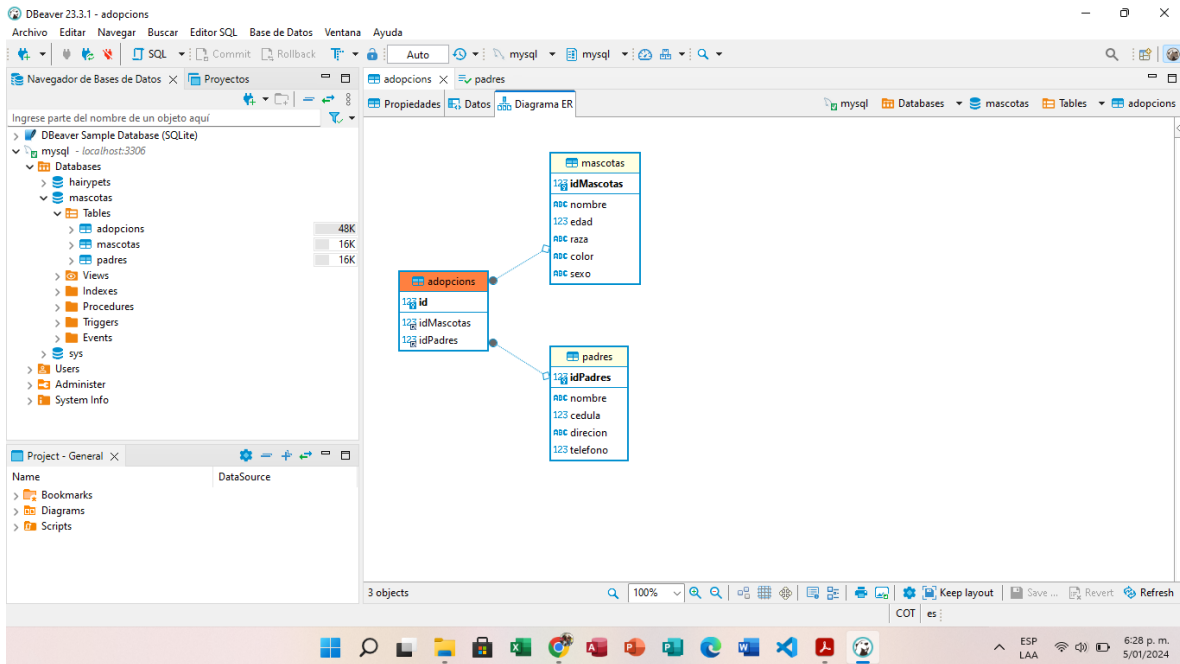


## TALLER BACKEND

Iniciamos creando la base de datos mascotas con sus respectivas tablas mascotas, padres y adopción en la siguiente imagen mostramos en el entono grafico las respectivas tablas

