Diplomado de actualización en nuevas tecnologías para el desarrollo de software

Taller Unidad 2 Backend

Hecho por:

Anthony Danilo Parra

Universidad de Nariño

San Juan de Pasto

2023

1. Crear una base de datos MYSQL que permita llevar el registro de mascotas (perros y gatos), así como también el proceso de solicitud de adopción de estas.

Primero se crea la base de datos:

```
-- Crear la base de datos
CREATE DATABASE adopcion;
use adopcion;
```

Se crea la tabla para almacenar información de las mascotas

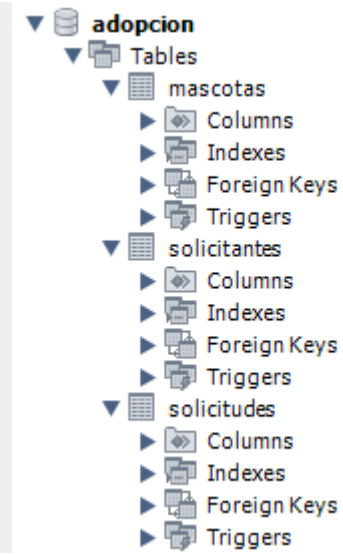
```
create table mascotas(
    id int primary key auto_increment,
    nombre varchar(100) not null,
    tipo varchar(10) not null,
    edad int,
    estado_adopcion ENUM('Disponible', 'Adoptado') DEFAULT 'Disponible'
);
```

Se crea la tabla para almacenar información de los solicitantes:

```
create table solicitantes(
    id INT PRIMARY KEY auto_increment,
    nombre varchar(100) not null,
    correo varchar(100) not null,
    telefono varchar(20),
    direccion TEXT
);
```

Tabla para almacenar información sobre las solicitudes de adopción:

```
create table solicitudes(
    id int primary key auto_increment,
    mascota_id int,
    solicitante_id int,
    fecha_solicitud date,
    estado ENUM('Pendiente', 'Aprobada', 'Rechazada') DEFAULT 'Pendiente',
    FOREIGN KEY (mascota_id) REFERENCES mascotas(id),
    FOREIGN KEY (solicitante_id) REFERENCES solicitantes(id)
);
```



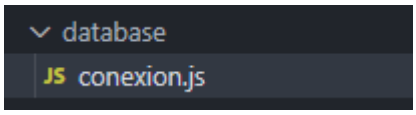
2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las mascotas dadas en adopción por la empresa (La empresa debe contar con un nombre).

   Se debe hacer uso correcto de los verbos HTTP dependiendo de la tarea a realizar.

Se instalan las dependencias necesarias:

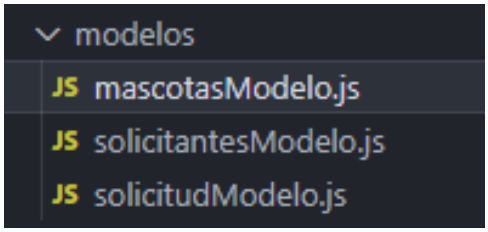
```
  "scripts": {
    "start": "nodemon ./src/app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.0.2"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.6.5",
    "sequelize": "^6.35.2"
  }
}
```

Se hace la conexión a la base de datos




```
Click here to ask Blackbox to help you code faster
import Sequelize from "sequelize";

const db = new Sequelize("adopcion", "mascotas", "mascotas2023",{
    host: 'localhost',
    dialect: 'mysql'
});

export {db}
```

A continuacion se crea el modelo para cada una de las tablas de la base de datos



JS mascotasModelo.js

modelos > JS mascotasModelo.js > ...

Click here to ask Blackbox to help you code faster

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

const mascotas = db.define("mascotas", {
    id: {
        type: Sequelize.INTEGER,
        primaryKey: true,
        autoIncrement: true,
        allowNull: false
    },
    nombre: {
        type: Sequelize.STRING,
        allowNull: false
    },
    tipo: {
        type: Sequelize.STRING(10),
        allowNull: false
    },
    edad: {
        type: Sequelize.INTEGER,
        allowNull: true
    },
    estado_adopcion: {
        type: Sequelize.ENUM('Disponible', 'Adoptado'),
        defaultValue: 'Disponible'
    }
});

export { mascotas };

```

JS solicitantesModelo.js

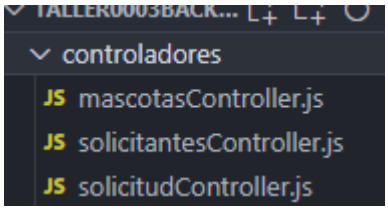
modelos > JS solicitantesModelo.js > solicitantes

Click here to ask Blackbox to help you code faster

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

const solicitantes = db.define('solicitantes', {
    nombre: {
      type: Sequelize.STRING,
      allowNull: false,
    },
    correo: {
      type: Sequelize.STRING,
      allowNull: false,
      unique: true,
      validate: {
        isEmail: true,
      },
    },
    telefono: {
      type: Sequelize.STRING,
    },
    direccion: {
      type: Sequelize.TEXT,
    },
  });

export { solicitantes };
```

