

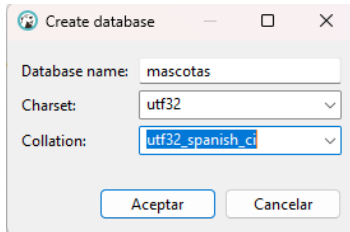
Taller Backend

**Universidad de Nariño.
Ingeniería de Sistemas.
Diplomado de actualización en nuevas tecnologías para el
Desarrollo de Software.**

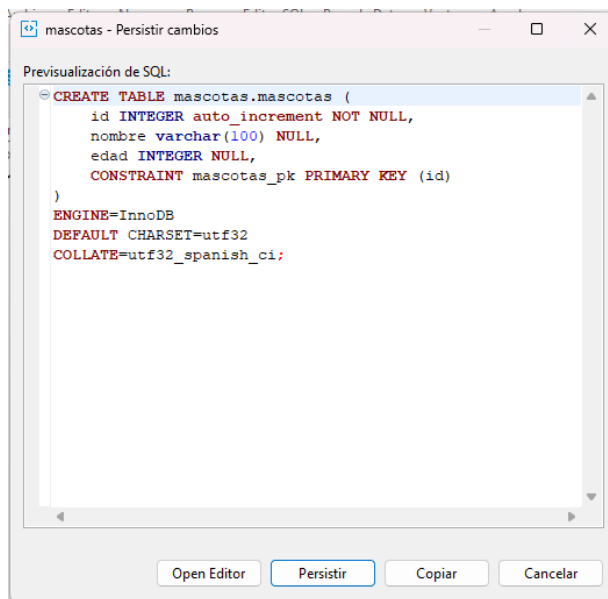
Bolivar David Bastidas Madroñero

1. Crear una base de datos MYSQL que permita llevar el registro de mascotas (perros y gatos), así como también el proceso de solicitud de adopción de estas.

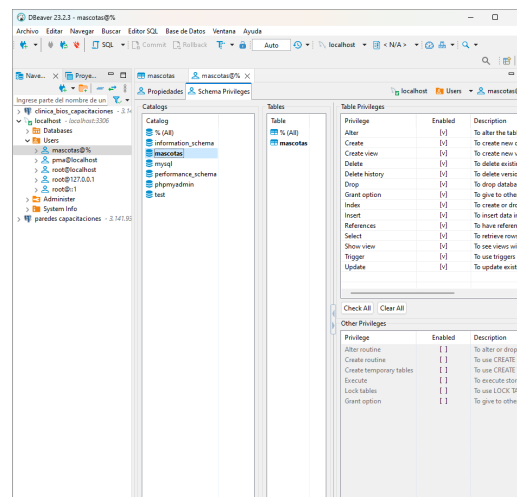
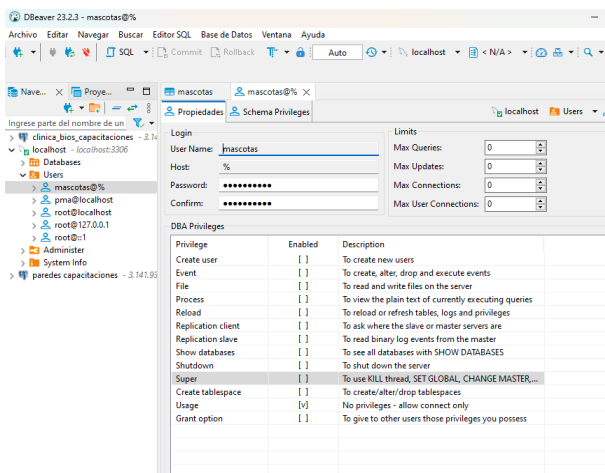
- a. Creamos la base de datos desde dbeaver