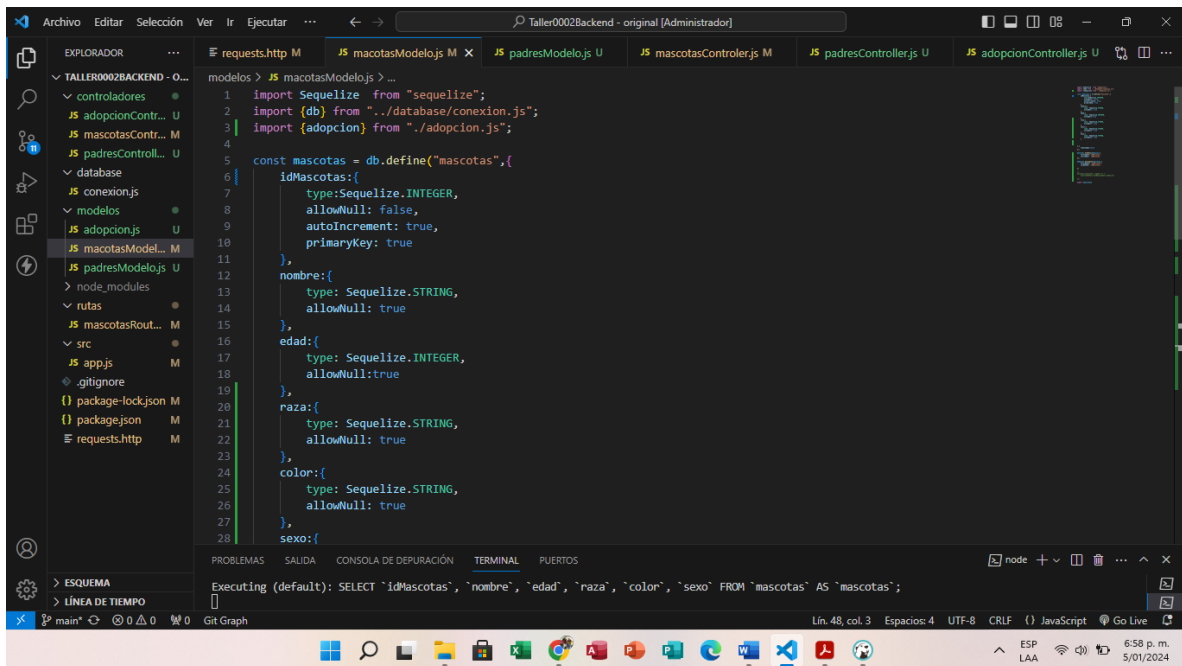


The screenshot shows a code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure, including 'TALLER0002BACKEND - O...', 'controladores', 'modelos', 'database', 'node\_modules', 'rutas', 'src', and 'package-lock.json'. The code editor shows the implementation of the 'adopcion' model in JavaScript. The code is as follows:

```
1 import Sequelize from "sequelize";
2 import {db} from "../database/conexion.js";
3
4 const adopcion = db.define("adopcion", {
5   id: {
6     type: Sequelize.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11 }, {
12   timestamps: false
13 });
14
15 export {adopcion}
```

En este apartado se hizo la instalación de todas las dependencias y que requiere el proyecto se crearon con ellos tres modelos y tres controladores los modelos estos son archivos javascript los se encargan de la iteración con la tabla de datos la cual corresponden a cada tabla de esta misma los

cuales son mascotaModelo.js el cual es el encargado de la iteración con la tabla mascotas y la asociación con la tabla adopción de uno a muchos



The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying the project structure. The main editor window shows the code for `mascotasModelo.js`. The code imports `Sequelize` and `db` from `../database/conexion.js`, and `adopcion` from `../adopcion.js`. It then defines the `mascotas` model with attributes: `idMascotas` (primary key, integer, auto-increment), `nombre` (string), `edad` (integer), `raza` (string), `color` (string), and `sexo` (string). The terminal at the bottom shows the command `node` being executed, and the output indicates that the server is initialized on port 8080.

```
models > JS mascotasModelo.js > ...
1 import Sequelize from "sequelize";
2 import {db} from "../database/conexion.js";
3 import {adopcion} from "../adopcion.js";
4
5 const mascotas = db.define("mascotas",{
6   idMascotas:{
7     type:Sequelize.INTEGER,
8     allowNull: false,
9     autoIncrement: true,
10    primaryKey: true
11  },
12  nombre:{
13    type: Sequelize.STRING,
14    allowNull: true
15  },
16  edad:{
17    type: Sequelize.INTEGER,
18    allowNull:true
19  },
20  raza:{
21    type: Sequelize.STRING,
22    allowNull: true
23  },
24  color:{
25    type: Sequelize.STRING,
26    allowNull: true
27  },
28  sexo:{
29    type: Sequelize.STRING,
30    allowNull: true
31  }
32 },{
33   timestamps:false
34 });
35
36 mascotas.hasMany(adopcion,{
37   foreignKey:'idMascotas',
38   sourceKey: 'idMascotas'
39 });
40
41 adopcion.belongsTo(mascotas,{
42   foreignKey: "idMascotas",
43   targetKey: "idMascotas",
44 });
45
46 export {mascotas}
```

padresModelos.js el cual es el encargado de la iteración con la tabla padres y la asociación con la tabla adopción de uno a muchos

