Diplomado de actualización en nuevas tecnologías para el desarrollo de Software.

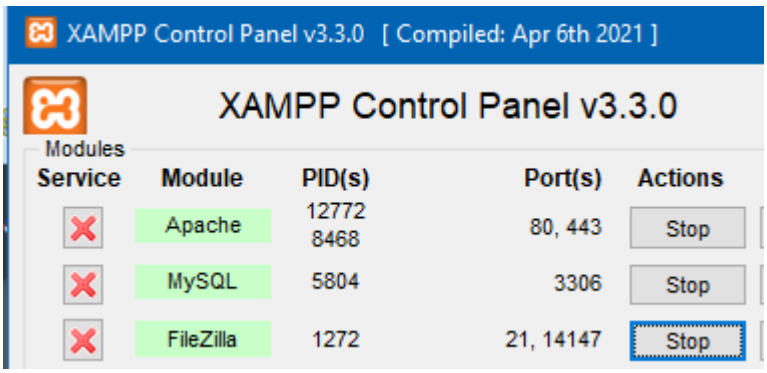
Tania Trejo – 220036029

Informe Taller Unidad 2 Backend

1. Crear una base de datos MYSQL/MARIADB que permita llevar el registro de una empresa de adopción de mascotas, debe soportar la administración de las mismas y la posibilidad de registrar solicitudes de adopción.

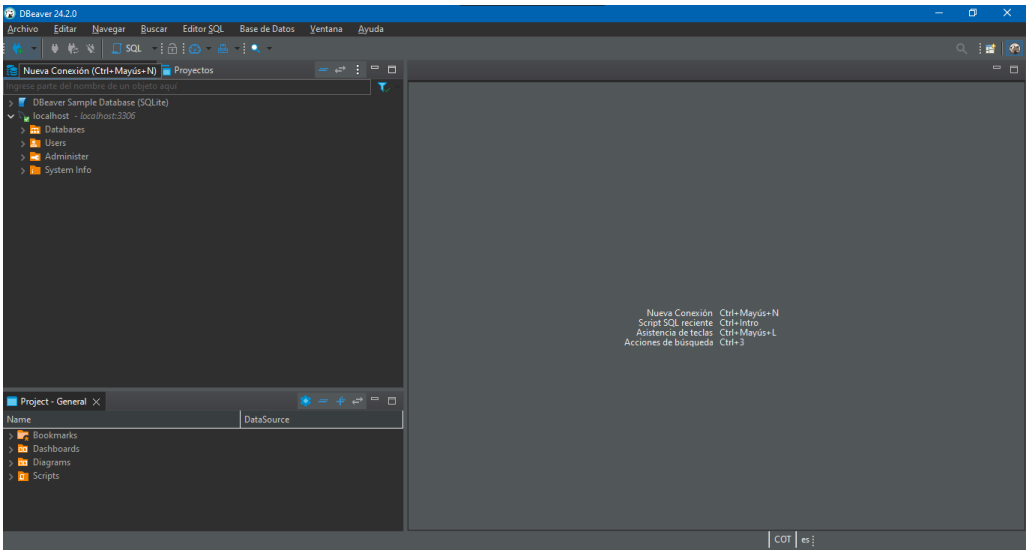
2. Inicializar XAMPP y levantar el servicio para la base de datos

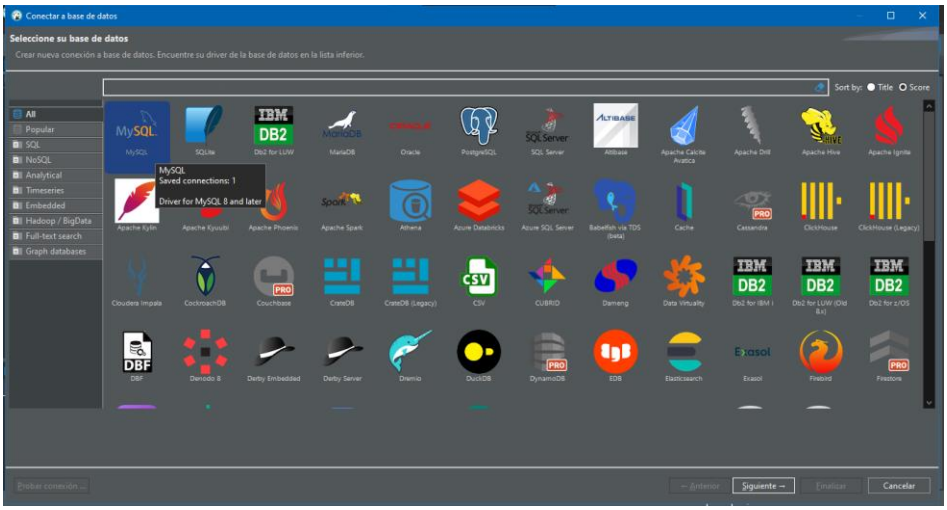
Inicia XAMPP



3. Conectar a la base de datos en DBeaver:

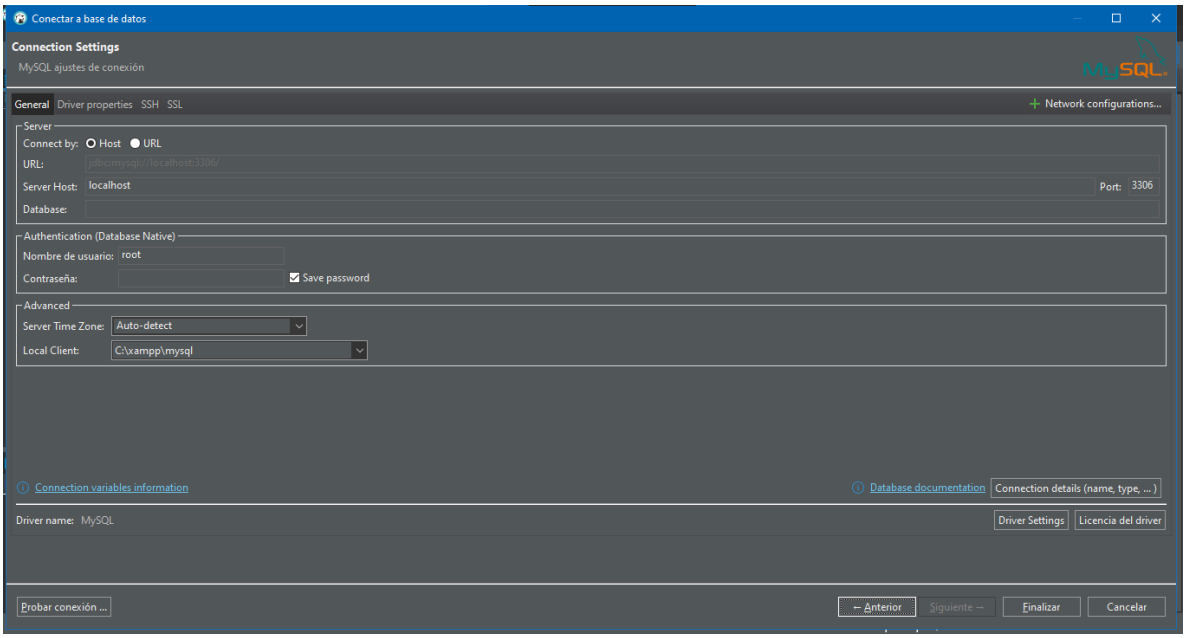
Abre DBeaver y crea una nueva conexión a MySQL.





En este caso se utilizó los siguientes parámetros:

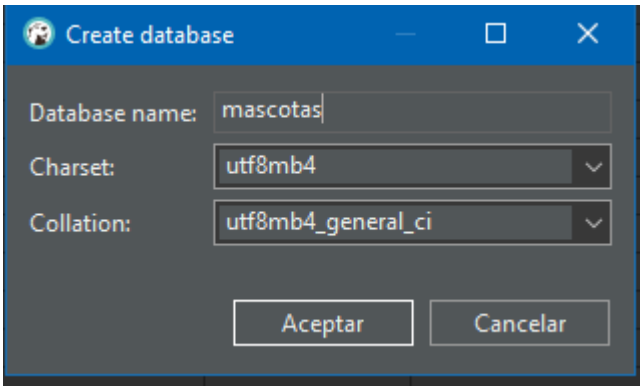
- Host: localhost
- Puerto: 3306 (por defecto)
- Usuario: root (o el usuario que hayas creado)
- Contraseña: (la contraseña que hayas configurado para el usuario root en XAMPP)



4. **Crear la base de datos:**

En el panel izquierdo de DBeaver, haz clic derecho sobre tu conexión de base de datos y selecciona Create -> Database.