En solicitudModelo.js se crea las asociaciones de las tablas de mascotas y solicitantes

Luego se crearon los controladores



Tomando como ejemplo mascotasController.js se hizo el CRUD

Crear

```
JS mascotasController.js ×
controladores > JS mascotasController.js > [⊙] crear
Click here to ask Blackbox to help you code faster
import {mascotas} from "../modelos/mascotasModelo.js";

//Crear un recurso
const crear = (req, res) => {
    if (!req.body.nombre) {
        return res.status(400).json({ mensaje: "El nombre no puede estar vacío." });
    }

    const dataset = {
        nombre: req.body.nombre,
        tipo: req.body.tipo,
        edad: req.body.edad

    };

    // Usar Sequelize para crear el recurso
    mascotas.create(dataset)
        .then((resultado) => {
            console.log("Registro creado correctamente:", resultado);
            return res.status(201).json({ mensaje: "Registro creado correctamente", resultado });
        })
        .catch((err) => {
            console.error("Error al crear el registro:", err);
            return res.status(500).json({ mensaje: "Error al crear el registro", error: err.message });
        });
};
```

Buscar por Id

```
const buscarId = (req, res) => {
    const mascotaId = req.params.id;

    mascotas.findByPk(mascotaId)
        .then((mascota) => {
            if (!mascota) {
                return res.status(404).json({ mensaje: "Mascota no encontrada" });
            }
            console.log("Mascota encontrada:", mascota);
            return res.status(200).json({ mensaje: "Mascota encontrada", mascota });
        })
        .catch((err) => {
            console.error("Error al buscar la mascota por id:", err);
            return res.status(500).json({ mensaje: `Error al buscar la mascota por id: ${err.message}` });
        });
};
```

Buscar todos

```
const buscar = (req, res) => {

    mascotas.findAll()
        .then((mascotas) => {
            if (!mascotas) {
                return res.status(404).json({ mensaje: "Mascotas no encontradas" });
            }
            console.log("Mascotas encontradas:", mascotas);
            return res.status(200).json({ mensaje: "Mascotas encontradas", mascotas });
        })
        .catch((err) => {
            console.error("Error al buscar la mascota por id:", err);
            return res.status(500).json({ mensaje: `Error al buscar la mascota por id: ${err.message}` });
        });
};
```

Actualizar

```
const actualizar = (req, res) => {
    const mascotaId = req.params.id;
    const nuevosDatos = {
        nombre: req.body.nombre,
        tipo: req.body.tipo,
        edad: req.body.edad
    };

    mascotas.findByPk(mascotaId)
        .then((mascota) => {
            if (!mascota) {
                return res.status(404).json({ mensaje: "Mascota no encontrada" });
            }

            // Actualizar los datos de la mascota
            return mascota.update(nuevosDatos);
        })
        .then((mascotaActualizada) => {
            console.log("Mascota actualizada:", mascotaActualizada);
            return res.status(200).json({ mensaje: "Mascota actualizada correctamente", mascotaActualizada });
        })
        .catch((err) => {
            console.error("Error al actualizar la mascota:", err);
            return res.status(500).json({ mensaje: `Error al actualizar la mascota: ${err.message}` });
        });
};
```

Eliminar

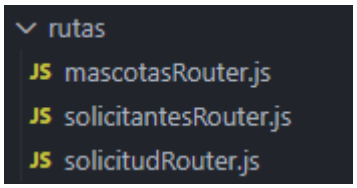
```
const eliminar = (req, res) => {
    const mascotaId = req.params.id;

    mascotas.findByPk(mascotaId)
        .then((mascota) => {
            if (!mascota) {
                return res.status(404).json({ mensaje: "Mascota no encontrada" });
            }

            // Eliminar la mascota
            return mascota.destroy();
        })
        .then(() => {
            console.log("Mascota eliminada correctamente");
            return res.status(200).json({ mensaje: "Mascota eliminada correctamente" });
        })
        .catch((err) => {
            console.error("Error al eliminar la mascota:", err);
            return res.status(500).json({ mensaje: `Error al eliminar la mascota: ${err.message}` });
        });
};

export { crear, buscarId, buscar, actualizar, eliminar};
```

Para los controladores solicitanteController.js y solicitudController.js se utilizó el mismo código,

Pero para solicitudController.js en “actualizar” se colocó el “include” para proporcionar información acerca de las tablas relacionadas a solicitud, como se muestra a continuación:

```
};
```

Luego se crearon las rutas para cada uno de los controladores:



mascotasRouter.js

rutas > mascotasRouter.js > routerMascotas.post("/crear") callback

Click here to ask Blackbox to help you code faster

```
import express from "express";
import { crear, buscarId, buscar, actualizar, eliminar } from "../controladores/mascotasController.js";

const routerMascotas = express.Router();

routerMascotas.get("/",(req,res) =>{
    res.send("Bienvenido a Mascotas");
});

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

routerMascotas.get("/buscar/",(req,res) =>{
    buscar(req,res);
});


routerMascotas.get("/buscar/:id",(req,res) =>{
    buscarId(req,res);
});

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

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

export {routerMascotas}
```

solicitantesRouter.js

rutas > solicitantesRouter.js > ...

