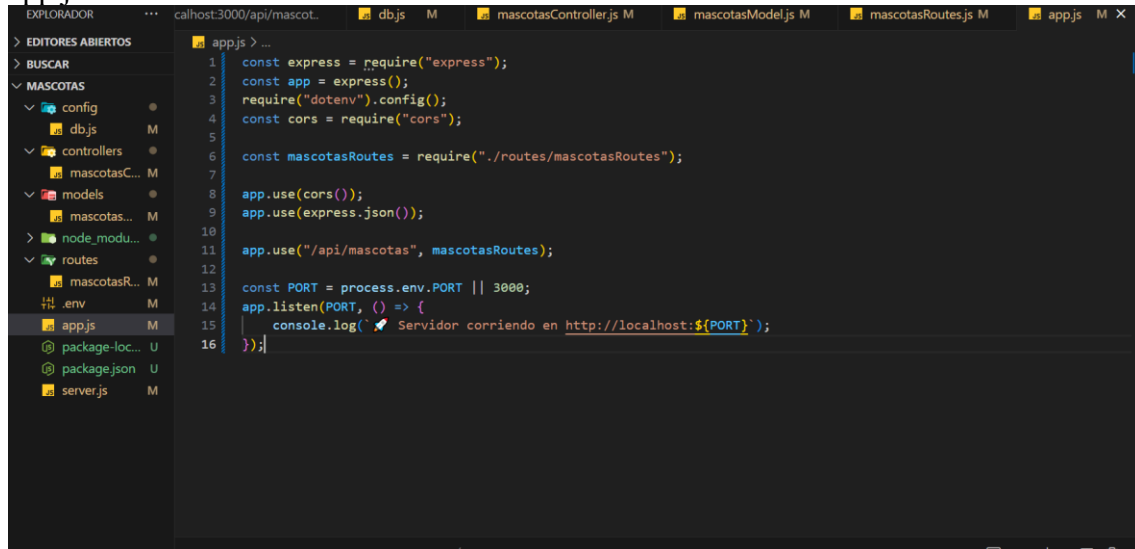

.env



```

1 DB_HOST=localhost
2 DB_USER=root
3 DB_PASSWORD=
4 DB_DATABASE=db_mascotas
5 DB_PORT=3306
  
```

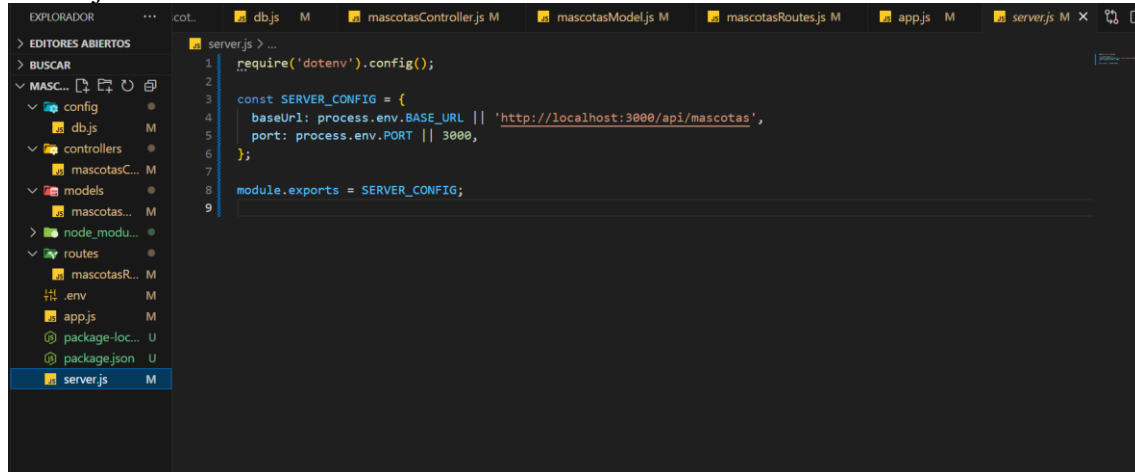
app.js



```

1 const express = require("express");
2 const app = express();
3 require("dotenv").config();
4 const cors = require("cors");
5
6 const mascotasRoutes = require("./routes/mascotasRoutes");
7
8 app.use(cors());
9 app.use(express.json());
10
11 app.use("/api/mascotas", mascotasRoutes);
12
13 const PORT = process.env.PORT || 3000;
14 app.listen(PORT, () => {
15   console.log(`Servidor corriendo en http://localhost:${PORT}`);
16 });
  
