TALLER N°2 BACKEND

PRESENTADO POR:

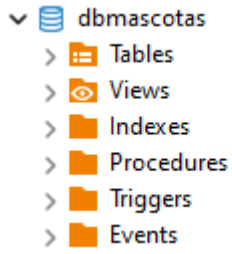
JOSE GABRIEL NASTUL NARVAEZ

UNIVERSIDAD DE NARIÑO

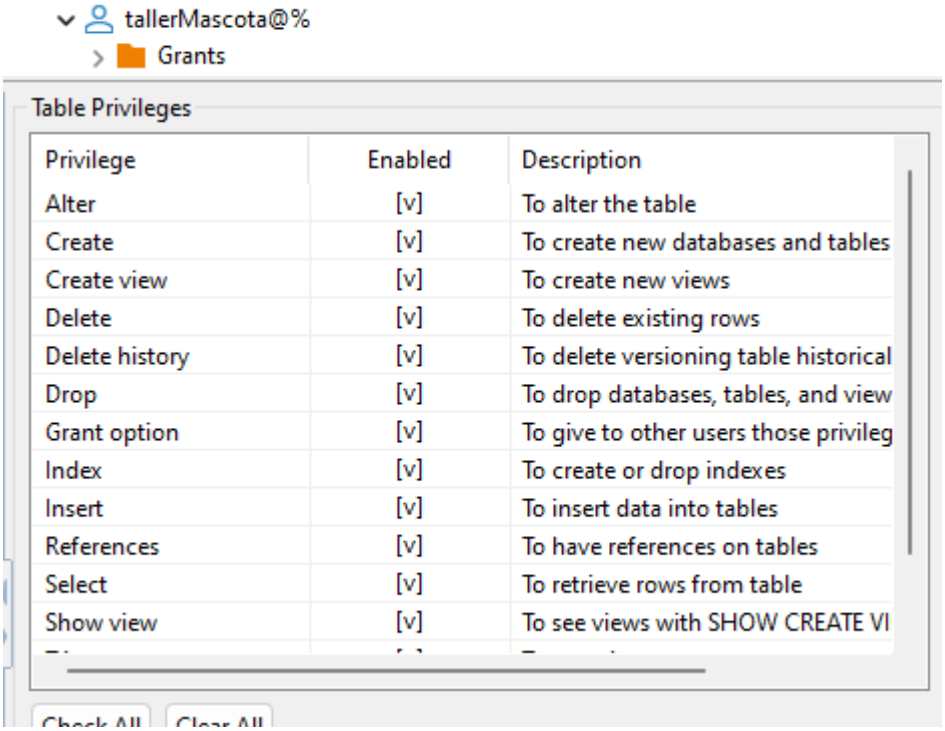
SEDE IPILAES

2024

1. Crear una base de datos en MySQL/MariaDB para registrar la información de una empresa de adopción de mascotas.



2. Después de que hayamos creado la base de datos creamos un usuario y le damos los privilegios. Y hacemos la conexión a la base de datos



3. Desarrollar una aplicación Backend en NodeJS y ExpressJS que interactúe con la base de datos. Esta aplicación debe permitir todas las tareas relacionadas con el registro y administración de mascotas, así como las solicitudes de adopción.

```
import  Sequelize from "sequelize";

const db = new Sequelize("dbmascotas", "tallerMascota", "tallerMascota",{
    dialect: "mysql",
    host: "localhost"
});

export {db}
```

4. Generamos la siguiente instrucción

```
PS C:\Users\DeLL\Desktop\Diplomado\TallerBackEnd> npm init -y
```

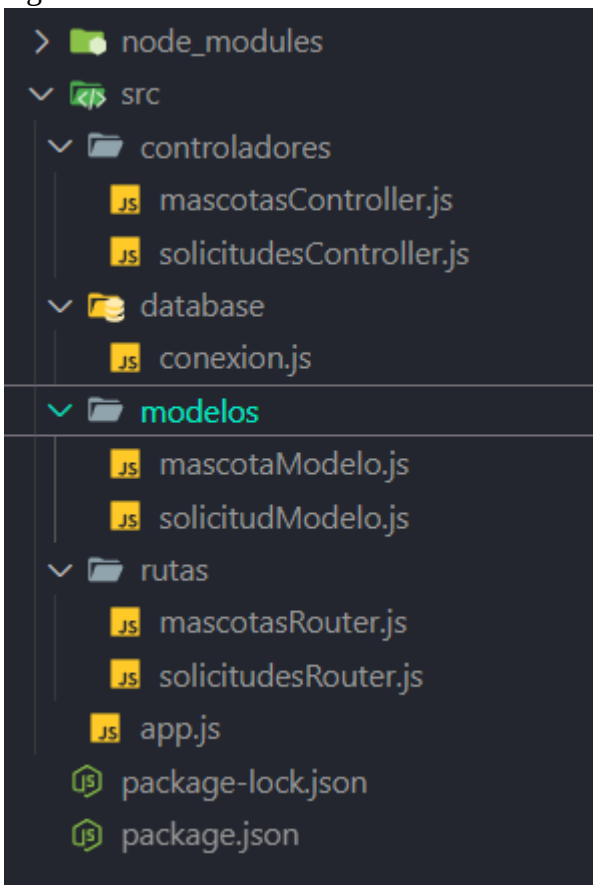
5. Después de iniciar esta instrucción nos aparece esta pantalla y realizamos algunos cambios, como lo son el type a module y el main para que inicie en app.js, y también el nodemon para que se ejecuten los cambios.

```
{
  "name": "tallerbackend",
  "version": "1.0.0",
  "type": "module",
  "main": "app.js",
  ▷ Depurar
  "scripts": {
    "start": "nodemon ./src/app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
```

