

Taller BackEnd

PRESENTADO POR:

VARGAS MARTINEZ DANIEL SANTIAGO

PRESENTADO A:

Vicente Aux Revelo

UNIVERSIDAD DE NARIÑO

INGENIERIA DE SISTEMAS

DIPLOMADO DE ACTUALIZACIÓN EN NUEVAS TECNOLOGÍAS PARA EL

DESARROLLO DE SOFTWARE

2024

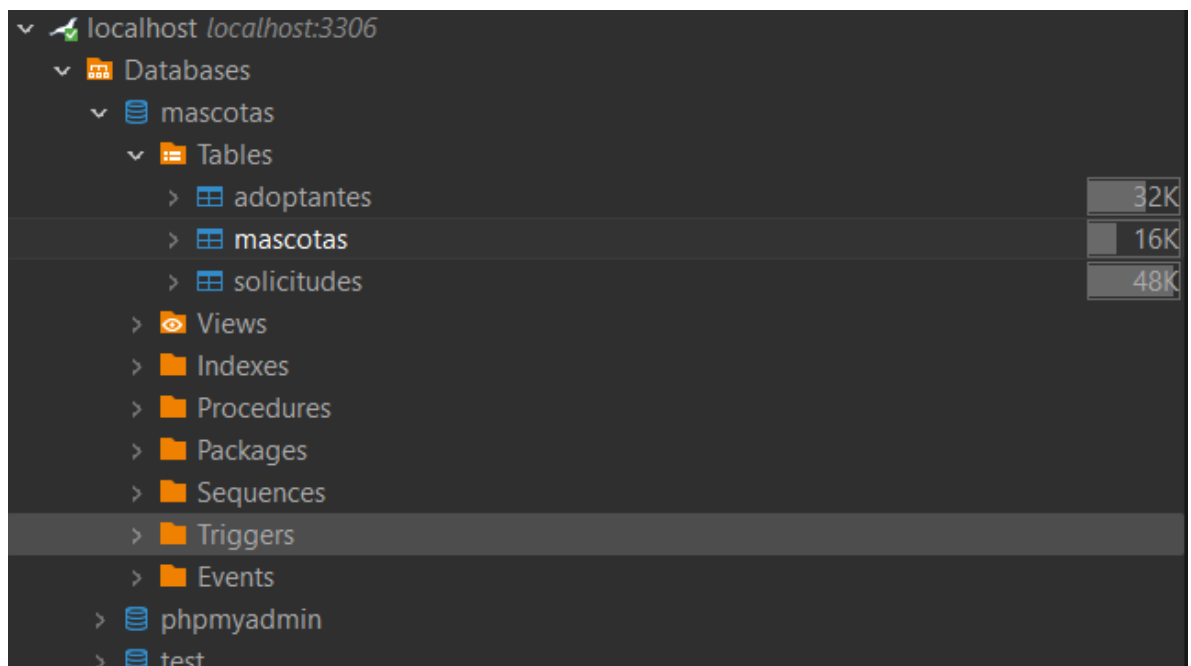
1. Creación de la Base de Datos para la Empresa de Adopción de Mascotas

En un entorno de desarrollo utilizando DBeaver, se creó una base de datos en MariaDB para gestionar la operación de una empresa dedicada a la adopción de mascotas, con el nombre "mascotas". Esta base de datos está diseñada para facilitar el manejo de las mascotas disponibles para adopción y las solicitudes enviadas por los posibles adoptantes. La estructura de las tablas se desarrollará posteriormente en el proyecto backend, y para ello, el servicio se inicializó previamente con XAMPP. A continuación, se muestra la evidencia de la creación de la base de datos y el esquema relacional que guiará su implementación.