```

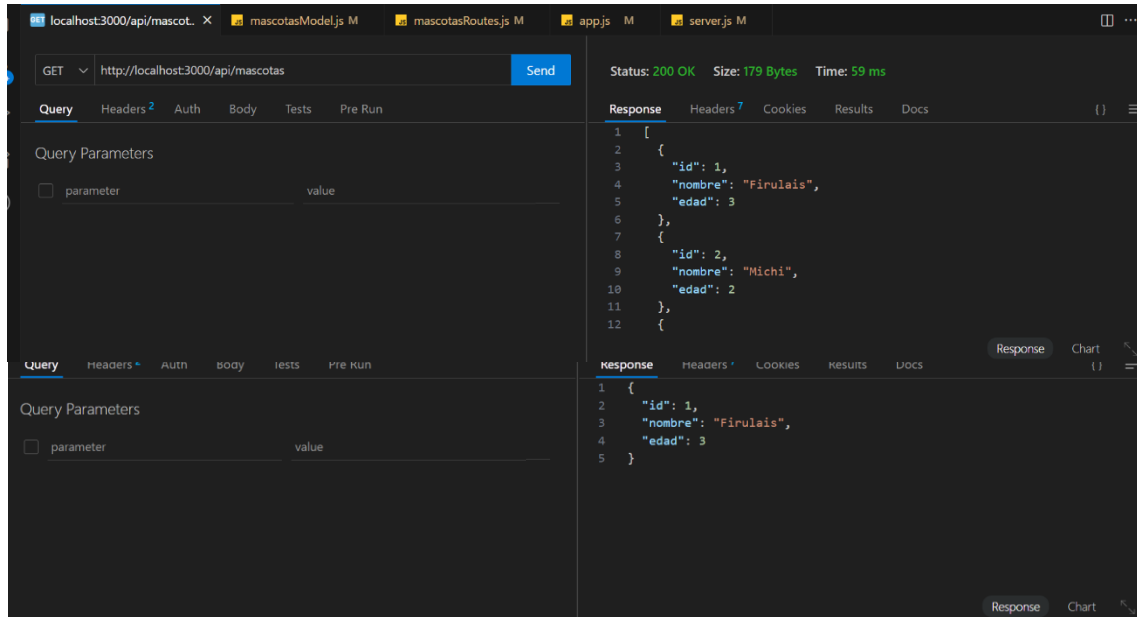
server.js



```

1 require('dotenv').config();
2
3 const SERVER_CONFIG = {
4   baseUrl: process.env.BASE_URL || 'http://localhost:3000/api/mascotas',
5   port: process.env.PORT || 3000,
6 };
7
8 module.exports = SERVER_CONFIG;
9
  
```

Método GET



GET

Status: 200 OK Size: 179 Bytes Time: 59 ms

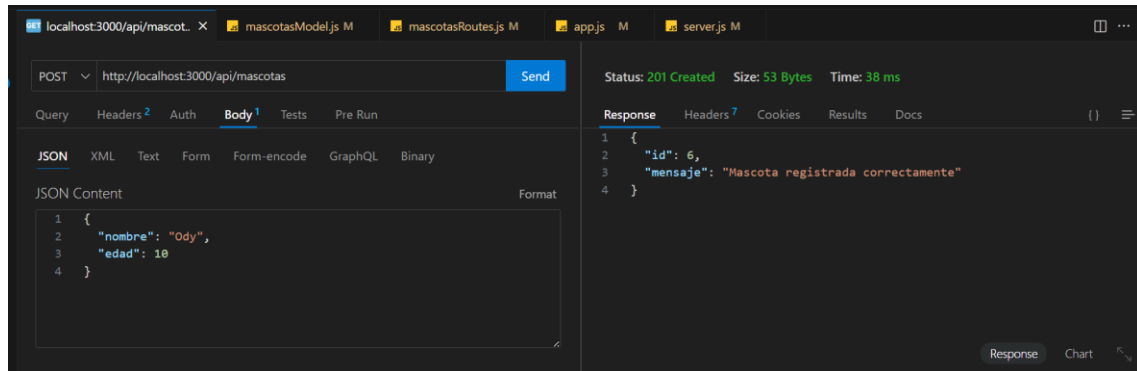
Response

```

1  [
2    {
3      "id": 1,
4      "nombre": "Firulais",
5      "edad": 3
6    },
7    {
8      "id": 2,
9      "nombre": "Michi",
10     "edad": 2
11   },
12  ]