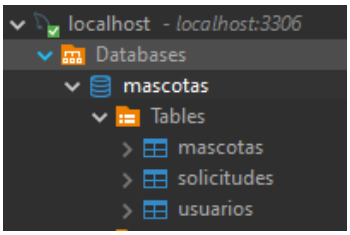
- Nombra la base de datos



5. **Crear las tablas:**

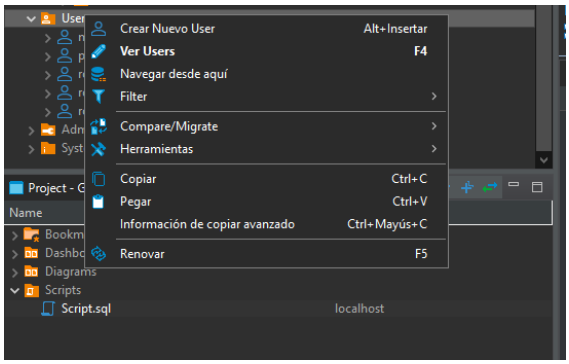
Haz clic derecho Nuevo Script y ejecutas.

```sql
CREATE TABLE usuarios (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombres VARCHAR(255) NOT NULL,
    telefono VARCHAR(15),
    usuario VARCHAR(100) NOT NULL UNIQUE,
    contraseña VARCHAR(255) NOT NULL,
    rol VARCHAR(50) NOT NULL,
    foto VARCHAR(255),
    edad INT
);

CREATE TABLE mascotas (
    id INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(255) NOT NULL,
    especie VARCHAR(100) NOT NULL,
    raza VARCHAR(100),
    edad INT,
    estado_adopcion ENUM('disponible', 'adoptado', 'en proceso') NOT NULL DEFAULT 'disponible',
    descripcion TEXT
);

CREATE TABLE solicitudes_adopcion (
    id INT PRIMARY KEY AUTO_INCREMENT,
    id_mascota INT,
    adoptante_id INT,
    estado_adopcion ENUM('disponible', 'adoptado', 'en proceso') NOT NULL DEFAULT 'en proceso',
    fecha_fin DATETIME,
    FOREIGN KEY (id_mascota) REFERENCES mascotas(id) ON DELETE CASCADE,
    FOREIGN KEY (adoptante_id) REFERENCES usuarios(id) ON DELETE CASCADE
);
```

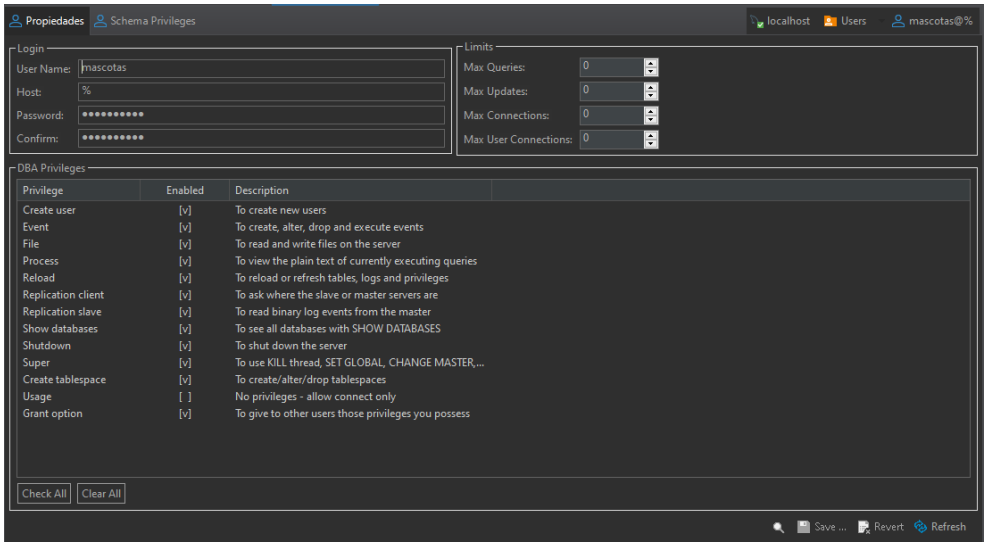


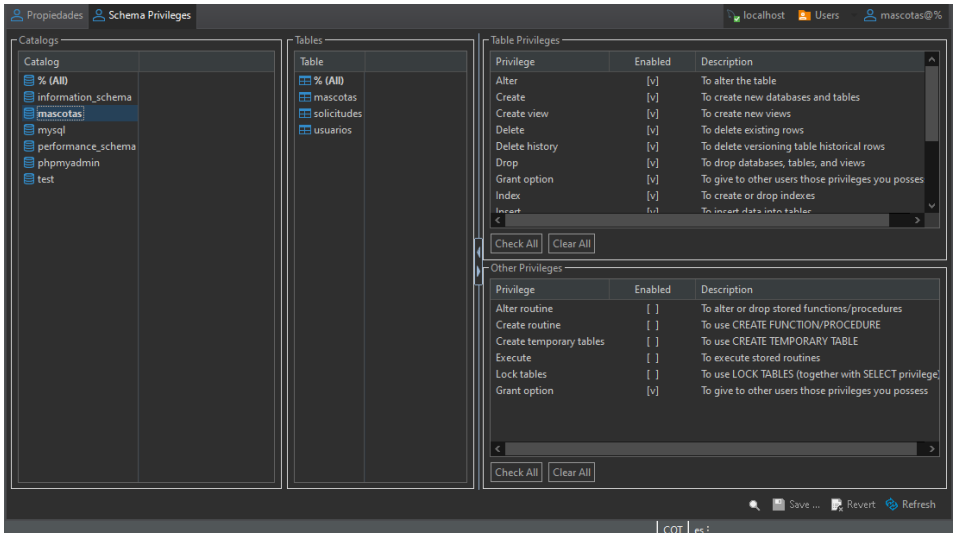
6. **Creación de usuario para acceder a base de datos**

Dar clic en user, luego en crear nuevo usuario.



Asignamos un nombre de usuario, contraseña, se selecciona los privilegios y se guarda.





7. **Verificar Versiones de Node.js y npm**

```
C:\Users\tania>node --version
v21.4.0

C:\Users\tania>npm --version
10.2.4
```

8. **Inicializa un nuevo proyecto Node.js**

Crear una carpeta y navegar hasta ella, luego ejecutar el comando.

```
C:\Users\tania\OneDrive\Documentos\Diplomado2024\rinconpeludo\backEnd>npm init -y
Wrote to C:\Users\tania\OneDrive\Documentos\Diplomado2024\rinconpeludo\backEnd\package.json:

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

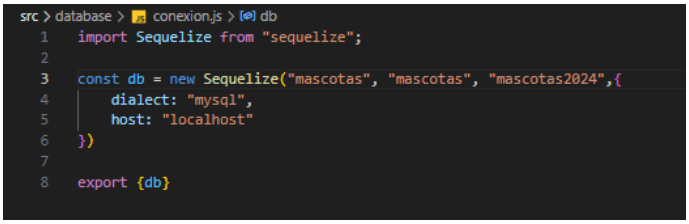
9. **Instala las dependencias necesarias.**
    - Instalar Nodemon para desarrollo
    - Instalar Express para Backend
    - Instalar soporte para MYSQL/MariaDB
    - Instalar Sequelize
    - Instalación de Cors

package.json