6. Después de ello instalamos las dependencias de nodemon, express, mysql2 y sequelize con npm

```
"keywords": [],
"author": "",
"license": "ISC",
"description": "",
"devDependencies": {
  "nodemon": "^3.1.6"
},
"dependencies": {
  "artisan": "^0.1.2",
  "cors": "^2.8.5",
  "express": "^4.21.0",
  "mysql2": "^3.11.3",
  "sequelize": "^6.37.3"
}
```

7. Después de ello formamos toda la estructura para hacer la lógica del proyecto, de la siguiente manera.



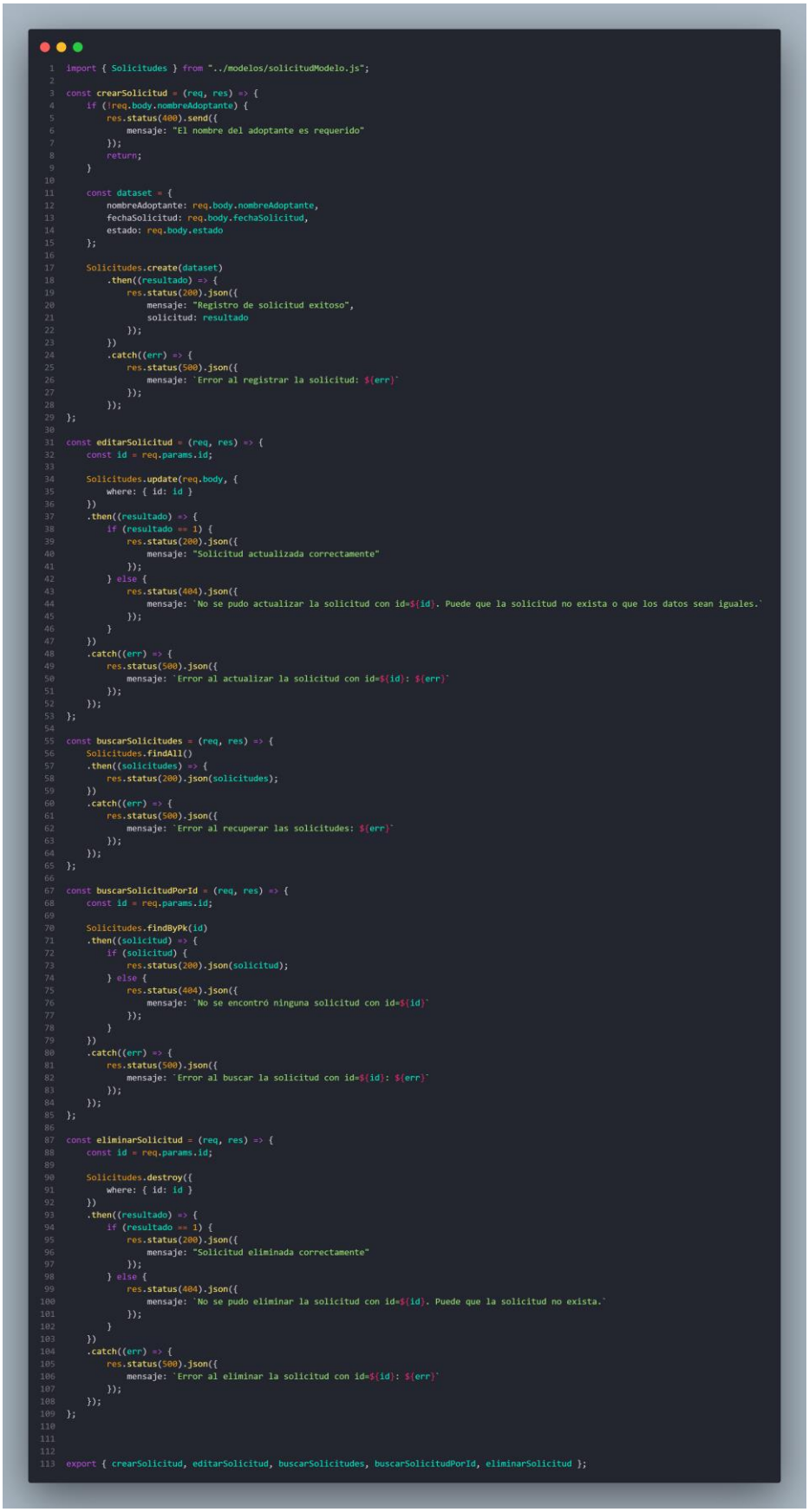
8. Creamos los modelos

```
import { Sequelize, DataTypes } from "sequelize";
import { db } from "../database/conexion.js";

const Mascotas = db.define("mascotas", {
    id: {
        type: DataTypes.INTEGER,
        allowNull: false,
        autoIncrement: true,
        primaryKey: true
    },
    nombre: {
        type: DataTypes.STRING,
        allowNull: false
    },
    descripcion: {
        type: DataTypes.TEXT,
        allowNull: true
    },
    edad: {
        type: DataTypes.INTEGER,
        allowNull: false
    },
    genero: {
        type: DataTypes.ENUM('Macho', 'Hembra'),
        allowNull: false
    },
    imagen: {
        type: DataTypes.STRING, // Ruta de la imagen o URL
        allowNull: true
    }
}, {
    tableName: 'mascotas', // Nombre de la tabla en la base de datos
    timestamps: false
});



export {Mascotas}

