

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

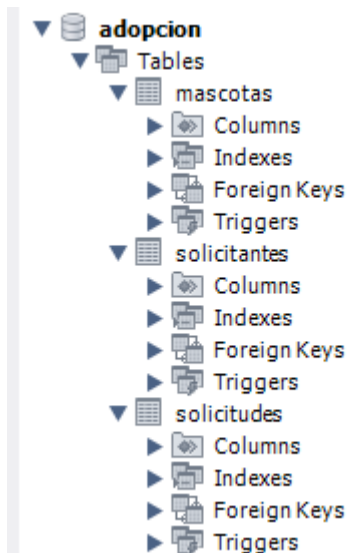
```
• create table mascotas(
    id int primary key auto_increment,
    nombre varchar(100) not null,
    tipo varchar(10) not null,
    edad int,
    estado_adopcion ENUM('Disponible', 'Adoptado') DEFAULT 'Disponible'
);
```

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)
);
```



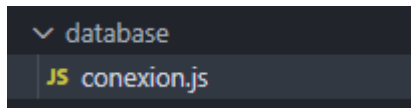
2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las mascotas dadas en adopción por la empresa (La empresa debe contar con un nombre).

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

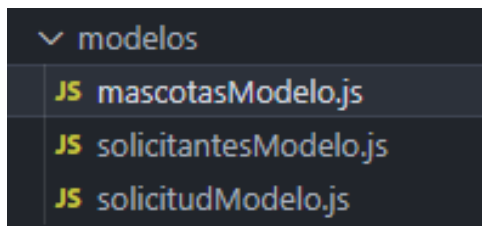


```
💡 Click here to ask Blackbox to help you code faster
import Sequelize from "sequelize";

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 X

modelos > JS mascotasModelo.js > ...

Click here to ask Blackbox to help you code faster

```
1 import Sequelize from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const mascotas = db.define("mascotas", {
5   id: {
6     type: Sequelize.INTEGER,
7     primaryKey: true,
8     autoIncrement: true,
9     allowNull: false
10  },
11  nombre: {
12    type: Sequelize.STRING,
13    allowNull: false
14  },
15  tipo: {
16    type: Sequelize.STRING(10),
17    allowNull: false
18  },
19  edad: {
20    type: Sequelize.INTEGER,
21    allowNull: true
22  },
23  estado_adopcion: {
24    type: Sequelize.ENUM('Disponible', 'Adoptado'),
25    defaultValue: 'Disponible'
26  }
27 });
28
29 export { mascotas };
30
```

JS solicitantesModelo.js X

modelos > JS solicitantesModelo.js > [🔍] solicitantes

Click here to ask Blackbox to help you code faster

```
1 import Sequelize from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const solicitantes = db.define('solicitantes', {
5   nombre: {
6     type: Sequelize.STRING,
7     allowNull: false,
8   },
9   correo: {
10    type: Sequelize.STRING,
11    allowNull: false,
12    unique: true,
13    validate: {
14      isEmpty: true,
15    },
16  },
17  telefono: {
18    type: Sequelize.STRING,
19  },
20  direccion: {
21    type: Sequelize.TEXT,
22  },
23 });
24
25 export { solicitantes };
```

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

```
modelos > JS solicitudModelo.js > ...
  Click here to ask Blackbox to help you code faster
1 import Sequelize from "sequelize";
2 import { db } from "../database/conexion.js";
3 import { mascotas } from "../mascotasModelo.js"
4 import { solicitantes } from "../solicitantesModelo.js"
5
6 const solicitud = db.define('solicitudes', {
7   mascota_id: {
8     type: Sequelize.INTEGER,
9     allowNull: false,
10  },
11  solicitante_id: {
12    type: Sequelize.INTEGER,
13    allowNull: false,
14  },
15  fecha_solicitud: {
16    type: Sequelize.DATE,
17    allowNull: false,
18  },
19  estado: {
20    type: Sequelize.ENUM('Pendiente', 'Aprobada', 'Rechazada'),
21    defaultValue: 'Pendiente',
22  },
23 });
24
25 // Asociaciones
26 solicitud.belongsTo(mascotas, { foreignKey: 'mascota_id' });
27 solicitud.belongsTo(solicitantes, { foreignKey: 'solicitante_id' });
28
29 export { solicitud };
```

