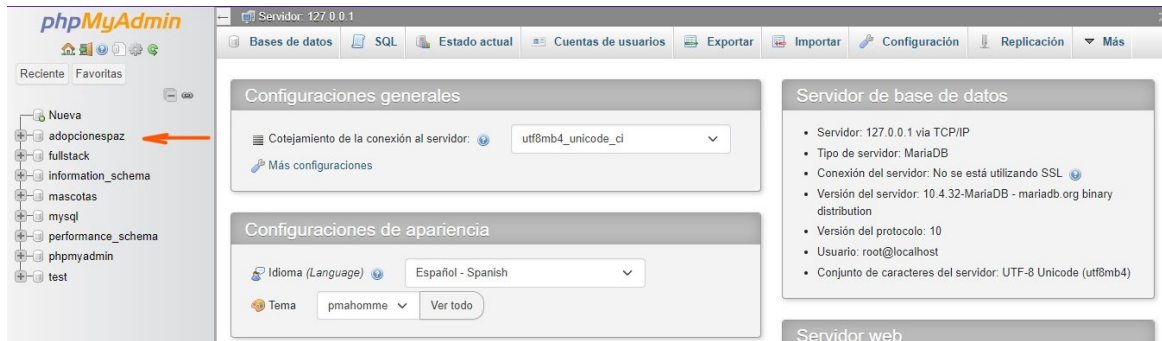
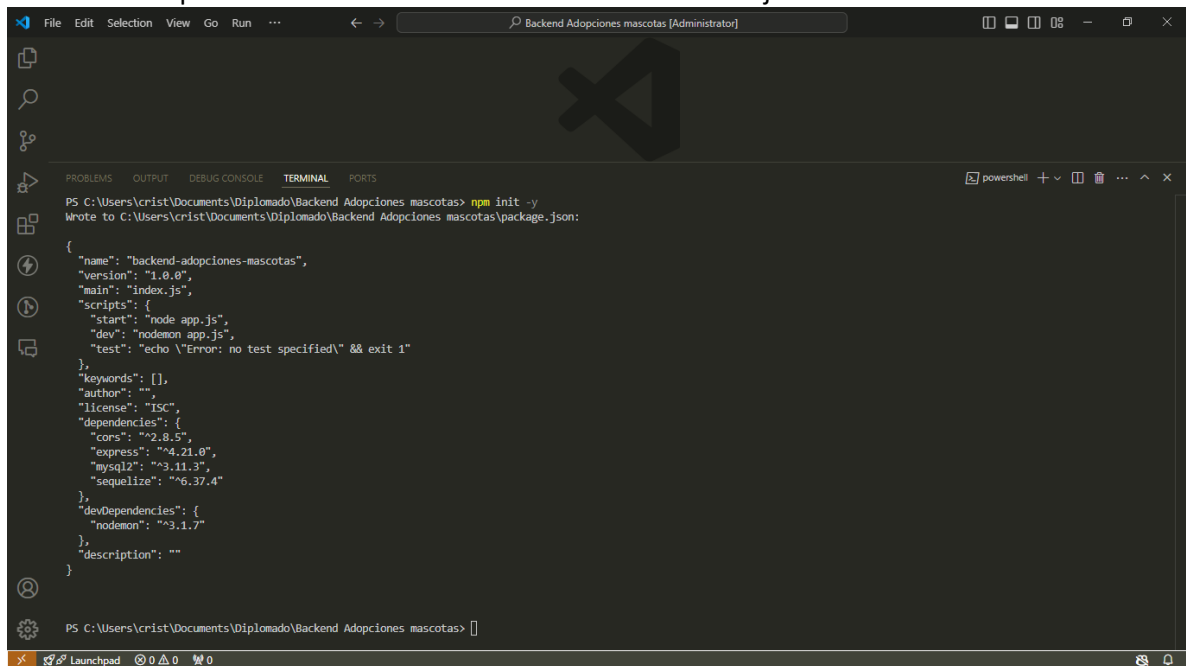


## INFORME DEL TALLER 2 (BACKEND) – 221036047

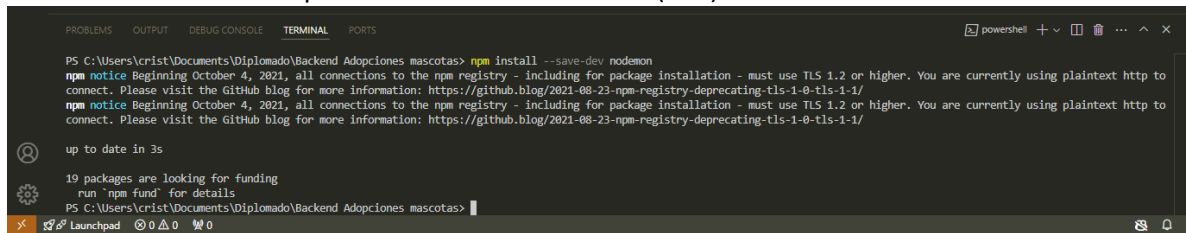
- Creación de la base de datos **adopcionespaz**:



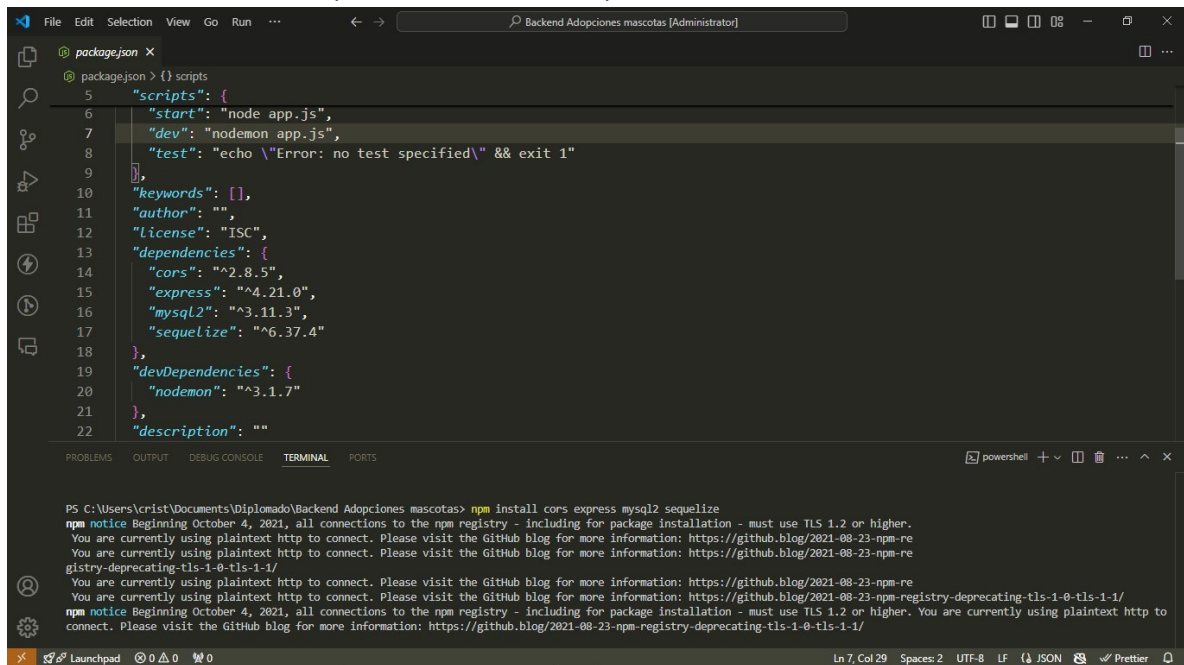
- Creación de repositorio local e iniciación del backend con node.js en vs.