```
package.json > {} dependencies
{
  "name": "rinconpeludo",
  "version": "1.0.0",
  "description": "Backend para adopcion de mascotas",
  "type": "module",
  "main": "app.js",
  Debug
  "scripts": {
    "start": "nodemon ./src/app.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.1.4"
  },
  "dependencies": {
    "backend": "file:",
    "cors": "^2.8.5",
    "express": "^4.19.2",
    "mysql2": "^3.11.0",
    "sequelize": "^6.37.3"
  }
}
```

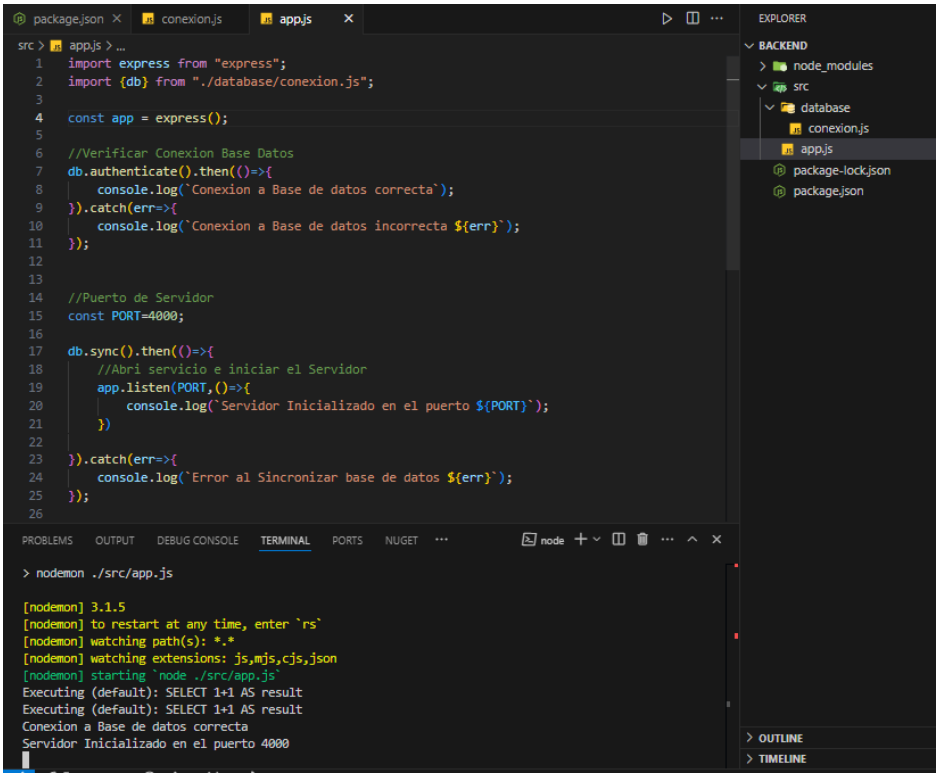
10. Crear una instancia de Sequelize

- En el proyecto, crea una carpeta llamada database dentro de tu carpeta src para efectos de orden.
- Dentro de la carpeta database, crea un archivo llamado conexion.js.
- Después configurar el archivo conexion.js para establecer la conexión con tu base de datos MySQL utilizando Sequelize.
    - Nombre de base de datos, usuario y contraseña



11. Configurar y ejecutar.

Crear un archivo app.js



12. Modelos

Crear modelo para la tabla mascotas

src > modelos > mascotaModelo.js > ...

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

// Definición del objeto que comunicará con la tabla
const mascotas = db.define('mascotas', {
    // Definición de los atributos
    id: {
        type: Sequelize.INTEGER,
        allowNull: false,
        primaryKey: true,
        autoIncrement: true
    },
    nombre: {
        type: Sequelize.STRING,
        allowNull: false
    },
    especie: {
        type: Sequelize.STRING,
        allowNull: false
    },
    raza: {
        type: Sequelize.STRING,
        allowNull: true
    },
    edad: {
        type: Sequelize.INTEGER,
        allowNull: true
    },
    estado_adopcion: {
        type: Sequelize.ENUM('disponible', 'adoptado', 'en proceso'),
        allowNull: false
    },
    descripcion: {
        type: Sequelize.TEXT,
        allowNull: true
    }
});

export { mascotas };
```

Crear modelo para la tabla usuarios

src > modelos > usuarioModelo.js > ...

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

// Definición de la variable que construirá la tabla
export const usuarios = db.define('usuarios', {
    // Definición de los atributos
    id: {
        type: Sequelize.INTEGER,
        allowNull: false,
        primaryKey: true,
        autoIncrement: true
    },
    nombres: {
        type: Sequelize.STRING,
        allowNull: false
    },
    telefono: {
        type: Sequelize.STRING,
        allowNull: true
    },
    usuario: {
        type: Sequelize.STRING,
        allowNull: false
    },
    contraseña: {
        type: Sequelize.STRING,
        allowNull: false
    },
    rol: {
        type: Sequelize.STRING,
        allowNull: false
    },
    foto: {
        type: Sequelize.STRING,
        allowNull: true
    },
    edad: {
        type: Sequelize.INTEGER,
        allowNull: true
    },
});

```

Crear modelo para la tabla solicitudes adopción

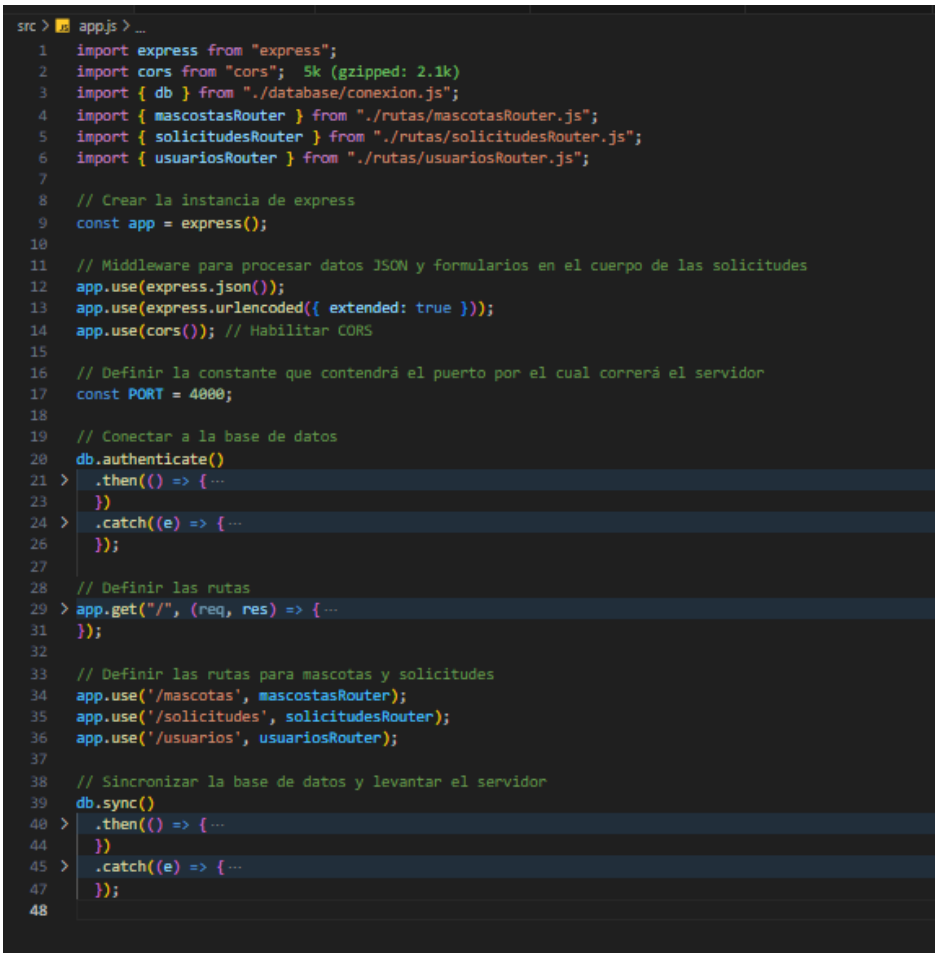
```
src > modelos > solicitudModelo.js > ...
1   import { Sequelize } from "sequelize";
2   import { db } from "../database/conexion.js";
3   import { mascotas } from "./mascotaModelo.js";
4   import { usuarios } from "./usuarioModelo.js";
5
6   // Definición del objeto que comunicará con la tabla
7   const solicitudes = db.define('solicitudes', {
8       // Definición de los atributos
9       id: {
10          type: Sequelize.INTEGER,
11          allowNull: false,
12          primaryKey: true,
13          autoIncrement: true
14      },
15      mascotaId: {
16          type: Sequelize.INTEGER,
17          allowNull: false
18      },
19      adoptanteId: {
20          type: Sequelize.INTEGER,
21          allowNull: false
22      },
23      estado_adopcion: {
24          type: Sequelize.ENUM('disponible', 'adoptado', 'en proceso'),
25          allowNull: false
26      },
27      fechaCreacion: {
28          type: Sequelize.DATE,
29          allowNull: false,
30          defaultValue: Sequelize.NOW
31      },
32      fechaFin: {
33          type: Sequelize.DATE,
34          allowNull: true
35      }
36  });
37
38  // Definir la relación de solicitud con mascota
39  solicitudes.belongsTo(mascotas, { foreignKey: 'mascotaId' });
40  // La relación con usuario
41  solicitudes.belongsTo(usuarios, { foreignKey: 'adoptanteId' });
42
43  export { solicitudes };
44
```

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 (La empresa debe contar con un nombre), así como también las solicitudes de adopción.

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

Rutas y Endpoints

Para interactuar con la aplicación, se han definido varias rutas.

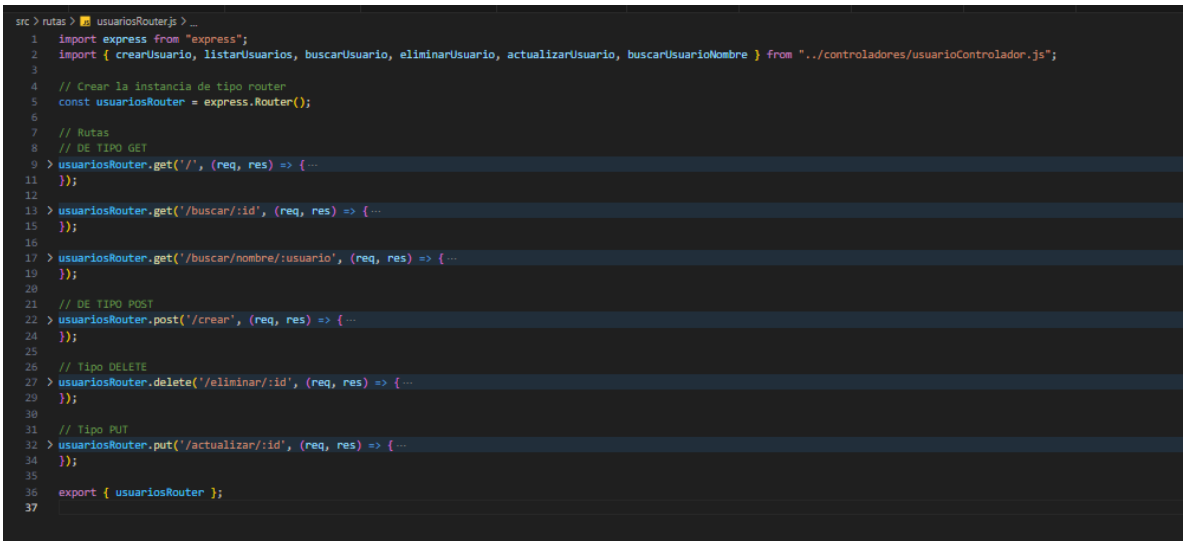


