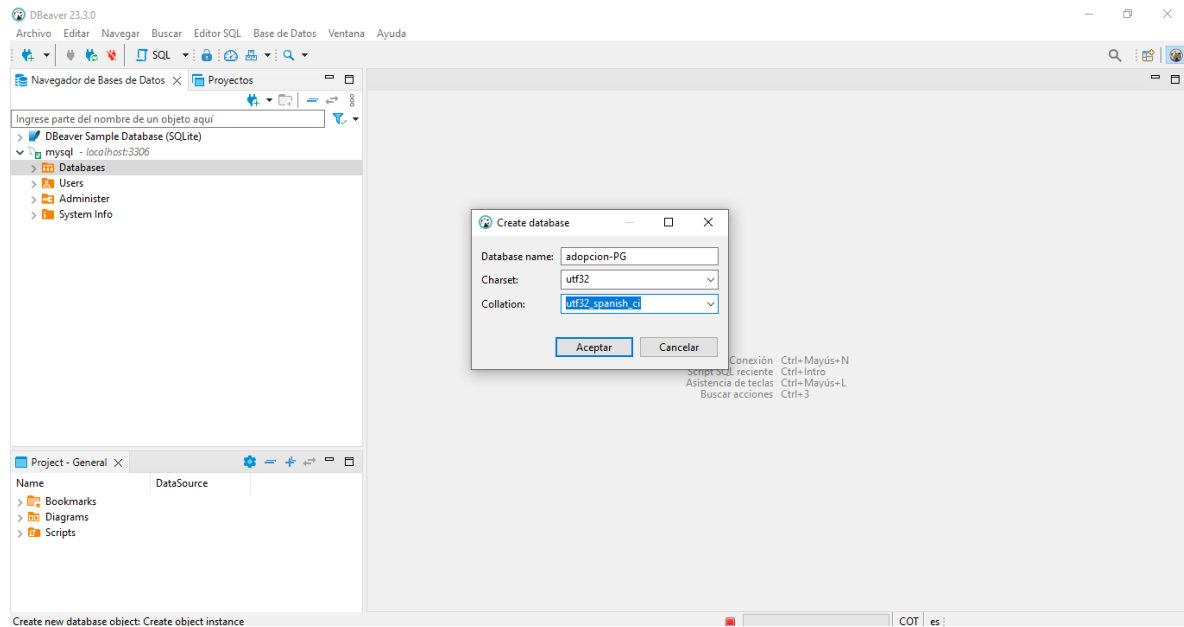
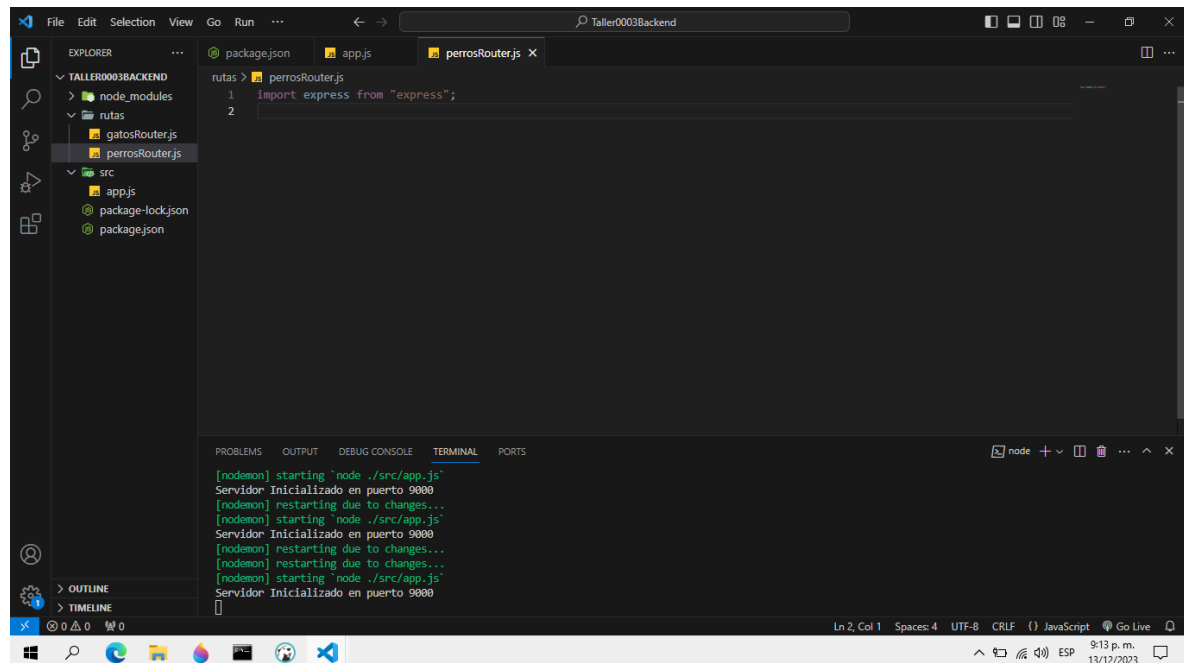


Taller Backend

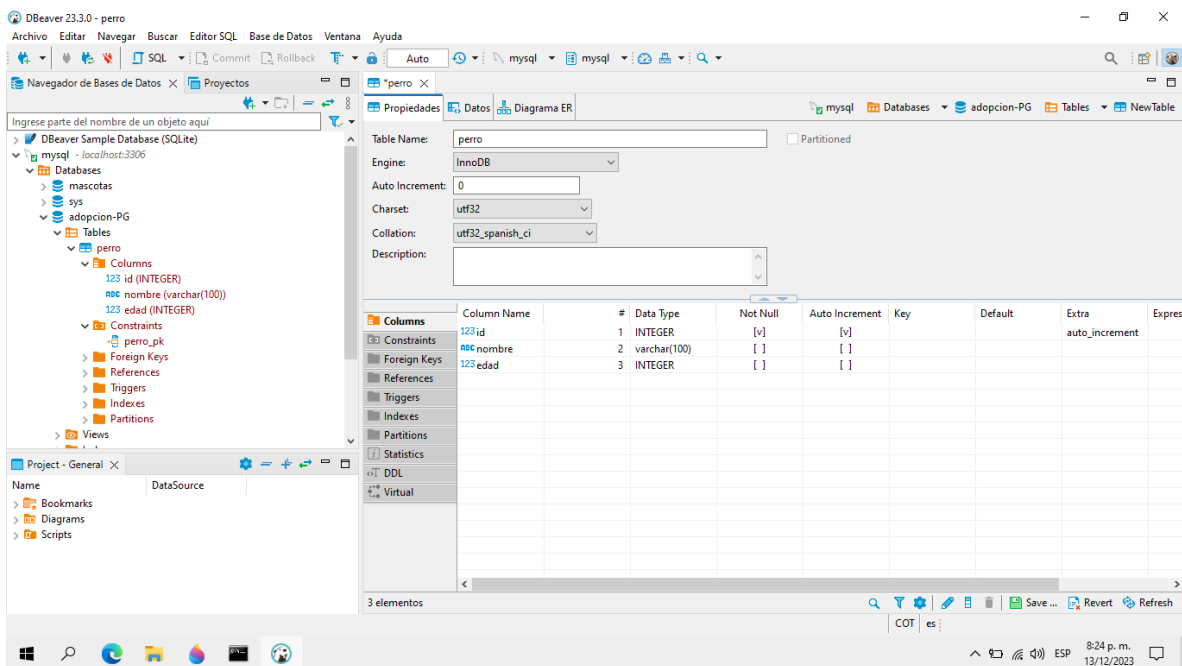
Creamos la base de datos



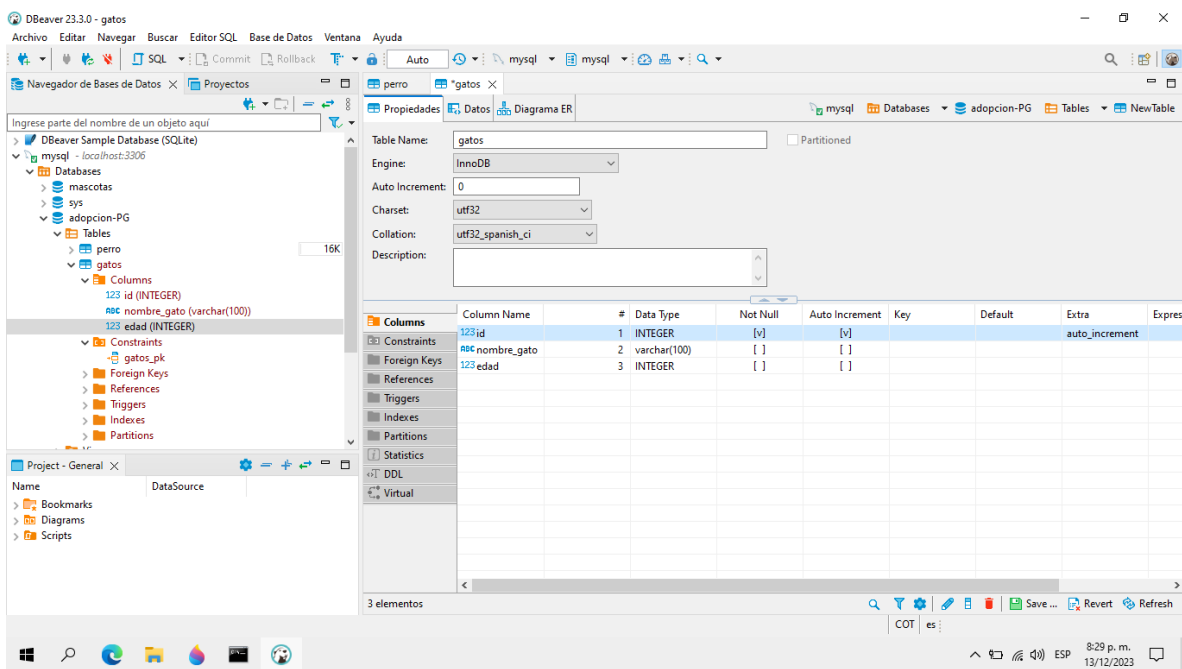
Creamos una carpeta donde se guardar las rutas para los llamados