- Instalación de nodemon para el entorno de desarrollo (dev)



- Instalación de las demás dependencias incluido express:

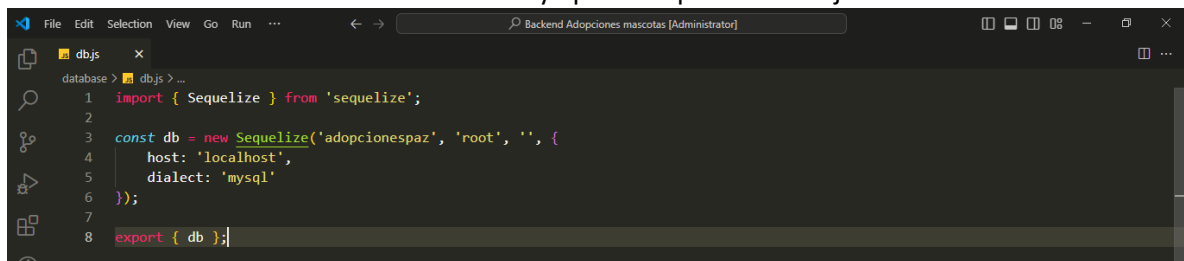


The screenshot shows a VS Code editor with a file named `package.json` open. The file contains the following JSON structure:

```
{
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js",
    "test": "echo \\\"Error: no test specified\\\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^4.21.0",
    "mysql2": "^3.11.3",
    "sequelize": "^6.37.4"
  },
  "devDependencies": {
    "nodemon": "^3.1.7"
  },
  "description": ""
}
```

Below the editor, the terminal shows the output of the command `npm install cors express mysql2 sequelize`. The output includes several notices from npm regarding TLS 1.2 requirements and the use of plaintext http for connections.

- Se crea la conexión de la base de datos de mysql con sequelize en db.js:



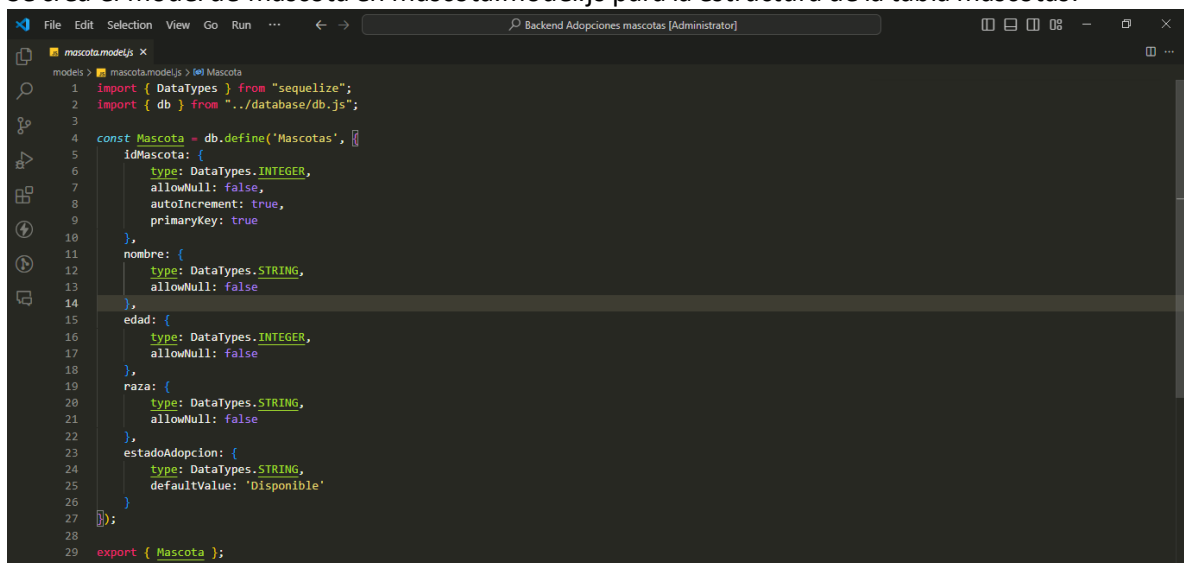
The screenshot shows a VS Code editor with a file named `db.js` open. The file contains the following JavaScript code:

```
import { Sequelize } from 'sequelize';

const db = new Sequelize('adopcionespaz', 'root', '', {
  host: 'localhost',
  dialect: 'mysql'
});

export { db };
```

- Se crea el model de mascota en mascota.model.js para la estructura de la tabla mascotas:



The screenshot shows a VS Code editor with a file named `mascota.model.js` open. The file contains the following JavaScript code:

```
import { DataTypes } from 'sequelize';
import { db } from '../database/db.js';

const Mascota = db.define('Mascotas', {
  idMascota: {
    type: DataTypes.INTEGER,
    allowNull: false,
    autoIncrement: true,
    primaryKey: true
  },
  nombre: {
    type: DataTypes.STRING,
    allowNull: false
  },
  edad: {
    type: DataTypes.INTEGER,
    allowNull: false
  },
  raza: {
    type: DataTypes.STRING,
    allowNull: false
  },
  estadoAdopcion: {
    type: DataTypes.STRING,
    defaultValue: 'Disponible'
  }
});

export { Mascota };
```

- Se crea el model de solicitud en solicitud.model.js para la estructura de la tabla solicitudes, con una relación de uno a varios ya que una mascota puede tener varias solicitudes de adopción.:

```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]
solicitud.model.js X
models > solicitud.model.js > ...
1 import { DataTypes } from "sequelize";
2 import { db } from "../database/db.js";
3 import { Mascota } from "../mascota.model.js";
4
5 const Solicitud = db.define('Solicitudes', {
6   idSolicitud: {
7     type: DataTypes.INTEGER,
8     allowNull: false,
9     autoIncrement: true,
10    primaryKey: true
11  },
12  identidadSolicitante: {
13    type: DataTypes.INTEGER,
14    allowNull: false,
15    unique: true,
16    validate: {
17      len: [6, 10]
18    }
19  },
20  fechaSolicitud: {
21    type: DataTypes.DATE,
22    defaultValue: DataTypes.NOW
23  },
24  estado: {
25    type: DataTypes.STRING,
26    defaultValue: 'Pendiente'
27  }
28 });
29
30 // Relación
31 Solicitud.belongsTo(Mascota, {
32   foreignKey: 'idMascota',
33   onDelete: 'CASCADE'
34 });
35
36 export { Solicitud };

