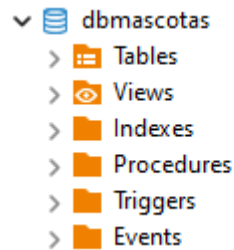


TALLER N°2 BACKEND

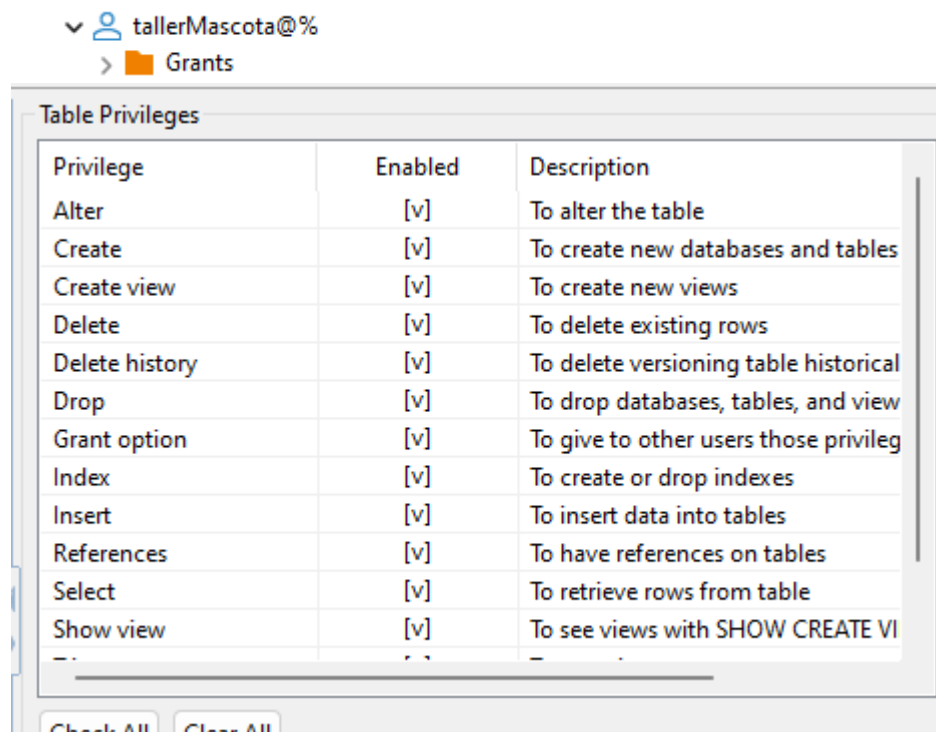
**PRESENTADO POR:
JOSE GABRIEL NASTUL NARVAEZ**

**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
```

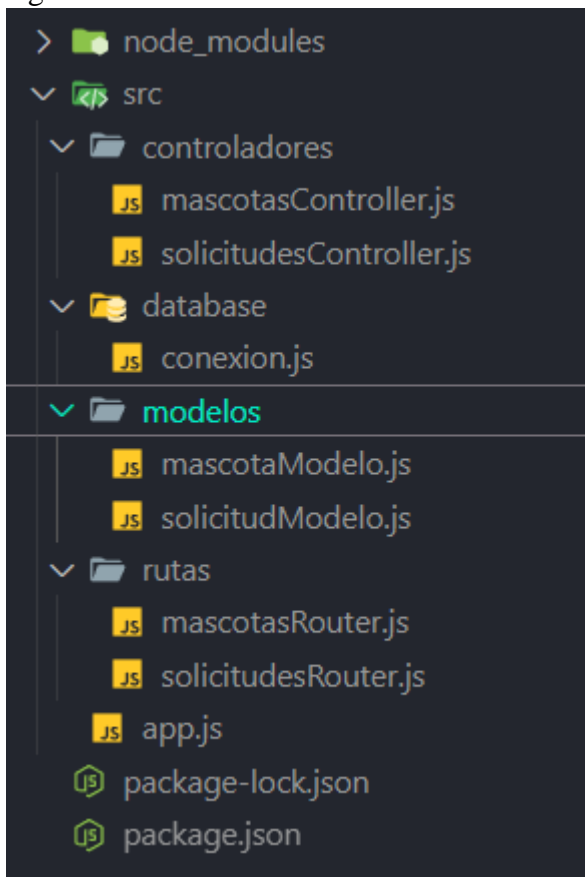
5. Después de iniciar esta instrucción nos aparece esta pantalla y realizamos algunos cambios, como lo son el type a module y el main para que inicie en app.js, y también el nodemon para que se ejecuten los cambios.