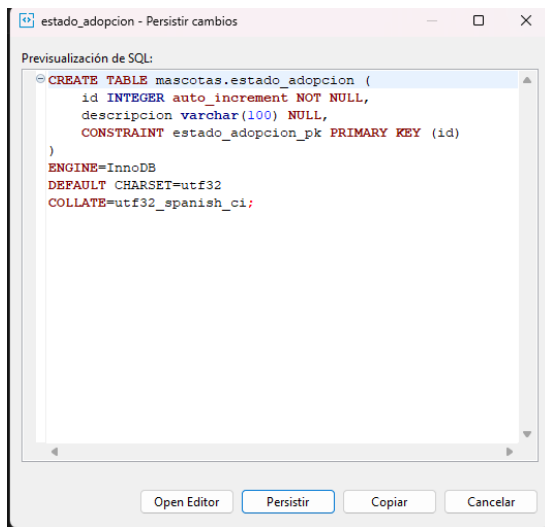
- b. Creamos la tabla mascotas



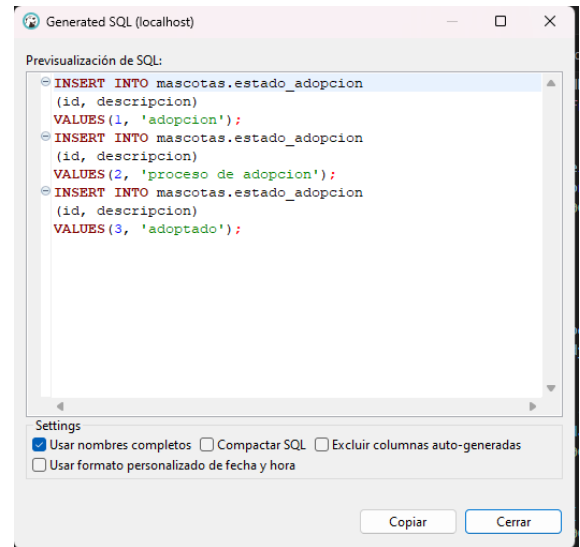
- c. creamos el usuario para el uso de la base de datos



- d. creamos la tabla estado_adopcion que tendra las opciones para la adopcion y insertamos datos

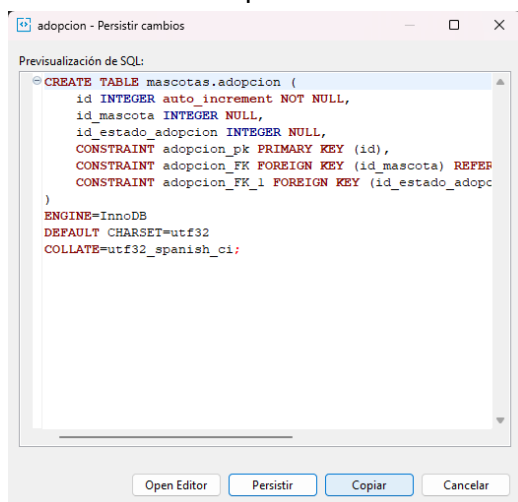


```
CREATE TABLE mascotas.estado_adopcion (
  id INTEGER auto_increment NOT NULL,
  descripcion varchar(100) NULL,
  CONSTRAINT estado_adopcion_pk PRIMARY KEY (id)
)
ENGINE=InnoDB
DEFAULT CHARSET=utf32
COLLATE=utf32_spanish_ci;
```



```
INSERT INTO mascotas.estado_adopcion
(id, descripcion)
VALUES (1, 'adopcion');
INSERT INTO mascotas.estado_adopcion
(id, descripcion)
VALUES (2, 'proceso de adopcion');
INSERT INTO mascotas.estado_adopcion
(id, descripcion)
VALUES (3, 'adoptado');
```

- e. creamos la tabla adopciones donde estara las mascotas y el estado de adopcion