```
src > app.js > ...
import express from "express";
import cors from "cors";  5k (gzipped: 2.1k)
import { db } from "./database/conexion.js";
import { mascotasRouter } from "./rutas/mascotasRouter.js";
import { solicitudesRouter } from "./rutas/solicitudesRouter.js";
import { usuariosRouter } from "./rutas/usuariosRouter.js";

// Crear la instancia de express
const app = express();

// Middleware para procesar datos JSON y formularios en el cuerpo de las solicitudes
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cors()); // Habilitar CORS

// Definir la constante que contendrá el puerto por el cual correrá el servidor
const PORT = 4000;

// Conectar a la base de datos
db.authenticate()
  .then(() => {···
  })
  .catch((e) => {···
  });

// Definir las rutas
app.get("/", (req, res) => {···
});

// Definir las rutas para mascotas y solicitudes
app.use('/mascotas', mascotasRouter);
app.use('/solicitudes', solicitudesRouter);
app.use('/usuarios', usuariosRouter);

// Sincronizar la base de datos y levantar el servidor
db.sync()
  .then(() => {···
  })
  .catch((e) => {···
  });

```

Rutas de Mascotas

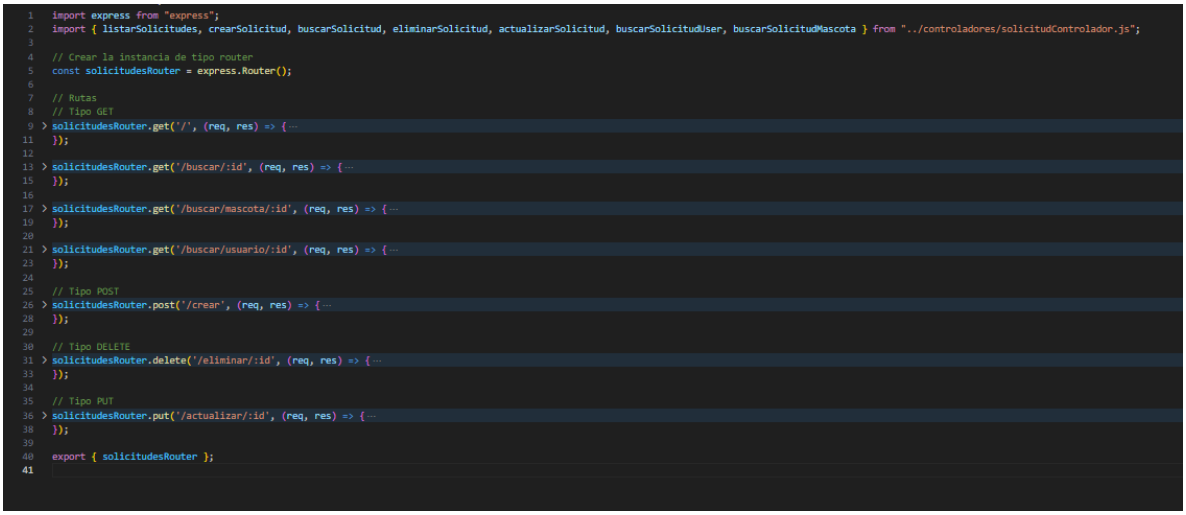
```
src > rutas > mascotasRouter.js > ...
import express from "express";
import { buscarMascota, crearMascota, listarMascotas, eliminarMascota, actualizarMascota } from "../controladores/mascotaControlador.js";

// Crear la instancia de tipo router
const mascotasRouter = express.Router();

// Rutas
// DE TIPO GET
mascotasRouter.get('/', (req, res) => {···
});

mascotasRouter.get('/buscar/:id', (req, res) => {···
});

// DE TIPO POST
mascotasRouter.post('/crear', (req, res) => {···
});

// Tipo DELETE
mascotasRouter.delete('/eliminar/:id', (req, res) => {···
});

// Tipo PUT
mascotasRouter.put('/actualizar/:id', (req, res) => {···
});

export { mascotasRouter };

```

Rutas de Usuarios



```
src > rutas > usuariosRouter.js > ...
import express from "express";
import { crearUsuario, listarUsuarios, buscarUsuario, eliminarUsuario, actualizarUsuario, buscarUsuarioNombre } from "../controladores/usuarioControlador.js";

// Crear la instancia de tipo router
const usuariosRouter = express.Router();

// Rutas
// DE TIPO GET
usuariosRouter.get('/', (req, res) => {···
});

usuariosRouter.get('/buscar/:id', (req, res) => {···
});

usuariosRouter.get('/buscar/nombre/:usuario', (req, res) => {···
});

// DE TIPO POST
usuariosRouter.post('/crear', (req, res) => {···
});

// Tipo DELETE
usuariosRouter.delete('/eliminar/:id', (req, res) => {···
});

// Tipo PUT
usuariosRouter.put('/actualizar/:id', (req, res) => {···
});