This screenshot shows the 'padresModelo.js' file in the 'modelos' directory. The code defines a Sequelize model named 'padres' with the following attributes:

```
1 import Sequelize from "sequelize";
2 import {db} from "../database/conexion.js";
3 import {adopcion} from "../adopcion.js";
4
5 const padres = db.define("padres",{
6   idPadres:{
7     type:Sequelize.INTEGER,
8     allowNull: false,
9     autoIncrement: true,
10    primaryKey: true
11  },
12  nombre:{
13    type: Sequelize.STRING,
14    allowNull: true
15  },
16  cedula:{
17    type: Sequelize.INTEGER,
18    allowNull: true
19  },
20  direccion:{
21    type: Sequelize.STRING,
22    allowNull: true
23  },
24  telefono:{
25    type: Sequelize.INTEGER,
26    allowNull: true
27  }
28 },{
29   timestamps: false
30 });
31
32 padres.hasMany(adopcion,{
33   foreignKey: 'idPadres',
34   sourceKey: 'idPadres'
35 });
36
37 adopcion.belongsTo(padres,{
38   foreignKey: "idPadres",
39   targetKey: "idPadres"
40 });
41
42 export {padres}
```

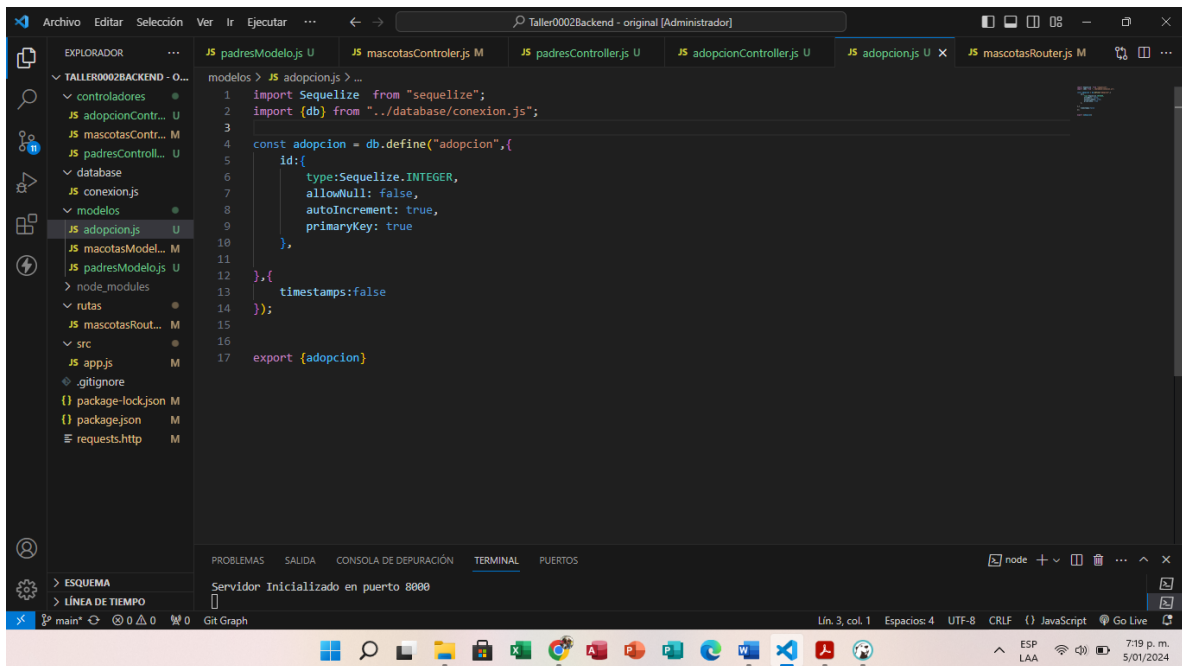
The terminal at the bottom shows 'Servidor Inicializado en puerto 8000'.