```

Metodo Post



POST

Status: 201 Created Size: 53 Bytes Time: 38 ms

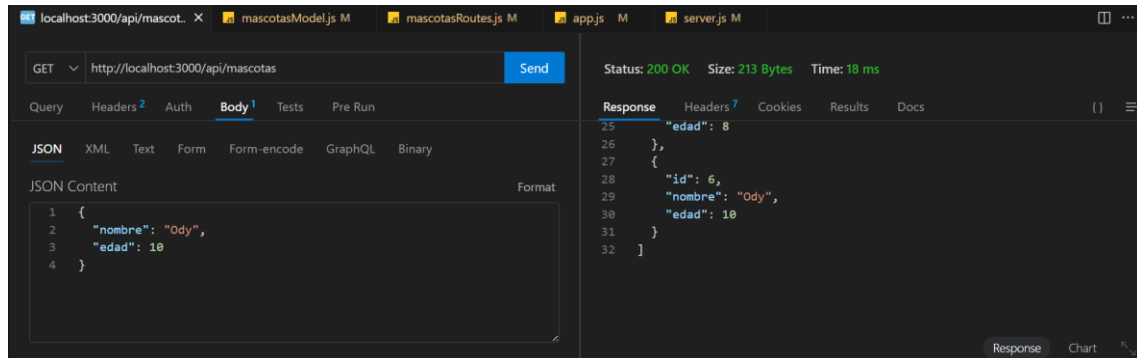
Response

```

1  {
2    "id": 6,
3    "mensaje": "Mascota registrada correctamente"
4  }

```

Creado correctamente



GET

Status: 200 OK Size: 213 Bytes Time: 18 ms

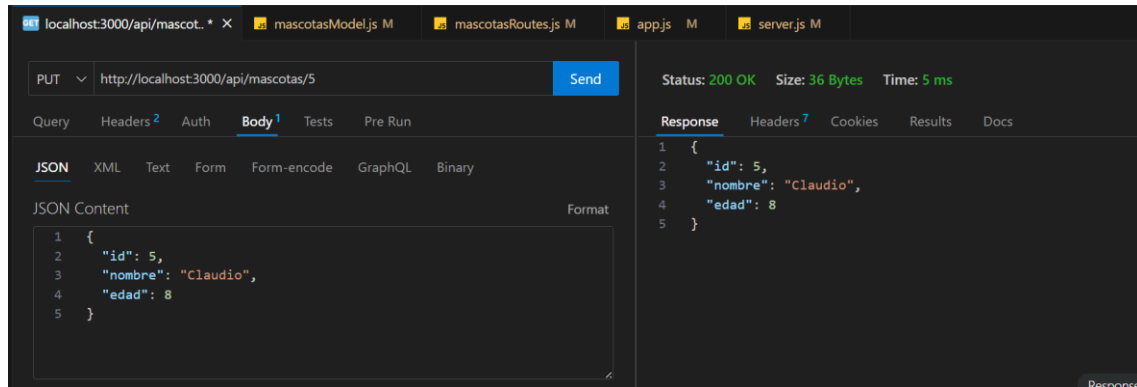
Response

```

25  "edad": 8
26  },
27  {
28    "id": 6,
29    "nombre": "Ody",
30    "edad": 10
31  }
32  ]