- Conexión en Xampp



- Conexión de DBeaver a Xampp



2. Configuración del Proyecto Node.js:

Para comenzar el desarrollo del proyecto en Node.js, se ejecutó el comando `npm init -y` en el directorio "TallerbackEnd". Este comando permitió generar automáticamente un archivo `package.json`, el cual contiene la configuración básica del proyecto, incluyendo detalles como el nombre del proyecto, la versión, descripción, y los scripts iniciales, facilitando así la gestión de dependencias y otros ajustes necesarios para el desarrollo.

```
PS C:\Users\So\Documents\Diplomado\backend> npm init -y
Wrote to C:\Users\So\Documents\Diplomado\backend\package.json:

{
  "name": "backend",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Posteriormente, se ejecutó el comando `npm install express`

```
PS C:\Users\So\Documents\Diplomado\backend> npm install express

added 65 packages, and audited 66 packages in 4s

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

2 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- mysql2

```
PS C:\Users\So\Documents\Diplomado\backend> npm install mysql2

added 12 packages, and audited 78 packages in 2s

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

2 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- sequelize

```
PS C:\Users\So\Documents\Diplomado\backend> npm install sequelize

added 21 packages, and audited 99 packages in 5s

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

2 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

- Nodemon

```
PS C:\Users\So\Documents\Diplomado\backend> npm install nodemon -D

added 28 packages, and audited 127 packages in 3s

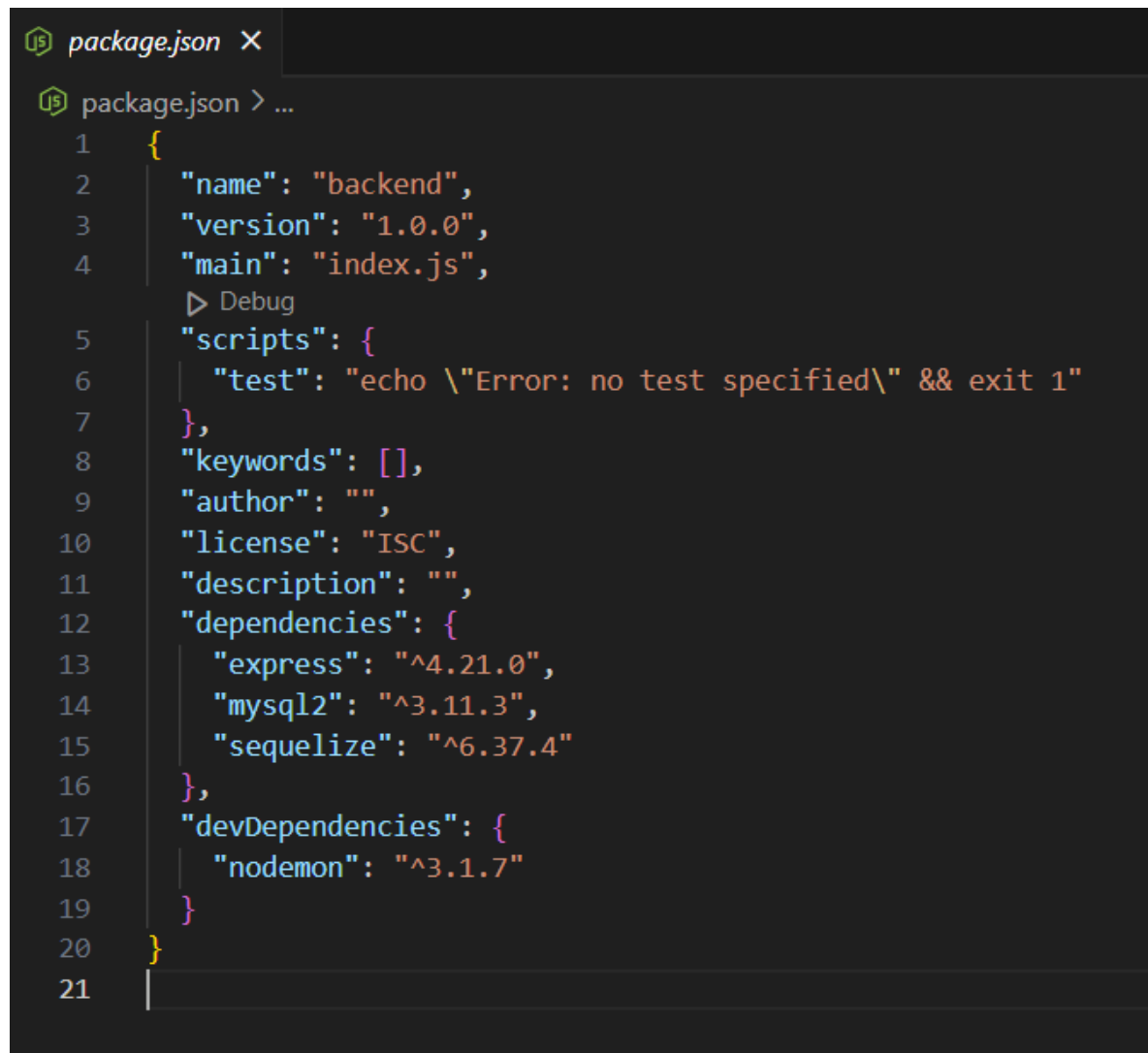
19 packages are looking for funding
  run `npm fund` for details

2 low severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Se puede observar que se instalaron correctamente y se muestra a continuación