export { usuariosRouter };
```

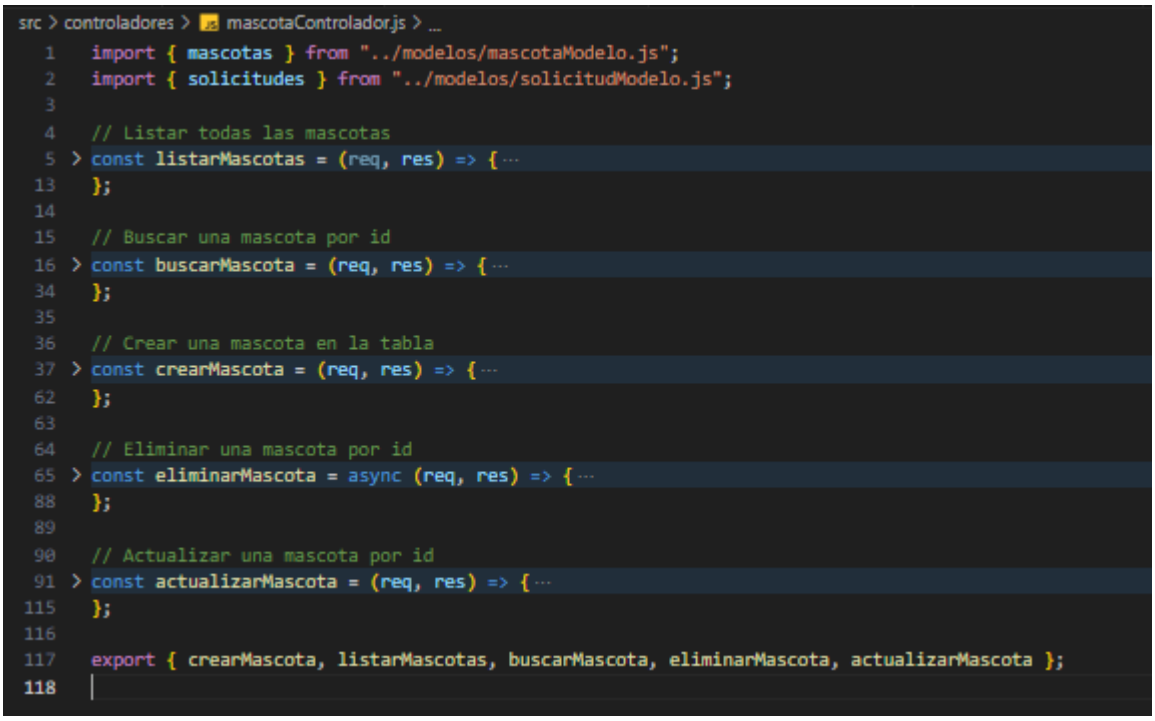
Rutas de Solicitudes



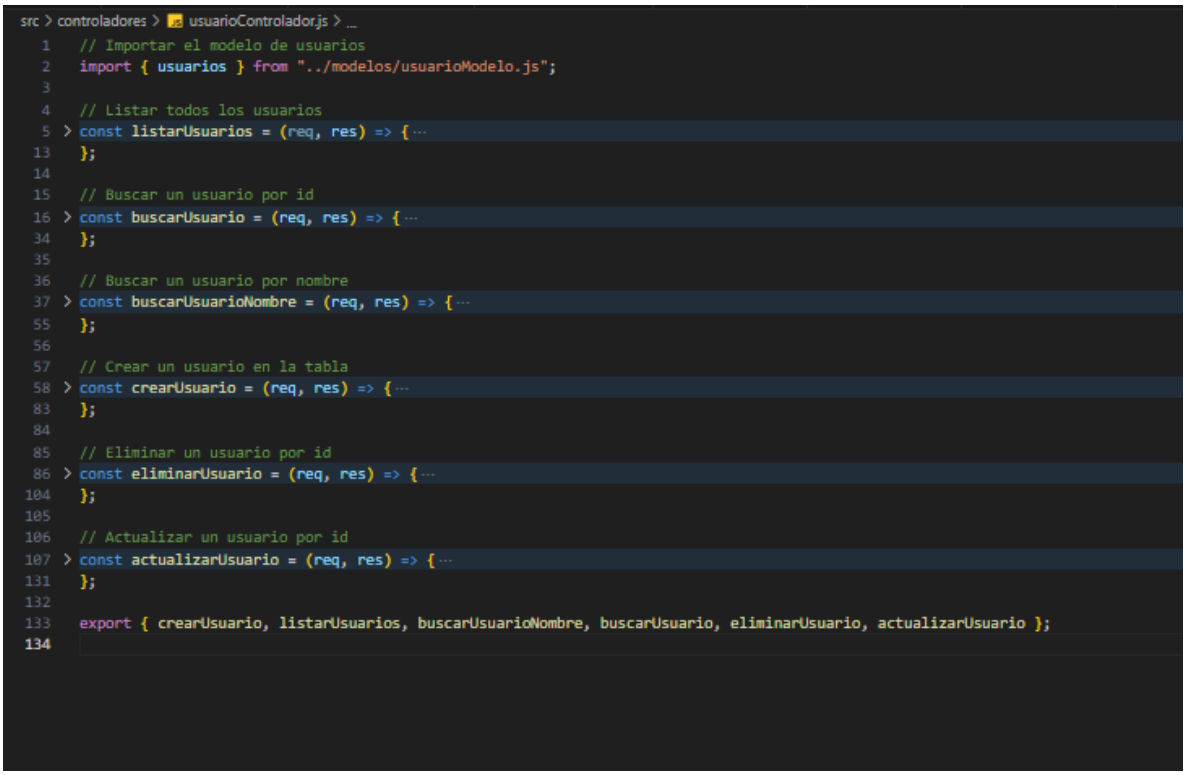
```
import express from "express";
import { listarSolicitudes, crearSolicitud, buscarSolicitud, eliminarSolicitud, actualizarSolicitud, buscarSolicitudUser, buscarSolicitudMascota } from "../controladores/solicitudControlador.js";

// Crear la instancia de tipo router
const solicitudesRouter = express.Router();

// Rutas
// Tipo GET
solicitudesRouter.get('/', (req, res) => {···
});

solicitudesRouter.get('/buscar/:id', (req, res) => {···
});

solicitudesRouter.get('/buscar/mascota/:id', (req, res) => {···
});

solicitudesRouter.get('/buscar/usuario/:id', (req, res) => {···
});

// Tipo POST
solicitudesRouter.post('/crear', (req, res) => {···
});

// Tipo DELETE
solicitudesRouter.delete('/eliminar/:id', (req, res) => {···
});

// Tipo PUT
solicitudesRouter.put('/actualizar/:id', (req, res) => {···
});

export { solicitudesRouter };
```

Controladores

Controladores de Mascotas



Controladores de Usuarios



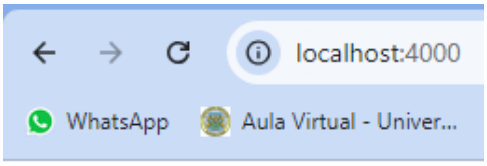
Controladores de Solicitud

src > controladores > solicitudControlador.js > crearSolicitud

```
// Importamos el modelo de solicitudes
import { solicitudes } from "../modelos/solicitudModelo.js";

// Listar todas las solicitudes
const listarSolicitudes = (req, res) => {···
};

// Buscar solicitud por id
const buscarSolicitud = (req, res) => {···
};

// Buscar solicitudes de un usuario
const buscarSolicitudUser = (req, res) => {···
};

// Buscar solicitudes de una mascota
const buscarSolicitudMascota = (req, res) => {···
};

// Eliminar solicitud por id
const eliminarSolicitud = (req, res) => {···
};

// Crear una solicitud
const crearSolicitud = (req, res) => {···
};

// Actualizar solicitud por id
const actualizarSolicitud = (req, res) => {···
};

export { listarSolicitudes, crearSolicitud, buscarSolicitudMascota, buscarSolicitudUser, buscarSolicitud, eliminarSolicitud, actualizarSolicitud };
```

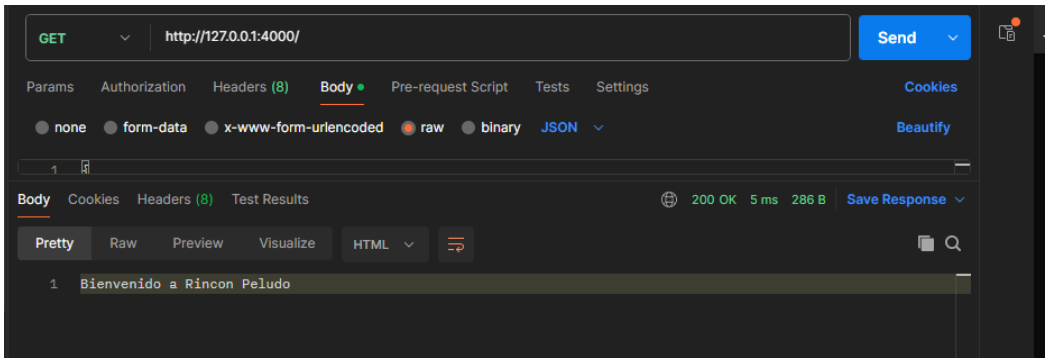
Dar un nombre de bienvenido para la empresa