```

- Se crea mascotas.routes.js en donde están todas las rutas y solicitudes para la gestión de la tabla mascotas:

```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]
mascotas.routes.js X
routes > mascotas.routes.js > router.post('/ingresarMascota') callback
1 import express from 'express';
2 import { Mascota } from '../models/mascota.model.js'; // Importamos el modelo Mascota
3
4 const router = express.Router();
5
6 // Obtener todas las mascotas
7 router.get('/mascotas', async (req, res) => {
8   try {
9     const mascotas = await Mascota.findAll();
10    res.json(mascotas);
11  } catch (error) {
12    res.status(500).json({ error: 'Error al obtener la lista de mascotas.' });
13  }
14 });
15
16 // Obtener una mascota por ID
17 router.get('/mascotas/:idMascota', async (req, res) => {
18   try {
19     const mascota = await Mascota.findByPk(req.params.idMascota);
20     if (mascota) {
21       res.json(mascota);
22     } else {
23       res.status(404).json({ error: 'Mascota no encontrada.' });
24     }
25   } catch (error) {
26     res.status(500).json({ error: 'Error al obtener la mascota.' });
27   }
28 });

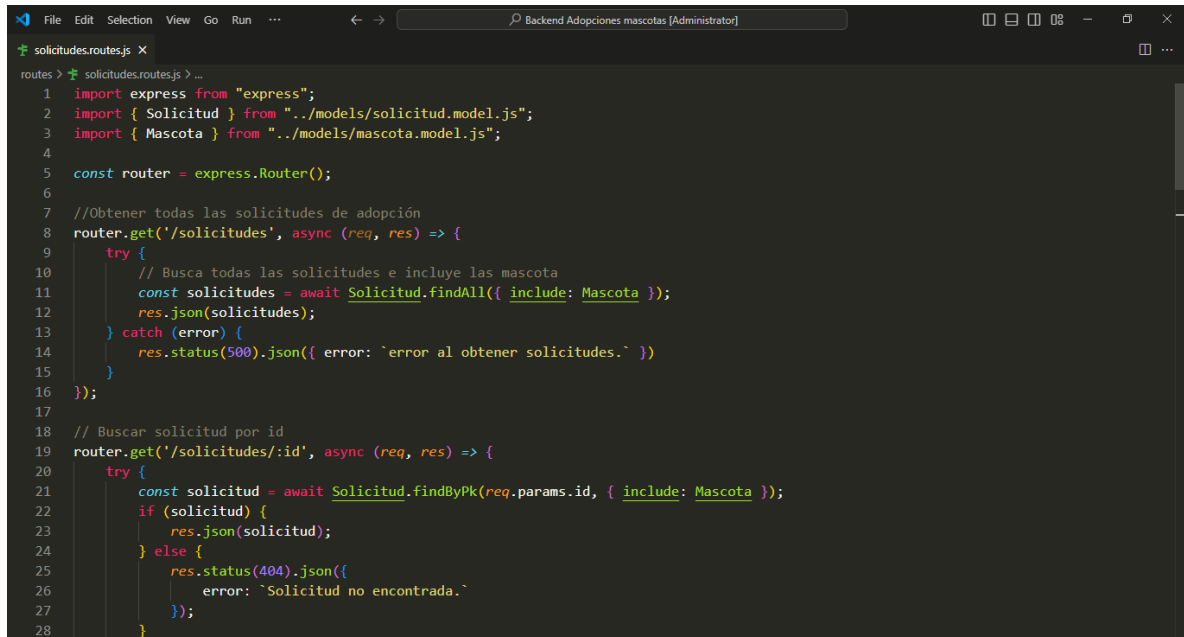
```

```

29
30 // Crear una nueva mascota
31 router.post('/ingresarmascota', async (req, res) => {
32   try {
33     const { nombre, edad, raza } = req.body; //Toma datos del cuerpo del JSON
34     const nuevaMascota = await Mascota.create({ nombre, edad, raza }); //Creación de mascota
35     res.status(201).json(nuevaMascota);
36   } catch (error) {
37     res.status(500).json({ error: 'Error al crear la mascota' });
38   }
39 });
40
41 // Actualizar una mascota por ID
42 router.put('/actualizarmascota/:idMascota', async (req, res) => {
43   try {
44     const mascota = await Mascota.findById(req.params.idMascota); // Buscar la mascota por ID
45     if (mascota) {
46       await mascota.update(req.body); // Actualizar los datos de la mascota
47       res.json(mascota); // Devolver la mascota actualizada
48     } else {
49       res.status(404).json({ error: 'Mascota no encontrada' });
50     }
51   } catch (error) {
52     res.status(500).json({ error: 'Error al actualizar la mascota' });
53   }
54 });
55
56 // Eliminar una mascota por ID
57 router.delete('/eliminarascota/:idMascota', async (req, res) => {
58   try {
59     const mascota = await Mascota.findById(req.params.idMascota); // Buscar la mascota por ID
60     if (mascota) {
61       await mascota.destroy(); // Eliminar la mascota
62       res.json({ mensaje: 'Mascota eliminada' }); // Confirmar la eliminación
63     } else {
64       res.status(404).json({ error: 'Mascota no encontrada' });
65     }
66   } catch (error) {
67     res.status(500).json({ error: 'Error al eliminar la mascota' });
68   }
69 });
70
71 export { router };

```

- Se crea solicitudes.routes.js en donde están todas las rutas y solicitudes para la tabla solicitudes:



```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrador]
solicitudes.routes.js X
routes > solicitudes.routes.js > ...
1 import express from "express";
2 import { Solicitud } from "../models/solicitud.model.js";
3 import { Mascota } from "../models/mascota.model.js";
4
5 const router = express.Router();
6
7 //Obtener todas las solicitudes de adopción
8 router.get('/solicitudes', async (req, res) => {
9   try {
10     // Busca todas las solicitudes e incluye las mascota
11     const solicitudes = await Solicitud.findAll({ include: Mascota });
12     res.json(solicitudes);
13   } catch (error) {
14     res.status(500).json({ error: `error al obtener solicitudes.` })
15   }
16 });
17
18 // Buscar solicitud por id
19 router.get('/solicitudes/:id', async (req, res) => {
20   try {
21     const solicitud = await Solicitud.findById(req.params.id, { include: Mascota });
22     if (solicitud) {
23       res.json(solicitud);
24     } else {
25       res.status(404).json({
26         error: `Solicitud no encontrada.`
27       });
28     }
29   }
30 });