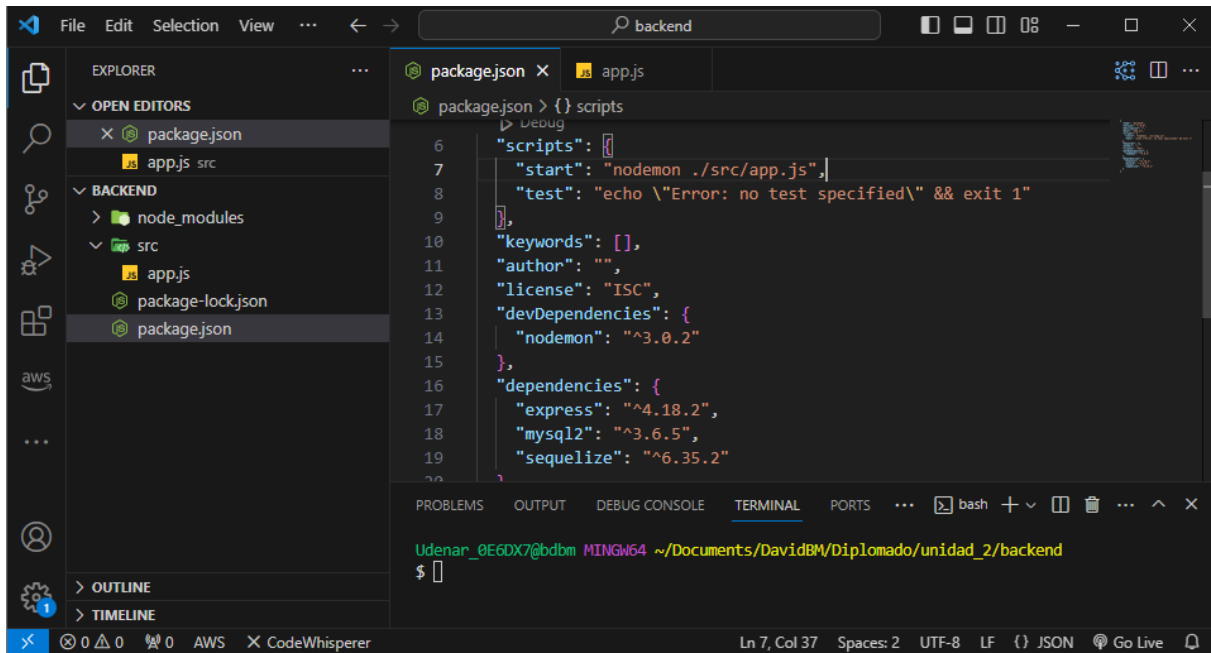


```
CREATE TABLE mascotas.adopcion (
  id INTEGER auto_increment NOT NULL,
  id_mascota INTEGER NULL,
  id_estado_adopcion INTEGER NULL,
  CONSTRAINT adopcion_pk PRIMARY KEY (id),
  CONSTRAINT adopcion_FK FOREIGN KEY (id_mascota) REFERENCES mascotas.mascotas (id),
  CONSTRAINT adopcion_FK_1 FOREIGN KEY (id_estado_adopcion) REFERENCES mascotas.estado_adopcion (id)
)
ENGINE=InnoDB
DEFAULT CHARSET=utf32
COLLATE=utf32_spanish_ci;
```