```

```
import { Sequelize, DataTypes } from "sequelize";
import { db } from "../database/conexion.js";

const Solicitudes = db.define("solicitudes", {
    id: {
        type: DataTypes.INTEGER,
        allowNull: false,
        autoIncrement: true,
        primaryKey: true
    },
    nombreAdoptante: {
        type: DataTypes.STRING,
        allowNull: false
    },
    fechaSolicitud: {
        type: DataTypes.DATE,
        allowNull: false,
        defaultValue: Sequelize.NOW
    },
    estado: {
        type: DataTypes.ENUM('Pendiente', 'Aprobada', 'Rechazada'),
        allowNull: false,
        defaultValue: 'Pendiente'
    }
}, {
    tableName: 'solicitudes',
    timestamps: false
});

export {Solicitudes}
```

9. Creamos los controladores para cada objeto y aplicamos toda la programación en todas las rutas.

```
import { Mascotas } from "../modelos/mascotaModelo.js";


//crear recurso mascota
const crear = (req, res)=>{
    if(!req.body.nombre){
        res.status(400).send({
            mensaje: "El nombre de la mascota es requerido"
        })
        return;
    }

    const dataset ={
        nombre: req.body.nombre,
        descripcion: req.body.descripcion,
        edad: req.body.edad,
        genero: req.body.genero,
        peso: req.body.peso,
        imagen: req.body.imagen
    }

    //usar sequelize bsae de datos
    Mascotas.create(dataset).then((resultado)=>{
        res.status(200).json({
            mensaje: "Registro de mascota Exitoso"
        });
    }).catch((err)=>{
        res.status(500).json({
            mensaje: `Error al registrar la mascota ::: ${err}`
        });
    });
}

const editar = (req, res) => {
    const id = req.params.id;

    Mascotas.update(req.body, {
        where: { id: id }
    }).then((resultado) => {
        if (resultado == 1) {
            res.status(200).json({
                mensaje: "Mascota actualizada correctamente"
            });
        } else {
            res.status(404).json({
                mensaje: `No se pudo actualizar la mascota con id=${id}. Puede que la mascota no exista o que los datos sean iguales a los actuales.`
            });
        }
    }).catch((err) => {
        res.status(500).json({
            mensaje: `Error al actualizar la mascota con id=${id}: ${err}`
        });
    });
}

const buscarTodos = (req, res) => {
    Mascotas.findAll()
    .then((mascotas) => {
        res.status(200).json(mascotas);
    }).catch((err) => {
        res.status(500).json({
            mensaje: `Error al recuperar las mascotas: ${err}`
        });
    });
}

const buscarPorId = (req, res) => {
    const id = req.params.id;

    Mascotas.findByPk(id)
    .then((mascota) => {
        if (mascota) {
            res.status(200).json(mascota);
        } else {
            res.status(404).json({
                mensaje: `No se encontró ninguna mascota con id=${id}`
            });
        }
    }).catch((err) => {
        res.status(500).json({
            mensaje: `Error al buscar la mascota con id=${id}: ${err}`
        });
    });
}

const eliminar = (req, res) => {
    const id = req.params.id;

    Mascotas.destroy({
        where: { id: id }
    }).then((resultado) => {
        if (resultado == 1) {
            res.status(200).json({
                mensaje: "Mascota eliminada correctamente"
            });
        } else {
            res.status(404).json({
                mensaje: `No se pudo eliminar la mascota con id=${id}. Puede que la mascota no exista.`
            });
        }
    }).catch((err) => {
        res.status(500).json({
            mensaje: `Error al eliminar la mascota con id=${id}: ${err}`
        });
    });
}



export {crear, editar, buscarTodos, buscarPorId, eliminar}
```



```
import { Solicitudes } from "../modelos/solicitudModelo.js";

const crearSolicitud = (req, res) => {
    if (!req.body.nombreAdoptante) {
        res.status(400).send({
            mensaje: "El nombre del adoptante es requerido"
        });
        return;
    }

    const dataset = {
        nombreAdoptante: req.body.nombreAdoptante,
        fechaSolicitud: req.body.fechaSolicitud,
        estado: req.body.estado
    };

    Solicitudes.create(dataset)
        .then((resultado) => {
            res.status(200).json({
                mensaje: "Registro de solicitud exitoso",
                solicitud: resultado
            });
        })
        .catch((err) => {
            res.status(500).json({
                mensaje: `Error al registrar la solicitud: ${err}`
            });
        });
};

const editarSolicitud = (req, res) => {
    const id = req.params.id;

    Solicitudes.update(req.body, {
        where: { id: id }
    })
    .then((resultado) => {
        if (resultado == 1) {
            res.status(200).json({
                mensaje: "Solicitud actualizada correctamente"
            });
        } else {
            res.status(404).json({
                mensaje: `No se pudo actualizar la solicitud con id=${id}. Puede que la solicitud no exista o que los datos sean iguales.`
            });
        }
    })
    .catch((err) => {
        res.status(500).json({
            mensaje: `Error al actualizar la solicitud con id=${id}: ${err}`
        });
    });
};

const buscarSolicitudes = (req, res) => {
    Solicitudes.findAll()
    .then((solicitudes) => {
        res.status(200).json(solicitudes);
    })
    .catch((err) => {
        res.status(500).json({
            mensaje: `Error al recuperar las solicitudes: ${err}`
        });
    });
};

const buscarSolicitudPorId = (req, res) => {
    const id = req.params.id;

    Solicitudes.findByPk(id)
    .then((solicitud) => {
        if (solicitud) {
            res.status(200).json(solicitud);
        } else {
            res.status(404).json({
                mensaje: `No se encontró ninguna solicitud con id=${id}`
            });
        }
    })
    .catch((err) => {
        res.status(500).json({
            mensaje: `Error al buscar la solicitud con id=${id}: ${err}`
        });
    });
};

const eliminarSolicitud = (req, res) => {
    const id = req.params.id;

    Solicitudes.destroy({
        where: { id: id }
    })
    .then((resultado) => {
        if (resultado == 1) {
            res.status(200).json({
                mensaje: "Solicitud eliminada correctamente"
            });
        } else {
            res.status(404).json({
                mensaje: `No se pudo eliminar la solicitud con id=${id}. Puede que la solicitud no exista.`
            });
        }
    })
    .catch((err) => {
        res.status(500).json({
            mensaje: `Error al eliminar la solicitud con id=${id}: ${err}`
        });
    });
};



export { crearSolicitud, editarSolicitud, buscarSolicitudes, buscarSolicitudPorId, eliminarSolicitud };
```