Click here to ask Blackbox to help you code faster

```
import express from "express";
import { crear, buscarId, buscar, actualizar, eliminar } from "../controladores/solicitantesController.js";

const routerPersonas = express.Router();

routerPersonas.get("/",(req,res) =>{
    res.send("Bienvenido");
});

routerPersonas.post("/crear",(req,res) =>{
   crear(req,res);
});

routerPersonas.get("/buscar/",(req,res) =>{
    buscar(req,res);
});


routerPersonas.get("/buscar/:id",(req,res) =>{
    buscarId(req,res);
});

routerPersonas.put("/actualizar/:id",(req,res) =>{
    actualizar(req,res);
});

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

export {routerPersonas}
```

solicitudRouter.js

rutas > solicitudRouter.js > ...

Click here to ask Blackbox to help you code faster

```
import express from "express";
import { crear, buscarId, buscar, actualizar, eliminar } from "../controladores/solicitudController.js";

const routerSolicitud = express.Router();

routerSolicitud.get("/",(req,res) =>{
    res.send("Bienvenido");
});

routerSolicitud.post("/crear",(req,res) =>{
   crear(req,res);
});

routerSolicitud.get("/buscar/",(req,res) =>{
    buscar(req,res);
});


routerSolicitud.get("/buscar/:id",(req,res) =>{
    buscarId(req,res);
});

routerSolicitud.put("/actualizar/:id",(req,res) =>{
    actualizar(req,res);
});

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

export {routerSolicitud}
```

Y por último el archivo app.js queda de esta manera:

Aquí se importan las rutas

```
import express from "express";
import { db } from "../database/conexion.js";
import { routerMascotas } from "../rutas/mascotasRouter.js";
import { routerPersonas } from "../rutas/solicitantesRouter.js";
import { routerSolicitud } from "../rutas/solicitudRouter.js";
```

Se crear una constante app, se verifica la conexión con la base de datos y se configura la constante PORT

```
const app = express();

app.use(express.json());

//Verificar Conexion a Base de Datos
db.authenticate().then(()=>{
    console.log(`Base de Datos conectada de manera exitosa`);
}).catch(err=>{
    console.log(`Error al conectarse a la Base de Datos ::: ${err}`);
})

const PORT = 8000;
```

Se llaman a esas rutas

```
app.use("/mascotas", routerMascotas);
app.use("/solicitantes", routerPersonas);
app.use("/solicitudes", routerSolicitud);
```

Se verifica la sincronización a la base de datos

```
//Verificar que pueda sincronizar con la base de datos
db.sync().then(()=>{
    app.listen(PORT,()=>{
        console.log(`Servidor Inicializado en puerto ${PORT}`);
    });
}).catch(err=>{
    console.log(`Error al sincronizar Base de Datos ${err}`);
});
```

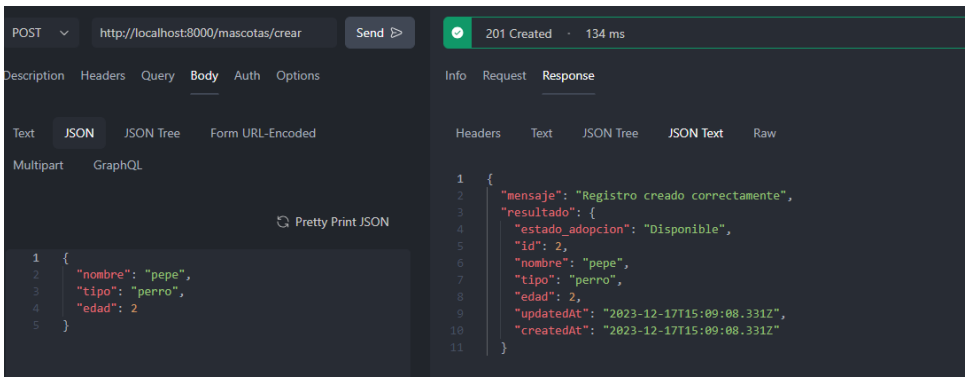
Se ejecuta la aplicación

```
PS C:\Users\ANTHONY\Desktop\Diplomado\Unidad2\Backend\Taller0003Backend> npm run start

> taller0003backend@1.0.0 start
> nodemon ./src/app.js

[nodemon] 3.0.2
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./src/app.js`
Executing (default): SELECT 1+1 AS result
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'mascotas' AND TABLE_SCHEMA = 'adopcion'
Base de Datos conectada de manera exitosa
Executing (default): SHOW INDEX FROM `mascotas`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'solicitantes' AND TABLE_SCHEMA = 'adopcion'
Executing (default): SHOW INDEX FROM `solicitantes`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'solicitudes' AND TABLE_SCHEMA = 'adopcion'
Executing (default): SHOW INDEX FROM `solicitudes`
Servidor Inicializado en puerto 8000
```