```
{
  "name": "tallerbackend",
  "version": "1.0.0",
  "type": "module",
  "main": "app.js",
  "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

```
1 import { Sequelize, DataTypes } from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const Mascotas = db.define("mascotas", {
5   id: {
6     type: DataTypes.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11  nombre: {
12    type: DataTypes.STRING,
13    allowNull: false
14  },
15  descripcion: {
16    type: DataTypes.TEXT,
17    allowNull: true
18  },
19  edad: {
20    type: DataTypes.INTEGER,
21    allowNull: false
22  },
23  genero: {
24    type: DataTypes.ENUM('Macho', 'Hembra'),
25    allowNull: false
26  },
27  imagen: {
28    type: DataTypes.STRING, // Ruta de la imagen o URL
29    allowNull: true
30  }
31 }, {
32   tableName: 'mascotas', // Nombre de la tabla en la base de datos
33   timestamps: false
34 });
35
36
37
38 export {Mascotas}
39
40
```



```
1 import { Sequelize, DataTypes } from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const Solicitudes = db.define("solicitudes", {
5   id: {
6     type: DataTypes.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11  nombreAdoptante: {
12    type: DataTypes.STRING,
13    allowNull: false
14  },
15  fechaSolicitud: {
16    type: DataTypes.DATE,
17    allowNull: false,
18    defaultValue: Sequelize.NOW
19  },
20  estado: {
21    type: DataTypes.ENUM('Pendiente', 'Aprobada', 'Rechazada'),
22    allowNull: false,
23    defaultValue: 'Pendiente'
24  }
25 }, {
26   tableName: 'solicitudes',
27   timestamps: false
28 });
29
30 export {Solicitudes}
```

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

```

1  import { Mascotas } from "../modelos/mascotaModelo.js";
2
3
4  //crear recurso mascota
5  const crear = (req, res) => {
6    if(!req.body.nombre){
7      res.status(400).send({
8        mensaje: "El nombre de la mascota es requerido"
9      })
10     return;
11   }
12
13   const dataset = {
14     nombre: req.body.nombre,
15     descripcion: req.body.descripcion,
16     edad: req.body.edad,
17     genero: req.body.genero,
18     peso: req.body.peso,
19     imagen: req.body.imagen
20   }
21
22   //usar sequelize bsae de datos
23   Mascotas.create(dataset).then((resultado) => {
24     res.status(200).json({
25       mensaje: "Registro de mascota Exitoso"
26     });
27   }).catch((err) => {
28     res.status(500).json({
29       mensaje: "Error al registrar la mascota ::: ${err}"
30     });
31   });
32 }
33
34 const editar = (req, res) => {
35   const id = req.params.id;
36
37   Mascotas.update(req.body, {
38     where: { id: id }
39   }).then((resultado) => {
40     if (resultado == 1) {
41       res.status(200).json({
42         mensaje: "Mascota actualizada correctamente"
43       });
44     } else {
45       res.status(404).json({
46         mensaje: "No se pudo actualizar la mascota con id=${id}. Puede que la mascota no exista o que los datos sean iguales a los actuales."
47       });
48     }
49   }).catch((err) => {
50     res.status(500).json({
51       mensaje: "Error al actualizar la mascota con id=${id}: ${err}"
52     });
53   });
54 }
55
56 const buscarTodos = (req, res) => {
57   Mascotas.findAll()
58   .then((mascotas) => {
59     res.status(200).json(mascotas);
60   }).catch((err) => {
61     res.status(500).json({
62       mensaje: "Error al recuperar las mascotas: ${err}"
63     });
64   });
65 }
66
67 const buscarPorId = (req, res) => {
68   const id = req.params.id;
69
70   Mascotas.findById(id)
71   .then((mascota) => {
72     if (mascota) {
73       res.status(200).json(mascota);
74     } else {
75       res.status(404).json({
76         mensaje: "No se encontró ninguna mascota con id=${id}"
77       });
78     }
79   }).catch((err) => {
80     res.status(500).json({
81       mensaje: "Error al buscar la mascota con id=${id}: ${err}"
82     });
83   });
84 }
85
86 const eliminar = (req, res) => {
87   const id = req.params.id;
88
89   Mascotas.destroy({
90     where: { id: id }
91   }).then((resultado) => {
92     if (resultado == 1) {
93       res.status(200).json({
94         mensaje: "Mascota eliminada correctamente"
95       });
96     } else {
97       res.status(404).json({
98         mensaje: "No se pudo eliminar la mascota con id=${id}. Puede que la mascota no exista."
99       });
100     }
101   }).catch((err) => {
102     res.status(500).json({
103       mensaje: "Error al eliminar la mascota con id=${id}: ${err}"
104     });
105   });
106 }
107
108
109
110 export { crear, editar, buscarTodos, buscarPorId, eliminar }