```
package.json X
package.json > ...
1  {
2    "name": "backend",
3    "version": "1.0.0",
4    "main": "index.js",
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1"
7    },
8    "keywords": [],
9    "author": "",
10   "license": "ISC",
11   "description": "",
12   "dependencies": {
13     "express": "^4.21.0",
14     "mysql2": "^3.11.3",
15     "sequelize": "^6.37.4"
16   },
17   "devDependencies": {
18     "nodemon": "^3.1.7"
19   }
20 }
21
```

App.js verificar conexión a base de datos

```
BackEnd > src > js app.js > then() callback
1  import express from 'express';
2  import cors from 'cors';
3  import { routerMascotas } from './rutas/mascotasRouter.js';
4  import { routerAdoptantes } from './rutas/adoptantesRouter.js';
5  import { routerSolicitud } from './rutas/solicitudRouter.js';
6  import { db } from './database/conexion.js';
7
8  const app = express();
9  const PORT = 4000;
10
11 app.use(cors());
12 app.use(express.json());
13
14 // Verificar conexión a la base de datos
15 db.authenticate()
16   .then(() => {
17     console.log('Conexión a Base de datos correcta');
18   })
19   .catch(err => {
20     console.log(`Conexión a Base de datos incorrecta: ${err}`);
21   });
22
23 db.sync({ force: false })
24   .then(() => {
25     console.log('Sincronización con la base de datos correcta');
26   })
27   .catch(err => {
28     console.log(`Error al sincronizar base de datos: ${err}`);
29   });
30
31 app.get('/', (req, res) => {
32   res.send('Hola Sitio Principal');
33 });
```

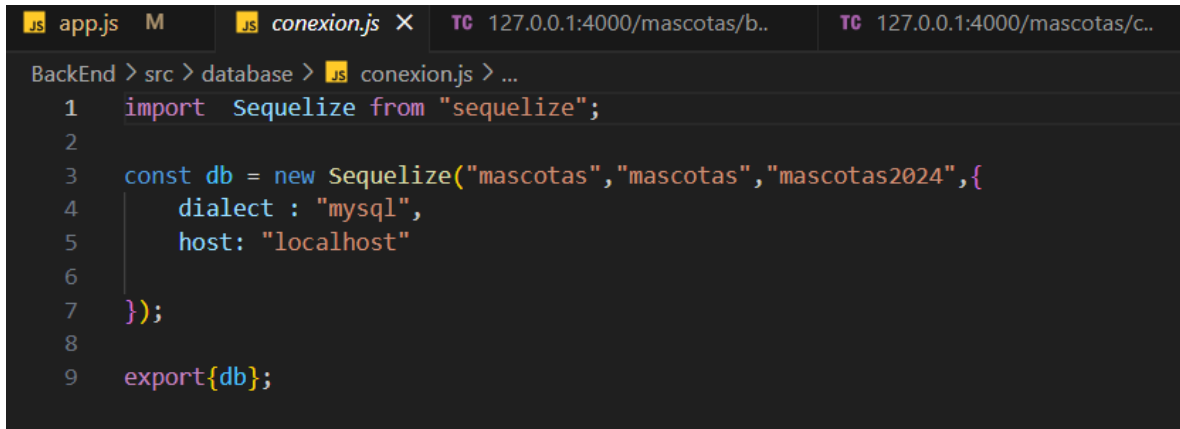
Se creó dentro de src la carpeta controlador los archivos .js de adoptantesController, mascotasController y solicitudController