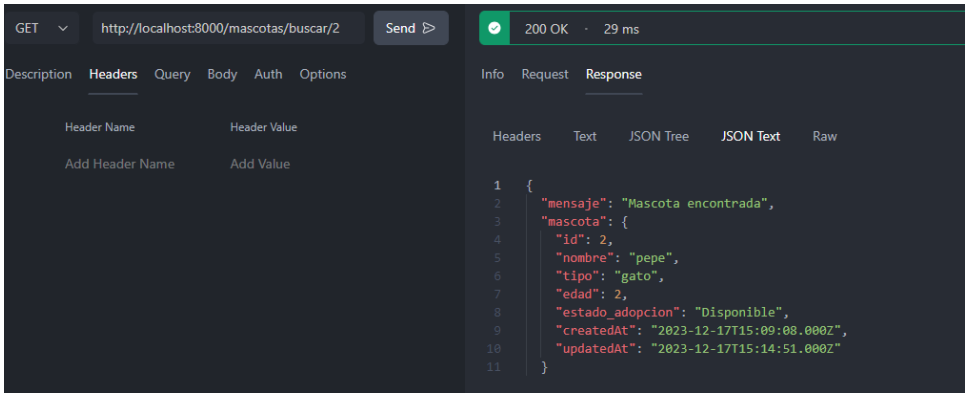
3. Realizar verificación de las diferentes operaciones a través de un cliente grafico (Postman, Imnsomia, etc.), tomar capturas de pantalla que evidencien el resultado de las solicitudes realizadas.