Luego se crearon los controladores

```
▼ TALLER0003BACK... L+ L+ O
  ▼ controladores
    JS mascotasController.js
    JS solicitantesController.js
    JS solicitudController.js
```

Tomando como ejemplo mascotasController.js se hizo el CRUD

Crear

```
JS mascotasController.js X
controladores > JS mascotasController.js > crear
Click here to ask Blackbox to help you code faster
1 import {mascotas} from "../modelos/mascotasModelo.js";
2
3 //Crear un recurso
4 const crear = (req, res) => {
5   if (!req.body.nombre) {
6     return res.status(400).json({ mensaje: "El nombre no puede estar vacío." });
7   }
8
9   const dataset = {
10     nombre: req.body.nombre,
11     tipo: req.body.tipo,
12     edad: req.body.edad
13   };
14
15   // Usar Sequelize para crear el recurso
16   mascotas.create(dataset)
17     .then((resultado) => {
18       console.log("Registro creado correctamente:", resultado);
19       return res.status(201).json({ mensaje: "Registro creado correctamente", resultado });
20     })
21     .catch((err) => {
22       console.error("Error al crear el registro:", err);
23       return res.status(500).json({ mensaje: "Error al crear el registro", error: err.message });
24     });
25 };
26
```

Buscar por Id

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

  mascotas.findPk(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

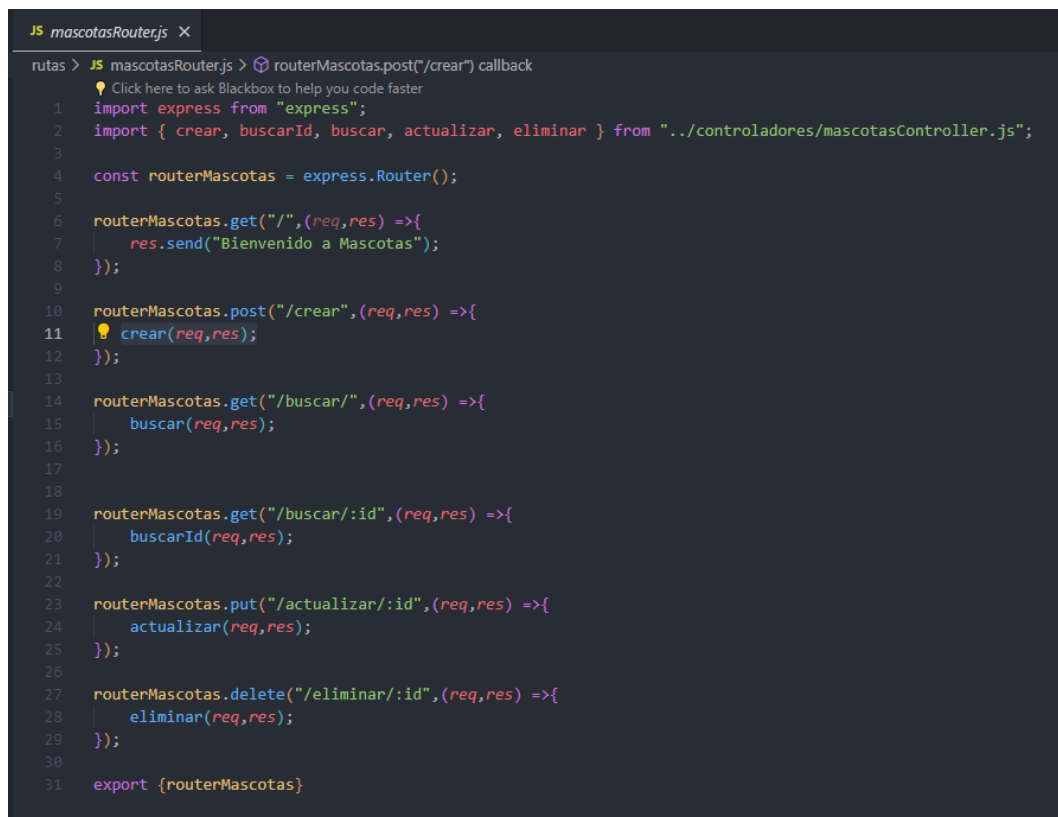
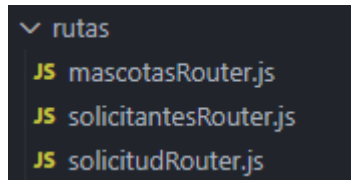
```
1 const actualizar = (req, res) => {
2   const mascotaId = req.params.id;
3   const nuevosDatos = {
4     nombre: req.body.nombre,
5     tipo: req.body.tipo,
6     edad: req.body.edad
7   };
8
9   mascotas.findByPk(mascotaId)
10    .then((mascota) => {
11      if (!mascota) {
12        return res.status(404).json({ mensaje: "Mascota no encontrada" });
13      }
14
15      // Actualizar Los datos de la mascota
16      return mascota.update(nuevosDatos);
17    })
18    .then((mascotaActualizada) => {
19      console.log("Mascota actualizada:", mascotaActualizada);
20      return res.status(200).json({ mensaje: "Mascota actualizada correctamente", mascotaActualizada });
21    })
22    .catch((err) => {
23      console.error("Error al actualizar la mascota:", err);
24      return res.status(500).json({ mensaje: `Error al actualizar la mascota: ${err.message}` });
25    });
26 };
27
```