10. Después de ello empezamos a maquetar las rutas

```
import express from "express";
import { buscarPorId, buscarTodos, crear, editar, eliminar } from "../controladores/mascotasController.js";

const routerMascotas = express.Router();

routerMascotas.get('/', (req, res) => {
    res.send('Bienvenido al Sitio Principal.');
});

routerMascotas.post('/crear', (req, res) => {
    crear(req,res);
});

routerMascotas.put('/editar/:id', (req, res)=>{
    editar(req,res);
});

routerMascotas.get('/buscarTodos', (req, res)=>{
    buscarTodos(req,res);
});

routerMascotas.get('/buscarPorId/:id',(req, res)=>{
    buscarPorId(req,res);
});

routerMascotas.delete('/eliminar/:id', (req, res)=>{
    eliminar(req,res);
});

export {routerMascotas}
```

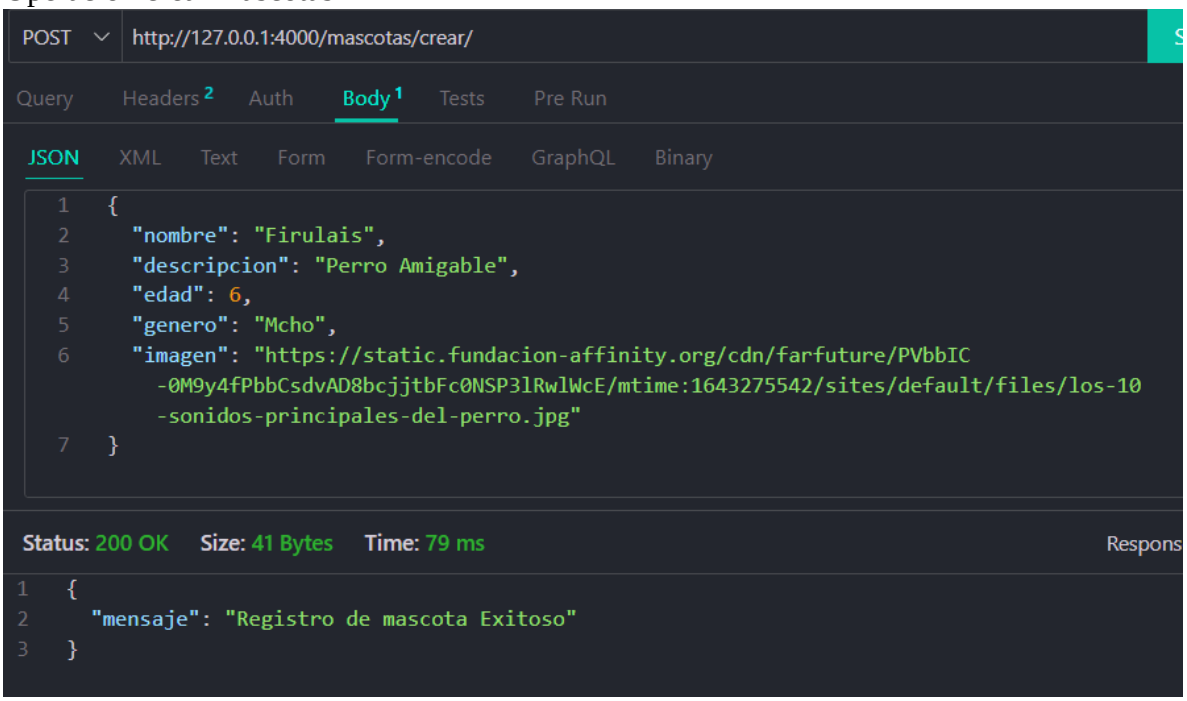
```
import express from "express";
import { buscarSolicitudPorId, buscarSolicitudes, crearSolicitud, editarSolicitud, eliminarSolicitud } from "../controladores/solicitudesController.js";

const routerSolicitudes = express.Router();

routerSolicitudes.get('/home', (req, res) => {
    res.send('Bienvenido al Sitio Principal.');
});

routerSolicitudes.post('/crearSolicitud', (req, res)=>{
    crearSolicitud(req,res);
});

routerSolicitudes.put('/editarSolicitud/:id', (req, res)=>{
    editarSolicitud(req,res);
});

routerSolicitudes.get('/buscarSolicitud', (req, res)=>{
    buscarSolicitudes(req,res);
});

routerSolicitudes.get('/buscarSolicitudPorId/:id', (req, res)=>{
    buscarSolicitudPorId(req,res);
})

routerSolicitudes.delete('/eliminarSolicitud/:id', (req, res)=>{
    eliminarSolicitud(req,res);
})
export {routerSolicitudes}
```