```

```

1 import { Solicitudes } from "../modelos/solicitudModelo.js";
2
3 const crearSolicitud = (req, res) => {
4   if (!req.body.nombreAdoptante) {
5     res.status(400).send({
6       mensaje: "El nombre del adoptante es requerido"
7     });
8     return;
9   }
10
11   const dataset = {
12     nombreAdoptante: req.body.nombreAdoptante,
13     fechaSolicitud: req.body.fechaSolicitud,
14     estado: req.body.estado
15   };
16
17   Solicitudes.create(dataset)
18     .then((resultado) => {
19       res.status(200).json({
20         mensaje: "Registro de solicitud exitoso",
21         solicitud: resultado
22       });
23     })
24     .catch((err) => {
25       res.status(500).json({
26         mensaje: "Error al registrar la solicitud: ${err}"
27       });
28     });
29 };
30
31 const editarSolicitud = (req, res) => {
32   const id = req.params.id;
33
34   Solicitudes.update(req.body, {
35     where: { id: id }
36   })
37     .then((resultado) => {
38       if (resultado == 1) {
39         res.status(200).json({
40           mensaje: "Solicitud actualizada correctamente"
41         });
42       } else {
43         res.status(404).json({
44           mensaje: "No se pudo actualizar la solicitud con id=${id}. Puede que la solicitud no exista o que los datos sean iguales."
45         });
46       }
47     })
48     .catch((err) => {
49       res.status(500).json({
50         mensaje: "Error al actualizar la solicitud con id=${id}: ${err}"
51       });
52     });
53 };
54
55 const buscarSolicitudes = (req, res) => {
56   Solicitudes.findAll()
57     .then((solicitudes) => {
58       res.status(200).json(solicitudes);
59     })
60     .catch((err) => {
61       res.status(500).json({
62         mensaje: "Error al recuperar las solicitudes: ${err}"
63       });
64     });
65 };
66
67 const buscarSolicitudPorId = (req, res) => {
68   const id = req.params.id;
69
70   Solicitudes.findById(id)
71     .then((solicitud) => {
72       if (solicitud) {
73         res.status(200).json(solicitud);
74       } else {
75         res.status(404).json({
76           mensaje: "No se encontró ninguna solicitud con id=${id}"
77         });
78       }
79     })
80     .catch((err) => {
81       res.status(500).json({
82         mensaje: "Error al buscar la solicitud con id=${id}: ${err}"
83       });
84     });
85 };
86
87 const eliminarSolicitud = (req, res) => {
88   const id = req.params.id;
89
90   Solicitudes.destroy({
91     where: { id: id }
92   })
93     .then((resultado) => {
94       if (resultado == 1) {
95         res.status(200).json({
96           mensaje: "Solicitud eliminada correctamente"
97         });
98       } else {
99         res.status(404).json({
100           mensaje: "No se pudo eliminar la solicitud con id=${id}. Puede que la solicitud no exista."
101         });
102       }
103     })
104     .catch((err) => {
105       res.status(500).json({
106         mensaje: "Error al eliminar la solicitud con id=${id}: ${err}"
107       });
108     });
109 };
110
111
112
113 export { crearSolicitud, editarSolicitud, buscarSolicitudes, buscarSolicitudPorId, eliminarSolicitud };