```

Método PUT



PUT `http://localhost:3000/api/mascotas/5` Send

Status: 200 OK Size: 36 Bytes Time: 5 ms

Response

```

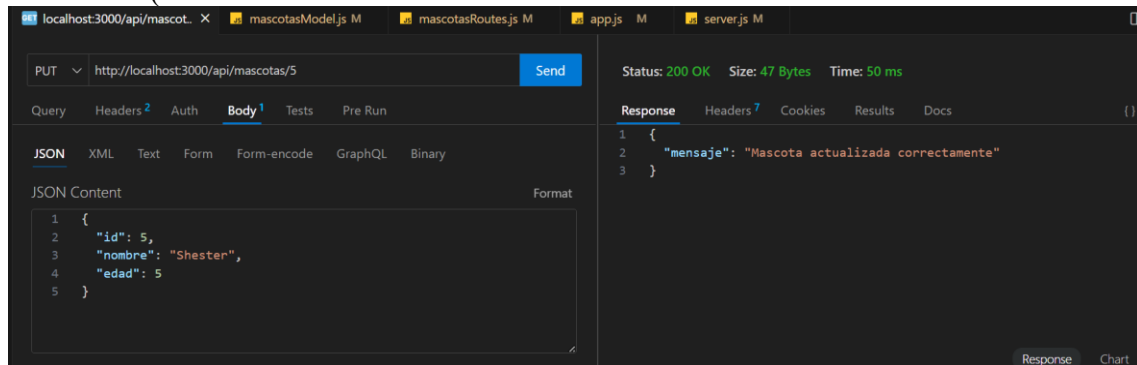
1 {
2   "id": 5,
3   "nombre": "Claudio",
4   "edad": 8
5 }
```

JSON Content

```

1 {
2   "id": 5,
3   "nombre": "Claudio",
4   "edad": 8
5 }
```

Actualizar (Mascota actualizada correctamente)



PUT `http://localhost:3000/api/mascotas/5` Send

Status: 200 OK Size: 47 Bytes Time: 50 ms

Response

```

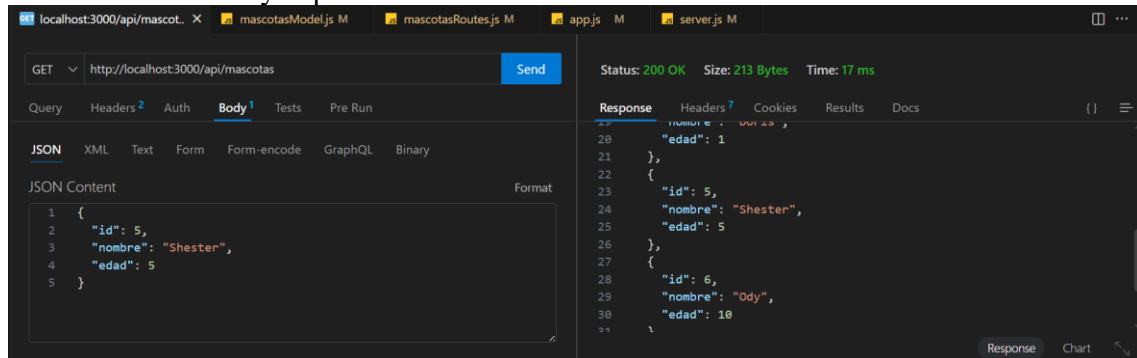
1 {
2   "mensaje": "Mascota actualizada correctamente"
3 }
```

JSON Content

```

1 {
2   "id": 5,
3   "nombre": "Shester",
4   "edad": 5
5 }
```

En el Metodo GET ya aparece actualizado



GET `http://localhost:3000/api/mascotas` Send

Status: 200 OK Size: 213 Bytes Time: 17 ms

Response

```

20 {
21   "id": 5,
22   "nombre": "Shester",
23   "edad": 5
24 },
25 {
26   "id": 6,
27   "nombre": "Ody",
28   "edad": 10
29 }
30 ]
```