```

```

28     }
29   } catch (error) {
30     res.status(500).json({
31       error: 'Error al obtener la solicitud.'
32     });
33   }
34 });
35
36 // Crear solicitud
37 router.post('/nuevasolicitud', async (req, res) => {
38   try {
39     const { idMascota, identidadSolicitante } = req.body;
40
41     const mascota = await Mascota.findById(idMascota);
42     if (!mascota) {
43       return res.status(404).json({
44         error: 'Mascota no encontrada.'
45       });
46     }
47
48     const solicitud = await Solicitud.create({
49       idMascota,
50       identidadSolicitante
51     });
52
53     // Enviar la solicitud creada.
54     res.status(201).json(solicitud);
55   } catch (error) {
56     console.log(error);
57     res.status(500).json({
58       error: 'Error al crear la solicitud.'
59     });
60   }
61 });
62
63 //Actualizar una solicitud
64 router.put('/actualizarsolicitud/:id', async (req, res) => {
65   try {
66     const solicitud = await Solicitud.findById(req.params.id);
67
68     if (!solicitud) {
69       return res.status(404).json({ error: 'Solicitud no encontrada.' });
70     }
71
72     // Si el estado de la solicitud es 'Aceptada', entonces se rechazan las
73     // demás solicitudes para esa mascota porque ya pasa a ser 'Adoptada'.
74     if (req.body.estado === 'Aceptada') {
75       const mascota = await Mascota.findById(solicitud.idMascota);
76
77       if (!mascota) {
78         return res.status(404).json({ error: 'Mascota no encontrada.' });
79       }
80
81       // Cambiar el estado de la mascota a 'Adoptada'
82       await mascota.update({ estadoAdopcion: 'Adoptada' });
83
84       // Actualizar la solicitud a 'Aceptada'
85       await solicitud.update({ estado: 'Aceptada' });
86
87       // Rechazar las otras solicitudes pendientes para esta mascota
88       await Solicitud.update(
89         { estado: 'Rechazada' },
90         { where: { idMascota: solicitud.idMascota, estado: 'Pendiente' } }
91       );
92
93       return res.json({ mensaje: 'Solicitud aceptada.' });
94     } else {
95       // Para cualquier otro cambio, solo actualiza la solicitud
96       await solicitud.update(req.body);
97       return res.json(solicitud);
98     }
99   } catch (error) {
100     console.error(error);
101     res.status(500).json({ error: 'Error al actualizar la solicitud.' });
102   }
103 });
104
105

```

```

106 // Eliminar solicitud.
107 router.delete('/eliminarSolicitud/:id', async (req, res) => {
108   try {
109     const solicitud = await Solicitud.findById(req.params.id);
110     if (solicitud) {
111       await solicitud.destroy();
112       res.json({ mensaje: 'Solicitud eliminada' });
113     } else {
114       res.status(404).json({ error: 'Solicitud no encontrada.' });
115     }
116   } catch (error) {
117     res.status(500).json({ error: 'Error al eliminar la solicitud' });
118   }
119 });
120
121
122 export { router };

```

- En el archivo app.js se configura un servidor backend con Express y Sequelize para gestionar solicitudes de adopción de mascotas para la empresa Adopcionespaz. Se agrega la inicialización el servidor, también se hace uso de CORS y el procesamiento de datos JSON. Se importa la base de datos hecha con sequelize anteriormente, asegurando que la conexión sea exitosa antes de iniciar el servidor en el puerto 3000. Las rutas para solicitudes y mascotas se integran bajo el mismo endpoint (/adopcionespaz) para gestionar las operaciones CRUD correspondientes:

```

1  import express from 'express';
2  import cors from 'cors';
3  import { db } from './database/db.js';
4  import { router as routerSolicitudes } from './routes/solicitudes.routes.js';
5  import { router as routerMascotas } from './routes/mascotas.routes.js';
6
7  // Instanciación de express.
8  const app = express();
9  // Se agrega middleware CORS e interprete de JSON.
10 app.use(cors());
11 app.use(express.json());
12
13 //conexión a la base de datos (db sequelize).
14 db.authenticate()
15   .then(() => {
16     console.log('■ Conexión exitosa a la base de datos!');
17     return db.sync();
18   })
19   .then(() => {
20     const PORT = 3000;
21     app.listen(PORT, () => {
22       // Confirmación del servidor corriendo.
23       console.log('■ Servidor corriendo en el puerto ${PORT}\n');
24     });
25   })
26   .catch(err => {
27     console.error('■ Conexión fallida a la base de datos: \n${err}');
28   });
29
30 app.use('/adopcionespaz', routerSolicitudes);
31 app.use('/adopcionespaz', routerMascotas);

```

- Al ejecutar “*npm run dev*” se crean las tablas (si no existen, como en este caso que las he eliminado para comprobarlo):