11. Por último, todas las configuraciones necesarias en app

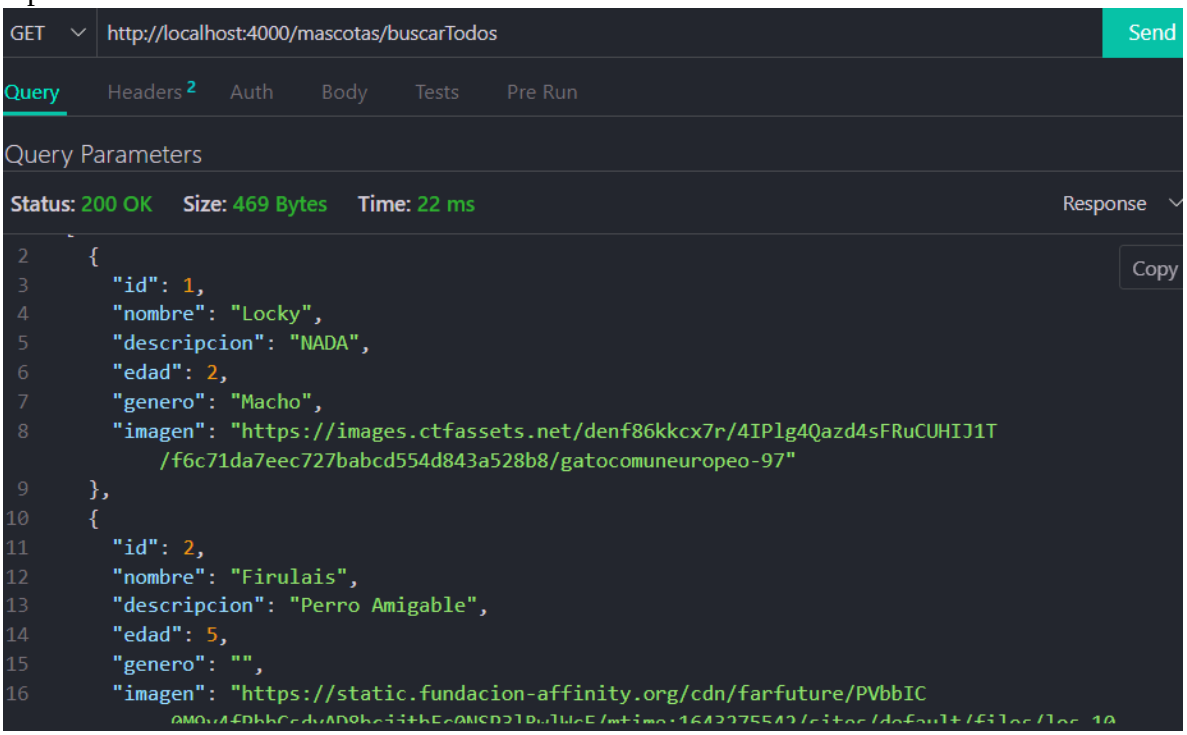
```
import express from "express";
import { routerMascotas } from "./rutas/mascotasRouter.js";
import { routerSolicitudes } from "./rutas/solicitudesRouter.js";
import { db } from "./database/conexion.js";
import cors from 'cors';

//crear instancia express
const app = express();


//json
app.use(express.json());

app.use(cors());


//verificar conexion base de datos
db.authenticate().then (()=>{
    console.log(`conexion a la  base de datos correcta`)
    }).catch(err=>{
        console.log(`conexion a la  base de datos incorrecta ${err}`)
});

//llamando rutas de mascotas
app.use("/mascotas", routerMascotas)

//llamado rutas de solicitudes
app.use("/solicitudes", routerSolicitudes)

//puerto del servidor
const PORT = 4000;

db.sync().then(()=>{
    //Abri servicio e iniciar el Servidor
    app.listen(PORT,()=>{
        console.log(`Servidor Inicializado en el puerto ${PORT}`);
    })
    }).catch(err=>{
        console.log(`Error al sincronizar la base de datos ${err}`);
});


```

12. Realizar verificación de las diferentes operaciones a través de un cliente grafico (Postman, Imnsomia, etc.), tomar capturas de pantalla que evidencien el resultado de las solicitudes realizadas.