This screenshot shows the 'adopcion.js' file in the 'modelos' directory. The code defines a Sequelize model named 'adopcion' with the following attributes:

```
1 import Sequelize from "sequelize";
2 import {db} from "../database/conexion.js";
3 import {padres} from "../padresModelo.js";
4
5 const adopcion = db.define("adopcion",{
6   idAdopcion:{
7     type:Sequelize.INTEGER,
8     allowNull: false,
9     autoIncrement: true,
10    primaryKey: true
11  },
12  nombreMascota:{
13    type: Sequelize.STRING,
14    allowNull: true
15  },
16  telefonoMascota:{
17    type: Sequelize.INTEGER,
18    allowNull: true
19  },
20  direccionMascota:{
21    type: Sequelize.STRING,
22    allowNull: true
23  },
24  telefonoMascota:{
25    type: Sequelize.INTEGER,
26    allowNull: true
27  }
28 },{
29   timestamps: false
30 });
31
32 adopcion.belongsTo(padres,{
33   foreignKey: "idPadres",
34   targetKey: "idPadres"
35 });
36
37 export {adopcion}
```

The terminal at the bottom shows 'Servidor Inicializado en puerto 8000'.

Adopción.js esta sirve como tabla intermedia para la iteración de la tabla padres y mascotas asociándose por medio de dos llaves foráneas



```
1 import Sequelize from "sequelize";
2 import {db} from "../database/conexion.js";
3
4 const adopcion = db.define("adopcion", {
5   id: {
6     type: Sequelize.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11 }, {
12   timestamps: false
13 });
14
15 export {adopcion}
```

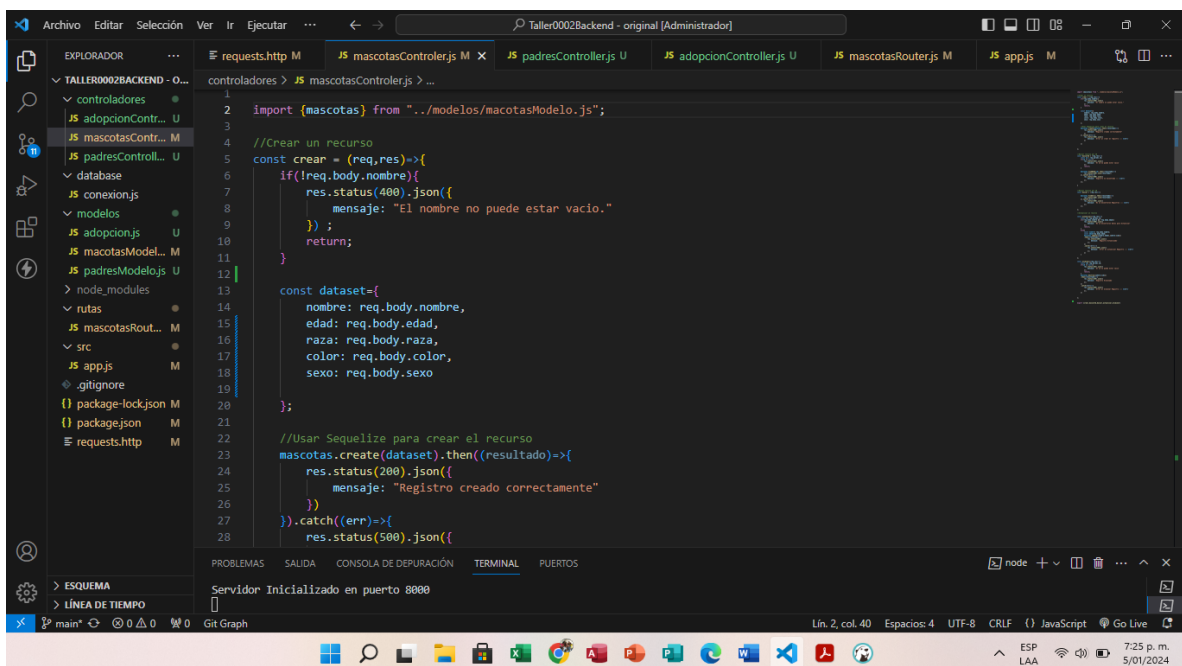
PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

node + v ...