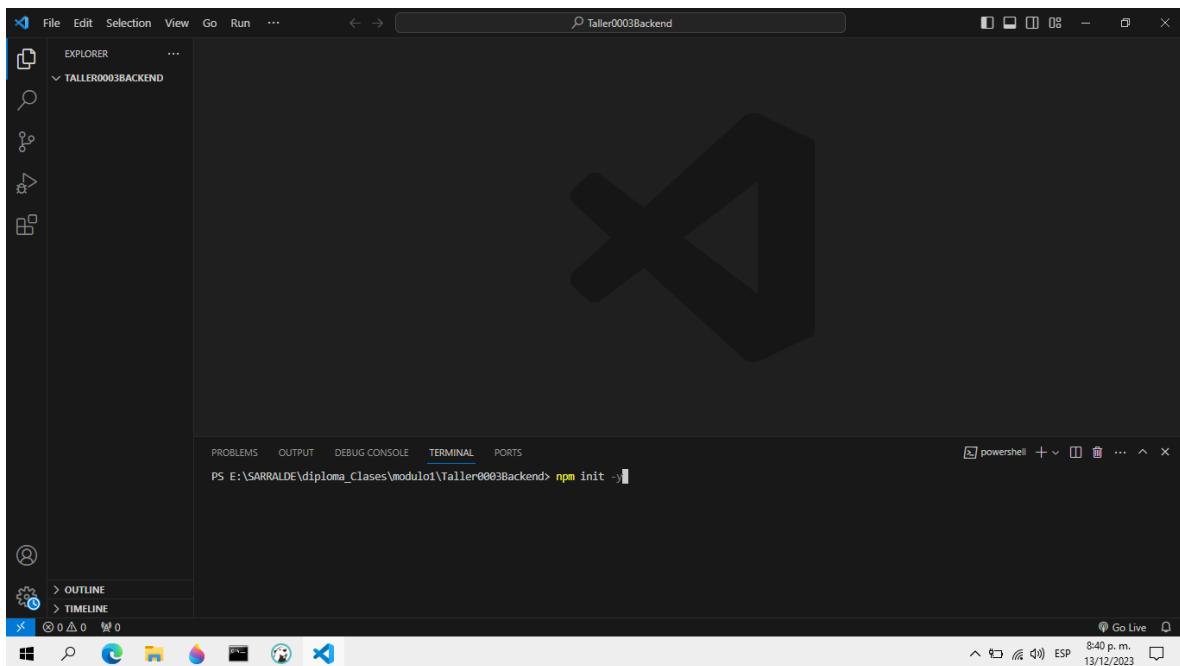
Creamos la tabla de perros en la base de datos



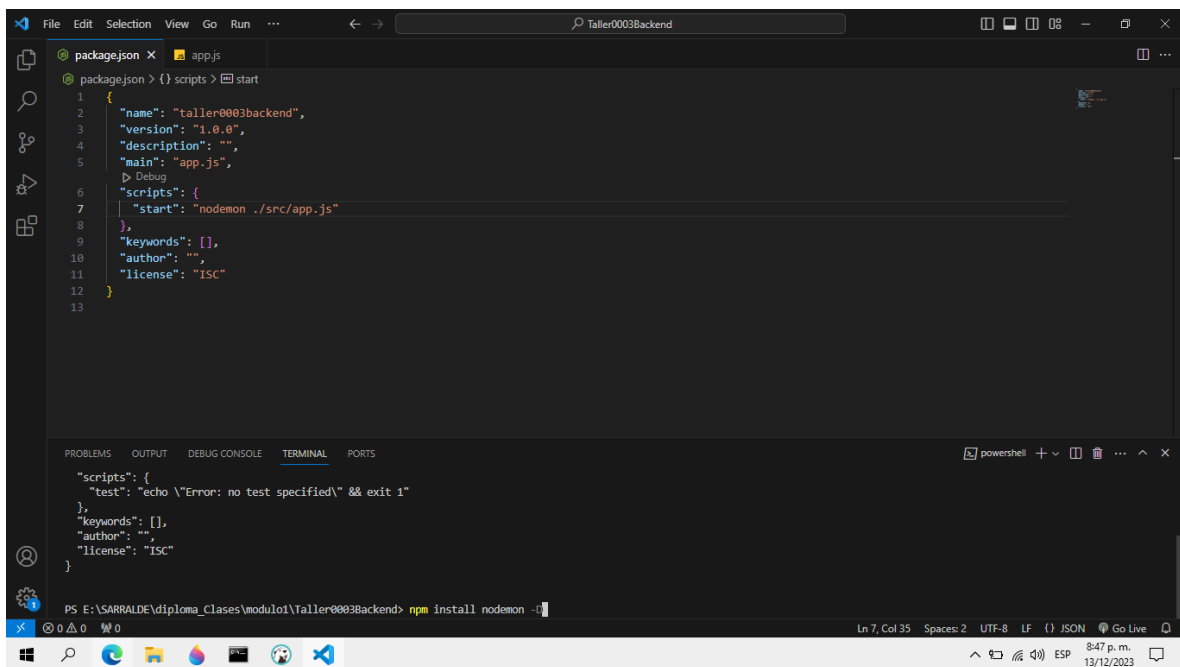
Creamos la tabla de gatos en la base de datos



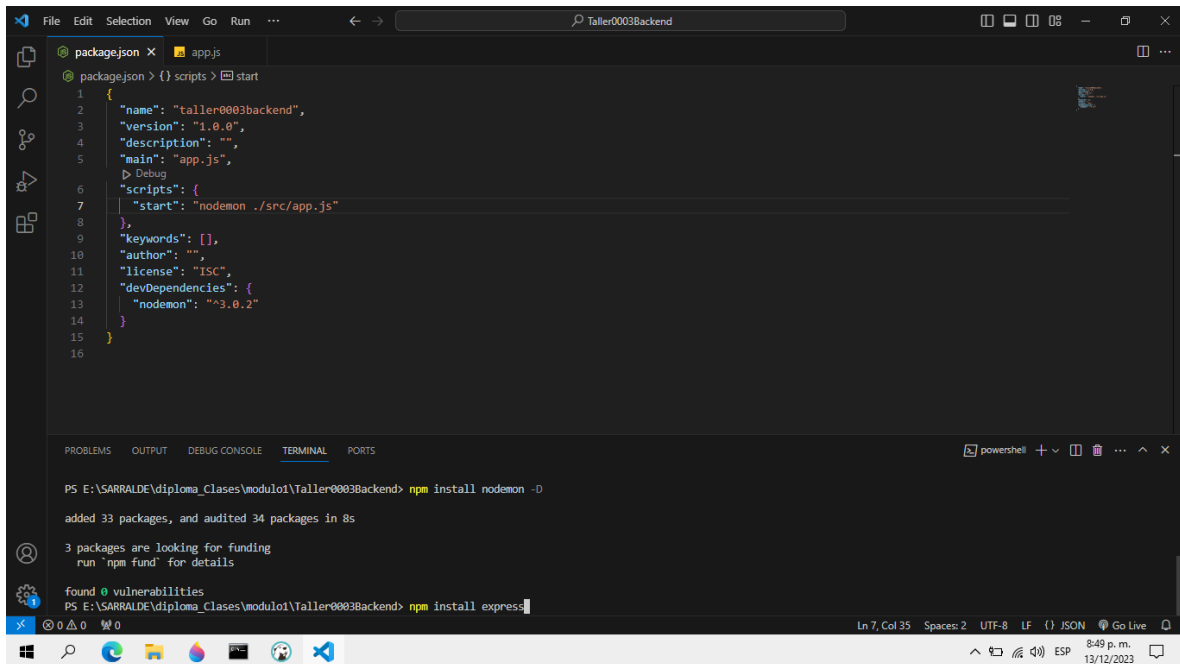
En esta parte vamos a inicializar el proyecto



