

TALLER UNIDAD 2 BACKEND



Presentado por:

LUIS FELIPE SANTACRUZ CHINCHAJOA (220034097)

Docente:

VICENTE AUX

SEPTIEMBRE 2024

UNIVERSIDAD DE NARIÑO

**DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS
PARA EL DESARROLLO DE SOFTWARE**

PASTO – NARIÑO

1. Introducción

En el presente informe se detalla el desarrollo de una aplicación backend destinada a gestionar una empresa de adopción de mascotas. Esta aplicación permite llevar el registro de las mascotas disponibles para adopción, así como gestionar las solicitudes de adopción de los usuarios. Se implementó usando tecnologías modernas, como **NodeJS**, **ExpressJS**, y una base de datos en **MySQL/MariaDB**, utilizando **XAMPP** como servidor para gestionar la base de datos localmente.

Además, se realizaron pruebas con **Postman** para asegurar el correcto funcionamiento del sistema y sus diferentes operaciones. El objetivo principal del proyecto es crear una solución eficiente para el manejo de los registros de mascotas y solicitudes de adopción, garantizando la integridad y consistencia de los datos.

2. Desarrollo de la base de datos

2.1 Diseño de la base de datos

La base de datos fue creada utilizando **MySQL/MariaDB** a través de **XAMPP**, que proporcionó un servidor local para gestionar los datos de la empresa de adopción de mascotas. La base de datos, llamada **AdopcionMascotas**, incluye dos tablas principales: una para el registro de las mascotas y otra para las solicitudes de adopción.

Tablas creadas:

- ❖ **Tabla Mascotas:** Contiene información sobre cada mascota registrada en la empresa.
 - **Estructura:**
 - **id_mascota:** Identificador único para cada mascota (clave primaria).
 - **nombre:** Nombre de la mascota.
 - **especie:** Especie a la que pertenece la mascota (perro, gato, etc.).
 - **raza:** Raza de la mascota.
 - **edad:** Edad de la mascota.
 - **estado_adopcion:** Indica si la mascota está disponible para adopción (0 para disponible, 1 para adoptada).

- ❖ **Tabla Solicitudes_Adopcion:** Almacena los detalles de las solicitudes de adopción realizadas por los usuarios interesados.
 - **Estructura:**
 - **id_solicitud:** Identificador único para cada solicitud (clave primaria).
 - **id_mascota:** Referencia al identificador de la mascota a la que corresponde la solicitud (clave foránea).
 - **nombre_solicitante:** Nombre del solicitante.
 - **contacto_solicitante:** Información de contacto del solicitante.
 - **fecha_solicitud:** Fecha en que se realizó la solicitud.
 - **estado_solicitud:** Estado actual de la solicitud (pendiente, aprobada, rechazada).

Script SQL para crear la base de datos y las tablas:

```
Ejecutar la(s) consulta(s) SQL en la base de datos adopcionmascotas: ⓘ

1 CREATE DATABASE IF NOT EXISTS AdopcionMascotas;
2 USE AdopcionMascotas;
3
4 CREATE TABLE Mascotas (
5     id_mascota INT AUTO_INCREMENT PRIMARY KEY,
6     nombre VARCHAR(50),
7     especie VARCHAR(50),
8     raza VARCHAR(50),
9     edad INT,
10    estado_adopcion BOOLEAN DEFAULT 0
11 );
12
13 CREATE TABLE Solicitudes_Adopcion (
14     id_solicitud INT AUTO_INCREMENT PRIMARY KEY,
15     id_mascota INT,
16     nombre_solicitante VARCHAR(100),
17     contacto_solicitante VARCHAR(100),
18     fecha_solicitud DATE,
19     estado_solicitud VARCHAR(50),
20     FOREIGN KEY (id_mascota) REFERENCES Mascotas(id_mascota)
21 );
22 |
```

Verificamos que las tablas se han creado de manera correcta:

Tables_in_adopcionmascotas
mascotas
solicitudes_adopcion

2.2 Población de tablas de la base de datos

Una vez creadas las tablas, se procedió a la inserción de datos iniciales en la base de datos para simular el entorno real de la empresa de adopción de mascotas. Se agregaron cuatro registros en la tabla de **Mascotas**, cada uno representando a un animal disponible para adopción, y se realizó una solicitud de adopción en la tabla **Solicitudes_Adopcion**. Los datos agregados fueron los siguientes:

←T→		id_mascota	nombre	especie	raza	edad	estado_adopcion
<input type="checkbox"/>	 Editar  Copiar  Borrar	1	Mateo	Perro	Bulldog	1	0
<input type="checkbox"/>	 Editar  Copiar  Borrar	2	Kitty	Gato	Persa	2	1
<input type="checkbox"/>	 Editar  Copiar  Borrar	3	Mono	Gato	Angora	1	0
<input type="checkbox"/>	 Editar  Copiar  Borrar	4	Doky	Perro	Husky	3	1

	id_solicitud	id_mascota	nombre_solicitante	contacto_solicitante	fecha_solicitud	estado_solicitud
<input type="checkbox"/> Editar <input type="checkbox"/> Copiar <input type="checkbox"/> Borrar	1	3	Matías Rosales	3224823651	2024-09-16	Pendiente

Con estos datos iniciales, se garantizó el correcto funcionamiento de las operaciones de inserción y consulta en las tablas correspondientes de la base de datos.

2.3 Conexión de la base de datos con el backend

Se utilizó el paquete **mysql2** para establecer la conexión entre la base de datos y la aplicación backend. La configuración de la conexión se realizó dentro del archivo principal **app.js** del proyecto, donde se especificaron los parámetros de acceso (host, usuario, contraseña, y nombre de la base de datos).

3. Desarrollo del Backend

3.1 Configuración del entorno de desarrollo

El backend fue desarrollado utilizando **NodeJS** y **ExpressJS**. Se configuró el entorno de desarrollo mediante la inicialización de un proyecto NodeJS en **Visual Studio Code**, seguido de la instalación de las dependencias necesarias, tales como **ExpressJS** y **mysql2**. A continuación, se presenta el archivo **app.js**, que define el servidor y las rutas principales.

Comando para inicializar el proyecto:

```
npm init -y
```

```

PROBLEMS  OUTPUT  TERMINAL  ...  powershell  +  -  [ ]  [X]  ...  ^  X

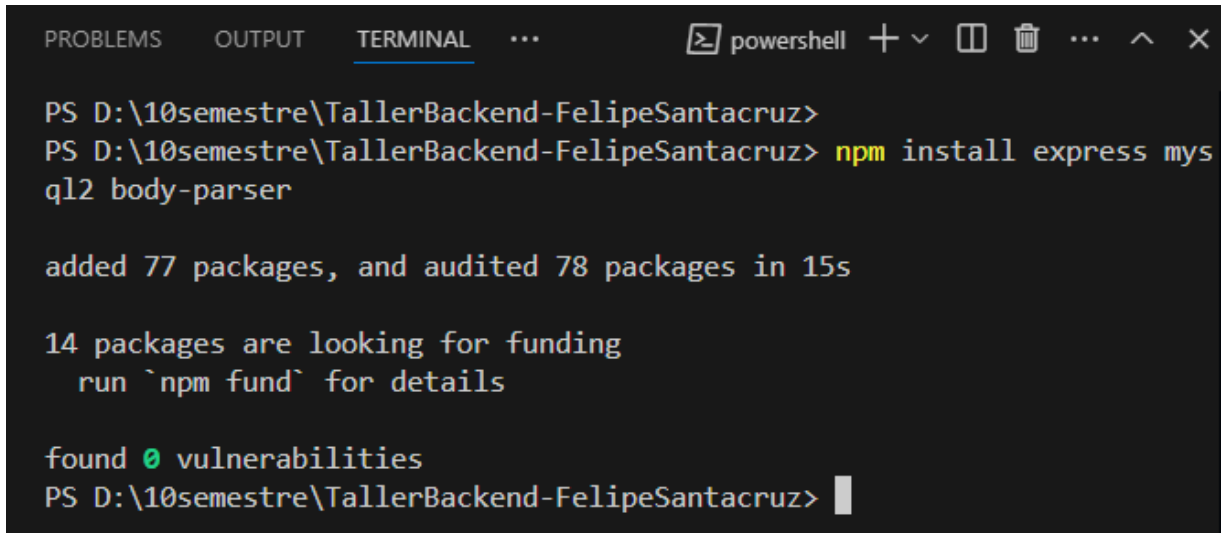
PS D:\10semestre\TallerBackend-FelipeSantacruz> npm init -y
Wrote to D:\10semestre\TallerBackend-FelipeSantacruz\package.json:

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