Solicitudes en mascotas

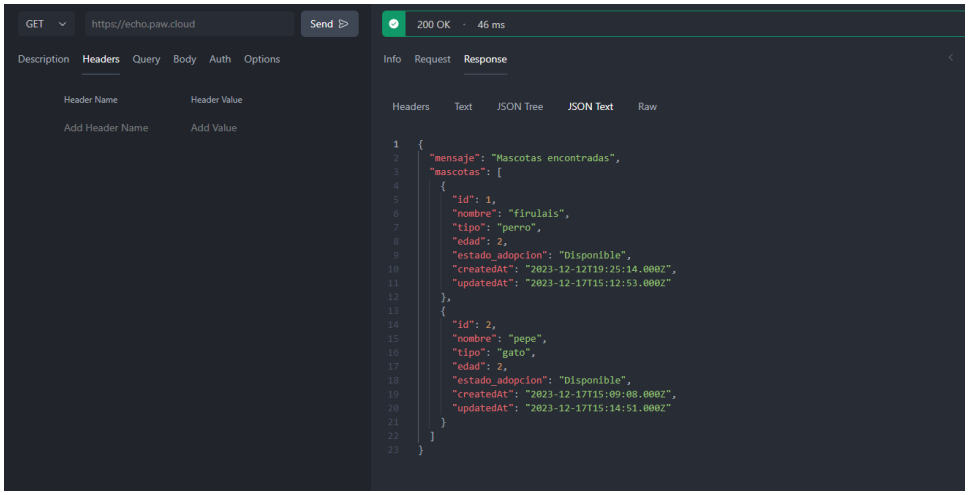
Crear



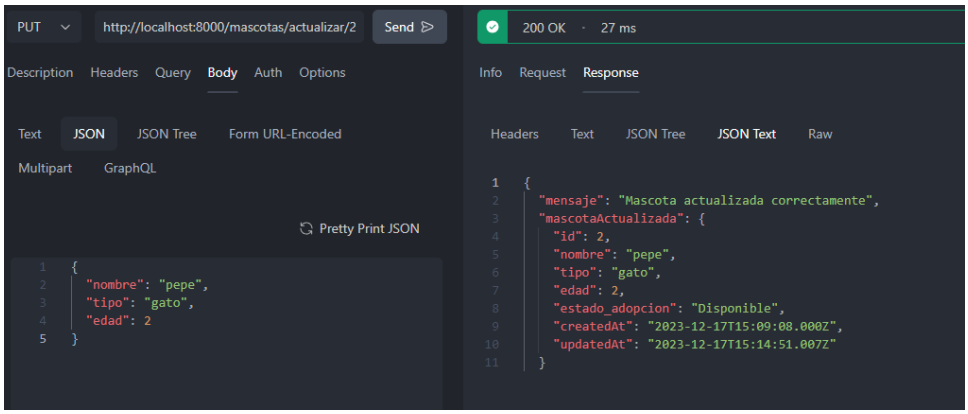
Buscar ID



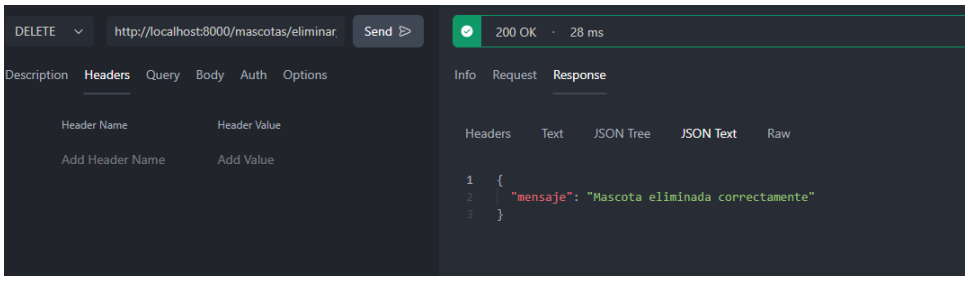
Buscar

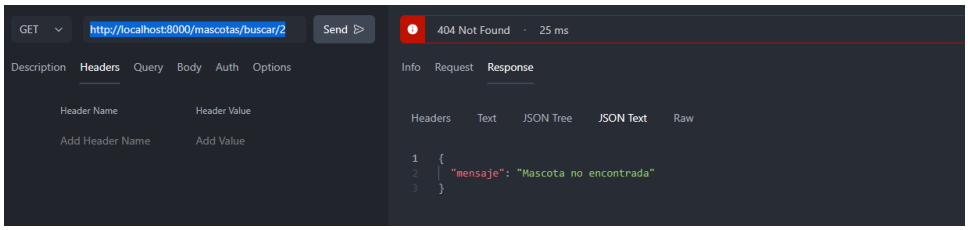


Actualizar



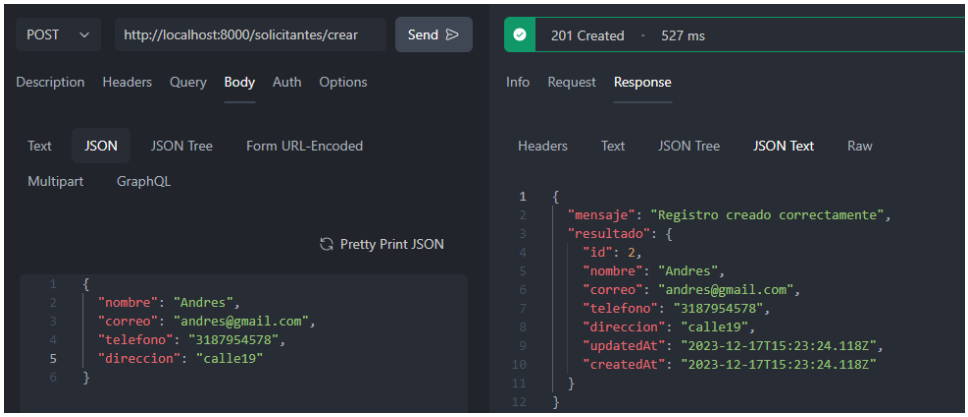
Eliminar



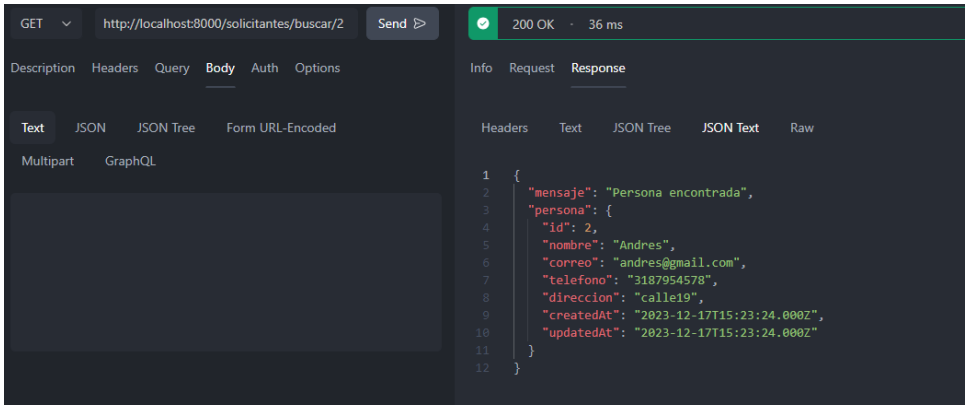


En solicitantes