Server Inicializado en puerto 8000

En las figuras siguientes se muestra los controladores los cuales se encargan de la programación funcional de la aplicación los cuales contamos con:

mascotaController.js



```
1 import {mascotas} from "../modelos/mascotasModelo.js";
2
3 //Crear un recurso
4 const crear = (req, res) => {
5   if(!req.body.nombre){
6     res.status(400).json({
7       mensaje: "El nombre no puede estar vacio."
8     });
9     return;
10   }
11
12   const dataset = {
13     nombre: req.body.nombre,
14     edad: req.body.edad,
15     raza: req.body.raza,
16     color: req.body.color,
17     sexo: req.body.sexo
18   };
19
20   //Usar Sequelize para crear el recurso
21   mascotas.create(dataset).then((resultado) => {
22     res.status(200).json({
23       mensaje: "Registro creado correctamente"
24     });
25   }).catch((err) => {
26     res.status(500).json({
27
28
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL PUERTOS

node + v ...

Server Inicializado en puerto 8000

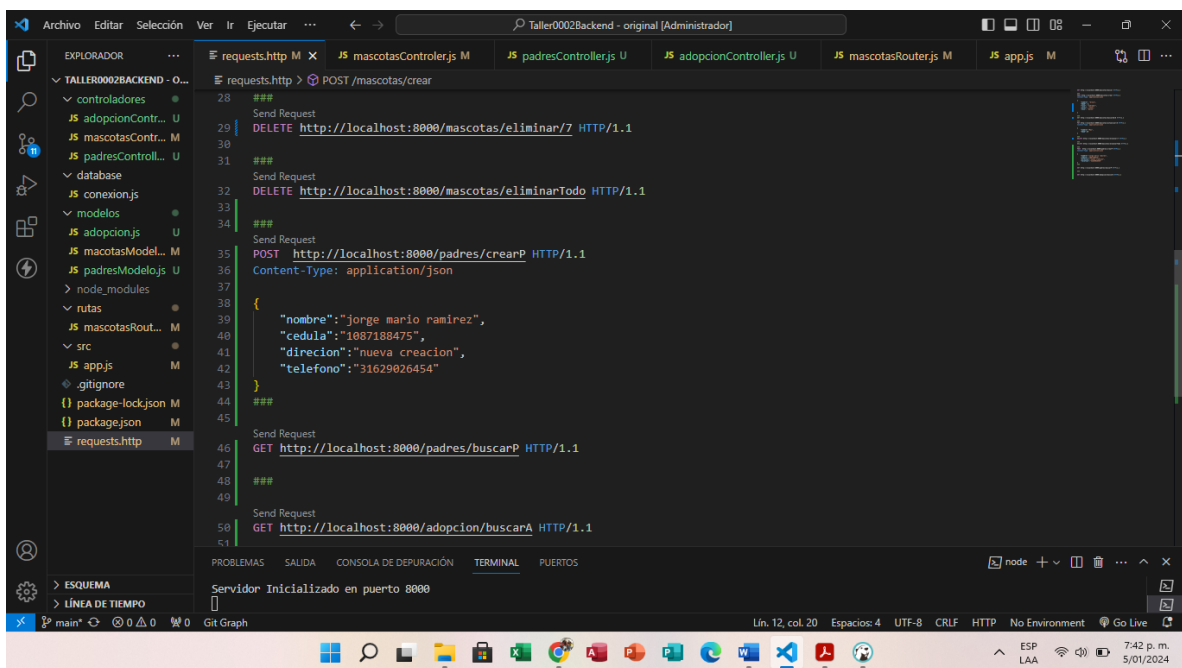
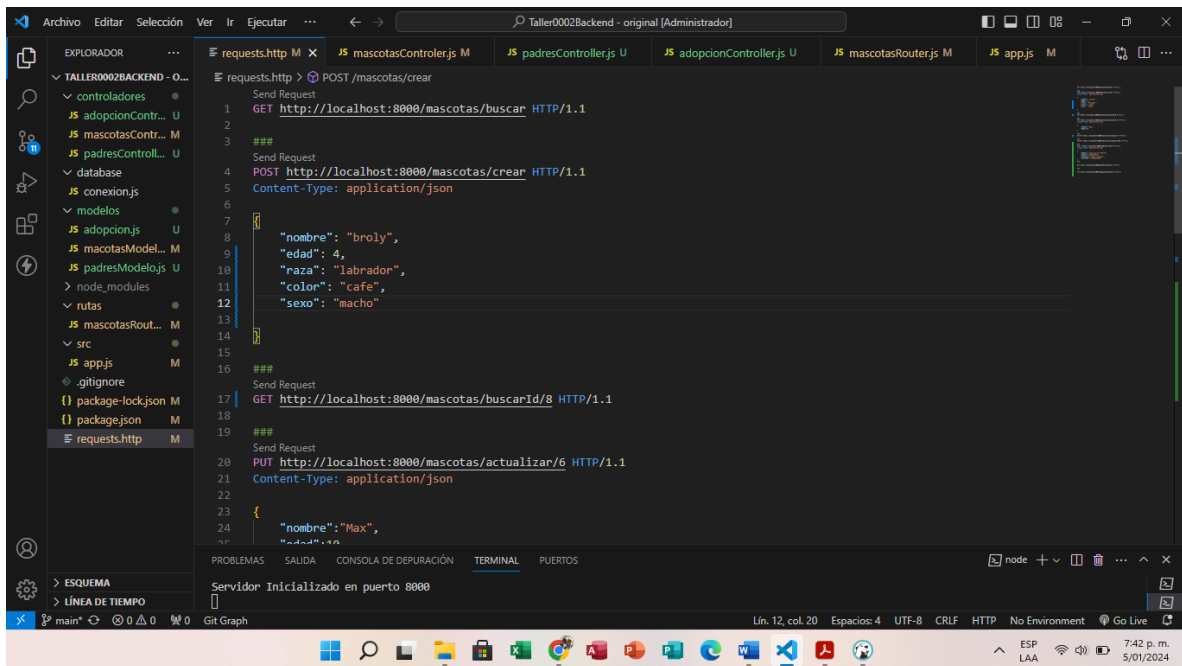
padresController.js