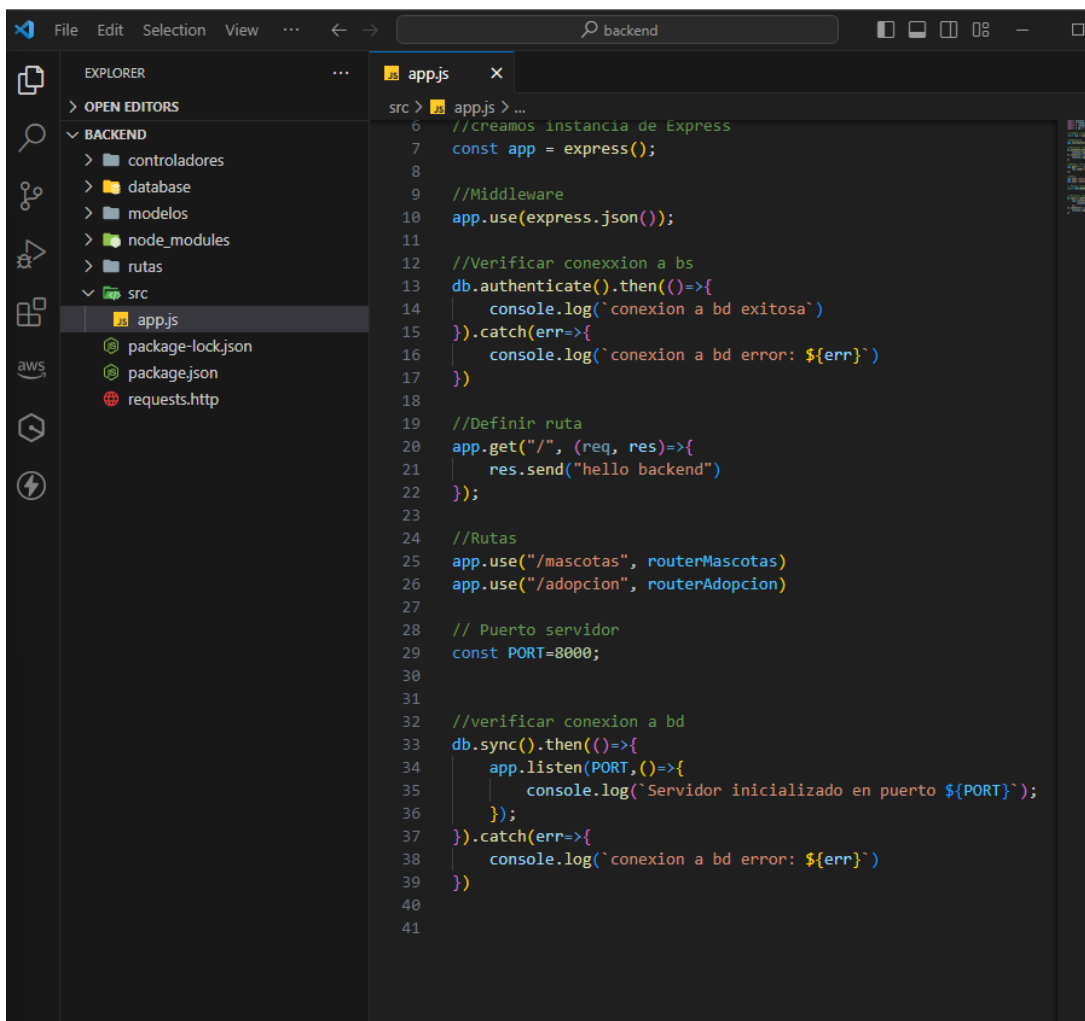
2. Desarrollar una aplicación Backend implementada en NodeJS y ExpressJS que haga uso de la base de datos del primer punto y que permita el desarrollo de todas las tareas asociadas al registro y administración de las mascotas dadas en adopción por la empresa (La empresa debe contar con un nombre).