```
✓ controladores
  js adoptantesController.js
  js mascotasController.js
  js solicitudController.js
```

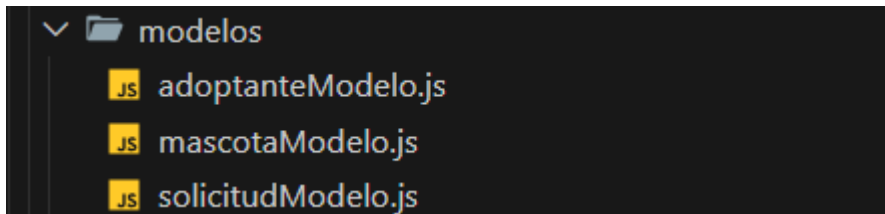
También se creó la carpeta database la cual contiene el archivo conexión.js el cual es donde se conecta a la base de datos



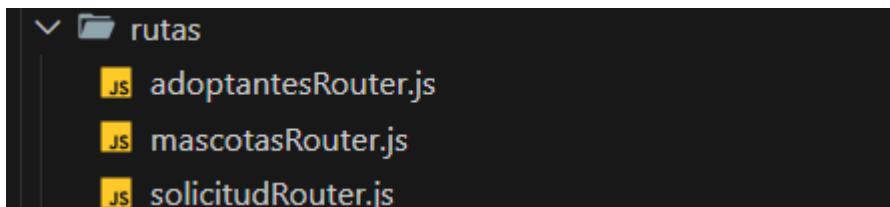
Manera en como se hizo la conexión a la base de datos



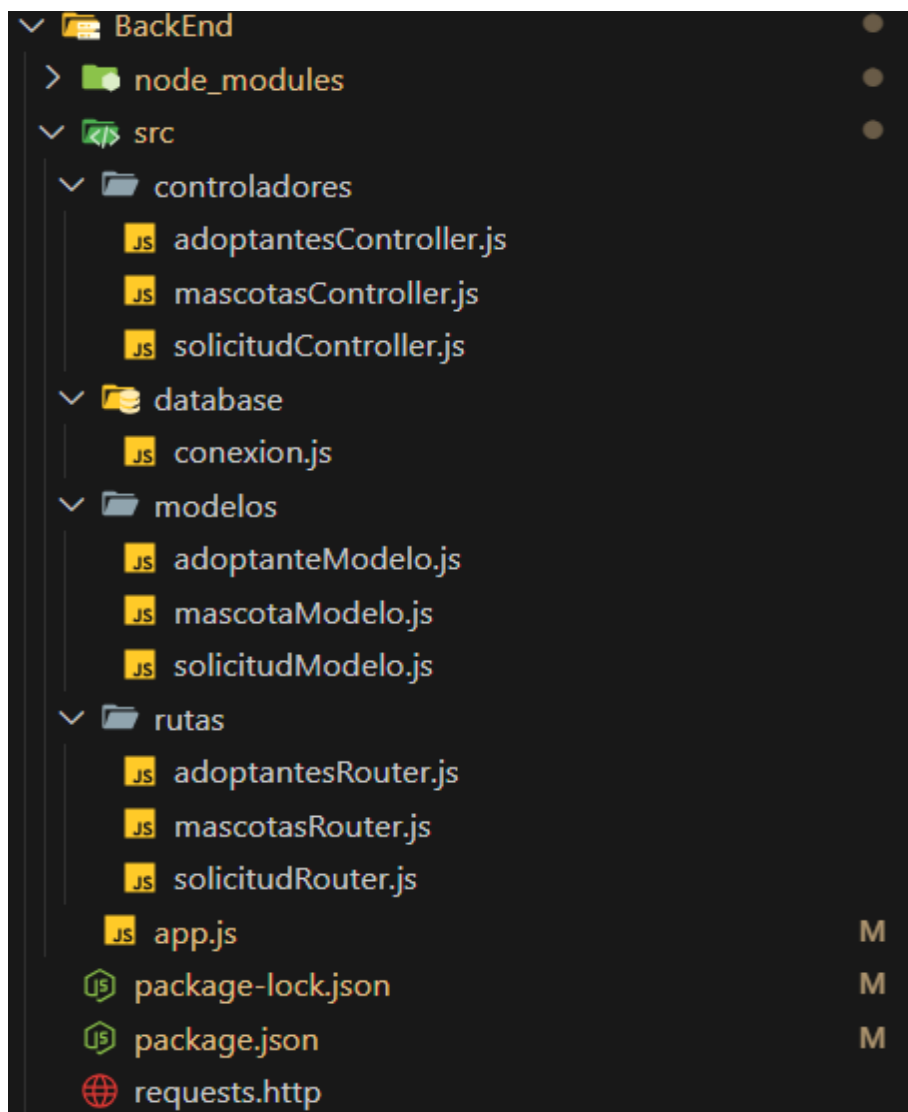
De igual manera se creó la carpeta modelos la cual contiene la estructura del modelo para adoptante, mascota y solicitud



Y por último se creó la carpeta rutas en la que están almacenadas los métodos para hacer consultas a la base de datos tanto como para adoptantes, mascotas y solicitud



Esta es la estructura completa del proyecto para el backEnd



3. Verificación de las diferentes operaciones a través de Thunder Client

Mascotas

- Método crear

The screenshot shows a Thunder Client interface with a POST request to `http://127.0.0.1:4000/mascotas/crear`. The status is 200 OK, size is 338 Bytes, and time is 12 ms. The request body is a JSON object with the following fields: `mascota_nombre` (Firulais), `mascota_edad` (5), `mascota_tipo` (perro), `mascota_raza` (labrador), `mascota_estado` (disponible), and `mascota_imagen` (http://example.com/imagen_de_firulais.jpg). The response is a JSON object with the following fields: `mensaje` (Registro de Mascota Creado con Exito), `datos` (an object with the same fields as the request), `updatedAt` (2024-10-06T18:22:19.450Z), and `createdAt` (2024-10-06T18:22:19.450Z).

Se comprueba que se creó el registro, en este caso el ID es el #7

	mascota_id	mascota_nombre	mascota_edad	mascota_tipo	mascota_raza	mascota_estado	mascota_imagen
1	3	pachos	9	Perro	Labrador	disponible	https://ichef.bbci.co.uk/
2	4	Pacho	3	Gato	FrencPoodle	adoptado	https://unamglobal.una
3	5	masha	7	perro	pinsher	disponible	https://as01.epimg.net/
4	6	bruno	10	gato	persa	adoptado	[NULL]
5	7	Firulais	5	perro	labrador	disponible	http://example.com/im

Verificación buscar todas las Mascotas

The screenshot shows a Thunder Client interface with a GET request to `http://127.0.0.1:4000/mascotas/buscar`. The status is 200 OK, size is 1.46 KB, and time is 20 ms. The response is a JSON array of pet objects. The first object is: `{ "mascota_id": 6, "mascota_nombre": "bruno", "mascota_edad": 10, "mascota_tipo": "gato", "mascota_raza": "persa", "mascota_estado": "adoptado", "mascota_imagen": null, "createdAt": "2024-10-06T21:57:05.000Z", "updatedAt": "2024-10-06T21:57:05.000Z" }`. The second object is: `{ "mascota_id": 7, "mascota_nombre": "Firulais", "mascota_edad": 5, "mascota_tipo": "perro", "mascota_raza": "labrador", "mascota_estado": "disponible", "mascota_imagen": "http://example.com/imagen_de_firulais.jpg", "createdAt": "2024-10-07T04:30:18.000Z", "updatedAt": "2024-10-07T04:30:18.000Z" }`.

- Método BuscarId

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:4000/mascotas/buscarId/7`. The response is a JSON object with the following structure:

```
{  "mascota_id": 7,  "mascota_nombre": "Firulais",  "mascota_edad": 5,  "mascota_tipo": "perro",  "mascota_raza": "labrador",  "mascota_estado": "disponible",  "mascota_imagen": "http://example.com/imagen_de_firulais.jpg",  "createdAt": "2024-10-07T04:30:18.000Z",  "updatedAt": "2024-10-07T04:30:18.000Z"}
```

- Método Actualizar

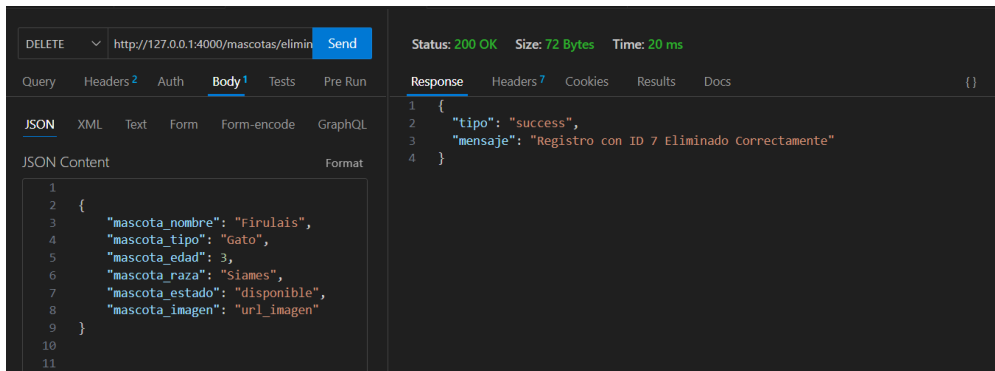
The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:4000/mascotas/actualizar/7`. The response is a JSON object with the following structure:

```
{  "tipo": "success",  "mensaje": "Registro Actualizado"}
```

Se comprueba en la base de datos la actualización de la mascota con el ID #7

mascotas							
Enter a SQL expression to filter results (use Ctrl+Space)							
	mascota_id	mascota_nombre	mascota_edad	mascota_tipo	mascota_raza	mascota_estado	mascota_imagen
1	3	pachos	9	Perro	Labrador	disponible	https://ichef.bbci.co.uk/10
2	4	Pacho	3	Gato	FrencPoodle	adoptado	https://unamglobal.una10
3	5	masha	7	perro	pinsher	disponible	https://as01.epimg.net/10
4	6	bruno	10	gato	persa	adoptado	[NULL]
5	7	Firulais	3	Gato	Siames	disponible	url_imagen

- Método Eliminar



Se comprueba que se ha eliminado el registro #7

The screenshot shows a database table view for the 'mascotas' table. The table has 8 columns: `mascota_id`, `mascota_nombre`, `mascota_edad`, `mascota_tipo`, `mascota_raza`, `mascota_estado`, and `mascota_imagen`. There are 4 records displayed:

	mascota_id	mascota_nombre	mascota_edad	mascota_tipo	mascota_raza	mascota_estado	mascota_imagen
1	3	pachos	9	Perro	Labrador	disponible	https://ichef.bbci.co.uk/...
2	4	Pacho	3	Gato	FrencPoodle	adoptado	https://unamglobal.una...
3	5	masha	7	perro	pinsher	disponible	https://as01.epimg.net/...
4	6	bruno	10	gato	persa	adoptado	[NULL]

Adoptante

- Método crear

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:4000/adoptantes/crear`. The request body is a JSON object with the following fields: `adoptante_nombre` (Daniel Santiago), `adoptante_email` (daniel@example.com), `adoptante_telefono` (1234567890), and `adoptante_direccion` (Calle Falsa 123, Ciudad). The response status is 201 Created, with a size of 320 Bytes and a time of 21 ms. The response body is a JSON object with the following fields: `mensaje` (Registro de Adoptante Creado con Éxito), `adoptante_id` (3), `adoptante_nombre` (Daniel Santiago), `adoptante_email` (daniel@example.com), `adoptante_telefono` (1234567890), `adoptante_direccion` (Calle Falsa 123, Ciudad), `updatedAt` (2024-10-07T04:42:52.848Z), and `createdAt` (2024-10-07T04:42:52.848Z).