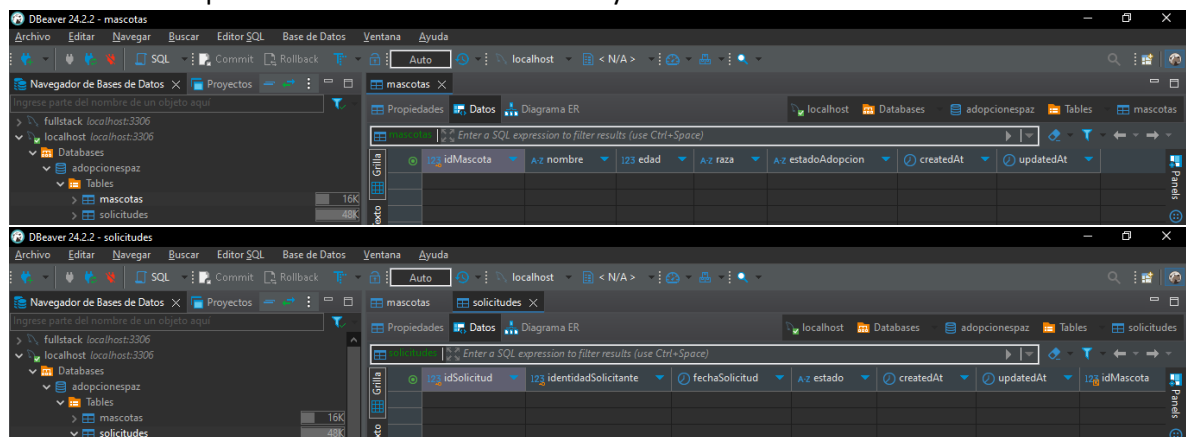
```

PS C:\Users\crist\Documents\Diplomado\Backend Adopciones mascotas> npm run dev

> backend-adopciones-mascotas@1.0.0 dev
> nodemon app.js

[nodemon] 3.1.7
[nodemon] to restart at any time, enter 'rs'
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting 'node app.js'
Executing (default): SELECT 1+1 AS result
■ Conexión exitosa a la base de datos!
■ Ejecutando (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'Mascotas' AND TABLE_SCHEMA = 'adopcionespaz'
■ Ejecutando (default): CREATE TABLE IF NOT EXISTS 'Mascotas' ('idMascota' INTEGER NOT NULL auto-increment, 'nombre' VARCHAR(255) NOT NULL, 'edad' INTEGER NOT NULL, 'raza' VARCHAR(255) NOT NULL, 'estadoAdopcion' VARCHAR(255) DEFAULT 'Disponible', 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, PRIMARY KEY ('idMascota')) ENGINE=InnoDB;
■ Ejecutando (default): SHOW INDEX FROM 'Mascotas'
■ Ejecutando (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'Solicitudes' AND TABLE_SCHEMA = 'adopcionespaz'
■ Ejecutando (default): CREATE TABLE IF NOT EXISTS 'Mascotas' ('idMascota' INTEGER NOT NULL auto-increment, 'nombre' VARCHAR(255) NOT NULL, 'edad' INTEGER NOT NULL, 'raza' VARCHAR(255) NOT NULL, 'estadoAdopcion' VARCHAR(255) DEFAULT 'Disponible', 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, PRIMARY KEY ('idMascota')) ENGINE=InnoDB;
■ Ejecutando (default): SHOW INDEX FROM 'Mascotas'
■ Ejecutando (default): CREATE TABLE IF NOT EXISTS 'Mascotas' ('idMascota' INTEGER NOT NULL auto-increment, 'nombre' VARCHAR(255) NOT NULL, 'edad' INTEGER NOT NULL, 'raza' VARCHAR(255) NOT NULL, 'estadoAdopcion' VARCHAR(255) DEFAULT 'Disponible', 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, PRIMARY KEY ('idMascota')) ENGINE=InnoDB;
■ Ejecutando (default): CREATE TABLE IF NOT EXISTS 'Mascotas' ('idMascota' INTEGER NOT NULL auto-increment, 'nombre' VARCHAR(255) NOT NULL, 'edad' INTEGER NOT NULL, 'raza' VARCHAR(255) NOT NULL, 'estadoAdopcion' VARCHAR(255) DEFAULT 'Disponible', 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, PRIMARY KEY ('idMascota')) ENGINE=InnoDB;
■ Ejecutando (default): SHOW INDEX FROM 'Mascotas'
■ Ejecutando (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'Solicitudes' AND TABLE_SCHEMA = 'adopcionespaz'
■ Ejecutando (default): CREATE TABLE IF NOT EXISTS 'Solicitudes' ('idSolicitud' INTEGER NOT NULL auto-increment, 'identidadSolicitante' INTEGER NOT NULL UNIQUE, 'fechaSolicitud' DATETIME, 'estado' VARCHAR(255) DEFAULT 'Pendiente', 'createdAt' DATETIME NOT NULL, 'updatedAt' DATETIME NOT NULL, 'idMascota' INTEGER, PRIMARY KEY ('idSolicitud'), FOREIGN KEY ('idMascota') REFERENCES 'Mascotas' ('idMascota') ON DELETE CASCADE ON UPDATE CASCADE) ENGINE=InnoDB;
■ Ejecutando (default): SHOW INDEX FROM 'Solicitudes'
■ Servidor corriendo en el puerto 3000
  
```

- En DBeaver compruebo si las tablas se han creado y satisfactoriamente ha sido así:



A continuación, se realizará la comprobación de las diferentes funcionalidades del crud implementado anteriormente con solicitudes realizadas con Thunder Client en visual studio code para la tabla mascotas:

- Se realiza pruebas de la creación o ingreso de mascotas (POST):

The image displays three sequential screenshots of the Thunder Client interface within Visual Studio Code, demonstrating the creation of three new pet records via POST requests to the endpoint `http://localhost:3000/adopcionespaz/ingresarmascota`.

**First Screenshot (Top):** Shows the first POST request. The JSON body contains the details for a pet named "Max". The response status is 201 Created, with a size of 170 Bytes and a time of 90 ms. The response body includes the pet's ID (1), name, age, breed, and timestamps.

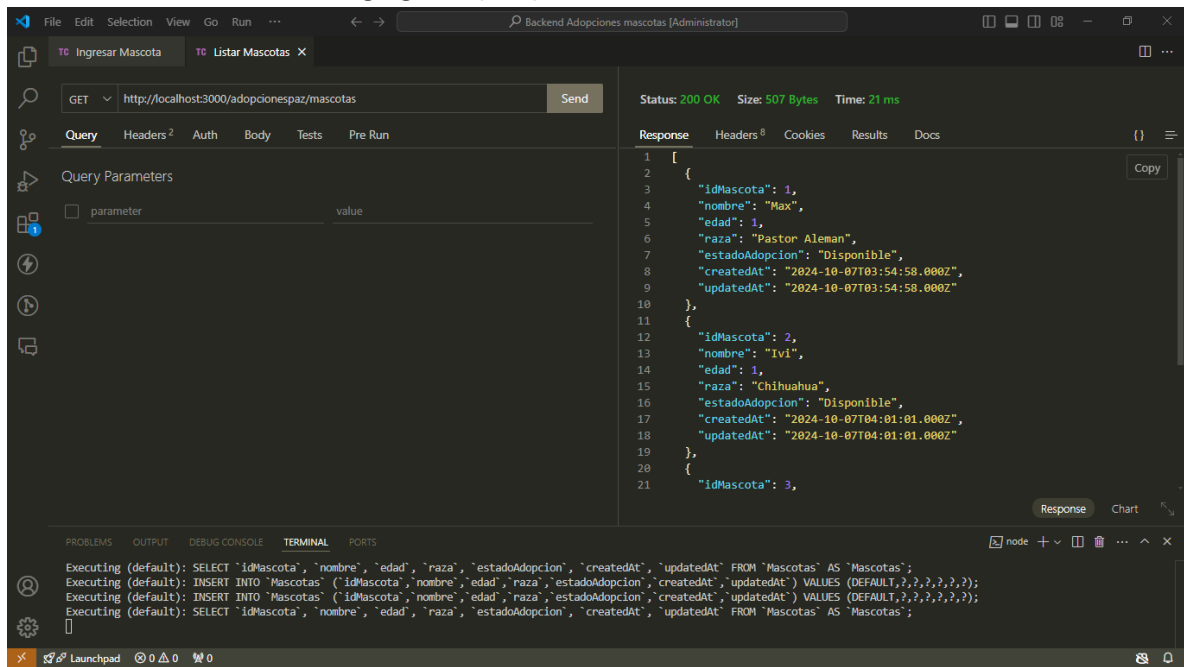
**Second Screenshot (Middle):** Shows the second POST request. The JSON body contains the details for a pet named "Ivi". The response status is 201 Created, with a size of 166 Bytes and a time of 21 ms. The response body includes the pet's ID (2), name, age, breed, and timestamps.