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

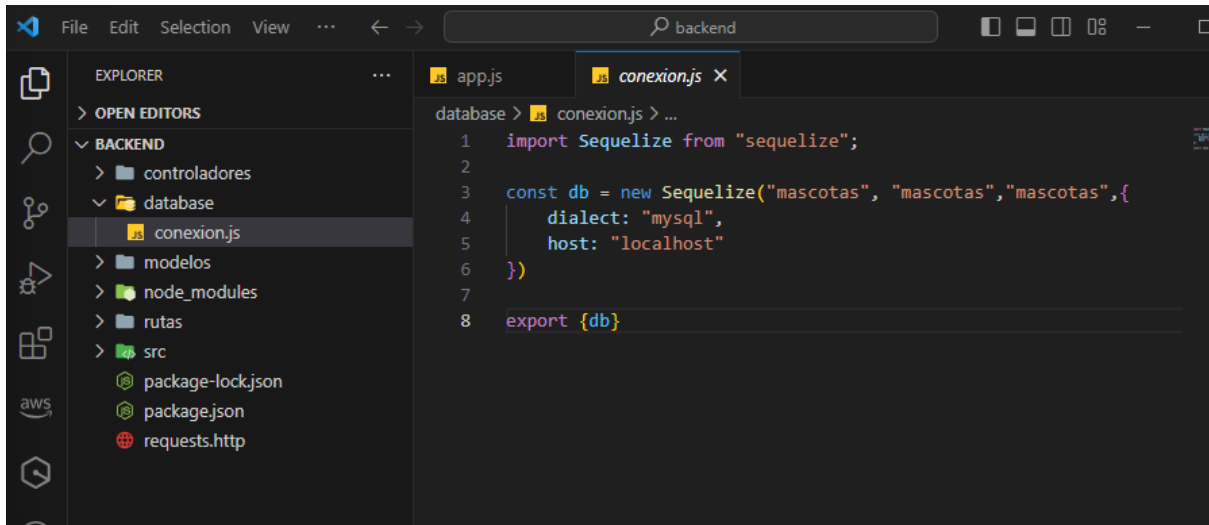
- a. creamos el proyecto backend e instalamos nodemon, expres, mysql y sequelize



- b. creamos el archivo app.js como inicio de la aplicacion especificando el puerto, iniciando coneccion a la base de datos y las rutas de nuestro backend