Eliminar

```
1 const eliminar = (req, res) => {
2   const mascotaId = req.params.id;
3
4   mascotas.findByPk(mascotaId)
5    .then((mascota) => {
6      if (!mascota) {
7        return res.status(404).json({ mensaje: "Mascota no encontrada" });
8      }
9
10     // Eliminar la mascota
11     return mascota.destroy();
12   })
13   .then(() => {
14     console.log("Mascota eliminada correctamente");
15     return res.status(200).json({ mensaje: "Mascota eliminada correctamente" });
16   })
17   .catch((err) => {
18     console.error("Error al eliminar la mascota:", err);
19     return res.status(500).json({ mensaje: `Error al eliminar la mascota: ${err.message}` });
20   });
21 };
22
23 export { crear, buscarId, buscar, actualizar, eliminar};
24
```

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:



```

JS solicitantesRouter.js X
rutas > JS solicitantesRouter.js > ...
  Click here to ask Blackbox to help you code faster
1  import express from "express";
2  import { crear, buscarId, buscar, actualizar, eliminar } from "../controladores/solicitudesController.js";
3
4  const routerPersonas = express.Router();
5
6  routerPersonas.get("/", (req, res) =>{
7    res.send("Bienvenido");
8  });
9
10 routerPersonas.post("/crear", (req, res) =>{
11   crear(req, res);
12 });
13
14 routerPersonas.get("/buscar/", (req, res) =>{
15   buscar(req, res);
16 });
17
18
19 routerPersonas.get("/buscar/:id", (req, res) =>{
20   buscarId(req, res);
21 });
22
23 routerPersonas.put("/actualizar/:id", (req, res) =>{
24   actualizar(req, res);
25 });
26
27 routerPersonas.delete("/eliminar/:id", (req, res) =>{
28   eliminar(req, res);
29 });
30
31 export {routerPersonas}