Instalamos el nodemon en el entorno de desarrollo



Instalamos el express

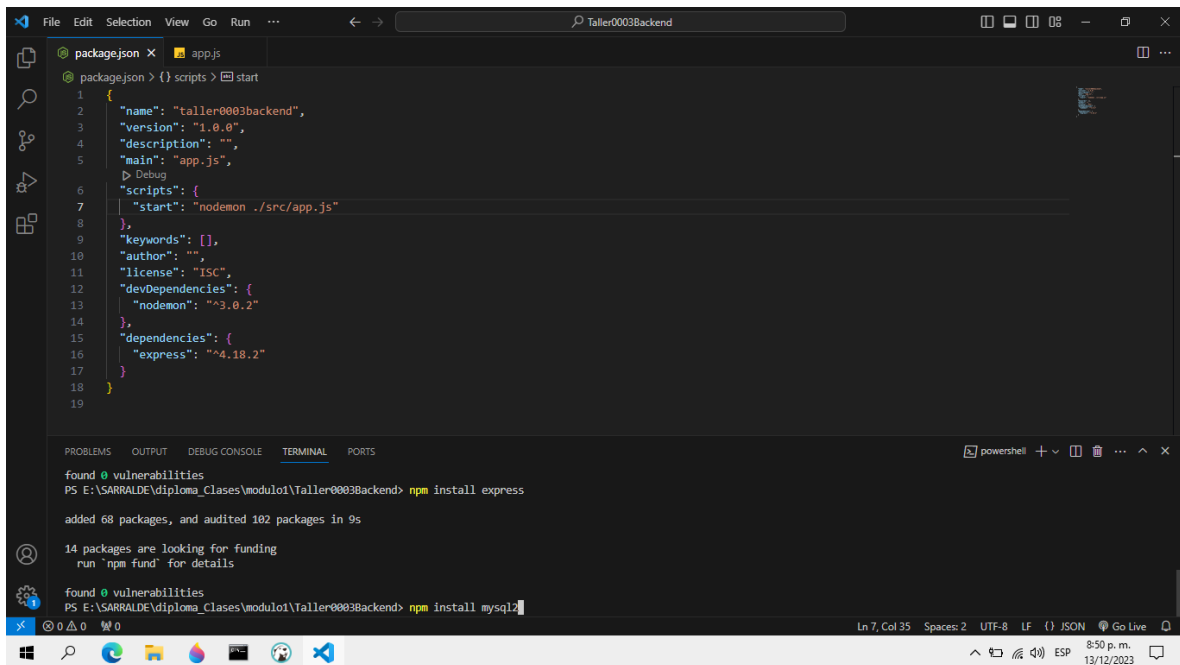


The screenshot shows the Visual Studio Code editor with a file named `package.json` open. The file contains the following JSON:

```
{
  "name": "taller0003backend",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodemon ./src/app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.0.2"
  }
}
```

The terminal at the bottom shows the command `npm install nodemon -D` being executed, which successfully installed 33 packages. The next command, `npm install express`, is partially visible at the bottom of the terminal.

Instalamos en mysql

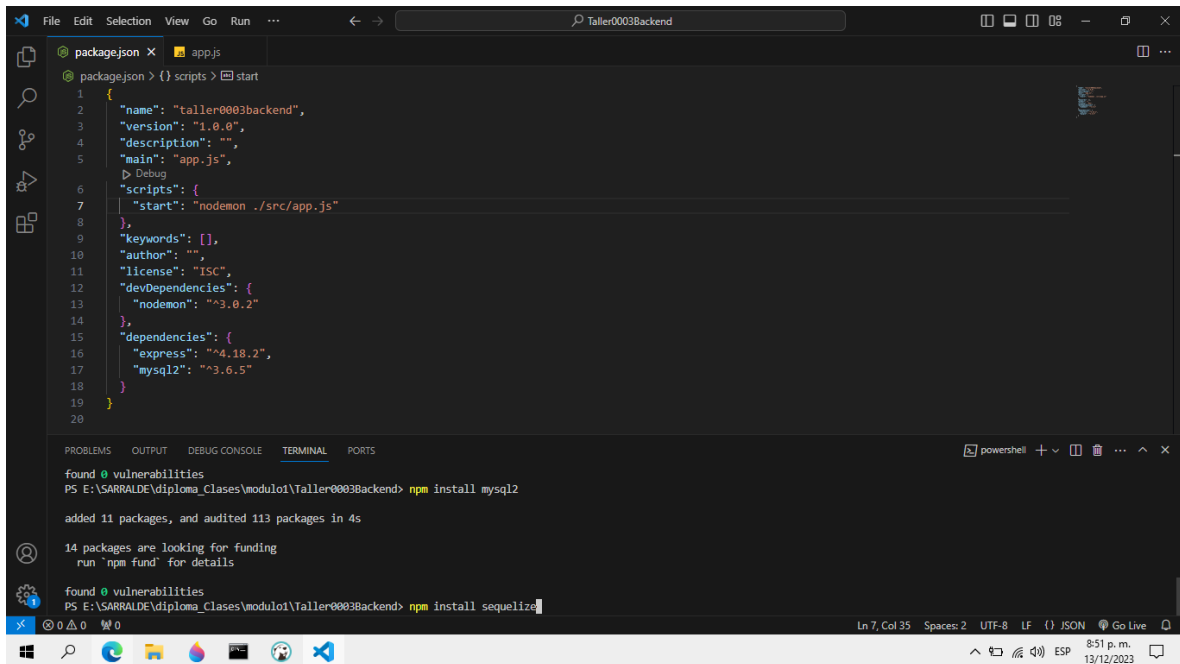


The screenshot shows the Visual Studio Code editor with the `package.json` file updated to include `express` in the `dependencies` section:

```
{
  "name": "taller0003backend",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodemon ./src/app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.0.2"
  },
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

The terminal shows the command `npm install express` being executed, which installed 68 packages. The next command, `npm install mysql2`, is partially visible at the bottom of the terminal.

Instalamos el sequelize

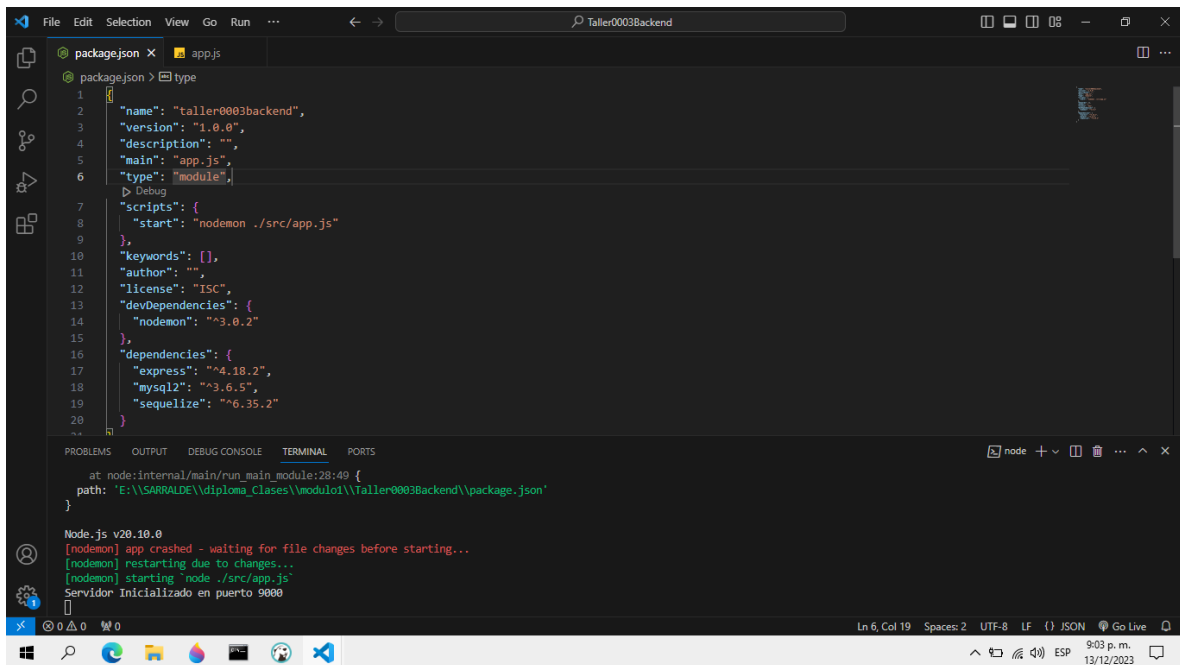


The screenshot shows a Visual Studio Code editor with a file named `package.json` open. The file contains the following JSON configuration:

```
{
  "name": "taller0003backend",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "start": "nodemon ./src/app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.0.2"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.6.5"
  }
}
```

The terminal at the bottom shows the command `npm install sequelize` being executed. The output indicates that 11 packages were added and 113 packages were audited in 4 seconds. The terminal also shows the command `npm fund` being run to check for funding information.

Configuramos el package.json

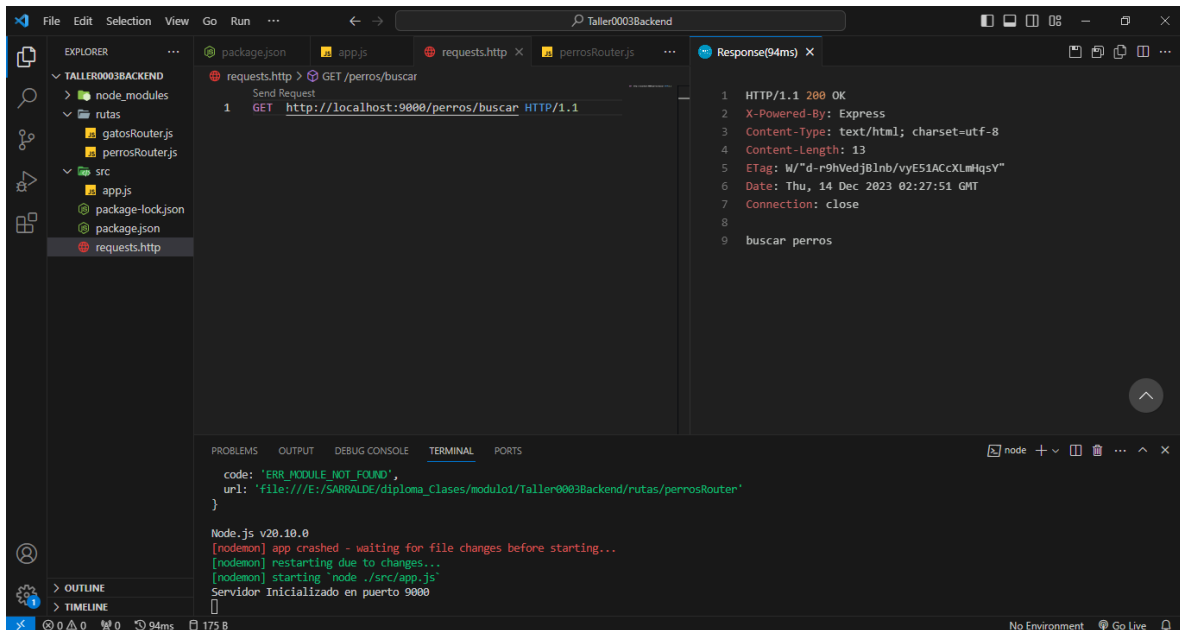


The screenshot shows the same Visual Studio Code editor with the `package.json` file. The file has been updated to include the `sequelize` dependency:

```
{
  "name": "taller0003backend",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "type": "module",
  "scripts": {
    "start": "nodemon ./src/app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^3.0.2"
  },
  "dependencies": {
    "express": "^4.18.2",
    "mysql2": "^3.6.5",
    "sequelize": "^6.35.2"
  }
}
```

The terminal shows the command `node` being executed. The output indicates that the application crashed and was restarted due to changes. The terminal also shows the command `nodemon` starting the application on port 9000.

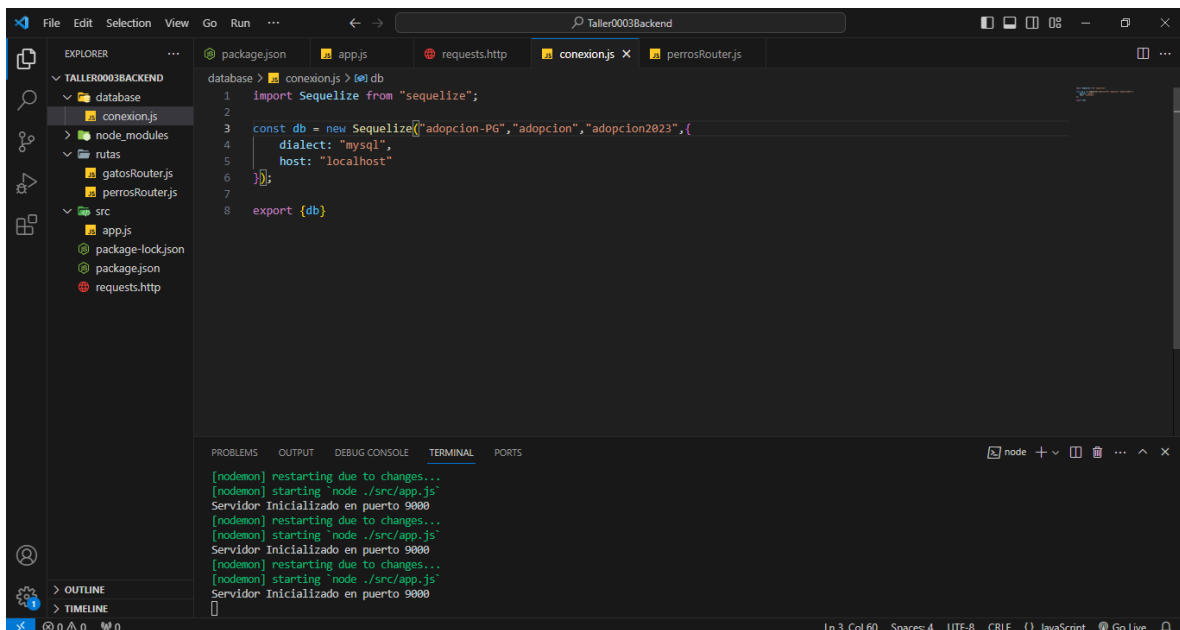
Creamos los verbos HTTP



The screenshot shows the VS Code editor with the 'requests.http' file open. The file contains a single GET request to 'http://localhost:9000/perros/buscar'. The response is shown in the 'Response(94ms)' panel, indicating a 200 OK status and a JSON body containing 'buscar perros'. The terminal at the bottom shows the server starting successfully on port 9000.