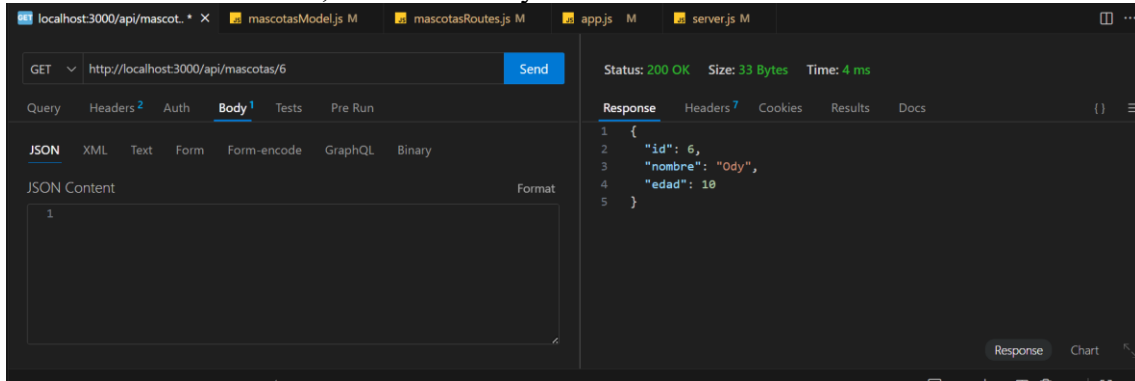
JSON Content

```

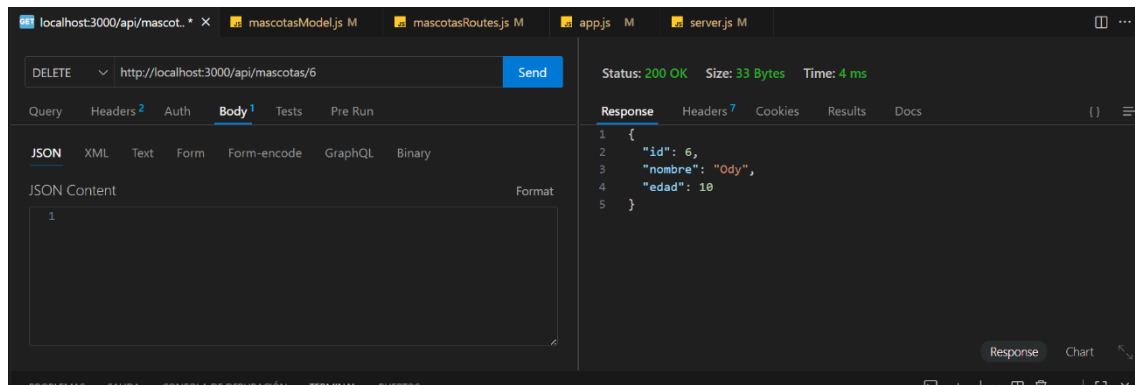
1 {
2   "id": 5,
3   "nombre": "Shester",
4   "edad": 5
5 }
```

Método DELETE

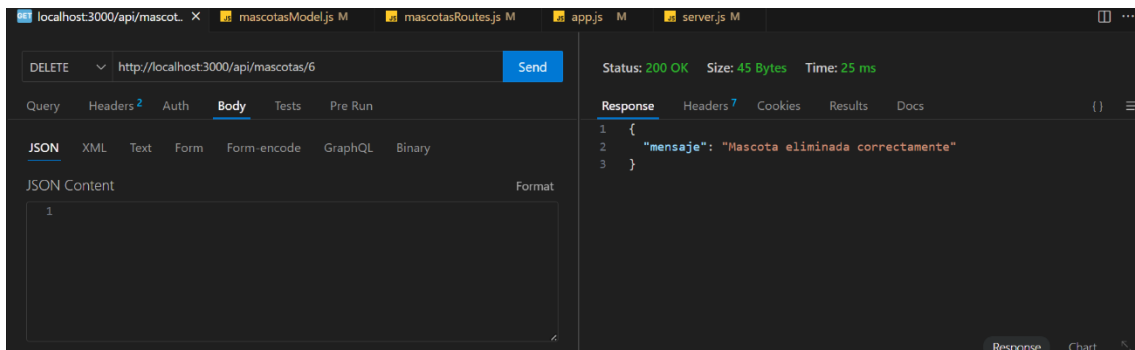
Vamos a eliminar el id 6, de nombre Ody con 10 años