```
POST http://127.0.0.1:4000/adoptantes/crear

{
  "adoptante_nombre": "Daniel Santiago",
  "adoptante_email": "daniel@example.com",
  "adoptante_telefono": "1234567890",
  "adoptante_direccion": "Calle Falsa 123, Ciudad"
}
```

Status: 201 Created Size: 320 Bytes Time: 21 ms

```
{
  "mensaje": "Registro de Adoptante Creado con Éxito",
  "adoptante": {
    "adoptante_id": 3,
    "adoptante_nombre": "Daniel Santiago",
    "adoptante_email": "daniel@example.com",
    "adoptante_telefono": "1234567890",
    "adoptante_direccion": "Calle Falsa 123, Ciudad",
    "updatedAt": "2024-10-07T04:42:52.848Z",
    "createdAt": "2024-10-07T04:42:52.848Z"
  }
}
```

- Método Buscar

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:4000/adoptantes/buscar`. The response status is 200 OK, with a size of 705 Bytes and a time of 28 ms. The response body is a JSON array of three adopter objects. The first object has `adoptante_id` 1, `adoptante_nombre` Luis, `adoptante_email` lucho@gmail.com, `adoptante_telefono` 7202020, `adoptante_direccion` calle 19, `createdAt` 2024-10-06T23:37:43.000Z, and `updatedAt` 2024-10-07T01:35:23.000Z. The second object has `adoptante_id` 2, `adoptante_nombre` andres, `adoptante_email` andres@gmail.com, `adoptante_telefono` 7202019, `adoptante_direccion` calle 20, `createdAt` 2024-10-07T00:42:25.000Z, and `updatedAt` 2024-10-07T00:42:25.000Z. The third object has `adoptante_id` 3, `adoptante_nombre` Daniel Santiago, `adoptante_email` daniel@example.com, `adoptante_telefono` 1234567890, `adoptante_direccion` Calle Falsa 123, Ciudad, `createdAt` 2024-10-07T04:42:52.000Z, and `updatedAt` 2024-10-07T04:42:52.000Z.

```
GET http://127.0.0.1:4000/adoptantes/buscar

{
  "adoptante_nombre": "Daniel Santiago",
  "adoptante_email": "daniel@example.com",
  "adoptante_telefono": "1234567890",
  "adoptante_direccion": "Calle Falsa 123, Ciudad"
}
```

Status: 200 OK Size: 705 Bytes Time: 28 ms

```
[
  {
    "adoptante_id": 1,
    "adoptante_nombre": "Luis",
    "adoptante_email": "lucho@gmail.com",
    "adoptante_telefono": "7202020",
    "adoptante_direccion": "calle 19",
    "createdAt": "2024-10-06T23:37:43.000Z",
    "updatedAt": "2024-10-07T01:35:23.000Z"
  },
  {
    "adoptante_id": 2,
    "adoptante_nombre": "andres",
    "adoptante_email": "andres@gmail.com",
    "adoptante_telefono": "7202019",
    "adoptante_direccion": "calle 20",
    "createdAt": "2024-10-07T00:42:25.000Z",
    "updatedAt": "2024-10-07T00:42:25.000Z"
  },
  {
    "adoptante_id": 3,
    "adoptante_nombre": "Daniel Santiago",
    "adoptante_email": "daniel@example.com",
    "adoptante_telefono": "1234567890",
    "adoptante_direccion": "Calle Falsa 123, Ciudad",
    "createdAt": "2024-10-07T04:42:52.000Z",
    "updatedAt": "2024-10-07T04:42:52.000Z"
  }
]
```

- Método Actualizar

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:4000/adoptantes/actualizar/3`. The request body is a JSON object with the following fields: `adoptante_nombre` (Santiago), `adoptante_email` (daniel@example.com), `adoptante_telefono` (1234567890), and `adoptante_direccion` (Calle 18). The response status is 200 OK, with a size of 51 Bytes and a time of 13 ms. The response body is a JSON object with the following fields: `tipo` (success) and `mensaje` (Registro Actualizado).