```

```

JS solicitudRouter.js X
rutas > JS solicitudRouter.js > ...
  Click here to ask Blackbox to help you code faster
1  import express from "express";
2  import { crear, buscarId, buscar, actualizar, eliminar } from "../controladores/solicitudController.js";
3
4  const routerSolicitud = express.Router();
5
6  routerSolicitud.get("/", (req, res) =>{
7    res.send("Bienvenido");
8  });
9
10 routerSolicitud.post("/crear", (req, res) =>{
11   crear(req, res);
12 });
13
14 routerSolicitud.get("/buscar/", (req, res) =>{
15   buscar(req, res);
16 });
17
18
19 routerSolicitud.get("/buscar/:id", (req, res) =>{
20   buscarId(req, res);
21 });
22
23 routerSolicitud.put("/actualizar/:id", (req, res) =>{
24   actualizar(req, res);
25 });
26
27 routerSolicitud.delete("/eliminar/:id", (req, res) =>{
28   eliminar(req, res);
29 });
30
31 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 crea 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

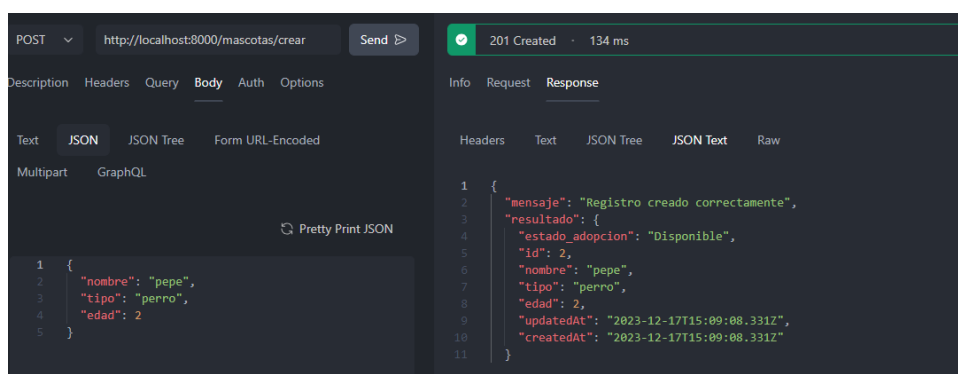
> taller0003backend@1.0.0 start
> nodemon ./src/app.js

[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 = 'solicitudes' AND TABLE_SCHEMA = 'adopcion'
Executing (default): SHOW INDEX FROM `solicitudes`
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
```

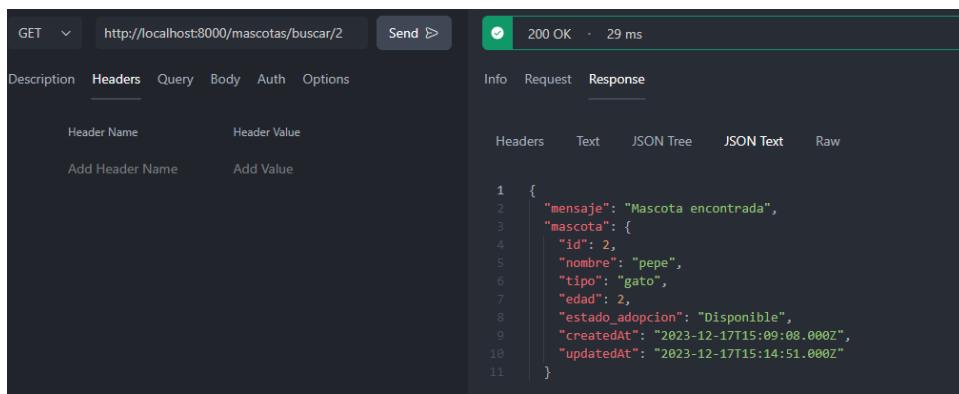
3. Realizar verificación de las diferentes operaciones a través de un cliente grafico (Postman, Imnsomia, etc.), tomar capturas de pantalla que evidencien el resultado de las solicitudes realizadas.