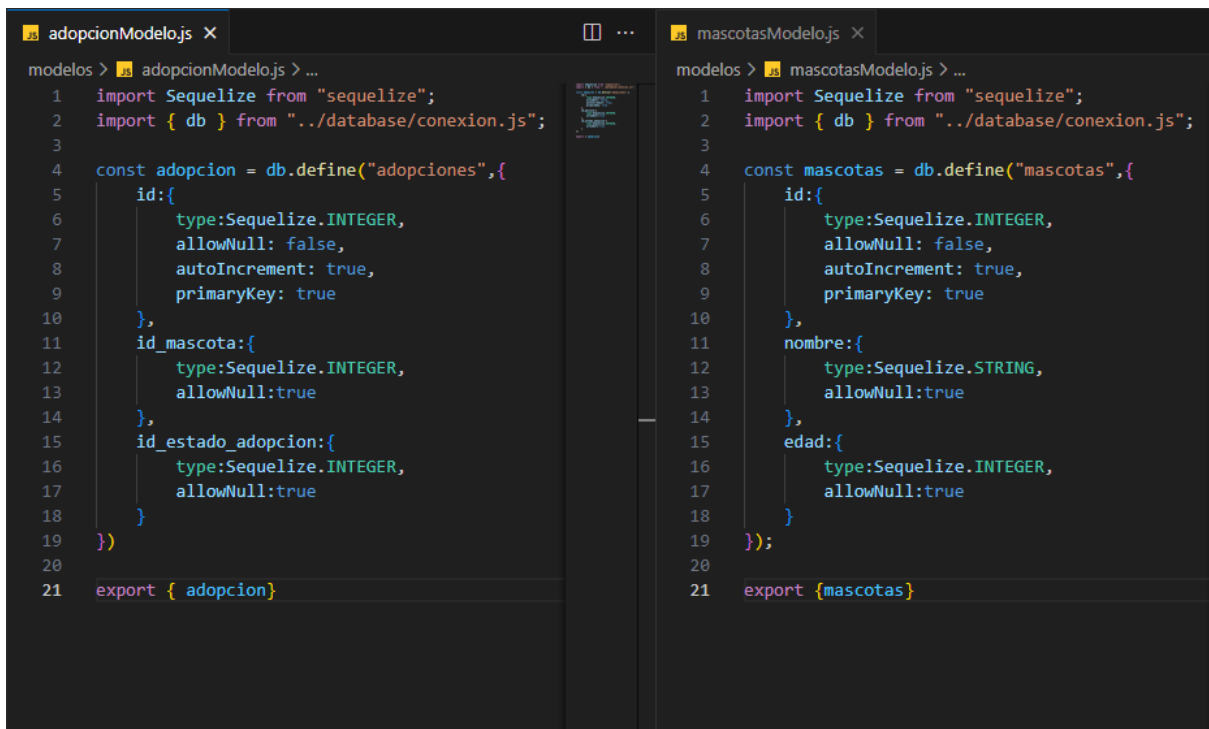


- c. creamos el archivo conexion.js donde damos las credenciales para la conexcion a la base de datos



```
1 import Sequelize from "sequelize";
2
3 const db = new Sequelize("mascotas", "mascotas", "mascotas", {
4   dialect: "mysql",
5   host: "localhost"
6 })
7
8 export {db}
```

- d. creamos los modelos donde se especifica la estructura de mascotas y adopcion



```
1 import Sequelize from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const adopcion = db.define("adopciones", {
5   id: {
6     type: Sequelize.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11   id_mascota: {
12     type: Sequelize.INTEGER,
13     allowNull: true
14  },
15   id_estado_adopcion: {
16     type: Sequelize.INTEGER,
17     allowNull: true
18  }
19 })
20
21 export { adopcion }
```

```
1 import Sequelize from "sequelize";
2 import { db } from "../database/conexion.js";
3
4 const mascotas = db.define("mascotas", {
5   id: {
6     type: Sequelize.INTEGER,
7     allowNull: false,
8     autoIncrement: true,
9     primaryKey: true
10  },
11   nombre: {
12     type: Sequelize.STRING,
13     allowNull: true
14  },
15   edad: {
16     type: Sequelize.INTEGER,
17     allowNull: true
18  }
19 })
20
21 export { mascotas }
```