```
package.json  app.js  requests.http  perrosRouter.js  ...  
Taller0003Backend  
node_modules  
rutas  
gatosRouter.js  
perrosRouter.js  
src  
app.js  
package-lock.json  
package.json  
requests.http  
Send Request  
1 GET http://localhost:9000/perros/buscar HTTP/1.1  
Response(94ms)  
1 HTTP/1.1 200 OK  
2 X-Powered-By: Express  
3 Content-Type: text/html; charset=utf-8  
4 Content-Length: 13  
5 ETag: W/"d-r9hVedj0lmb/vyE51ACcXLMHqsY"  
6 Date: Thu, 14 Dec 2023 02:27:51 GMT  
7 Connection: close  
8  
9 buscar perros  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
code: 'ERR_MODULE_NOT_FOUND',  
url: 'file:///E:/SARRALDE/diploma_Clases/modulo1/Taller0003Backend/rutas/perrosRouter'  
Node.js v20.10.0  
[nodemon] app crashed - waiting for file changes before starting...  
[nodemon] restarting due to changes...  
[nodemon] starting 'node ./src/app.js'  
Servidor Inicializado en puerto 9000
```

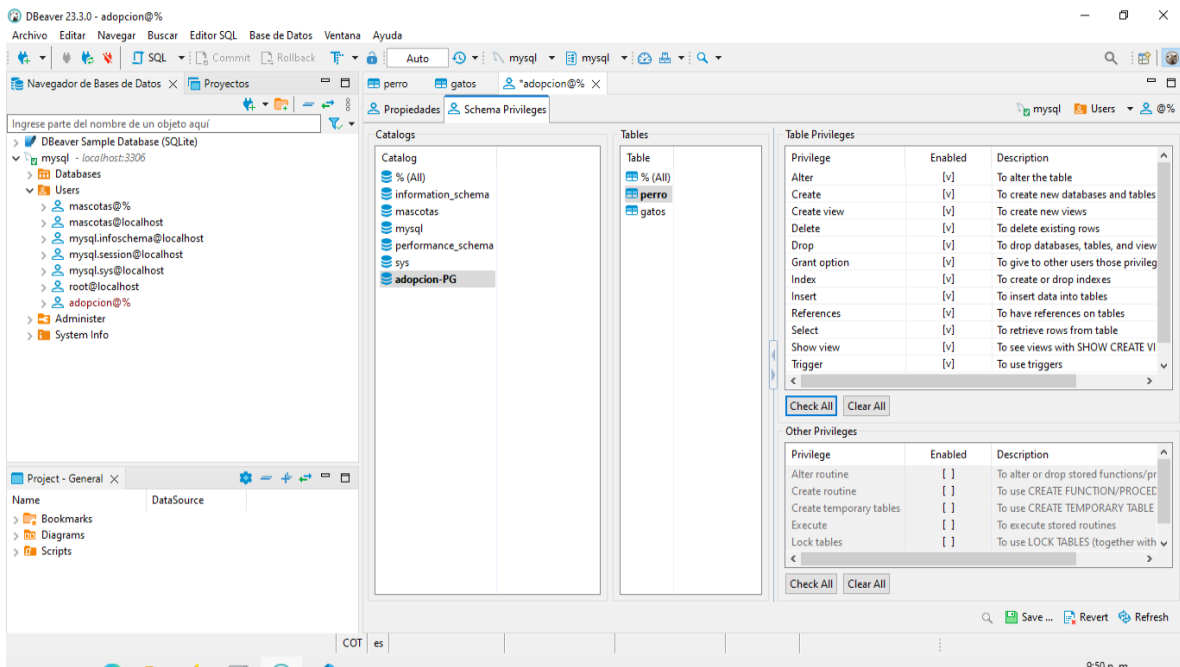
Creamos la conexión con la base de datos



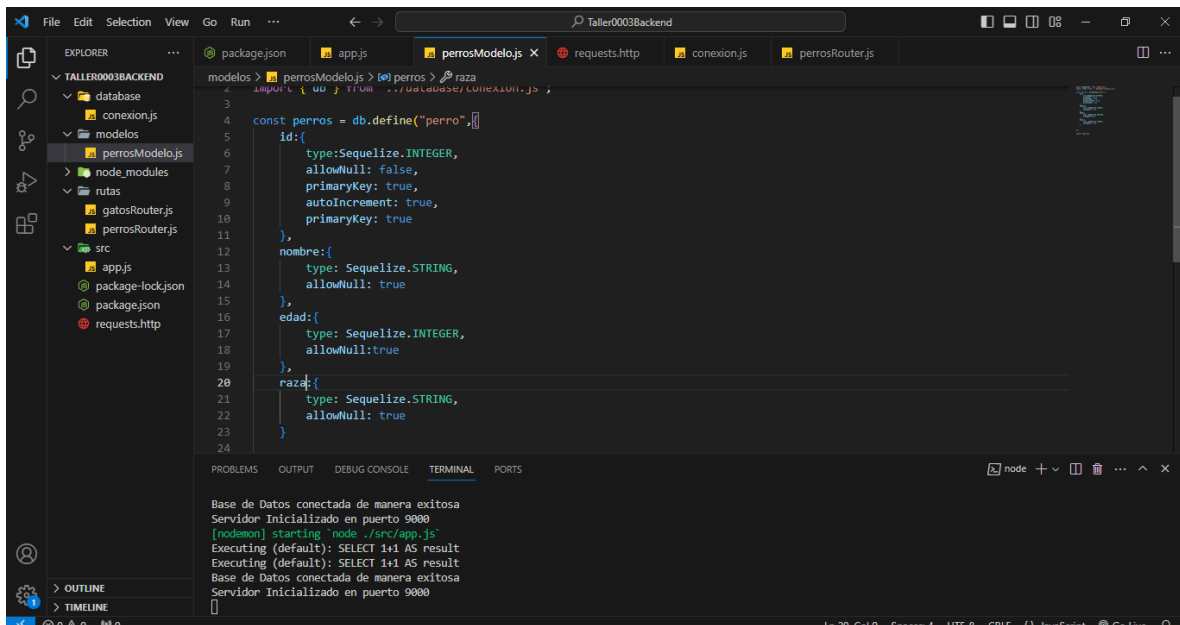
The screenshot shows the VS Code editor with the 'conexion.js' file open. The file contains the Sequelize database connection setup, including the import of Sequelize, the creation of a new Sequelize instance with the database name 'adopcion-PG', and the export of the database instance. The terminal at the bottom shows the server starting successfully on port 9000.

```
package.json  app.js  requests.http  perrosRouter.js  ...  
Taller0003Backend  
node_modules  
rutas  
gatosRouter.js  
perrosRouter.js  
src  
app.js  
package-lock.json  
package.json  
requests.http  
database > conexion.js > db  
1 import Sequelize from "sequelize";  
2  
3 const db = new Sequelize("adopcion-PG", "adopcion", "adopcion2023", {  
4   dialect: "mysql",  
5   host: "localhost"  
6 });  
7  
8 export {db};  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
[nodemon] restarting due to changes...  
[nodemon] starting 'node ./src/app.js'  
Servidor Inicializado en puerto 9000  
[nodemon] restarting due to changes...  
[nodemon] starting 'node ./src/app.js'  
Servidor Inicializado en puerto 9000  
[nodemon] restarting due to changes...  
[nodemon] starting 'node ./src/app.js'  
Servidor Inicializado en puerto 9000  
Ln 3, Col 60 Spaces: 4 UTF-8 CRLF JavaScript Go Live
```

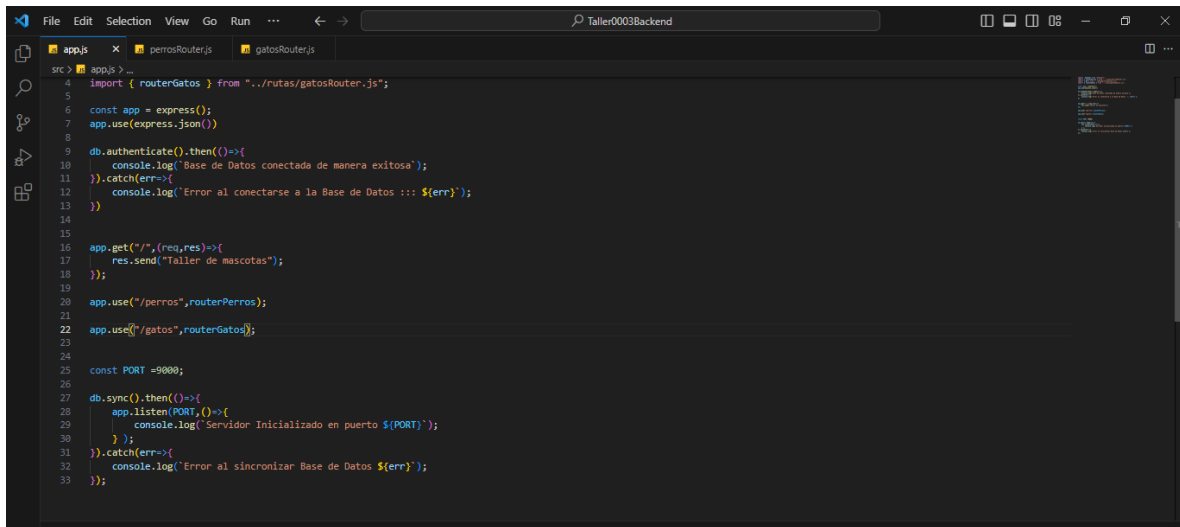
Creamos el usuario para la base de datos y le damos los permisos para las tablas



En esta sesión creamos la estructura para los datos de la base

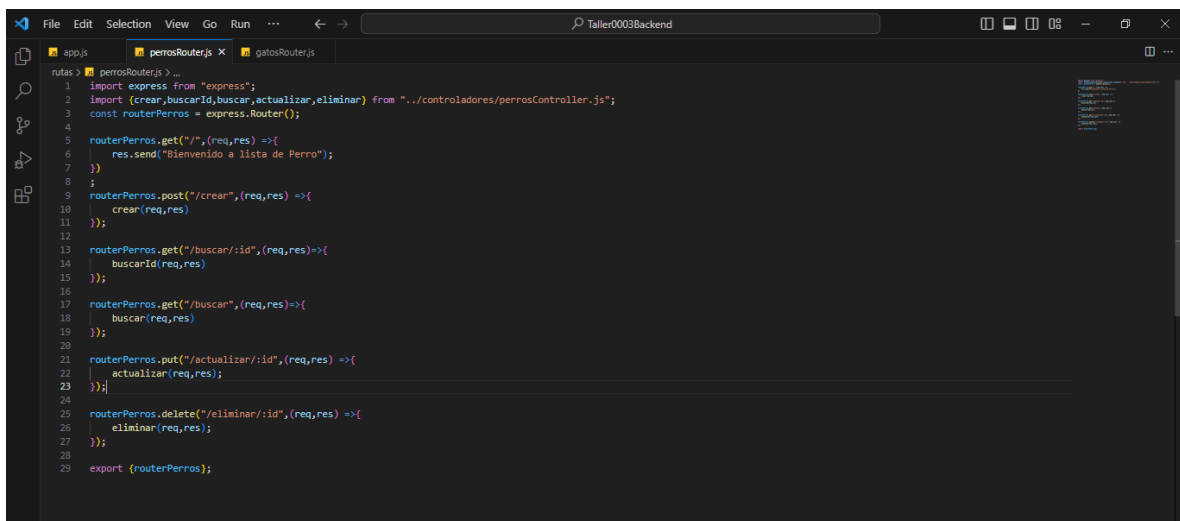


En app vamos a configurar en que puerto se va a establecer nuestra conexión y en donde también se va a verificar si la conexión a la base de datos fue exitosa



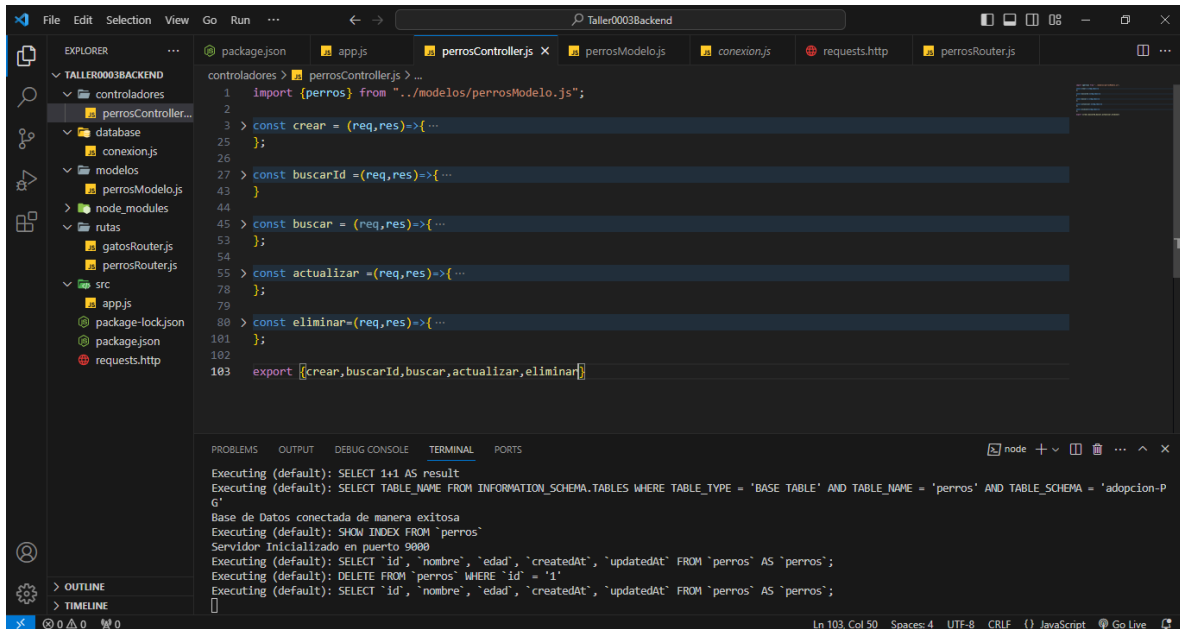
```
4 import { routerGatos } from "../rutas/gatosRouter.js";
5
6 const app = express();
7 app.use(express.json());
8
9 db.authenticate().then(()=>{
10   console.log('Base de Datos conectada de manera exitosa');
11 }).catch(err=>{
12   console.log('Error al conectarse a la Base de Datos ::: ${err}');
13 })
14
15
16 app.get("/",(req,res)=>{
17   res.send("Taller de mascotas");
18 });
19
20 app.use("/perros",routerPerros);
21
22 app.use("/gatos",routerGatos);
23
24
25 const PORT =9000;
26
27 db.sync().then(()=>{
28   app.listen(PORT,()=>{
29     console.log('Servidor Inicializado en puerto ${PORT}');
30   });
31 }).catch(err=>{
32   console.log('Error al sincronizar Base de Datos ${err}');
33 });
```

En esta sesión se van a configurar las rutas donde se hacen el llamado de las operaciones