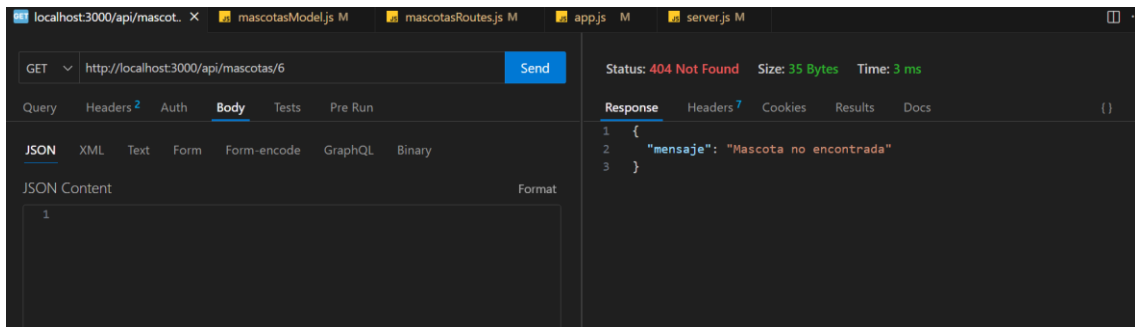
Cambiamos al método delete



Mascota eliminada correctamente



Se elimino correctamente



La base de datos

```

1 • CREATE DATABASE db_mascotas;
2
3 • USE db_mascotas;
4
5 • CREATE TABLE mascotas (
6     id INT AUTO_INCREMENT PRIMARY KEY,
7     nombre VARCHAR(100),
8     edad SMALLINT
9 );
10
11 • INSERT INTO mascotas (nombre, edad) VALUES
12     ('Firulais', 3),
13     ('Michi', 2),
14     ('Rex', 5);
15
16 • SELECT *FROM mascotas;
17

```

Result Grid			
Filter Rows:			
Edit: Export/Import: Wrap Cell Content:			
id	nombre	edad	
1	Firulais	3	
2	Michi	2	
3	Rex	5	
•	NULL	NULL	

5. CONTROLAR

- Verificar el cumplimiento de los procesos desarrollados en la propuesta de solución del caso práctico.

EVIDENCIAS	CUMPLE	NO CUMPLE
• ¿Se identificó claramente la problemática del caso práctico?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se desarrolló las condiciones de los requerimientos solicitados?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se formularon respuestas claras y fundamentadas a todas las preguntas guía?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
• ¿Se elaboró un cronograma claro de actividades a ejecutar?	<input type="checkbox"/>	<input checked="" type="checkbox"/>
• ¿Se identificaron y listaron los recursos (máquinas, equipos, herramientas, materiales) necesarios para ejecutar la propuesta?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se ejecutó la propuesta de acuerdo con la planificación y cronograma establecidos?	<input type="checkbox"/>	<input type="checkbox"/>
• ¿Se describieron todas las operaciones y pasos seguidos para garantizar la correcta ejecución?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se consideran las normativas técnicas, de seguridad y medio ambiente en la propuesta de solución?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿La propuesta es pertinente con los requerimientos solicitados?	<input checked="" type="checkbox"/>	<input type="checkbox"/>
• ¿Se evaluó la viabilidad de la propuesta para un contexto real?	<input checked="" type="checkbox"/>	<input type="checkbox"/>

6. VALORAR

- Califica el impacto que representa la propuesta de solución ante la situación planteada en el caso práctico.

CRITERIO DE EVALUACIÓN	DESCRIPCIÓN DEL CRITERIO	PUNTUACIÓN MÁXIMA	PUNTAJE CALIFICADO POR EL ESTUDIANTE
Identificación del problema	Claridad en la identificación del problema planteado.	3	
Relevancia de la propuesta de solución	La propuesta responde adecuadamente al problema planteado y es relevante para el contexto del caso práctico.	8	
Viabilidad técnica	La solución es técnicamente factible, tomando en cuenta los recursos y conocimientos disponibles.	6	
Cumplimiento de Normas	La solución cumple con todas las normas técnicas de seguridad, higiene y medio ambiente.	3	
PUNTAJE TOTAL		20	