- e. creamos los archivos controller donde se maneja las validaciones y las acciones a la base de datos

```
adopcionController.js X
controladores > adopcionController.js > ...
1 import { adopcion } from "../modelos/adopcionModelo.js";
2
3 //buscar todos
4 const buscar= (req, res)=>{
5
6     adopcion.findAll().then((resultado)->{
7         res.status(200).json(resultado);
8     }).catch((err)->{
9         res.status(500).json({
10             mensaje: "error: ${err}"
11         })
12     })
13 }
14
15 //buscar por id
16 const buscarId= (req, res)->{
17     const id = req.params.id;
18     if(id==null){
19         res.status(203).json({
20             mensaje: "id vacio"
21         });
22         return;
23     }
24     adopcion.findById(id).then((resultado)->{
25         res.status(200).json(resultado);
26     }).catch((err)->{
27         res.status(500).json({
28             mensaje: "error: ${err}"
29         })
30     })
31 }
32
33 //crear un recurso
34 const crear = (req,res)->{
35     if(!req.body.id_mascota && !req.body.id_estado_adopcion){
36         res.status(400).json({
37             mensaje: "dato en vacio"
38         });
39         return;
40     }
41     const dataset={
42         id_mascota: req.body.id_mascota,
43         id_estado_adopcion: req.body.id_estado_adopcion
44     }
45 }
46
47 adopcion.create(dataset).then((resultado)->{
48     res.status(200).json({
49         mensaje: "registrado"
50     })
51 }).catch((err)->{
52     res.status(500).json({
53         mensaje: "error: ${err}"
54     })
55 })
56
57 //actualizar
58 const actualizar=(req,res)->{
59     const id = req.params.id;
60     if(!req.body.id_mascota && !req.body.id_estado_adopcion){
61         res.status(400).json({
62             mensaje: "no se encontraron datos para actualizar"
63         });
64         return;
65     }
66     else{
67         const id_mascota= req.body.id_mascota;
68         const id_estado_adopcion= req.body.id_estado_adopcion
69         adopcion.update({id_mascota, id_estado_adopcion}, {where:{id}}).then((resultado)->{
70             res.status(200).json({
71                 mensaje: "actualizado"
72             })
73         }).catch((err)->{
74             res.status(500).json({
75                 mensaje: "error: ${err}"
76             })
77         })
78     }
79 }
80
81 //eliminar
82 const eliminar=(req,res)->{
83     const id = req.params.id;
84     if(id==null){
85         res.status(203).json({
86             mensaje: "id vacio"
87         });
88         return;
89     }
90     else{
91         adopcion.destroy({ where:{id} })
92         .then((resultado)->{
93             res.status(200).json({
94                 mensaje: "eliminado"
95             })
96         }).catch((err)->{
97             res.status(500).json({
98                 mensaje: "error: ${err}"
99             })
100         })
101     }
102 }
103
104 export { buscar, buscarId, crear, actualizar, eliminar}
```

```
mascoasController.js X
controladores > mascotasController.js > (0) buscar > catch() callback
1 import { mascotas } from "../modelos/mascotasModelo.js";
2
3 //buscar todos
4 const buscar= (req, res)=>{
5
6     mascotas.findAll().then((resultado)->{
7         res.status(200).json(resultado);
8     }).catch((err)->{
9         res.status(500).json({
10             mensaje: "error: ${err}"
11         })
12     })
13 }
14
15 //buscar por id
16 const buscarId= (req, res)->{
17     const id = req.params.id;
18     if(id==null){
19         res.status(203).json({
20             mensaje: "id vacio"
21         });
22         return;
23     }
24     mascotas.findById(id).then((resultado)->{
25         res.status(200).json(resultado);
26     }).catch((err)->{
27         res.status(500).json({
28             mensaje: "error: ${err}"
29         })
30     })
31 }
32
33 //crear un recurso
34 const crear = (req,res)->{
35     if(!req.body.nombre){
36         res.status(400).json({
37             mensaje: "nombre en vacio"
38         });
39         return;
40     }
41     const dataset={
42         nombre: req.body.nombre,
43         edad: req.body.edad
44     }
45 }
46
47 mascotas.create(dataset).then((resultado)->{
48     res.status(200).json({
49         mensaje: "registrado"
50     })
51 }).catch((err)->{
52     res.status(500).json({
53         mensaje: "error: ${err}"
54     })
55 })
56
57 //actualizar
58 const actualizar=(req,res)->{
59     const id = req.params.id;
60     if(!req.body.nombre && !req.body.edad){
61         res.status(400).json({
62             mensaje: "no se encontraron datos para actualizar"
63         });
64         return;
65     }
66     else{
67         const nombre= req.body.nombre;
68         const edad= req.body.edad;
69         mascotas.update({nombre, edad}, {where:{id}}).then((resultado)->{
70             res.status(200).json({
71                 mensaje: "actualizado"
72             })
73         }).catch((err)->{
74             res.status(500).json({
75                 mensaje: "error: ${err}"
76             })
77         })
78     }
79 }
80
81 //eliminar
82 const eliminar=(req,res)->{
83     const id = req.params.id;
84     if(id==null){
85         res.status(203).json({
86             mensaje: "id vacio"
87         });
88         return;
89     }
90     else{
91         mascotas.destroy({ where:{id} })
92         .then((resultado)->{
93             res.status(200).json({
94                 mensaje: "eliminado"
95             })
96         }).catch((err)->{
97             res.status(500).json({
98                 mensaje: "error: ${err}"
99             })
100         })
101     }
102 }
103
104 export { crear, buscar, buscarId, actualizar, eliminar}
```