```
1 import express from "express";
2 import {crear, buscarId, buscar, actualizar, eliminar} from "../controladores/perrosController.js";
3 const routerPerros = express.Router();
4
5 routerPerros.get("/",(req,res) =>{
6   res.send("Bienvenido a lista de Perro");
7 })
8
9 routerPerros.post("/crear", (req,res) =>{
10   crear(req,res)
11 });
12
13 routerPerros.get("/buscar/:id",(req,res)=>{
14   buscarId(req,res)
15 });
16
17 routerPerros.get("/buscar", (req,res)=>{
18   buscar(req,res)
19 });
20
21 routerPerros.put("/actualizar/:id", (req,res) =>{
22   actualizar(req,res);
23 });
24
25 routerPerros.delete("/eliminar/:id", (req,res) =>{
26   eliminar(req,res);
27 });
28
29 export {routerPerros};
```


En esta sesión haremos la programación para los diferentes llamados



The screenshot shows a Visual Studio Code editor with the file explorer on the left displaying the project structure for 'Taller0003BACKEND'. The main editor window shows the 'perrosController.js' file with the following code:

```
1 import {perros} from "../modelos/perrosModelo.js";
2
3 > const crear = (req,res)=>{ ...
25   };
26
27 > const buscarId =(req,res)=>{ ...
43   }
44
45 > const buscar = (req,res)=>{ ...
53   };
54
55 > const actualizar =(req,res)=>{ ...
78   };
79
80 > const eliminar=(req,res)=>{ ...
101   };
102
103 export {crear,buscarId,buscar,actualizar,eliminar}
```

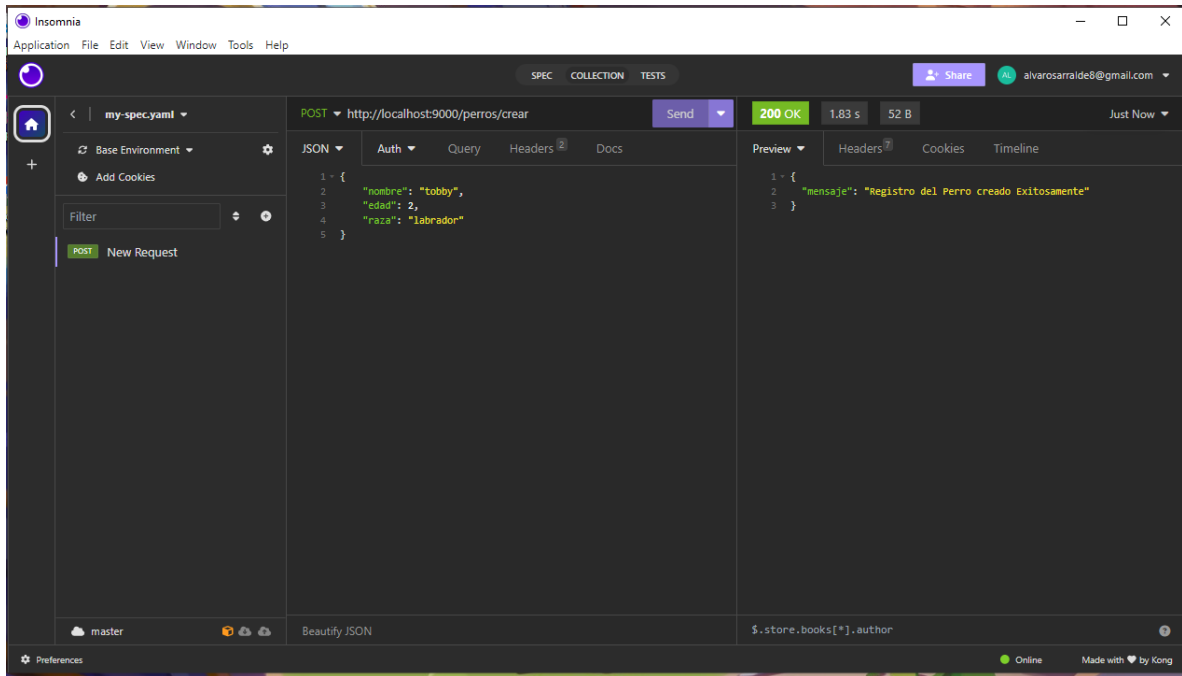
Below the code editor, the terminal window is open, showing the following output:

```
Executing (default): SELECT 1+1 AS result
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'perros' AND TABLE_SCHEMA = 'adopcion-P
G'
Base de Datos conectada de manera exitosa
Executing (default): SHOW INDEX FROM `perros`
Servidor Inicializado en puerto 9000
Executing (default): SELECT `id`,`nombre`,`edad`,`createdAt`,`updatedAt` FROM `perros` AS `perros`;
Executing (default): DELETE FROM `perros` WHERE `id` = '1'
Executing (default): SELECT `id`,`nombre`,`edad`,`createdAt`,`updatedAt` FROM `perros` AS `perros`;
```

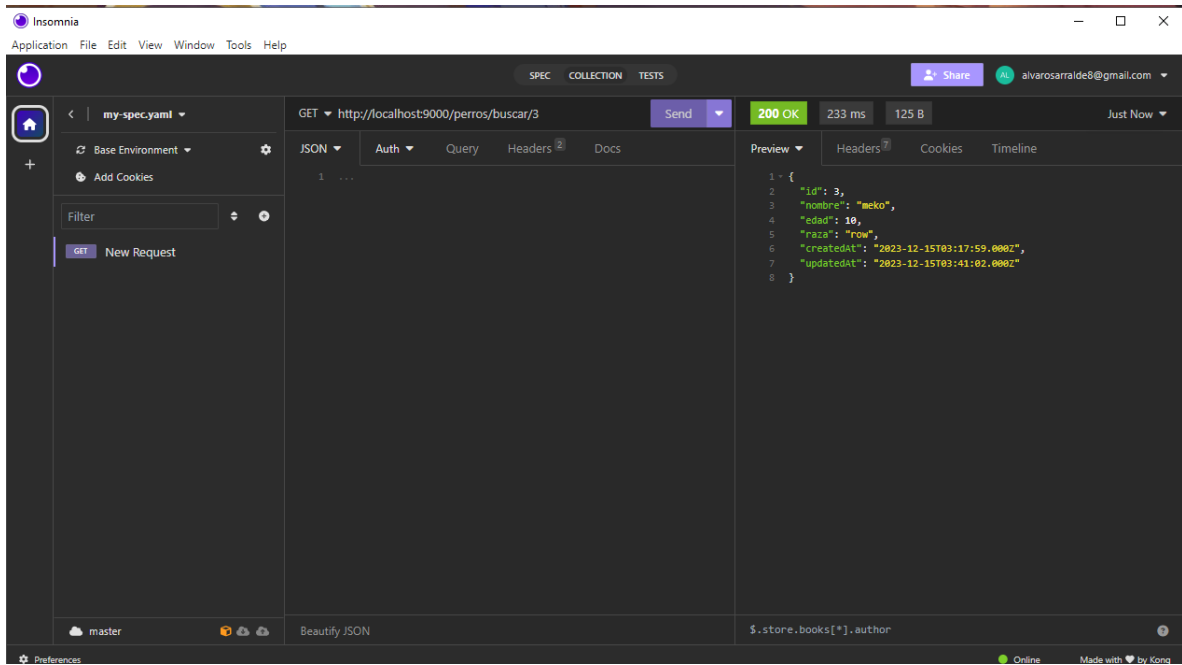
Mostraremos lo resultado de los verbos HTTP en imnsomia

Mostraremos las diferentes operaciones en la tabla del perro

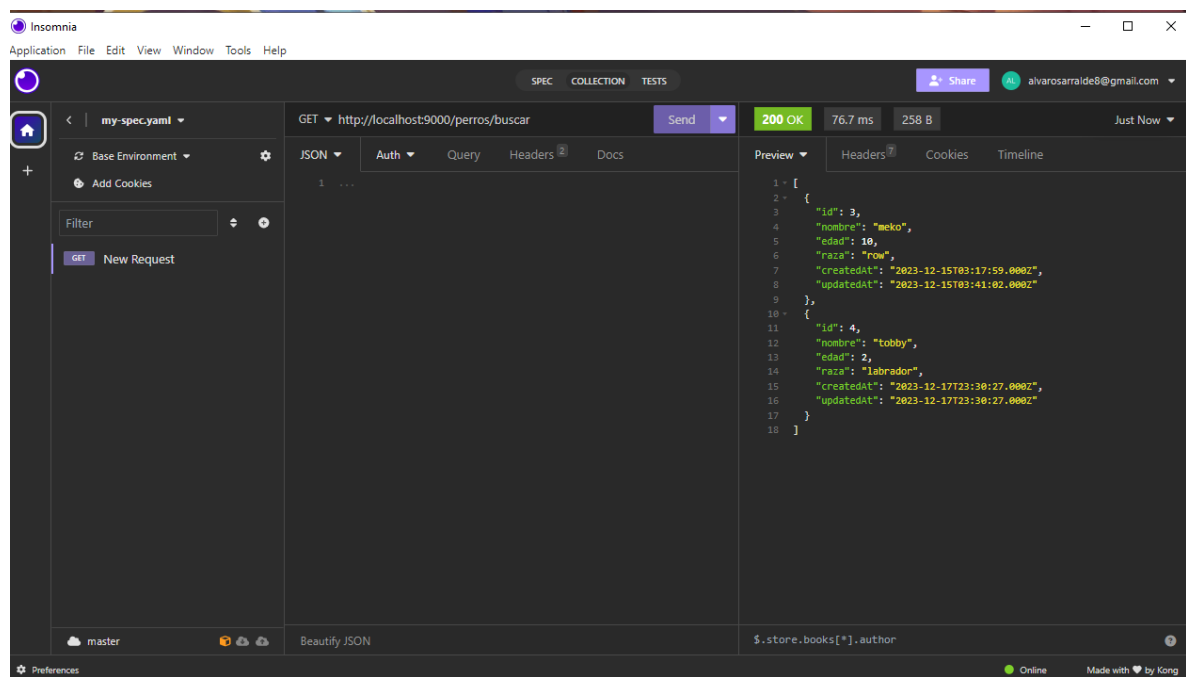
En esta parte hacemos el registro del perro



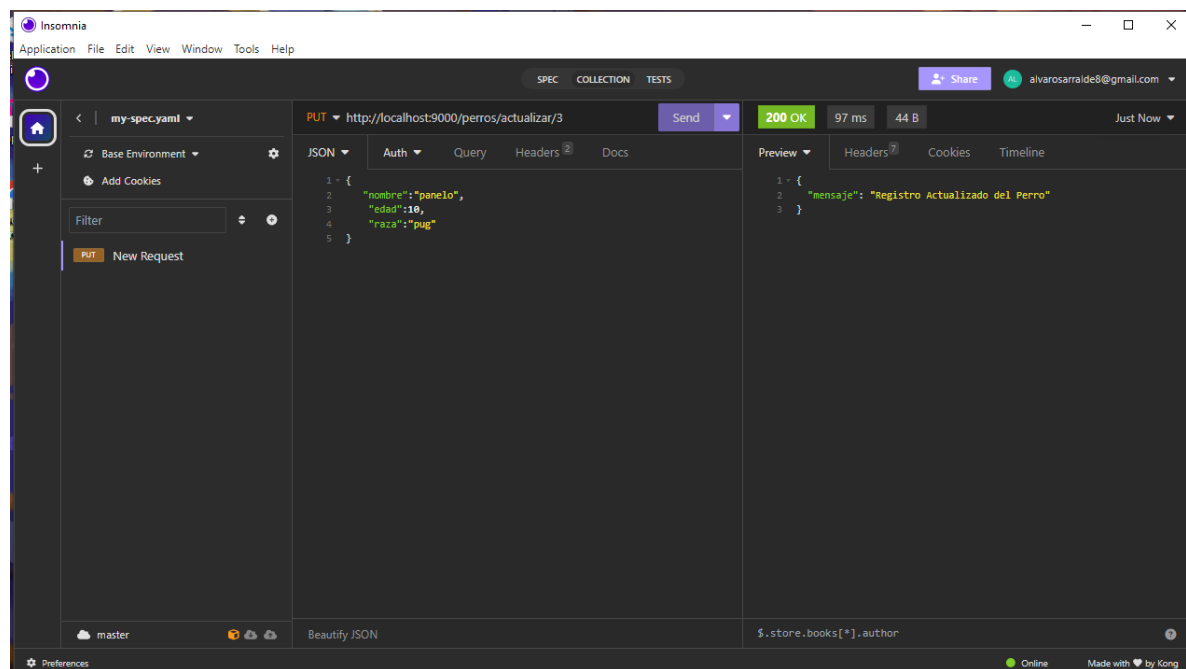
En esta parte buscamos el perro por el ID



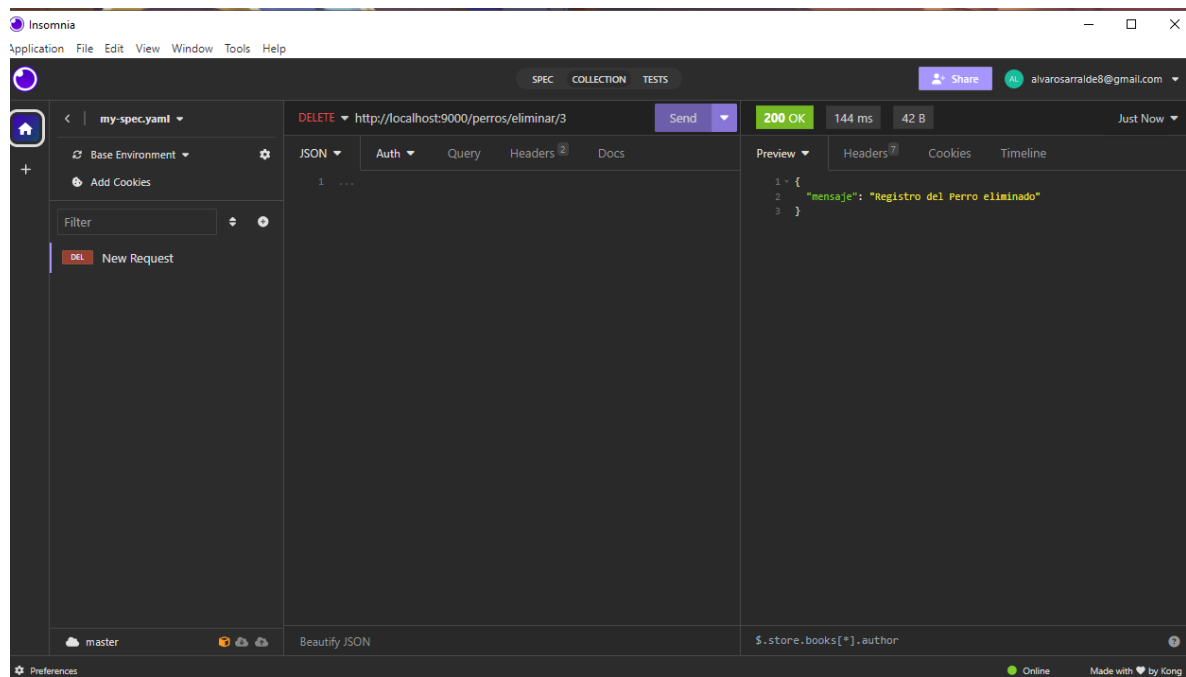
En esta parte buscamos todos los perros de forma global



En esta parte actualizamos el registro del perro

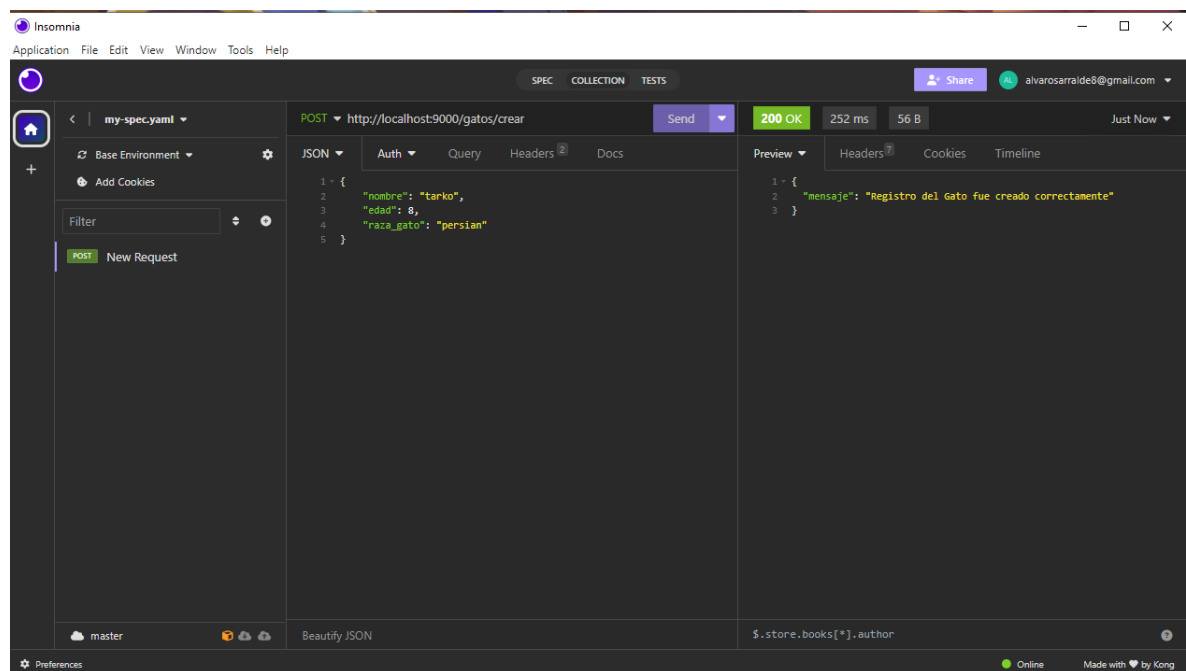


En esta parte eliminamos el registro del perro por el ID



Mostraremos las diferentes operaciones en la tabla de gato

Creamos el registro del gato



Buscamos los registros de los gatos de forma global

The screenshot shows the Insomnia application interface. The top bar includes the Insomnia logo, menu items (Application, File, Edit, View, Window, Tools, Help), and tabs for SPEC, COLLECTION, and TESTS. A user profile for 'alvarosaralde@gmail.com' is visible in the top right. The main workspace is divided into three panels. The left panel shows a project named 'my-spec.yaml' with a 'Base Environment' and an 'Add Cookies' button. The middle panel displays a REST client request: 'GET http://localhost:9000/gatos/buscar'. The right panel shows the response status '200 OK' with a response time of '49.5 ms' and a size of '267 B'. The response body is a JSON array of two cat objects, displayed in the 'Preview' tab. The bottom status bar shows 'master', 'Beautify JSON', and a path '\$.store.books[*].author'.