    1. Operación crear mascotas

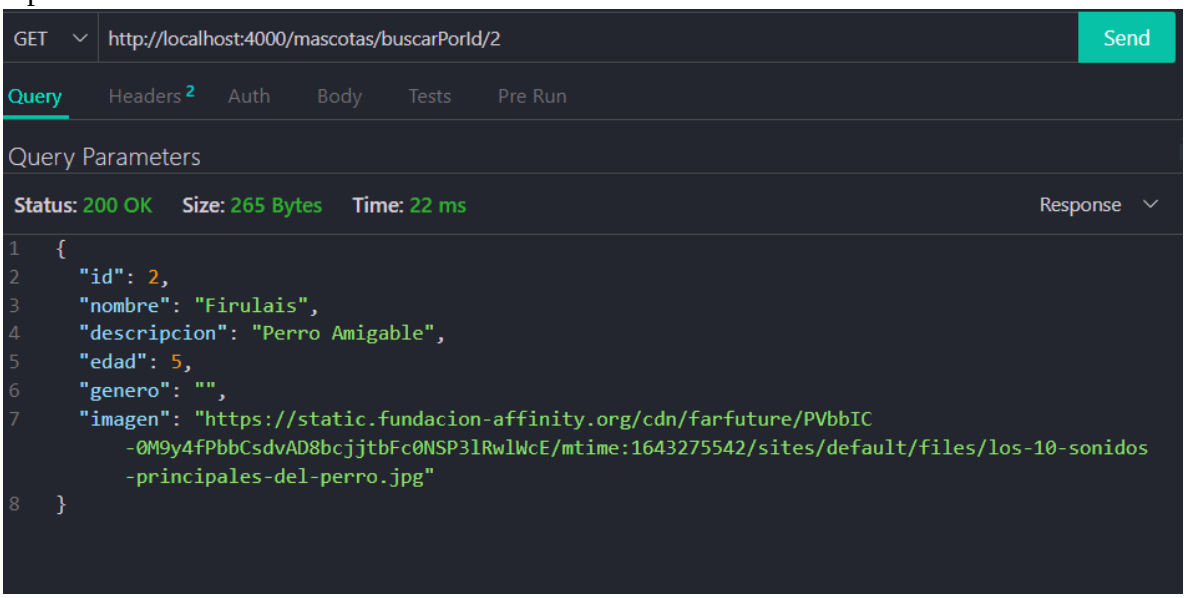


    2. Operación Buscar todas las Mascotas

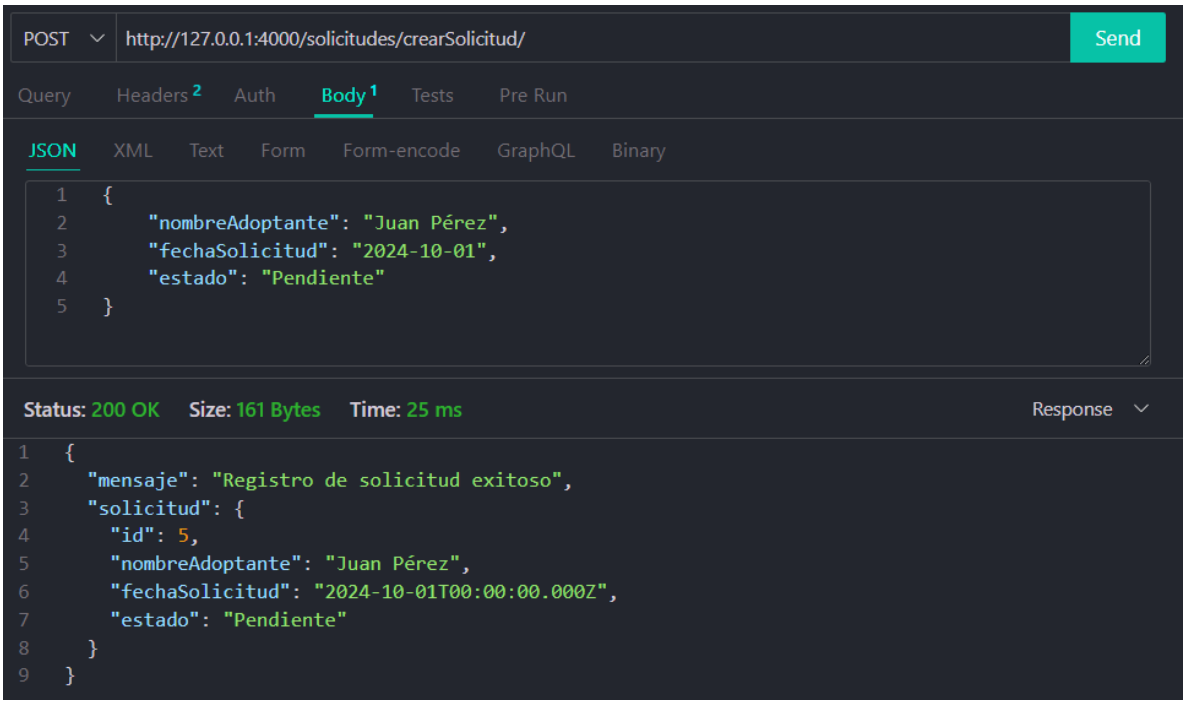


3. Operación editar mascotas