Solicitudes en mascotas

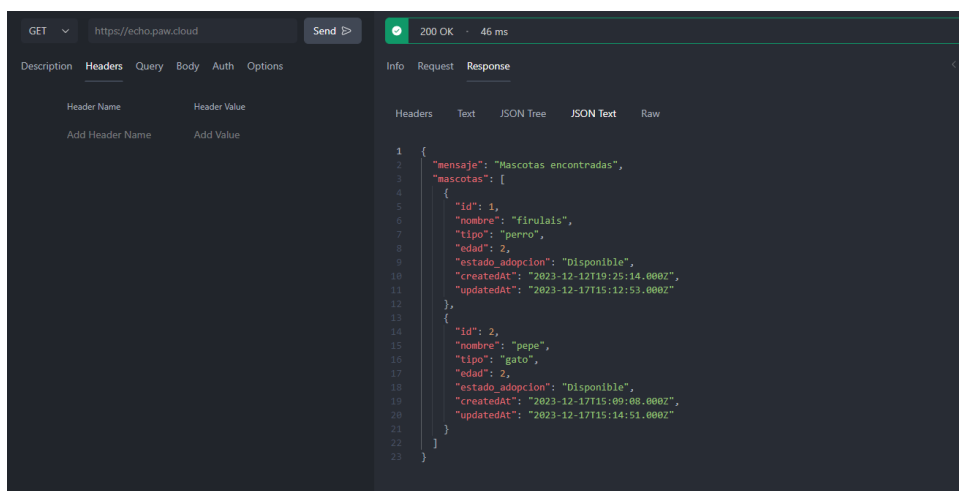
Crear



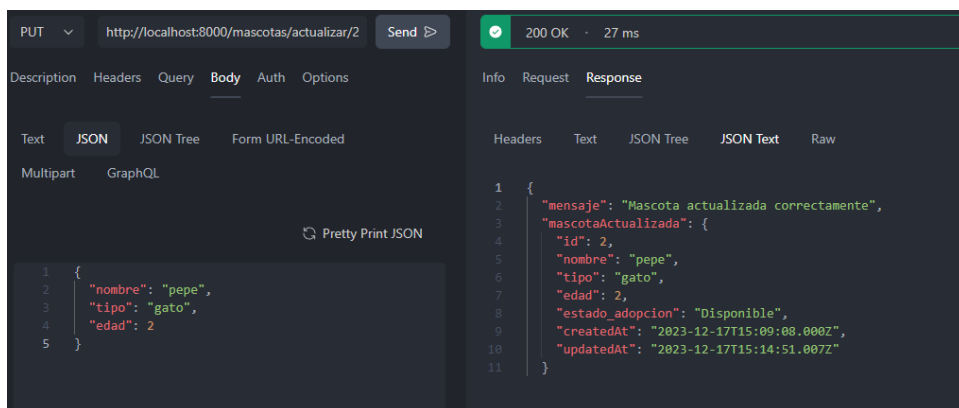
Buscar ID



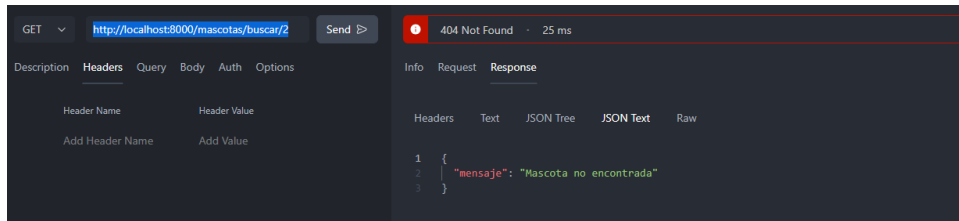
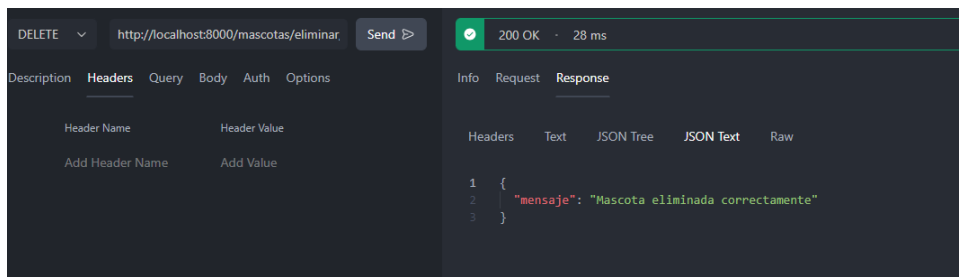
Buscar