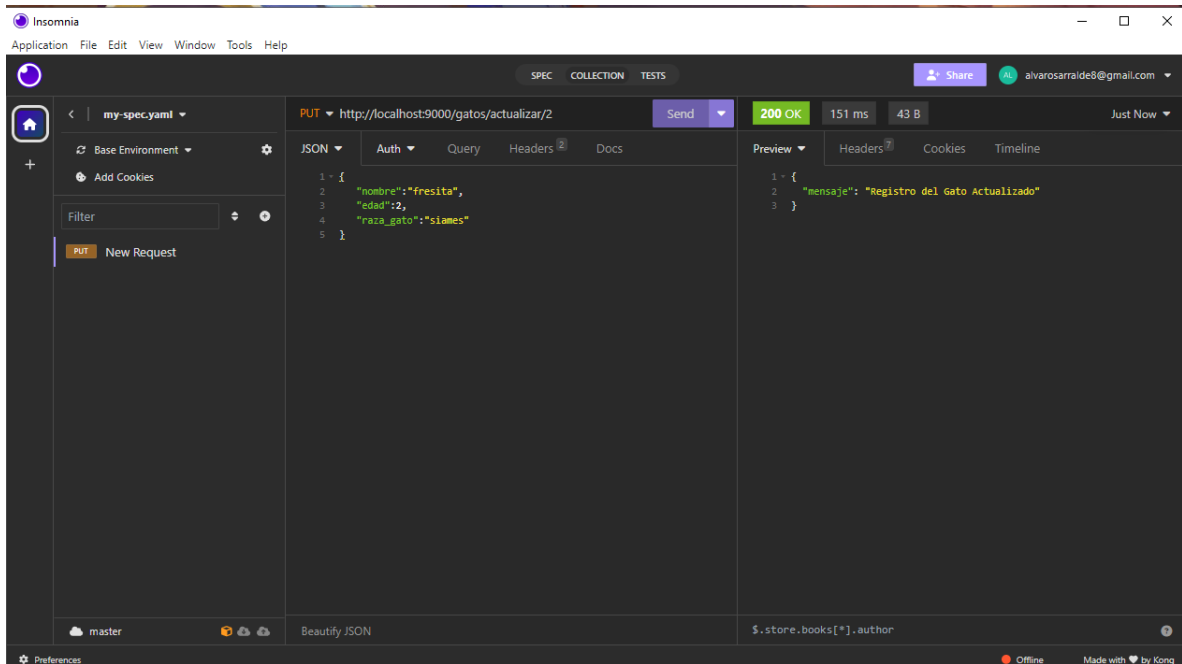
```
1 [
2   {
3     "id": 2,
4     "nombre": "laku",
5     "edad": 10,
6     "raza_gato": "lok",
7     "createdAt": "2023-12-17T01:21:30.000Z",
8     "updatedAt": "2023-12-17T01:23:32.000Z"
9   },
10  {
11    "id": 3,
12    "nombre": "tarko",
13    "edad": 8,
14    "raza_gato": "persian",
15    "createdAt": "2023-12-17T23:42:31.000Z",
16    "updatedAt": "2023-12-17T23:42:31.000Z"
17  }
18 ]
```

Buscamos el registro del gato por el ID

The screenshot shows the Insomnia application interface. The top bar includes the Insomnia logo, menu items (Application, File, Edit, View, Window, Tools, Help), and tabs for SPEC, COLLECTION, and TESTS. A user profile for 'alvarosaralde@gmail.com' is visible in the top right. The main workspace is divided into three panels. The left panel shows a project named 'my-spec.yaml' with a 'Base Environment' and an 'Add Cookies' button. The middle panel displays a REST client request: 'GET http://localhost:9000/gatos/buscar/2'. The right panel shows the response status '200 OK' with a response time of '64 ms' and a size of '130 B'. The response body is a single cat object, displayed in the 'Preview' tab. The bottom status bar shows 'master', 'Beautify JSON', and a path '\$.store.books[*].author'.

```
1 {
2   "id": 2,
3   "nombre": "laku",
4   "edad": 10,
5   "raza_gato": "lok",
6   "createdAt": "2023-12-17T01:21:30.000Z",
7   "updatedAt": "2023-12-17T01:23:32.000Z"
8 }
```

Actualizamos el registro del gato por el ID



Eliminamos el registro del gato por el ID