**Third Screenshot (Bottom):** Shows the third POST request. The JSON body contains the details for a pet named "Stuart". The response status is 201 Created, with a size of 167 Bytes and a time of 28 ms. The response body includes the pet's ID (3), name, age, breed, and timestamps.

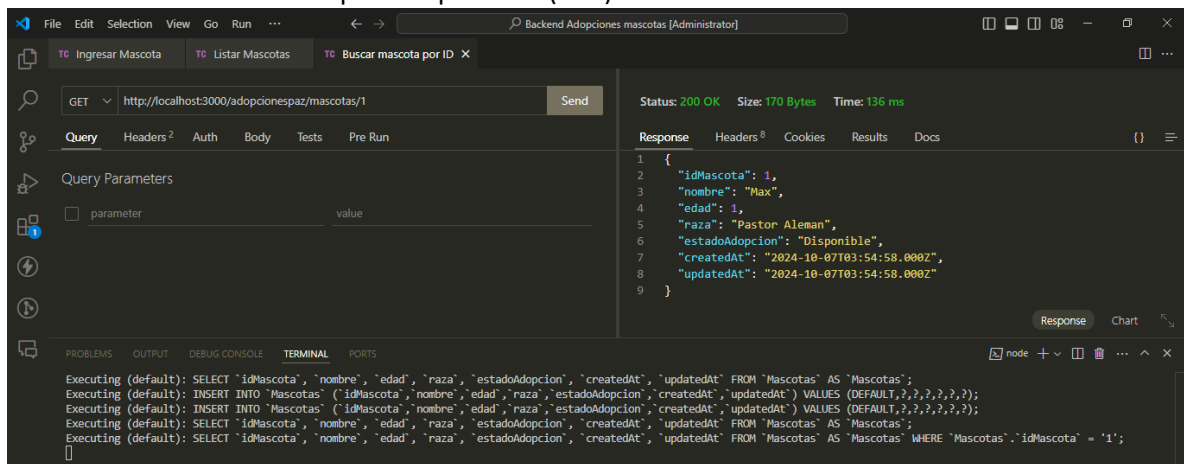
In all three screenshots, the terminal at the bottom shows the server running on port 3000 and the execution of SQL commands to insert the new records into the `Mascotas` table.



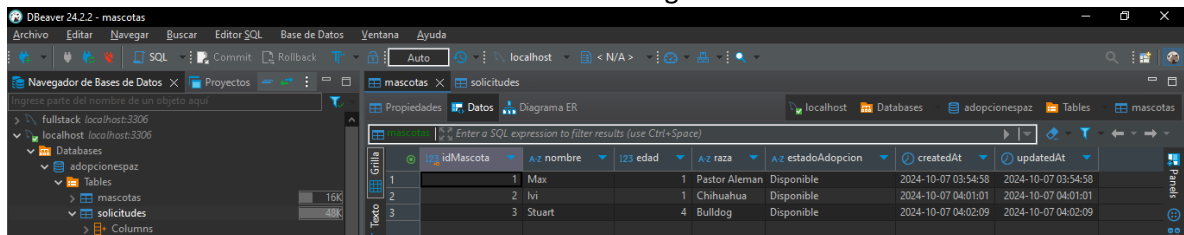
- Ahora se listan las mascotas agregadas (GET):



- Se busca una mascota en específico por su id (GET):



- Al momento la tabla mascotas en DBeaver se ve de la siguiente manera:



- Ahora se realizarán pruebas de actualización de datos para las 3 tuplas (PUT):

The following tables represent the data shown in the screenshots, organized by the ID of the record being updated.

Request ID	URL	Request Body (JSON)	Response Body (JSON)
1	http://localhost:3000/adopcionespaz/actualizarmascota/1	<pre>{   "nombre": "Maxi",   "edad": 2 }</pre>	<pre>{   "idMascota": 1,   "nombre": "Maxi",   "edad": 2,   "raza": "Pastor Aleman",   "estadoAdopcion": "Disponible",   "createdAt": "2024-10-07T03:54:58.000Z",   "updatedAt": "2024-10-07T04:15:05.000Z" }</pre>
2	http://localhost:3000/adopcionespaz/actualizarmascota/2	<pre>{   "estadoAdopcion": "Adoptada" }</pre>	<pre>{   "idMascota": 2,   "nombre": "Ivli",   "edad": 1,   "raza": "Chihuahua",   "estadoAdopcion": "Adoptada",   "createdAt": "2024-10-07T04:01:01.000Z",   "updatedAt": "2024-10-07T04:18:42.468Z" }</pre>
3	http://localhost:3000/adopcionespaz/actualizarmascota/3	<pre>{   "raza": "Beagle" }</pre>	<pre>{   "idMascota": 3,   "nombre": "Stuart",   "edad": 4,   "raza": "Beagle",   "estadoAdopcion": "Disponible",   "createdAt": "2024-10-07T04:02:09.000Z",   "updatedAt": "2024-10-07T04:19:50.770Z" }</pre>

- Al momento así se ven los cambios efectuados de la tabla mascotas en DBeaver:

	idMascota	nombre	edad	raza	estadoAdopcion	createdAt	updatedAt
1	1	Maxi	2	Pastor Aleman	Disponible	2024-10-07 03:54:58	2024-10-07 04:15:05
2	2	Ivi	1	Chihuahua	Adoptada	2024-10-07 04:01:01	2024-10-07 04:18:42
3	3	Stuart	4	Beagle	Disponible	2024-10-07 04:02:09	2024-10-07 04:19:50

- Y por último para la tabla mascotas se prueba la eliminación de una mascota con id (DELETE):

**DELETE** `http://localhost:3000/adopcionespaz/eliminarmascota/2`

Status: 200 OK Size: 31 Bytes Time: 24 ms

```

{
  "mensaje": "Mascota eliminada"
}

```

Terminal output:

```

Conexión exitosa a la base de datos!
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'Mascotas' AND TABLE_SCHEMA = 'adopcionespaz'
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 = 'adopcionespaz'
Executing (default): SHOW INDEX FROM 'Solicitudes'
Servidor corriendo en el puerto 3000

Executing (default): SELECT `idMascota`, `nombre`, `edad`, `raza`, `estadoAdopcion`, `createdAt`, `updatedAt` FROM `Mascotas` AS `Mascotas` WHERE `Mascotas`.`idMascota` = '1';
Executing (default): SELECT `idMascota`, `nombre`, `edad`, `raza`, `estadoAdopcion`, `createdAt`, `updatedAt` FROM `Mascotas` AS `Mascotas` WHERE `Mascotas`.`idMascota` = '2';
Executing (default): SELECT `idMascota`, `nombre`, `edad`, `raza`, `estadoAdopcion`, `createdAt`, `updatedAt` FROM `Mascotas` AS `Mascotas` WHERE `Mascotas`.`idMascota` = '3';
Executing (default): UPDATE `Mascotas` SET `estadoAdopcion`=?, `updatedAt`=? WHERE `idMascota` = ?
Executing (default): SELECT `idMascota`, `nombre`, `edad`, `raza`, `estadoAdopcion`, `createdAt`, `updatedAt` FROM `Mascotas` AS `Mascotas` WHERE `Mascotas`.`idMascota` = '2';
Executing (default): UPDATE `Mascotas` SET `raza`=?, `updatedAt`=? WHERE `idMascota` = ?
Executing (default): SELECT `idMascota`, `nombre`, `edad`, `raza`, `estadoAdopcion`, `createdAt`, `updatedAt` FROM `Mascotas` AS `Mascotas` WHERE `Mascotas`.`idMascota` = '2';
Executing (default): DELETE FROM `Mascotas` WHERE `idMascota` = 2