Actualizar

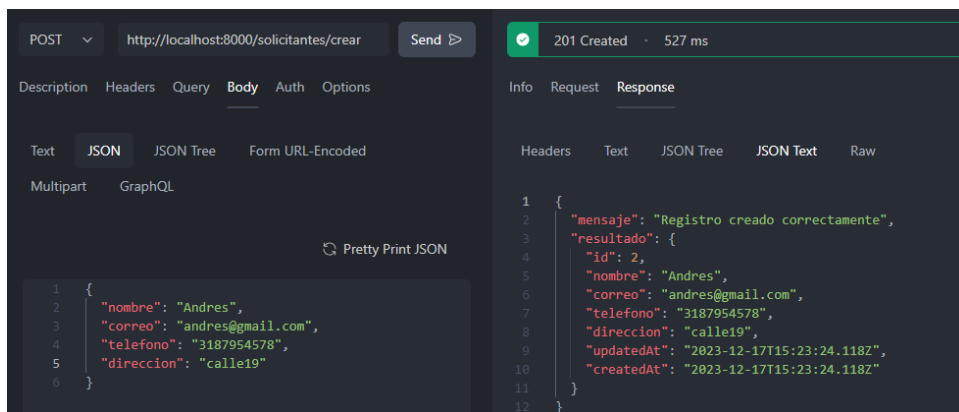


Eliminar

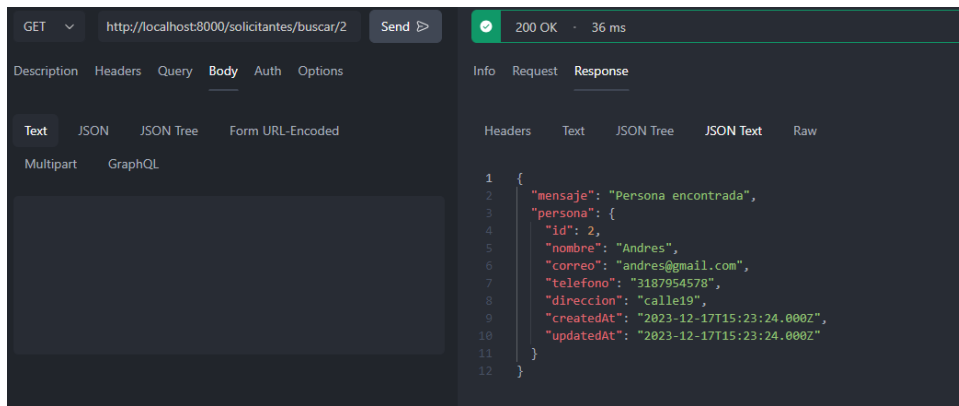


En solicitantes

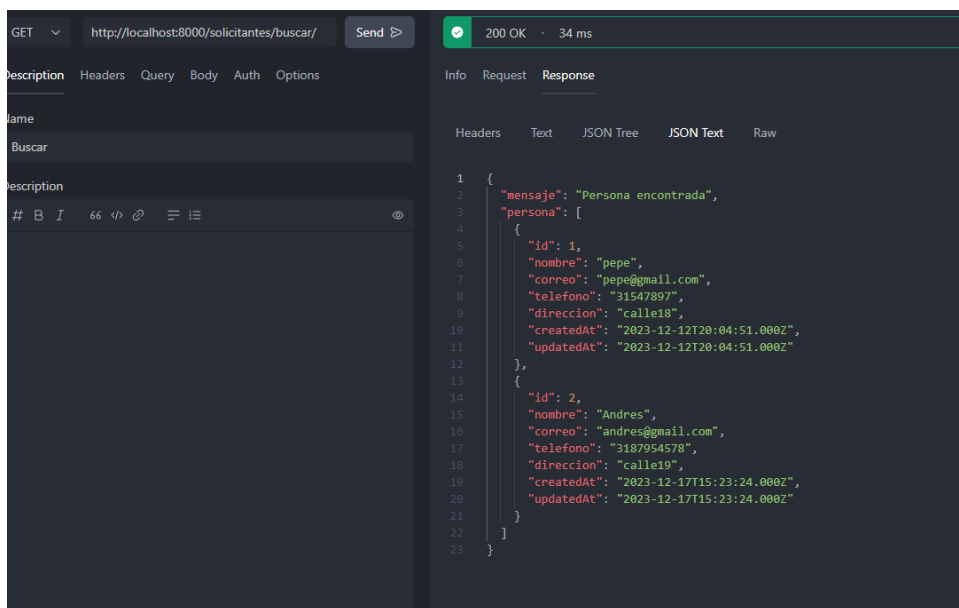
Crear



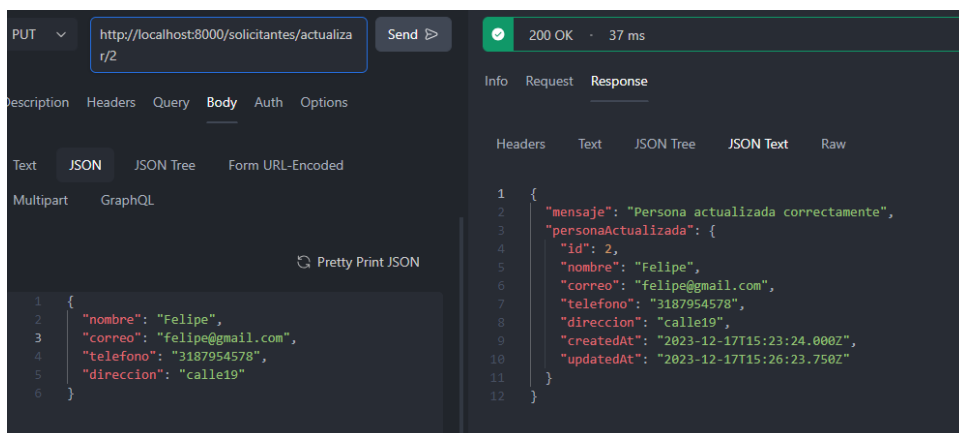
Buscar ID



Buscar



Actualizar



Eliminar

DELETE

http://localhost:8000/solicitudes/eliminar

Send

Description

Headers

Query

Body

Auth

Options

Name

Eliminar

Description

B I 66 </> @ ≡ ≡

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

1 {

2 | "mensaje": "persona eliminada correctamente"

3 }

GET

http://localhost:8000/solicitudes/buscar/2

Send

Description

Headers

Query

Body

Auth

Options

Text

JSON

JSON Tree

Form URL-Encoded

Multipart

GraphQL

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

1 {

2 | "mensaje": "Persona no encontrada"

3 }

En solicitud

Crear

POST

http://localhost:8000/solicitudes/crear

Send

Description

Headers

Query

Body

Auth

Options

Text

JSON

JSON Tree

Form URL-Encoded

Multipart

GraphQL

1 {

2 | "mascota_id": 1,

3 | "solicitante_id": 3,

4 | "fecha_solicitud": "2023-12-13",

5 | "estado": "Pendiente"

6 }

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

1 {

2 | "mensaje": "Solicitud creada correctamente",

3 | "solicitud": {

4 | | "id": 2,

5 | | "mascota_id": 1,

6 | | "solicitante_id": 3,

7 | | "fecha_solicitud": "2023-12-17T15:33:14.047Z",

8 | | "estado": "Pendiente",

9 | | "updatedAt": "2023-12-17T15:33:14.047Z",

10 | | "createdAt": "2023-12-17T15:33:14.047Z"