```
// Definir las rutas
app.get("/", (req, res) => {
  res.send("Bienvenido a Rincon Peludo");
});
```

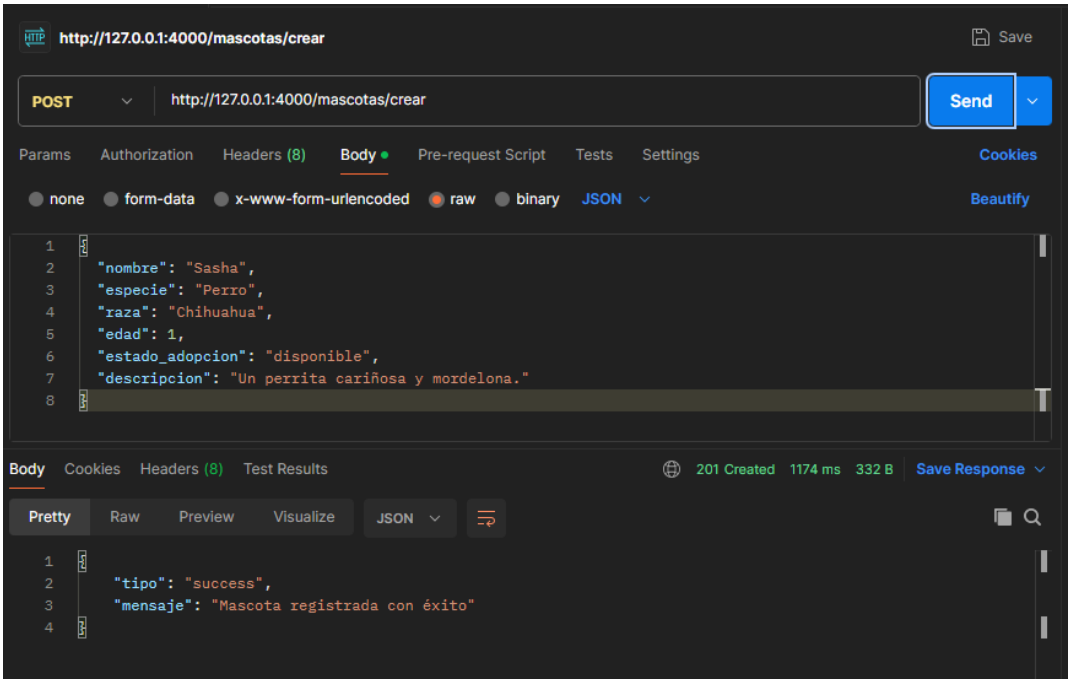


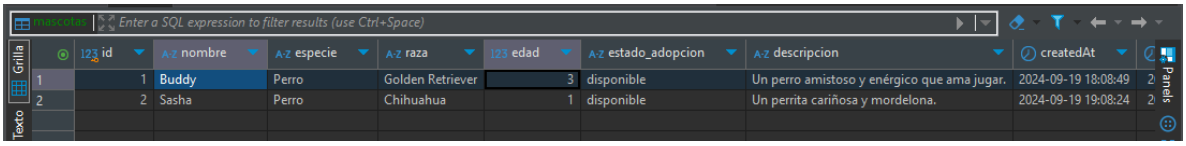
Bienvenido a Rincon Peludo

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.

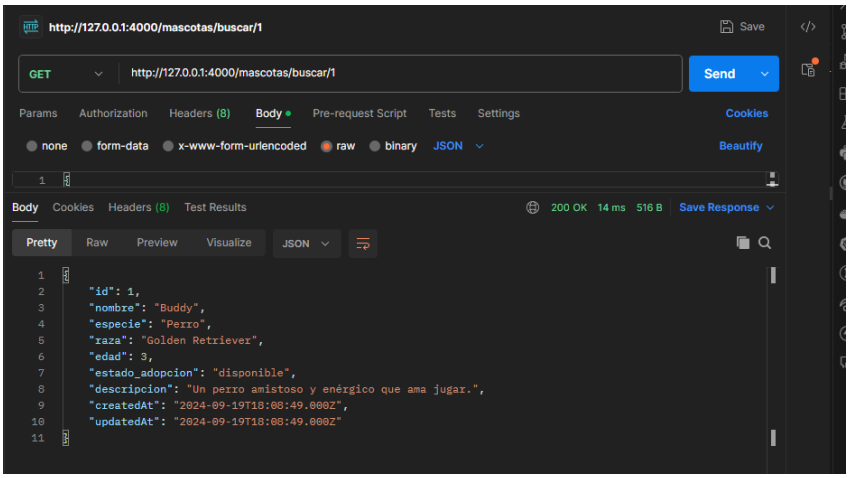


Mascotas: Crear

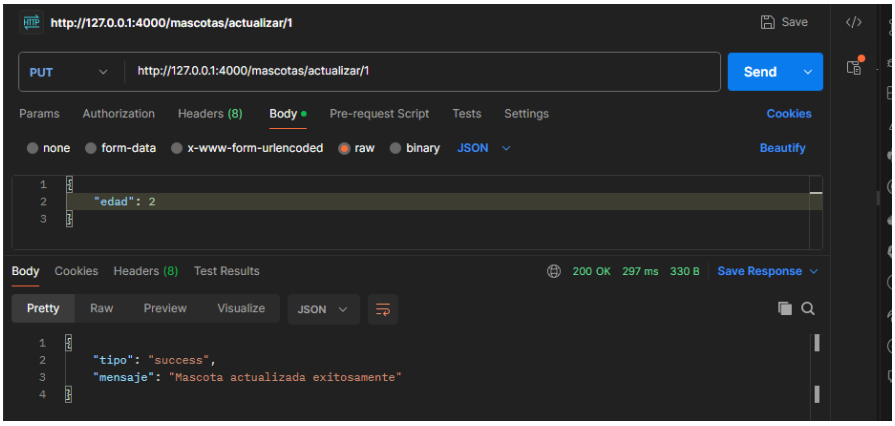