```

- Quedando la tabla mascotas de la siguiente manera en DBeaver:

	idMascota	nombre	edad	raza	estadoAdopcion	createdAt	updatedAt
1	1	Maxi	2	Pastor Aleman	Disponible	2024-10-07 03:54:58	2024-10-07 04:15:05
2	3	Stuart	4	Beagle	Disponible	2024-10-07 04:02:09	2024-10-07 04:19:50

Por último, se realizará la comprobación de las diferentes funcionalidades del crud implementado anteriormente con solicitudes realizadas con Thunder Client en visual studio code para la tabla solicitudes:

- Se realiza la creación de nuevas solicitudes de adopción para una mascota en específico:

**POST** `http://localhost:3000/adopcionespaz/nuevasolicitud`

Status: 201 Created Size: 208 Bytes Time: 97 ms

```

{
  "fechaSolicitud": "2024-10-07T04:34:57.135Z",
  "estado": "Pendiente",
  "idSolicitud": 1,
  "idMascota": 1,
  "IdentidadSolicitante": 1004236301,
  "updatedAt": "2024-10-07T04:34:57.136Z",
  "createdAt": "2024-10-07T04:34:57.136Z"
}

```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]

TC Crear solicitud X

POST http://localhost:3000/adopcionespaz/nuevasolicitud Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "idMascota": 1,
3   "identidadSolicitante": 1004236302
4 }
```

Status: 201 Created Size: 208 Bytes Time: 25 ms

Response Headers<sup>8</sup> Cookies Results Docs {}

```
1 {
2   "fechaSolicitud": "2024-10-07T04:36:43.456Z",
3   "estado": "Pendiente",
4   "idSolicitud": 2,
5   "idMascota": 1,
6   "identidadSolicitante": 1004236302,
7   "updatedAt": "2024-10-07T04:36:43.457Z",
8   "createdAt": "2024-10-07T04:36:43.457Z"
9 }
```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]

TC Crear solicitud X

POST http://localhost:3000/adopcionespaz/nuevasolicitud Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "idMascota": 1,
3   "identidadSolicitante": 1004236303
4 }
```

Status: 201 Created Size: 208 Bytes Time: 27 ms

Response Headers<sup>8</sup> Cookies Results Docs {}

```
1 {
2   "fechaSolicitud": "2024-10-07T04:37:06.106Z",
3   "estado": "Pendiente",
4   "idSolicitud": 3,
5   "idMascota": 1,
6   "identidadSolicitante": 1004236303,
7   "updatedAt": "2024-10-07T04:37:06.107Z",
8   "createdAt": "2024-10-07T04:37:06.107Z"
9 }
```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]

TC Crear solicitud X

POST http://localhost:3000/adopcionespaz/nuevasolicitud Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "idMascota": 3,
3   "identidadSolicitante": 1004236304
4 }
```

Status: 201 Created Size: 208 Bytes Time: 25 ms

Response Headers<sup>8</sup> Cookies Results Docs {}

```
1 {
2   "fechaSolicitud": "2024-10-07T04:37:41.549Z",
3   "estado": "Pendiente",
4   "idSolicitud": 4,
5   "idMascota": 3,
6   "identidadSolicitante": 1004236304,
7   "updatedAt": "2024-10-07T04:37:41.549Z",
8   "createdAt": "2024-10-07T04:37:41.549Z"
9 }
```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]

TC Crear solicitud X

POST http://localhost:3000/adopcionespaz/nuevasolicitud Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "idMascota": 3,
3   "identidadSolicitante": 1004236305
4 }
```

Status: 201 Created Size: 208 Bytes Time: 25 ms

Response Headers<sup>8</sup> Cookies Results Docs {}

```
1 {
2   "fechaSolicitud": "2024-10-07T04:39:17.350Z",
3   "estado": "Pendiente",
4   "idSolicitud": 5,
5   "idMascota": 3,
6   "identidadSolicitante": 1004236305,
7   "updatedAt": "2024-10-07T04:39:17.350Z",
8   "createdAt": "2024-10-07T04:39:17.350Z"
9 }
```

File Edit Selection View Go Run ... Backend Adopciones mascotas [Administrator]

TC Crear solicitud X

POST http://localhost:3000/adopcionespaz/nuevasolicitud Send

Query Headers<sup>2</sup> Auth Body<sup>1</sup> Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

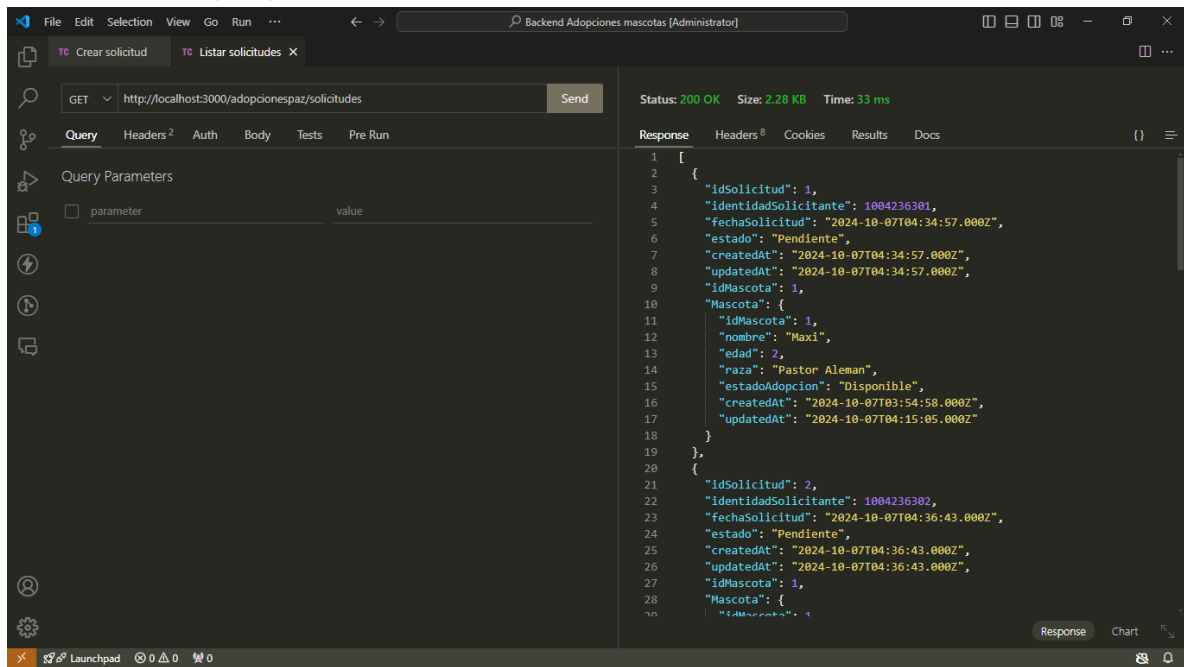
```
1 {
2   "idMascota": 3,
3   "identidadSolicitante": 1004236306
4 }
```