```


10. Después de ello empezamos a maquetar las rutas

```
1 import express from "express";
2 import { buscarPorId, buscarTodos, crear, editar, eliminar } from "../controladores/mascotasController.js";
3
4 const routerMascotas = express.Router();
5
6 routerMascotas.get('/', (req, res) => {
7   res.send('Bienvenido al Sitio Principal.');
```

```
8 });
9
10 routerMascotas.post('/crear', (req, res) => {
11   crear(req,res);
12 });
13
14 routerMascotas.put('/editar/:id', (req, res)=>{
15   editar(req,res);
16 });
17
18 routerMascotas.get('/buscarTodos', (req, res)=>{
19   buscarTodos(req,res);
20 });
21
22 routerMascotas.get('/buscarPorId/:id', (req, res)=>{
23   buscarPorId(req,res);
24 });
25
26 routerMascotas.delete('/eliminar/:id', (req, res)=>{
27   eliminar(req,res);
28 });
29
30 export {routerMascotas}
```

```
1 import express from "express";
2 import { buscarSolicitudPorId, buscarSolicitudes, crearSolicitud, editarSolicitud, eliminarSolicitud } from "../controladores/solicitudesController.js";
3
4 const routerSolicitudes = express.Router();
5
6 routerSolicitudes.get('/home', (req, res) => {
7   res.send('Bienvenido al Sitio Principal.');
```

```
8 });
9
10 routerSolicitudes.post('/crearSolicitud', (req, res)=>{
11   crearSolicitud(req,res);
12 });
13
14 routerSolicitudes.put('/editarSolicitud/:id', (req, res)=>{
15   editarSolicitud(req,res);
16 });
17
18 routerSolicitudes.get('/buscarSolicitud', (req, res)=>{
19   buscarSolicitudes(req,res);
20 });
21
22 routerSolicitudes.get('/buscarSolicitudPorId/:id', (req, res)=>{
23   buscarSolicitudPorId(req,res);
24 })
25
26 routerSolicitudes.delete('/eliminarSolicitud/:id', (req, res)=>{
27   eliminarSolicitud(req,res);
28 })
29 export {routerSolicitudes}
```

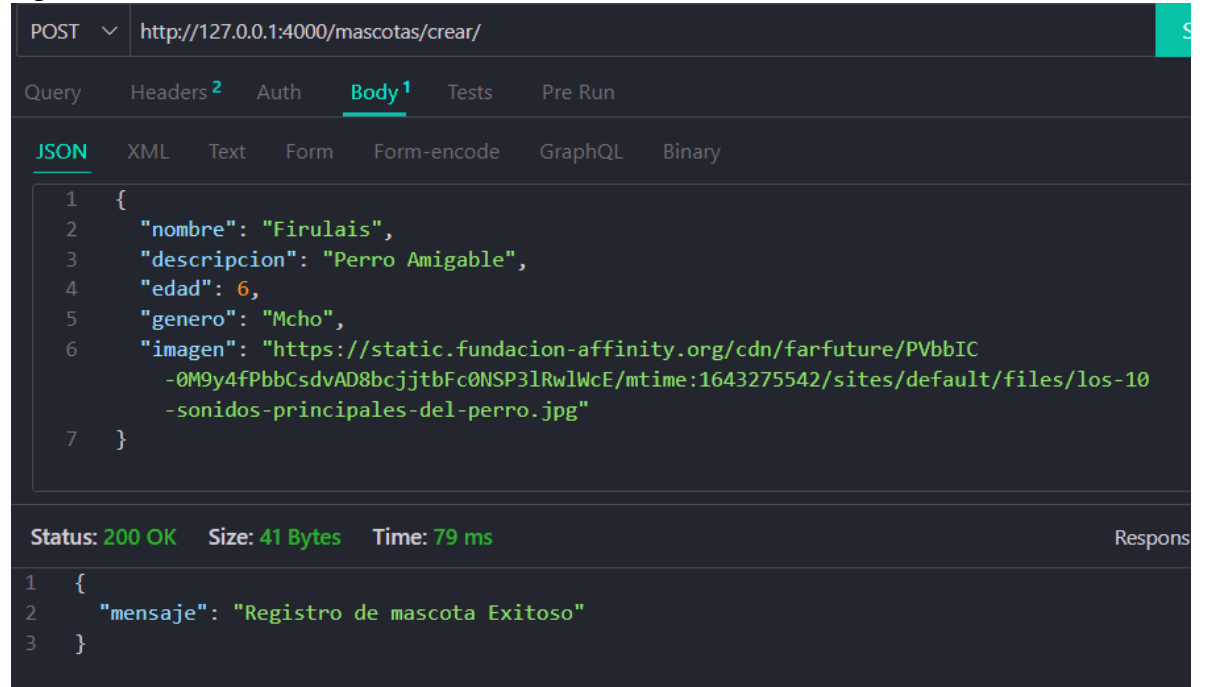
11. Por último, todas las configuraciones necesarias en app