11 | }

12 }

Buscar ID

GET

http://localhost:8000/solicitudes/buscar/2

Send

Description

Headers

Query

Body

Auth

Options

Text

JSON

JSON Tree

Form URL-Encoded

Multipart

GraphQL

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

1 {

2 | "mensaje": "Solicitud encontrada",

3 | "solicitud": {

4 | | "id": 2,

5 | | "mascota_id": 1,

6 | | "solicitante_id": 3,

7 | | "fecha_solicitud": "2023-12-17",

8 | | "estado": "Pendiente",

9 | | "createdAt": "2023-12-17T15:33:14.000Z",

10 | | "updatedAt": "2023-12-17T15:33:14.000Z"

11 | }

12 }

Buscar

GET

http://localhost:8000/solicitudes/buscar/

Send

200 OK

29 ms

escripion

Headers

Query

Body

Auth

Options

ame

Buscar

escripion

#

B

I

66

<

>

@

≡

≡

@

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

```
1 {
2   "mensaje": "Solicitud encontrada",
3   "solicitud": [
4     {
5       "id": 1,
6       "mascota_id": 1,
7       "solicitante_id": 1,
8       "fecha_solicitud": "2023-12-13",
9       "estado": "Pendiente",
10      "createdAt": "2023-12-13T21:23:18.000Z",
11      "updatedAt": "2023-12-13T21:23:18.000Z"
12    },
13    {
14      "id": 2,
15      "mascota_id": 1,
16      "solicitante_id": 3,
17      "fecha_solicitud": "2023-12-17",
18      "estado": "Pendiente",
19      "createdAt": "2023-12-17T15:33:14.000Z",
20      "updatedAt": "2023-12-17T15:33:14.000Z"
21    }
22  ]
23 }
```

Actualizar

PUT

http://localhost:8000/solicitudes/actualizar/

Send

200 OK

66 ms

Description

Headers

Query

Body

Auth

Options

Text

JSON

JSON Tree

Form URL-Encoded

Multipart

GraphQL

Pretty Print JSON

```
1 {
2   "estado": "Aprobada"
3 }
```

Info

Request

Response

Headers

Text

JSON Tree

JSON Text

Raw

```
1 {
2   "mensaje": "Solicitud actualizada correctamente",
3   "solicitud": {
4     "id": 2,
5     "mascota_id": 1,
6     "solicitante_id": 3,
7     "fecha_solicitud": "2023-12-17",
8     "estado": "Aprobada",
9     "createdAt": "2023-12-17T15:33:14.000Z",
10    "updatedAt": "2023-12-17T15:48:28.342Z",
11    "mascota": {
12      "id": 1,
13      "nombre": "firulais",
14      "tipo": "perro",
15      "edad": 2,
16      "estado_adopcion": "Disponible",
17      "createdAt": "2023-12-12T19:25:14.000Z",
18      "updatedAt": "2023-12-17T15:12:53.000Z"
19    },
20    "solicitante": {
21      "id": 3,
22      "nombre": "Andres",
23      "correo": "andres@gmail.com",
24      "telefono": "3187954578",
25      "direccion": "calle19",
26      "createdAt": "2023-12-17T15:31:05.000Z",
27      "updatedAt": "2023-12-17T15:31:05.000Z"
28    }
29  }
30 }
```

Eliminar

DELETE Send ➤

Description Headers Query Body Auth Options

Name

Eliminar

Description

B I *66* ↵ 🔗 ☰ ☷

Info Request **Response**

Headers Text JSON Tree **JSON Text** Raw

```
1 {
2   "mensaje": "Solicitud eliminada correctamente"
3 }
```

GET Send ➤

Description Headers Query **Body** Auth Options

Text JSON JSON Tree Form URL-Encoded

Multipart GraphQL

Info Request **Response**

Headers Text JSON Tree **JSON Text** Raw

```
1 {
2   "mensaje": "Solicitud no encontrada"
3 }
```