Status: 201 Created Size: 208 Bytes Time: 29 ms

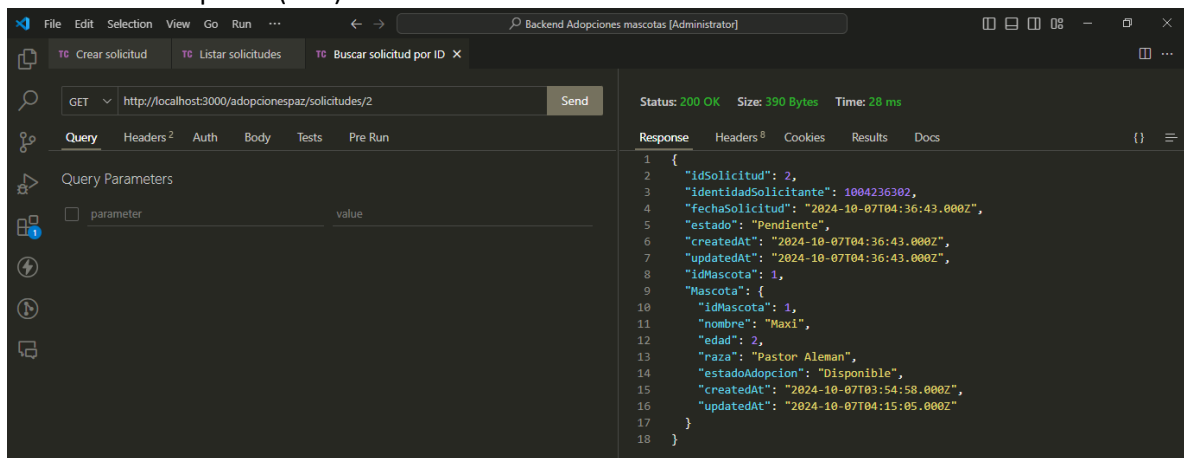
Response Headers<sup>8</sup> Cookies Results Docs {}

```
1 {
2   "fechaSolicitud": "2024-10-07T04:39:43.028Z",
3   "estado": "Pendiente",
4   "idSolicitud": 6,
5   "idMascota": 3,
6   "identidadSolicitante": 1004236306,
7   "updatedAt": "2024-10-07T04:39:43.028Z",
8   "createdAt": "2024-10-07T04:39:43.028Z"
9 }
```

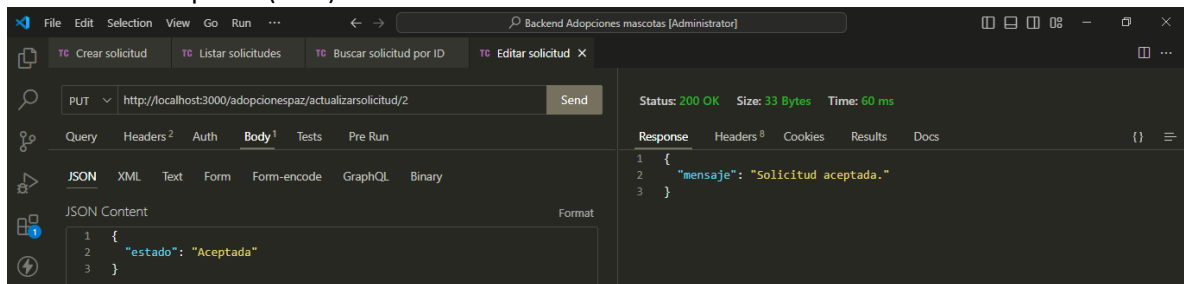
- Listar solicitudes (GET):



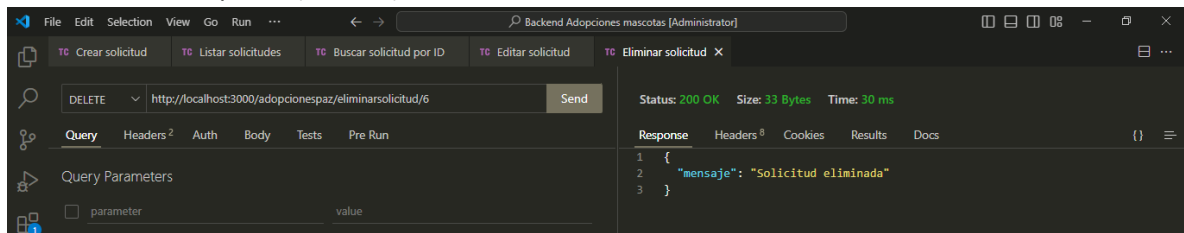
- Buscar solicitud por id (GET):



- Editar solicitud por id (PUT):



- Eliminar solicitud por id (DELETE):



- Después de los cambios realizados en la tabla solicitudes, se lo visualiza en DBeaver, cuando se acepta una solicitud para un animal quiere decir que el animal será adoptado por lo que las demás solicitudes de adopción para el mismo animal en automático se rechazan como se puede visualizar en el resultado final:

The screenshot shows the DBeaver interface with the 'solicitudes' table selected. The table contains the following data:

	idSolicitud	identidadSolicitante	fechaSolicitud	estado	createdAt	updatedAt	idMascota
1	1	1.004.236.301	2024-10-07 04:34:57	Rechazada	2024-10-07 04:34:57	2024-10-07 04:53:30	1
2	2	1.004.236.302	2024-10-07 04:36:43	Aceptada	2024-10-07 04:36:43	2024-10-07 04:53:30	1
3	3	1.004.236.303	2024-10-07 04:37:06	Rechazada	2024-10-07 04:37:06	2024-10-07 04:53:30	1
4	4	1.004.236.304	2024-10-07 04:37:41	Pendiente	2024-10-07 04:37:41	2024-10-07 04:53:30	2
5	5	1.004.236.305	2024-10-07 04:39:17	Pendiente	2024-10-07 04:39:17	2024-10-07 04:53:30	3
6	6	1.004.236.306	2024-10-07 04:39:43	Pendiente	2024-10-07 04:39:43	2024-10-07 04:53:30	3