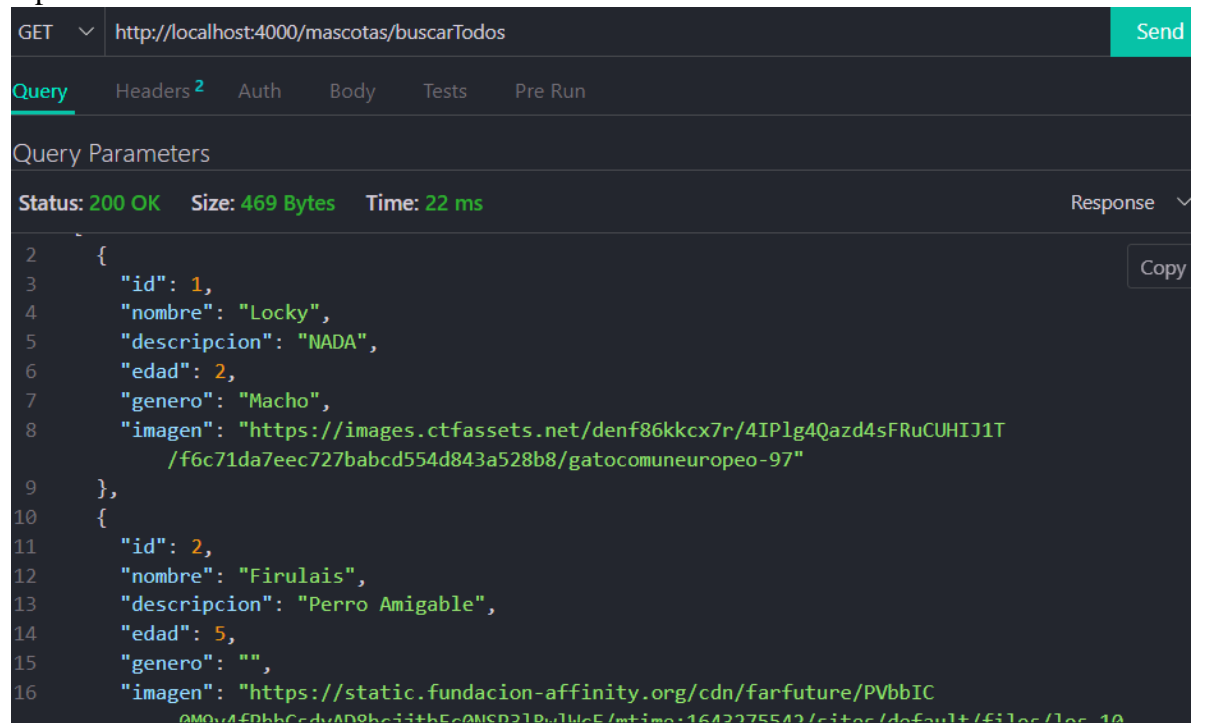
```
1  import express from "express";
2  import { routerMascotas } from "../rutas/mascotasRouter.js";
3  import { routerSolicitudes } from "../rutas/solicitudesRouter.js";
4  import { db } from "../database/conexion.js";
5  import cors from 'cors';
6
7  //crear instancia express
8  const app = express();
9
10
11  //json
12  app.use(express.json());
13
14  app.use(cors());
15
16
17  //verificar conexion base de datos
18  db.authenticate().then (()=>{
19      console.log(`conexion a la base de datos correcta`)
20  }).catch(err=>{
21      console.log(`conexion a la base de datos incorrecta ${err}`)
22  });
23
24  //llamando rutas de mascotas
25  app.use("/mascotas", routerMascotas)
26
27  //llamado rutas de solicitudes
28  app.use("/solicitudes", routerSolicitudes)
29
30  //puerto del servidor
31  const PORT = 4000;
32
33  db.sync().then(()=>{
34      //Abri servicio e iniciar el Servidor
35      app.listen(PORT,()=>{
36          console.log(`Servidor Inicializado en el puerto ${PORT}`);
37      })
38  }).catch(err=>{
39      console.log(`Error al sincronizar la base de datos ${err}`);
40  });
41
42
```

12. Realizar verificación de las diferentes operaciones a través de un cliente gráfico (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

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:4000/mascotas/editar/2`. The 'Body' tab is selected, showing a JSON payload: `{ "nombre": "Firulais", "edad": 5 }`. The status bar indicates a successful response: `Status: 200 OK`, `Size: 47 Bytes`, and `Time: 21 ms`. The response body is a JSON object: `{ "mensaje": "Mascota actualizada correctamente" }`.

```
PUT http://127.0.0.1:4000/mascotas/editar/2
```