```

Comando para la instalación de las dependencias:

```
npm install express mysql2 body-parser
```



```
PROBLEMS OUTPUT TERMINAL ... powershell + - [ ] [ ] ... ^ X

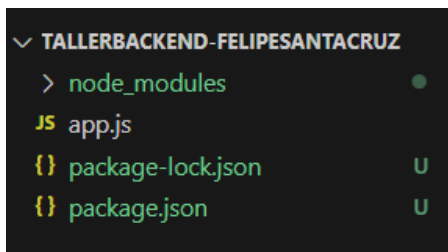
PS D:\10semestre\TallerBackend-FelipeSantacruz>
PS D:\10semestre\TallerBackend-FelipeSantacruz> npm install express mysql2 body-parser

added 77 packages, and audited 78 packages in 15s

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

found 0 vulnerabilities
PS D:\10semestre\TallerBackend-FelipeSantacruz> |
```

Comprobamos que se han creado de manera correcta:



```
▼ TALLERBACKEND-FELIPESANTACRUZ
  > node_modules
  JS app.js
  {} package-lock.json U
  {} package.json U
```

3.2 Desarrollo de las rutas para mascotas y solicitudes

El archivo **app.js** contiene la configuración básica para las rutas que permiten gestionar las mascotas y las solicitudes de adopción. Estas rutas incluyen las operaciones básicas para listar mascotas, registrar nuevas mascotas, y registrar solicitudes de adopción.

Código principal del servidor:

```
JS app.js M X
JS app.js > ...
1  const express = require('express');
2  const mysql = require('mysql2');
3  const app = express();
4  app.use(express.json());
5
6  // Conexión a la base de datos
7  const db = mysql.createConnection({
8    host: 'localhost',
9    user: 'root',
10    password: '',
11    database: 'AdopcionMascotas'
12  });
13
14  db.connect(err => {
15    if (err) throw err;
16    console.log('Conectado a la base de datos');
17  });
18
19  // Obtener todas las mascotas
20  app.get('/mascotas', (req, res) => {
21    db.query('SELECT * FROM Mascotas', (err, result) => {
22      if (err) throw err;
23      res.json(result);
24    });
25  });
26
27  // Agregar una nueva mascota
28  app.post('/mascotas', (req, res) => {
29    const { nombre, especie, raza, edad } = req.body;
30    const sql = 'INSERT INTO Mascotas (nombre, especie, raza, edad) VALUES (?, ?, ?, ?)';
31    db.query(sql, [nombre, especie, raza, edad], (err, result) => {
32      if (err) throw err;
33      res.json({ mensaje: 'Mascota agregada', id_mascota: result.insertId });
34    });
35  });
36
37  // Iniciar el servidor
38  app.listen(3000, () => {
39    console.log('Servidor corriendo en http://localhost:3000');
40  });
```

3.3 Ejecución del servidor

El servidor se ejecuta en el puerto 3000 y responde a las solicitudes HTTP enviadas a través de **Postman**. Para iniciar el servidor, se utilizó el siguiente comando en la terminal:

node app.js

```
PROBLEMS  OUTPUT  TERMINAL  ...  node + v  [icon]  [icon]  ...  ^  X

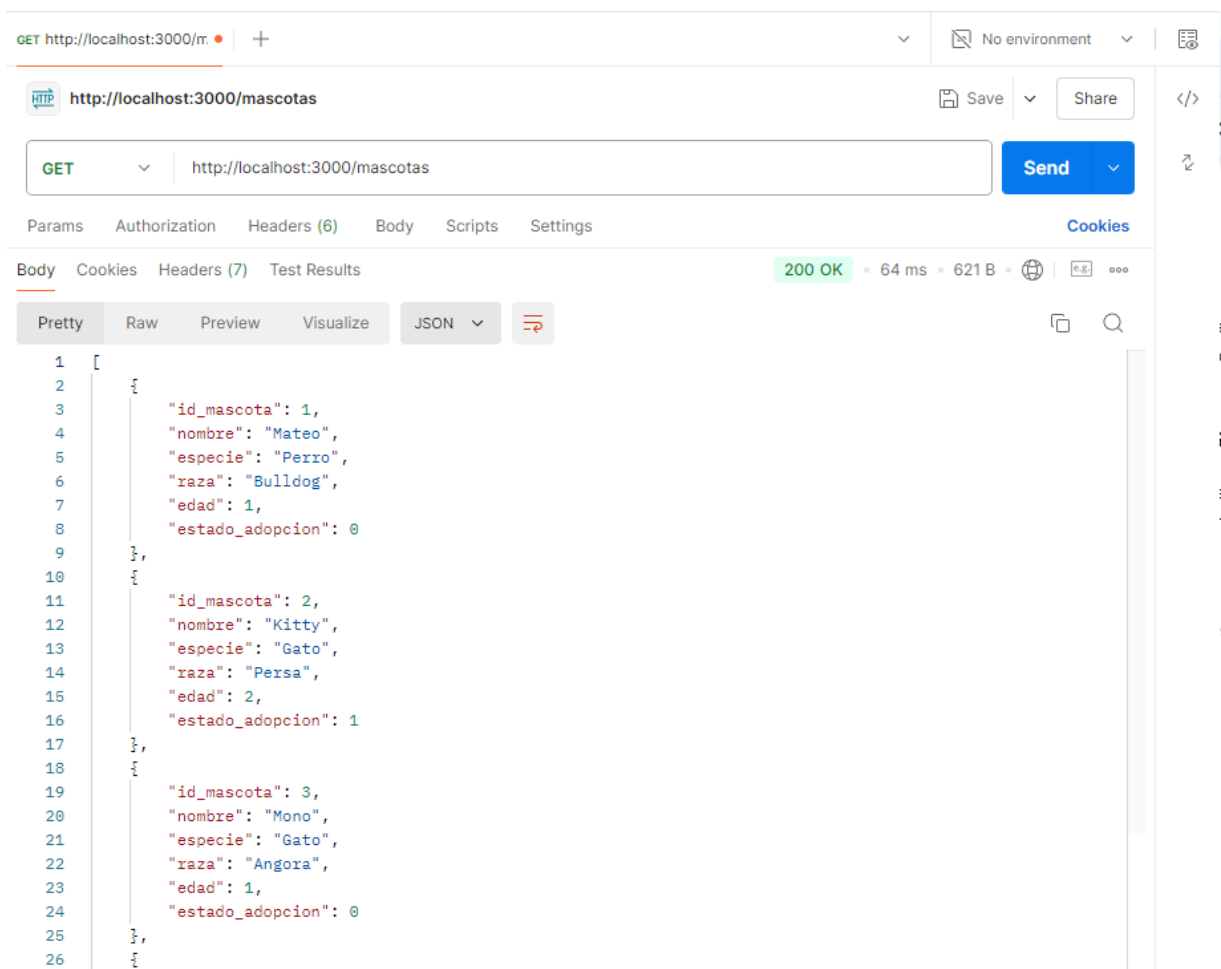
PS D:\10semestre\TallerBackend-FelipeSantacruz> node app.js
Servidor corriendo en http://localhost:3000
Conectado a la base de datos
```

4. Pruebas realizadas con Postman

Se realizaron diversas pruebas para validar el correcto funcionamiento de las rutas implementadas. Estas pruebas se ejecutaron en **Postman**, enviando solicitudes HTTP a las rutas **GET** y **POST** desarrolladas en el backend.

4.1 Solicitud para listar todas las mascotas (GET /mascotas)

Se probó la ruta **GET /mascotas** para obtener el listado de todas las mascotas registradas en la base de datos. Se seleccionó el método **GET** y en el campo de URL se escribió la siguiente dirección: <http://localhost:3000/mascotas>