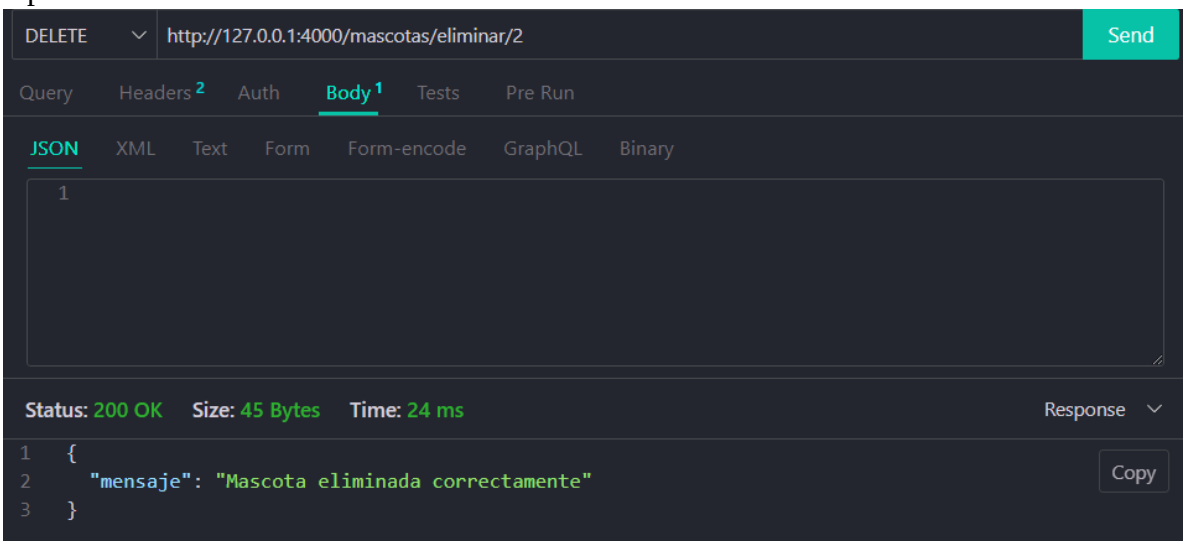
4. Operación buscar solo uno



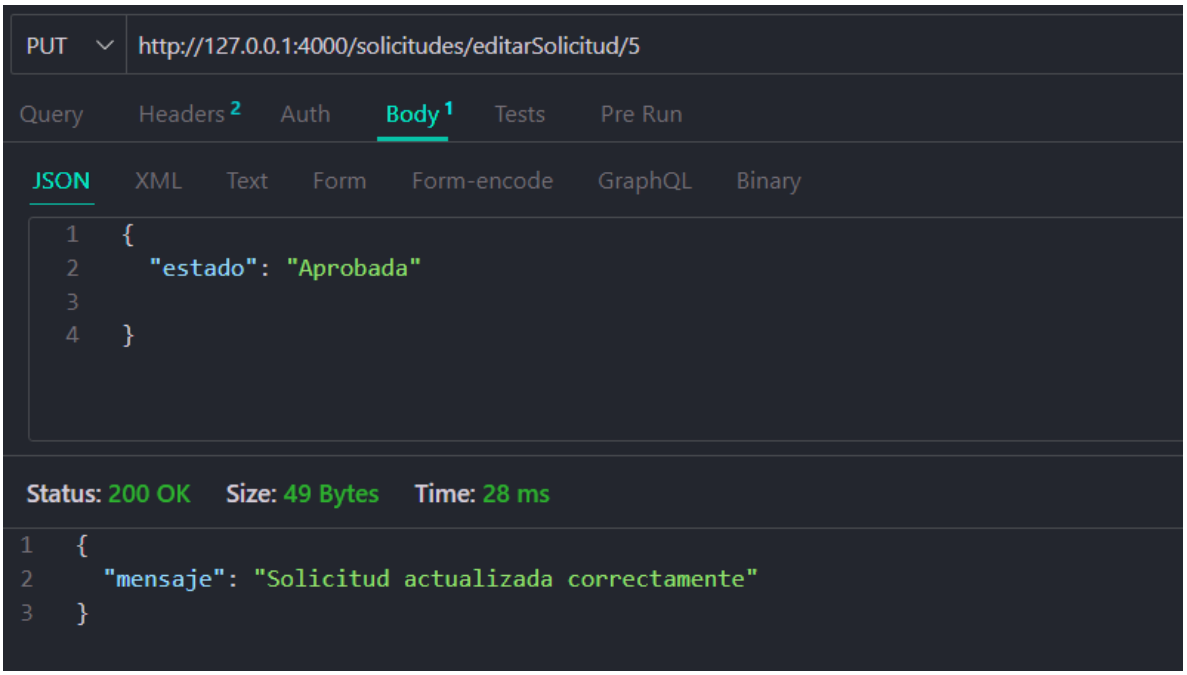
5. Operación eliminar mascota



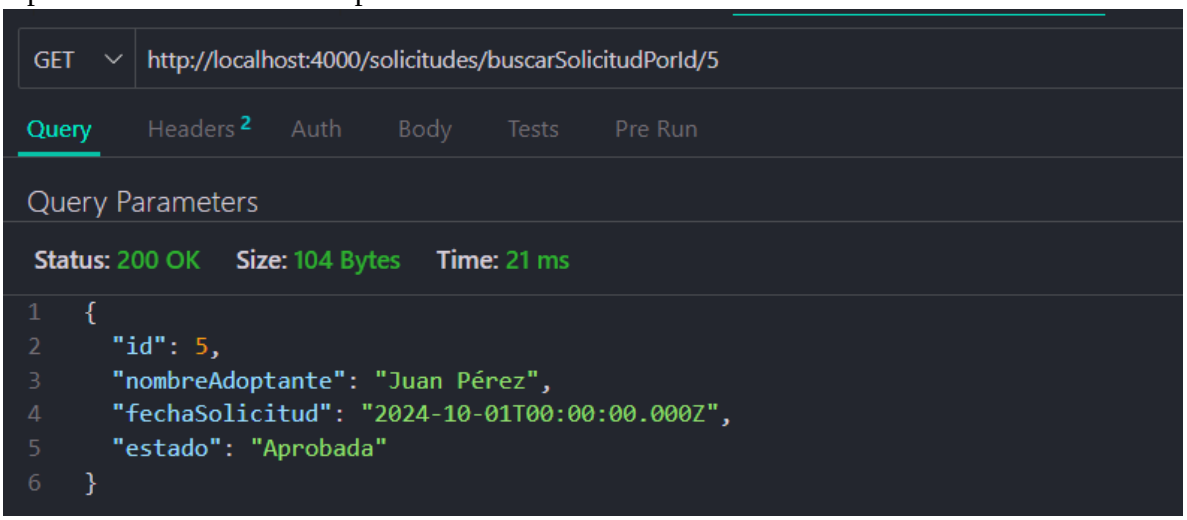
6. Operación crear Solicitud