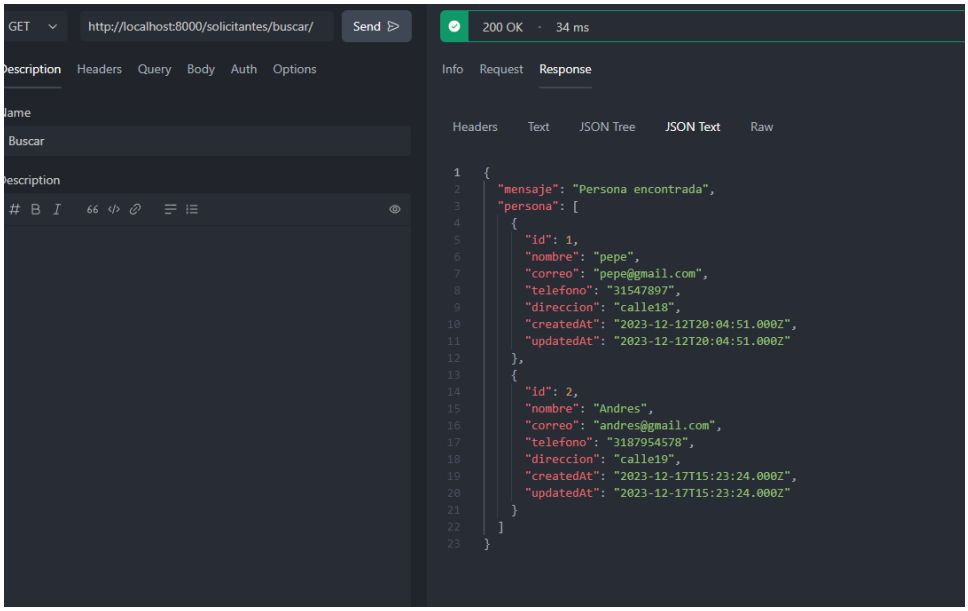
Crear



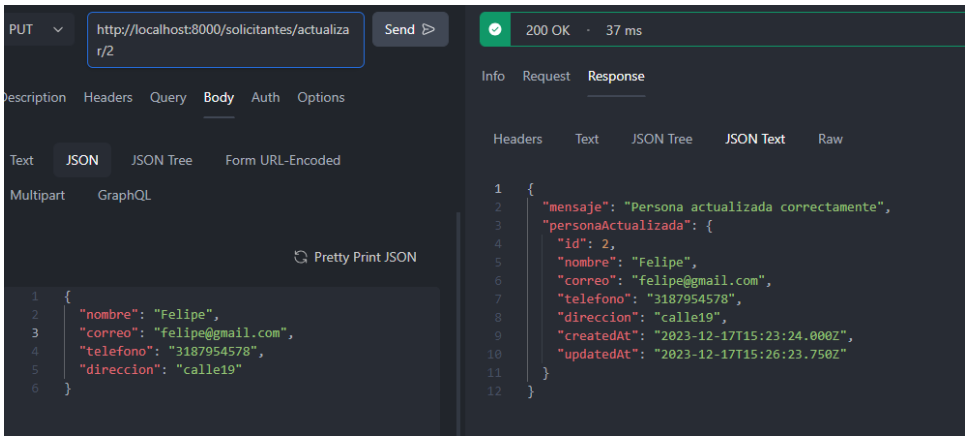
Buscar ID



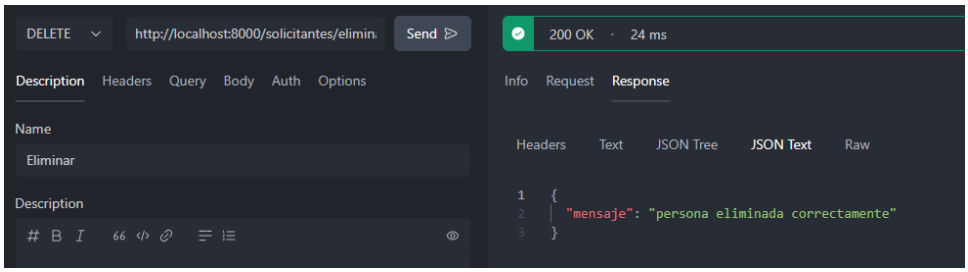
Buscar

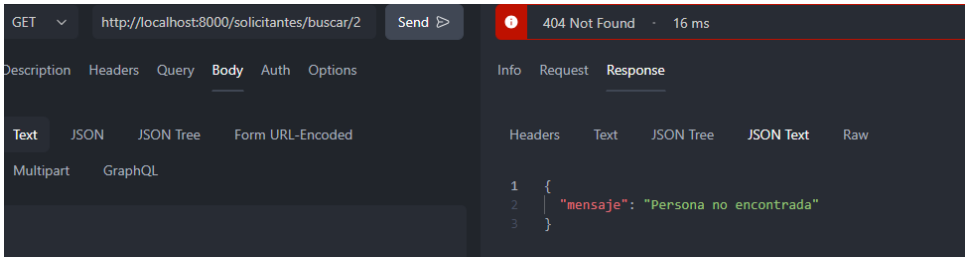


Actualizar



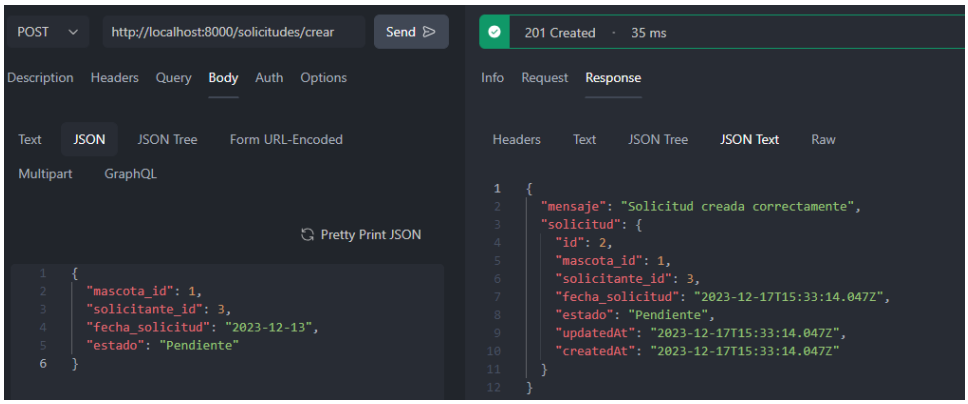
Eliminar



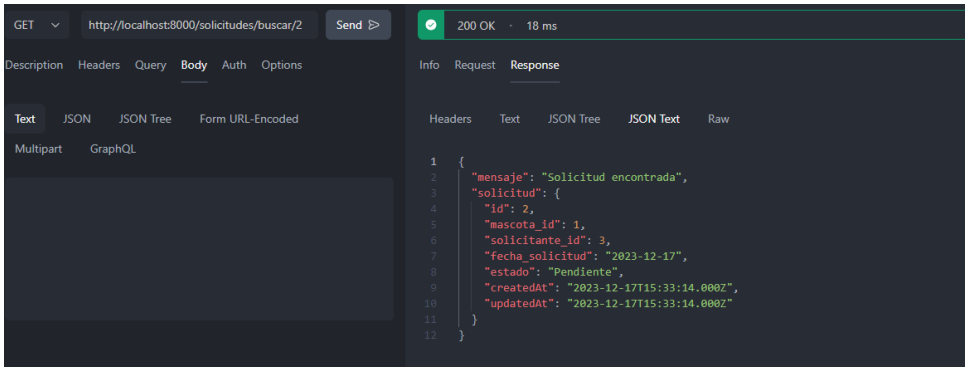


En solicitud

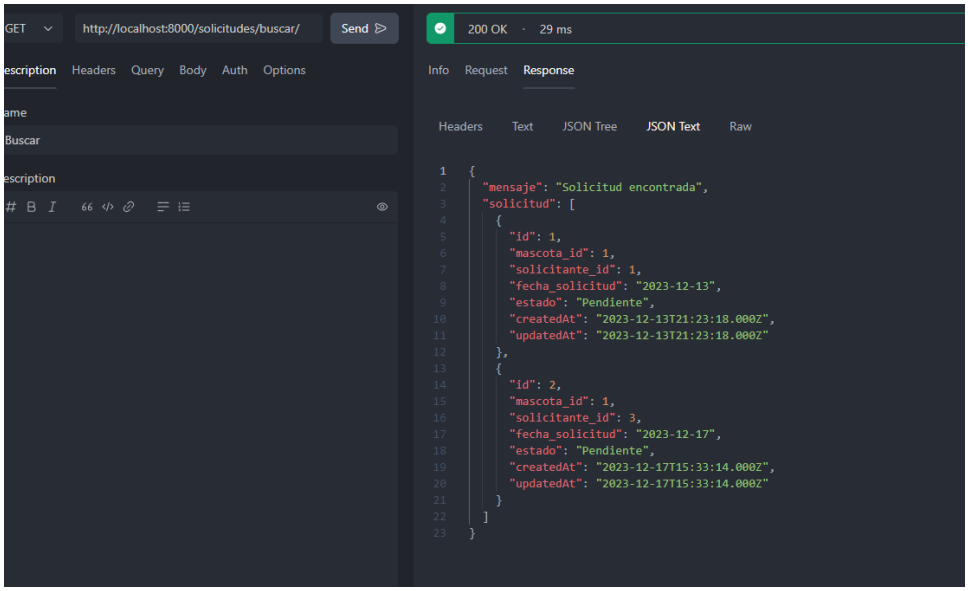