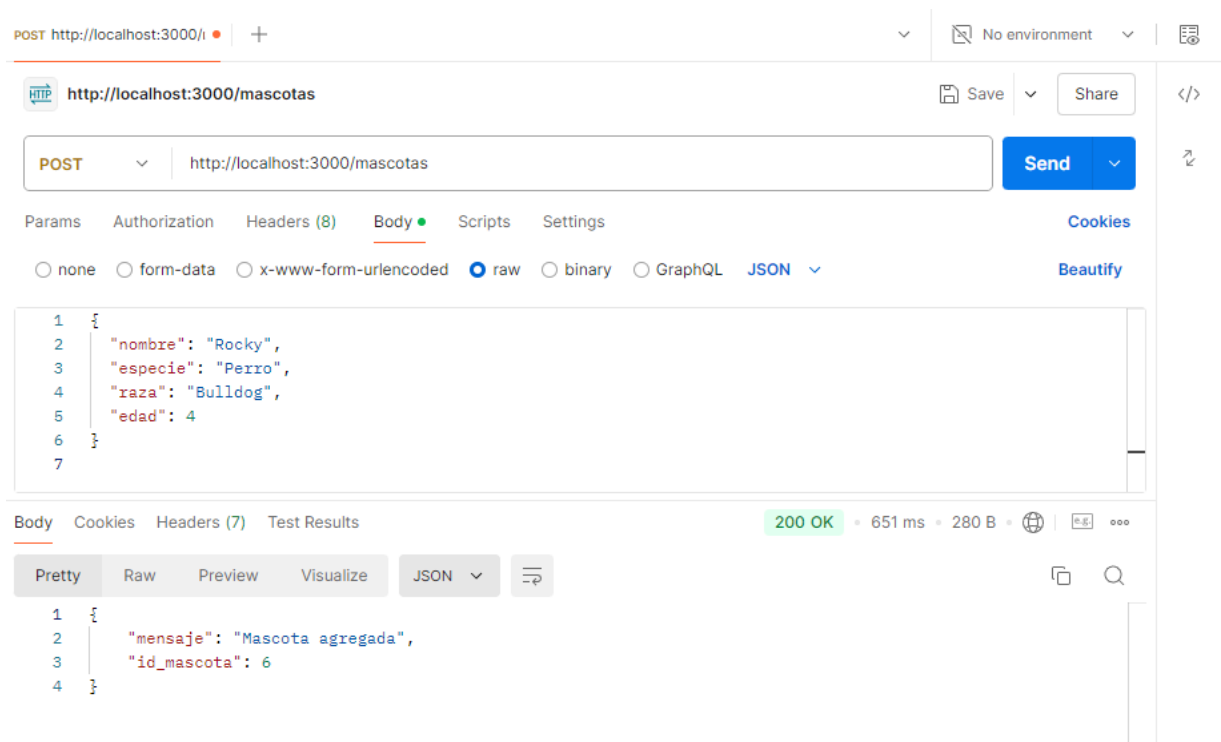


La solicitud devolvió los registros correctamente en formato JSON.

4.2 Solicitud para agregar una nueva mascota (POST /mascotas)

Se probó la ruta **POST /mascotas** para agregar una nueva mascota a la base de datos, enviando la información en formato JSON en el cuerpo de la solicitud. Se seleccionó el método **POST** y en el campo de URL se escribió la siguiente dirección:

<http://localhost:3000/mascotas>



La operación fue exitosa y la nueva mascota se agregó a la base de datos.

			id_mascota	nombre	especie	raza	edad	estado_adopcion
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	1	Mateo	Perro	Bulldog	1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	2	Kitty	Gato	Persa	2
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	3	Mono	Gato	Angora	1
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	4	Doky	Perro	Husky	3
<input type="checkbox"/>	 Editar	 Copiar	 Borrar	6	Rocky	Perro	Bulldog	4

5. Conclusión

El desarrollo de este backend para la gestión de una empresa de adopción de mascotas permitió aplicar los conocimientos adquiridos en NodeJS y ExpressJS, así como la integración con una base de datos MySQL/MariaDB. La creación de las tablas, la configuración del servidor y la implementación de las rutas fueron pasos clave para alcanzar los objetivos del proyecto. Las pruebas realizadas a través de Postman evidencian que el sistema es capaz de manejar correctamente los registros de mascotas y solicitudes de adopción, garantizando un flujo eficiente de información.

Este proyecto no solo permitió mejorar las habilidades técnicas, sino también el entendimiento del ciclo de vida de una aplicación backend, desde la creación de la base de datos hasta la interacción con los usuarios mediante rutas HTTP.