Query Headers² Auth **Body¹** Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

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

Status: 200 OK Size: 47 Bytes Time: 21 ms

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

4. Operación buscar solo uno

The screenshot shows a REST client interface with a GET request to `http://localhost:4000/mascotas/buscarPorId/2`. The 'Query' tab is selected, showing the request parameters. The status bar indicates a successful response: `Status: 200 OK`, `Size: 265 Bytes`, and `Time: 22 ms`. The response body is a JSON object containing pet details: `{ "id": 2, "nombre": "Firulais", "descripcion": "Perro Amigable", "edad": 5, "genero": "", "imagen": "https://static.fundacion-affinity.org/cdn/farfuture/PVbbIC-0M9y4fPbbCsdrvAD8bcjtbFc0NSP3lRwIWcE/mtime:1643275542/sites/default/files/los-10-sonidos-principales-del-perro.jpg" }`.

```
GET http://localhost:4000/mascotas/buscarPorId/2 Send
```

Query Headers² Auth Body Tests Pre Run

Query Parameters

Status: 200 OK Size: 265 Bytes Time: 22 ms Response ▾

```
1 {
2   "id": 2,
3   "nombre": "Firulais",
4   "descripcion": "Perro Amigable",
5   "edad": 5,
6   "genero": "",
7   "imagen": "https://static.fundacion-affinity.org/cdn/farfuture/PVbbIC-0M9y4fPbbCsdrvAD8bcjtbFc0NSP3lRwIWcE/mtime:1643275542/sites/default/files/los-10-sonidos-principales-del-perro.jpg"
8 }
```

5. Operación eliminar mascota

```
POST http://127.0.0.1:4000/solicitudes/crearSolicitud/
Body
1 {
2   "nombreAdoptante": "Juan Pérez",
3   "fechaSolicitud": "2024-10-01",
4   "estado": "Pendiente"
5 }

Status: 200 OK Size: 161 Bytes Time: 25 ms
Response
1 {
2   "mensaje": "Registro de solicitud exitoso",
3   "solicitud": {
4     "id": 5,
5     "nombreAdoptante": "Juan Pérez",
6     "fechaSolicitud": "2024-10-01T00:00:00.000Z",
7     "estado": "Pendiente"
8   }
9 }
```

6. Operación crear Solicitud

```
DELETE http://127.0.0.1:4000/mascotas/eliminar/2
Body
1

Status: 200 OK Size: 45 Bytes Time: 24 ms
Response
1 {
2   "mensaje": "Mascota eliminada correctamente"
3 }
```

7. Operación Editar Solicitud

PUT ▼ http://127.0.0.1:4000/solicitudes/editarSolicitud/5

Query Headers ² Auth Body ¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

```
1 {
2   "estado": "Aprobada"
3 }
4 }
```

Status: 200 OK Size: 49 Bytes Time: 28 ms

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

8. Operación buscar solicitud por id

GET ▼ http://localhost:4000/solicitudes/buscarSolicitudPorId/5

Query Headers ² Auth Body Tests Pre Run

Query Parameters

Status: 200 OK Size: 104 Bytes Time: 21 ms

```
1 {
2   "id": 5,
3   "nombreAdoptante": "Juan Pérez",
4   "fechaSolicitud": "2024-10-01T00:00:00.000Z",
5   "estado": "Aprobada"
6 }
```

9. Operación para buscar todas las solicitudes

GET ⌵ http://localhost:4000/solicitudes/buscarSolicitud

Query Headers² Auth Body Tests Pre Run

Query Parameters

Status: 200 OK Size: 206 Bytes Time: 17 ms

```
1  [  
2    {  
3      "id": 4,  
4      "nombreAdoptante": "Maria",  
5      "fechaSolicitud": "2024-10-06T00:00:00.000Z",  
6      "estado": "Pendiente"  
7    },  
8    {  
9      "id": 5,  
10     "nombreAdoptante": "Juan Pérez",  
11     "fechaSolicitud": "2024-10-01T00:00:00.000Z",  
12     "estado": "Aprobada"  
13   }  
14 ]
```

10. Operación para eliminar solicitud

DELETE ⌵ http://127.0.0.1:4000/solicitudes/eliminarSolicitud/5

Query Headers² Auth Body Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

1

Status: 200 OK Size: 47 Bytes Time: 24 ms

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