```
1 import { padres } from "../modelos/padresModelo.js";
2
3
4 const crearP = (req,res)=>{
5   if(!req.body.nombre){
6     res.status(400).json({
7       mensaje: "El nombre no puede estar vacio."
8     });
9     return;
10  }
11  const dataset={
12    nombre: req.body.nombre,
13    cedula: req.body.cedula,
14    direccion: req.body.direccion,
15    telefono: req.body.telefono
16  };
17
18  //Usar Sequelize para crear el recurso
19  padres.create(dataset).then((resultado)=>{
20    res.status(200).json({
21      mensaje: "Registro creado correctamente"
22    });
23  }).catch((err)=>{
24    res.status(500).json({
25      mensaje: `Error al crear el registro ::: ${err}`
26    });
27  })
28 }
```

## adopcionController.js

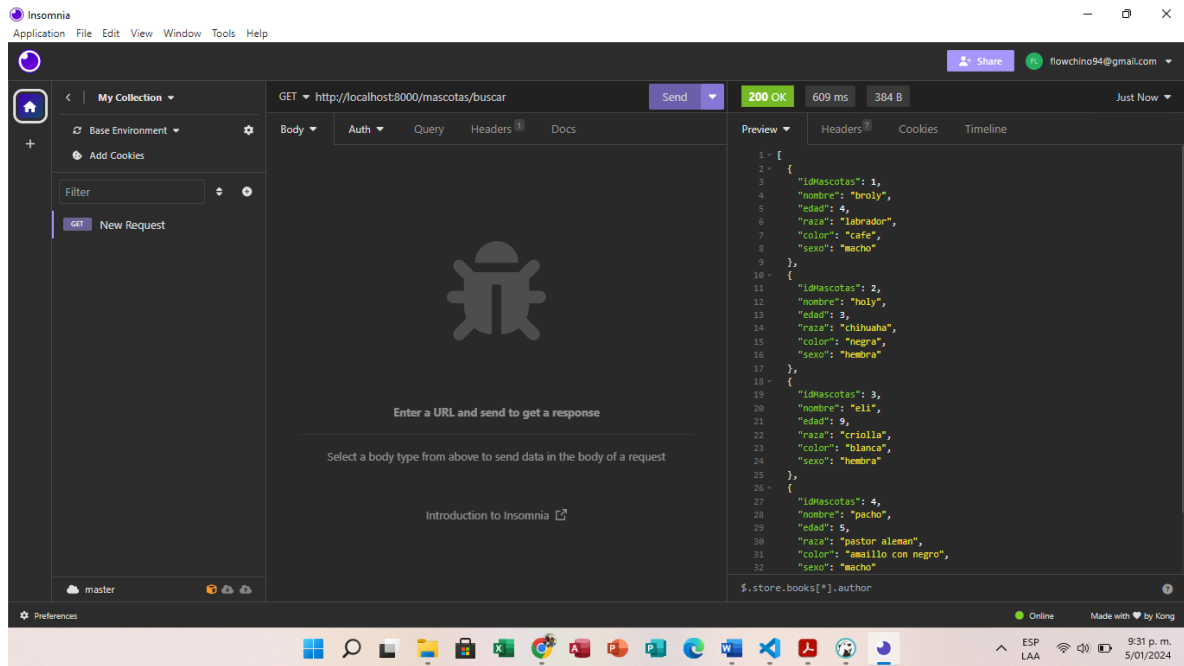