Crear



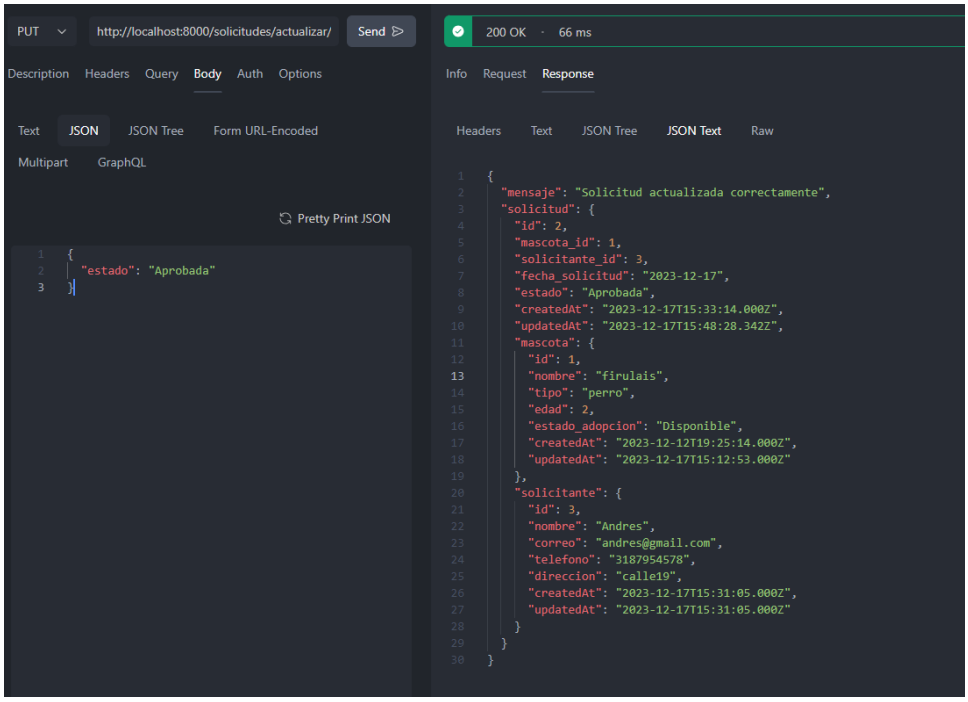
Buscar ID



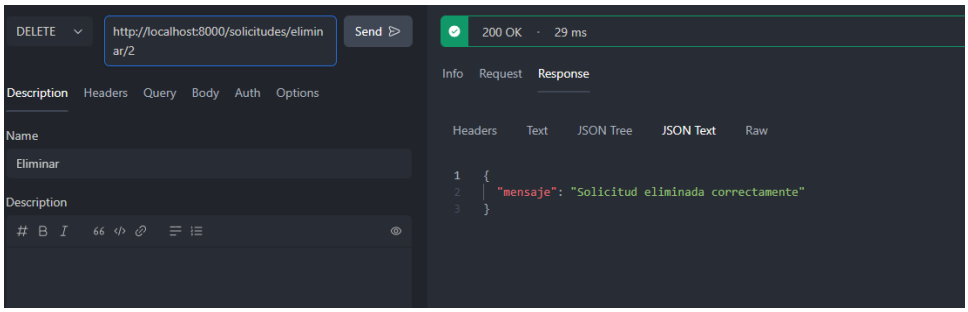
Buscar

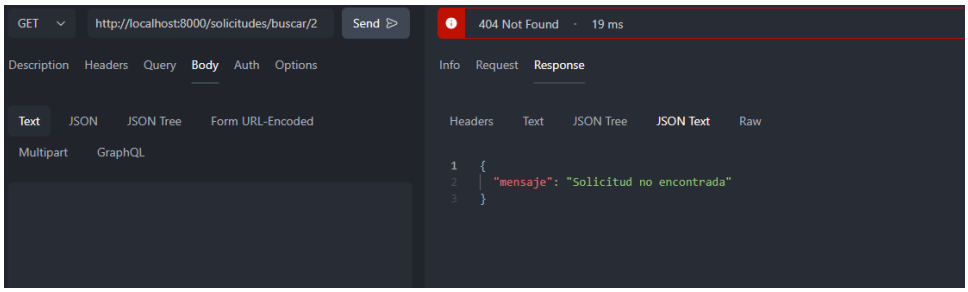


Actualizar



Eliminar


DELETE
http://localhost:8000/solicitudes/eliminar/2
Send
200 OK · 29 ms
Description Headers Query Body Auth Options
Info Request Response
Name
Eliminar
Description
Headers Text JSON Tree JSON Text Raw
{
"mensaje": "Solicitud eliminada correctamente"
}


GET
http://localhost:8000/solicitudes/buscar/2
Send
404 Not Found · 19 ms
Description Headers Query Body Auth Options
Info Request Response
Text JSON JSON Tree Form URL-Encoded
Multipart GraphQL
Headers Text JSON Tree JSON Text Raw
{
"mensaje": "Solicitud no encontrada"
}