```
PUT http://127.0.0.1:4000/adoptantes/actualizar/3

{
  "adoptante_nombre": "Santiago",
  "adoptante_email": "daniel@example.com",
  "adoptante_telefono": "1234567890",
  "adoptante_direccion": "Calle 18"
}
```

Status: 200 OK Size: 51 Bytes Time: 13 ms

```
{
  "tipo": "success",
  "mensaje": "Registro Actualizado"
}
```

- Método Eliminar

DELETE

http://127.0.0.1:4000/adoptantes/eliminar/3

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "adoptante_nombre": "Santiago",
3   "adoptante_email": "daniel@example.com",
4   "adoptante_telefono": "1234567890",
5   "adoptante_direccion": "Calle 18"
6 }
7 |
```

Status: 200 OK

Size: 72 Bytes

Time: 12 ms

Response

Headers 7

Cookies

Results

Docs

{}

≡

```
1 {
2   "tipo": "success",
3   "mensaje": "Registro con ID 3 Eliminado Correctamente"
4 }
```

Solicitudes

- Método Crear

POST

http://127.0.0.1:4000/solicitudes/crear

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "mascota_id": 5,
3   "adoptante_id": 1,
4   "solicitud_fecha": "2024-10-06",
5   "solicitud_estado": "en revisión"
6 }
7 |
```

Status: 201 Created

Size: 185 Bytes

Time: 46 ms

Response

Headers 7

Cookies

Results

Docs

{}

≡

```
1 {
2   "mensaje": "Solicitud creada con éxito.",
3   "solicitud": {
4     "solicitud_fecha": {
5       "val": "CURRENT_TIMESTAMP"
6     },
7     "solicitud_estado": "en revisión",
8     "solicitud_id": 7,
9     "mascota_id": 5,
10    "adoptante_id": 1
11  }
12 }
```

- Método Buscar

GET

http://127.0.0.1:4000/solicitudes/buscar

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "mascota_id": 5,
3   "adoptante_id": 1,
4   "solicitud_fecha": "2024-10-06",
5   "solicitud_estado": "en revisión"
6 }
7 |
```

Status: 200 OK

Size: 391 Bytes

Time: 14 ms

Response

Headers 7

Cookies

Results

Docs

{}

≡

```
1 [
2   {
3     "solicitud_id": 1,
4     "mascota_id": 3,
5     "adoptante_id": 1,
6     "solicitud_fecha": "2024-10-07T03:01:59.000Z",
7     "solicitud_estado": "en revisión"
8   },
9   {
10    "solicitud_id": 6,
11    "mascota_id": 5,
12    "adoptante_id": 2,
13    "solicitud_fecha": "2024-10-07T03:41:29.000Z",
14    "solicitud_estado": "en revisión"
15  },
16  {
17    "solicitud_id": 7,
18    "mascota_id": 5,
19    "adoptante_id": 1,
20    "solicitud_fecha": "2024-10-07T04:47:23.000Z",
21    "solicitud_estado": "en revisión"
22  }
23 ]
```

- Método Actualizar

PUT

http://127.0.0.1:4000/solicitudes/actualizar/7

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "mascota_id": 5,
3   "adoptante_id": 6,
4   "solicitud_fecha": "2024-10-06",
5   "solicitud_estado": "denegada"
6 }
7
```

Status: 200 OK

Size: 47 Bytes

Time: 20 ms

Response

Headers 7

Cookies

Results

Docs

{}

≡

```
1 {
2   "mensaje": "Solicitud actualizada con éxito."
3 }
```

- Método Eliminar

DELETE

http://127.0.0.1:4000/solicitudes/eliminar/7

Send

Query

Headers 2

Auth

Body 1

Tests

Pre Run

JSON

XML

Text

Form

Form-encode

GraphQL

Binary

JSON Content

Format

```
1 {
2   "mascota_id": 5,
3   "adoptante_id": 6,
4   "solicitud_fecha": "2024-10-06",
5   "solicitud_estado": "denegada"
6 }
7
```

Status: 200 OK

Size: 45 Bytes

Time: 18 ms

Response

Headers 7

Cookies

Results

Docs

{}

≡

```
1 {
2   "mensaje": "Solicitud eliminada con éxito."
3 }
```