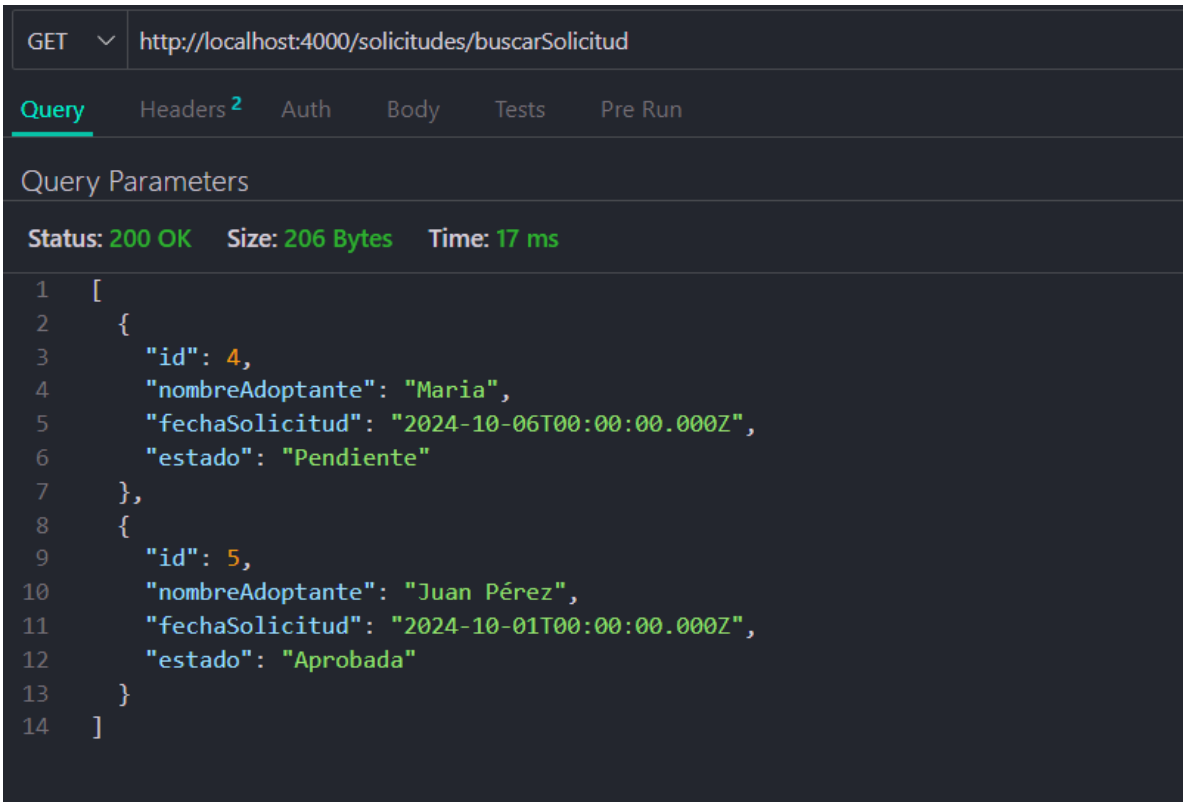
7. Operación Editar Solicitud



8. Operación buscar solicitud por id



9. Operación para buscar todas las solicitudes



10. Operación para eliminar solicitud