```
1 import { adopcion } from "../modelos/adopcion.js";
2
3 const buscarA = (req,res)=>{
4   try {
5     adopcion.findAll({ include: { all: true, nested: true } })
6       .then((resultado)=>{
7         res.status(200).json(resultado);
8       });
9   } catch (error) {
10     // Handle error
11   }
12 };
13
14 export { buscarA }
```

En este apartado podemos ver la el uso de los verbos http

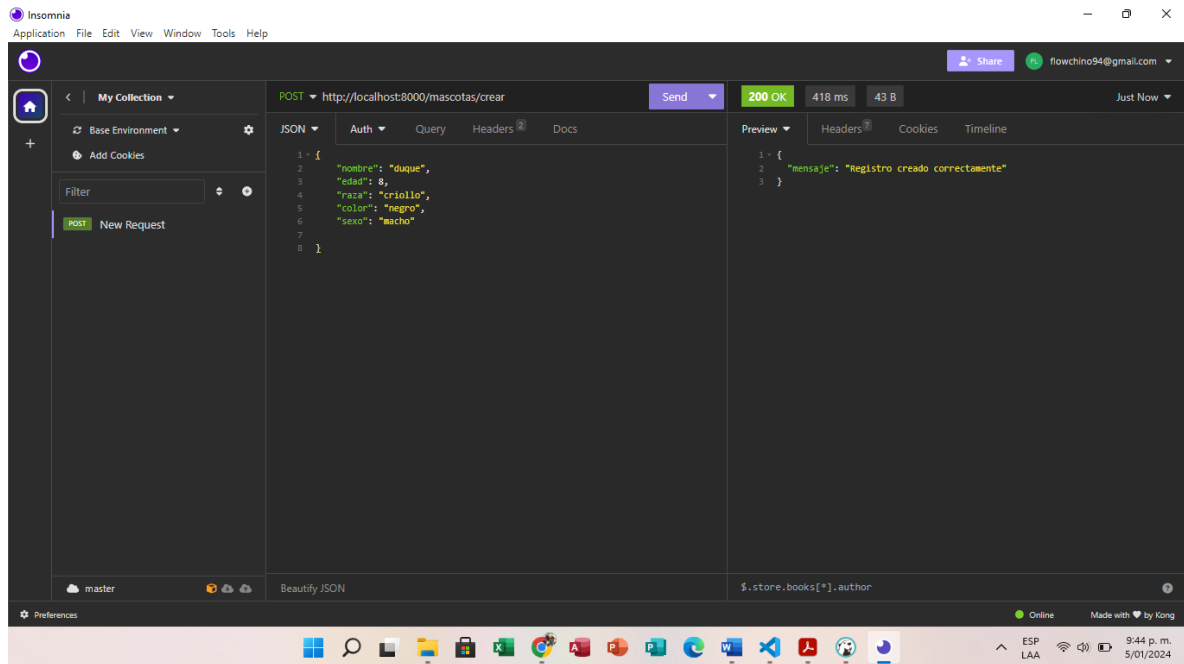


En esta mostramos la verificamos las operaciones usando el cliente grafico insomnia

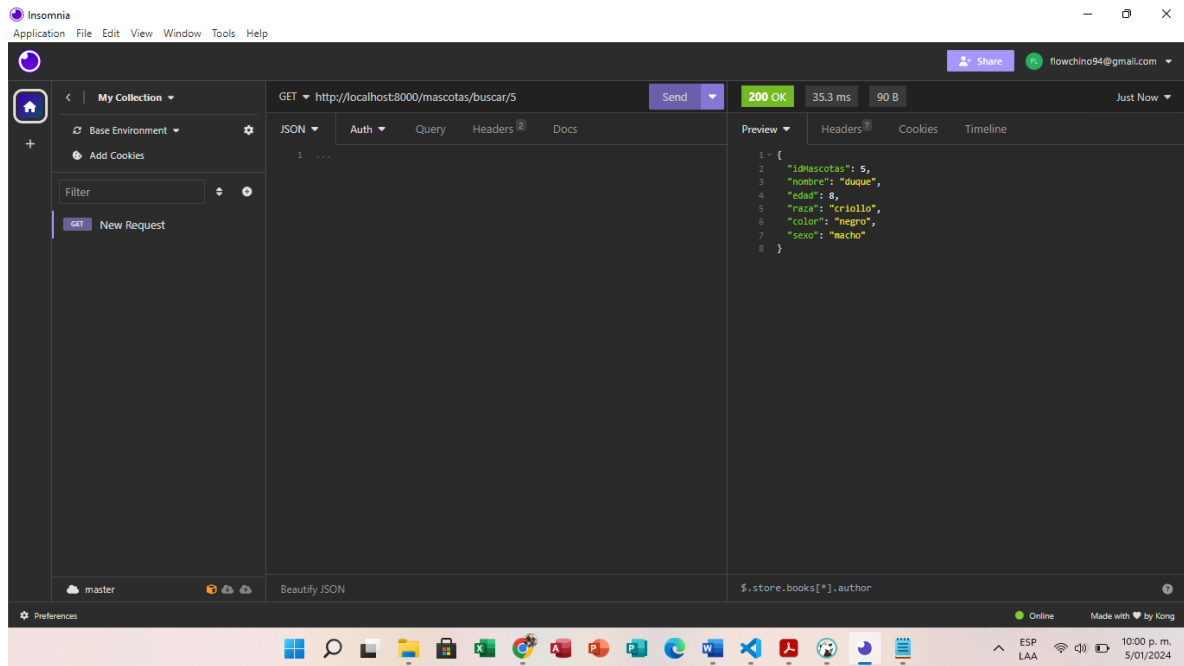
Primera La cual seria listar todas las mascotas



## 2 crear una mascota



## 3 buscar un registro por su id "identificador"



5 en este se muestra la asociación de las 3 tablas de la base de datos