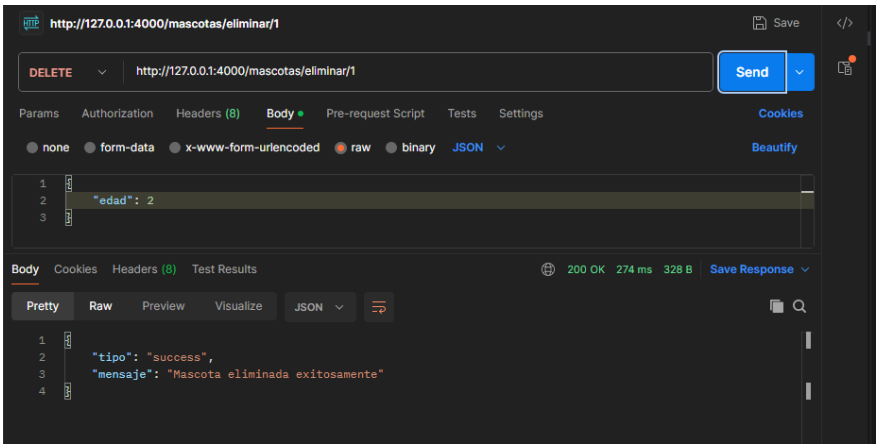
Mascotas: Buscar por id



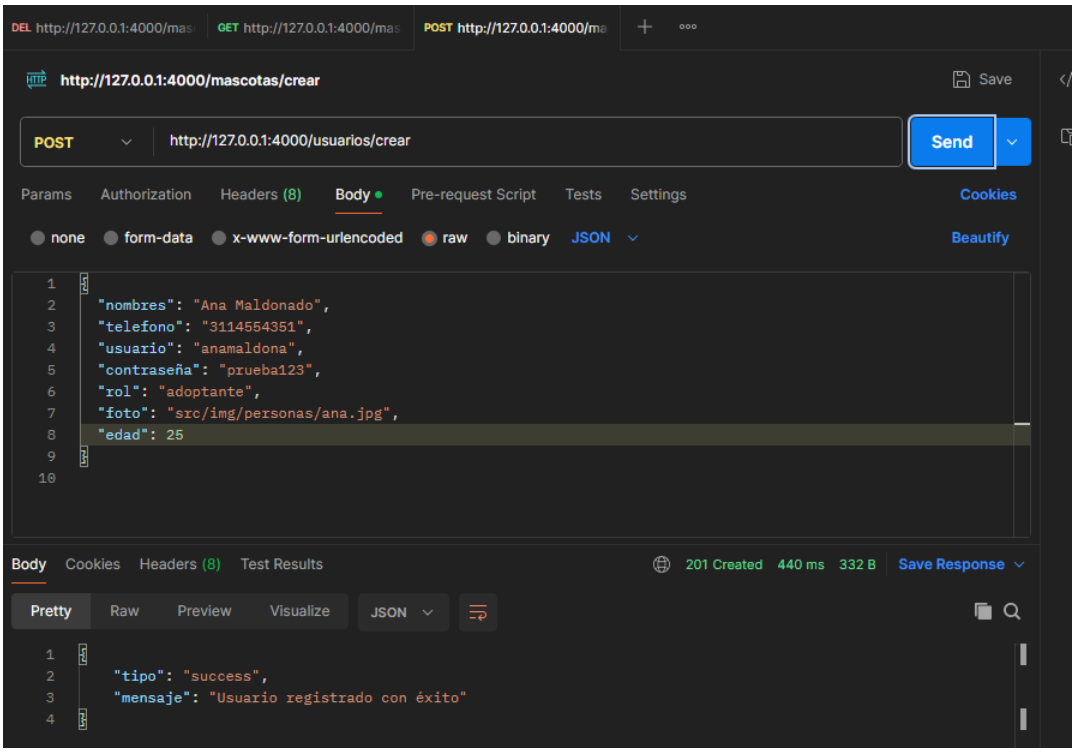
Mascotas: Actualizar

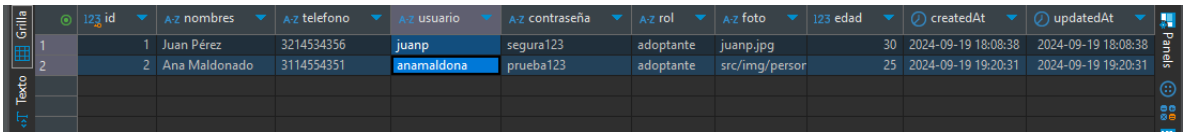


Mascotas: Eliminar

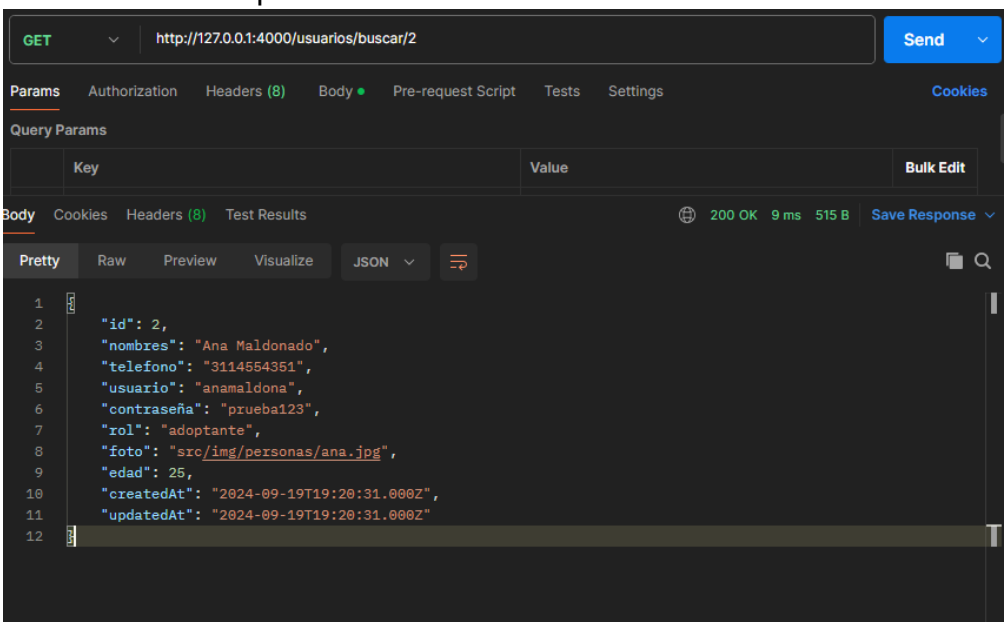


Usuarios: Crear

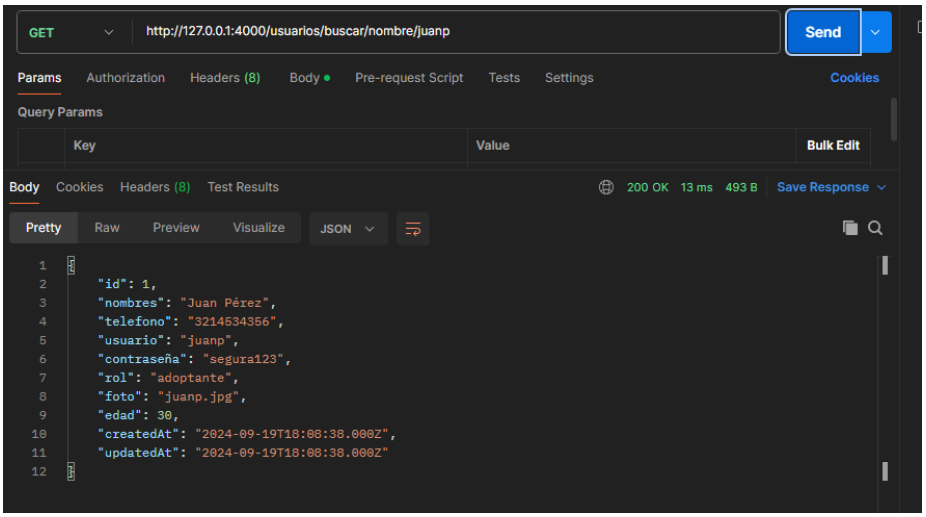




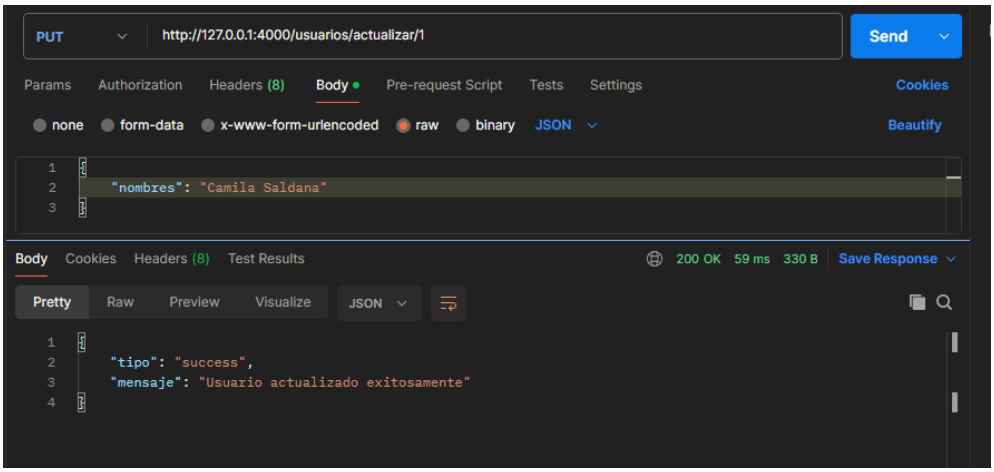
Usuarios: Buscar por id



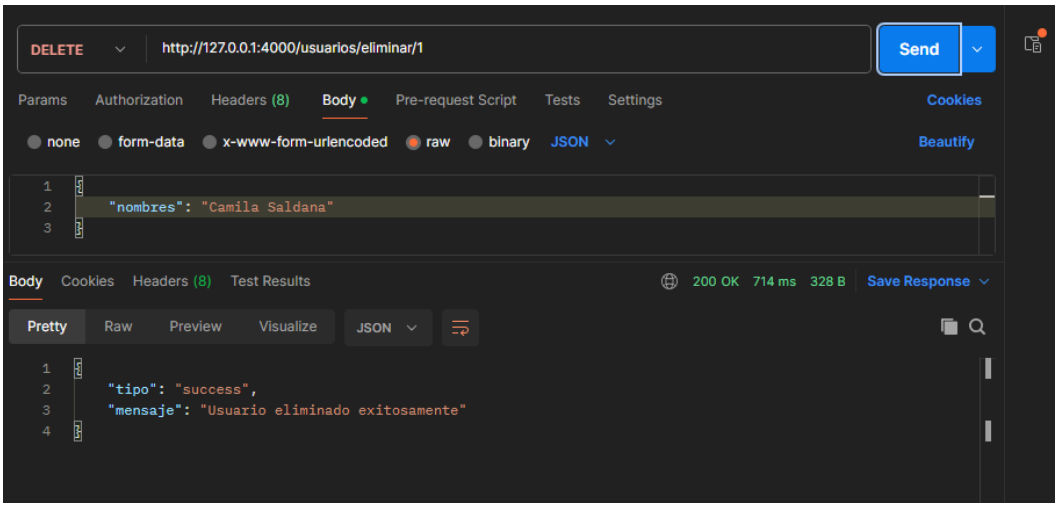
Usuarios: Buscar por usuario



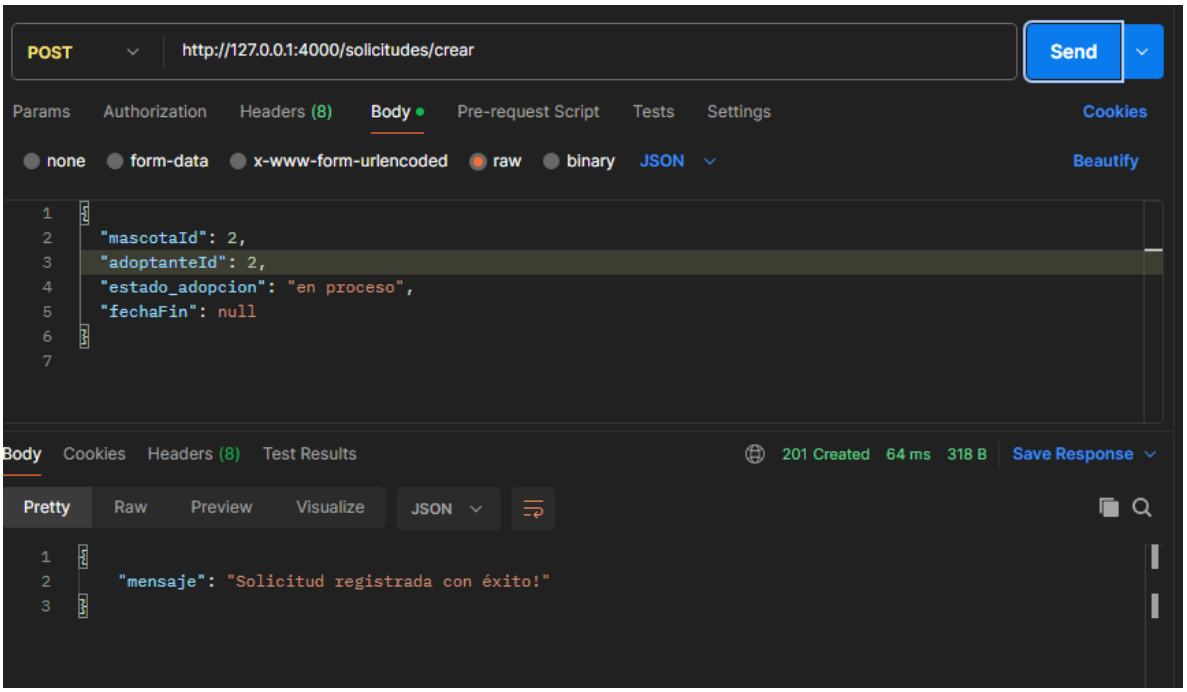
Usuario: Actualizar

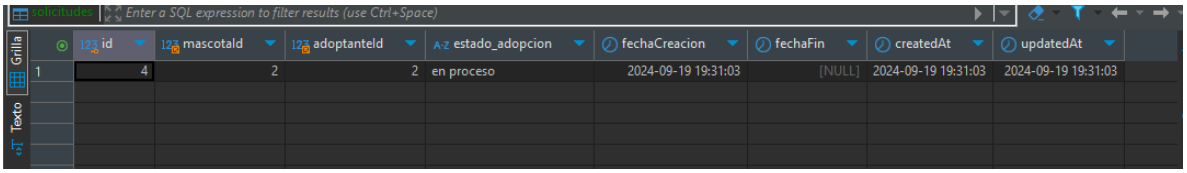


Usuarios: Eliminar

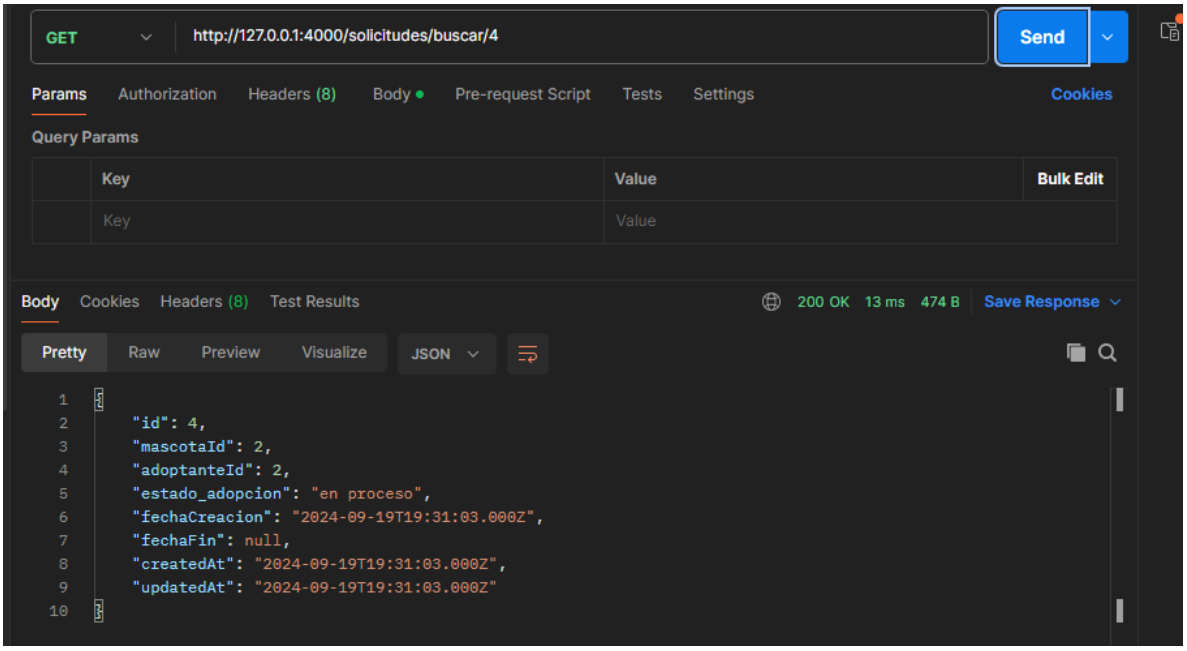


Solicitudes: Crear

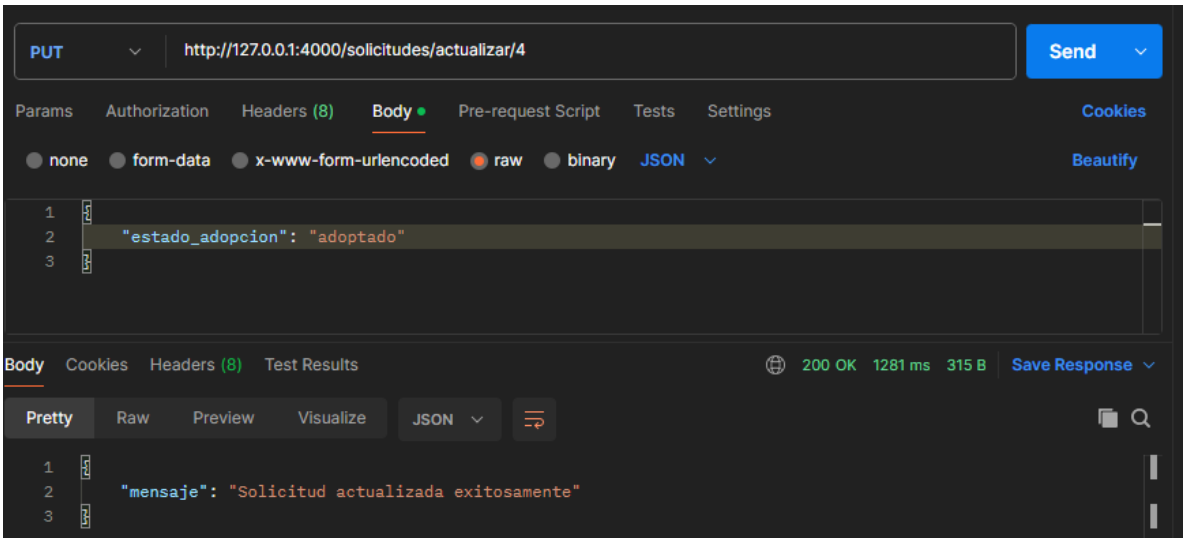




Solicitudes: Buscar por id



Solicitudes: Actualizar



Solicitudes: Eliminar