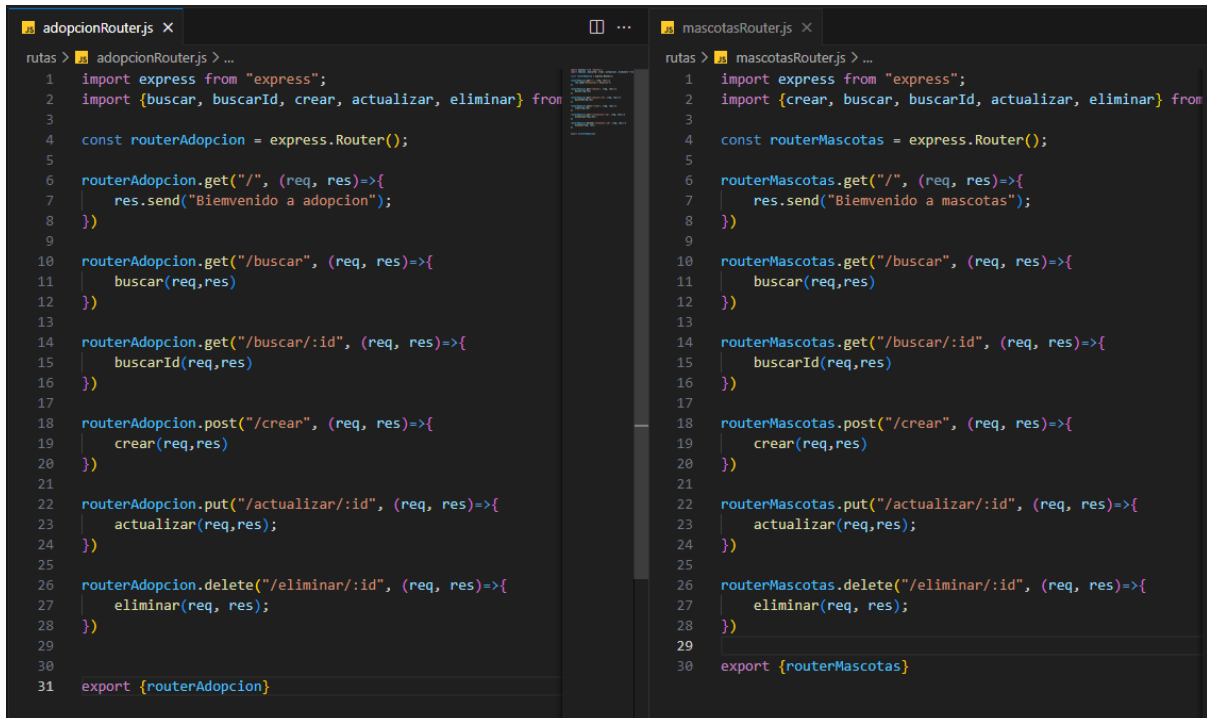
```
adopcionController.js X
controladores > adopcionController.js > ...
44 }
45
46 adopcion.create(dataset).then((resultado)->{
47     res.status(200).json({
48         mensaje: "registrado"
49     })
50 }).catch((err)->{
51     res.status(500).json({
52         mensaje: "error: ${err}"
53     })
54 })
55
56 //actualizar
57 const actualizar=(req,res)->{
58     const id = req.params.id;
59     if(!req.body.id_mascota && !req.body.id_estado_adopcion){
60         res.status(400).json({
61             mensaje: "no se encontraron datos para actualizar"
62         });
63         return;
64     }
65     else{
66         const id_mascota= req.body.id_mascota;
67         const id_estado_adopcion= req.body.id_estado_adopcion
68         adopcion.update({id_mascota, id_estado_adopcion}, {where:{id}}).then((resultado)->{
69             res.status(200).json({
70                 mensaje: "actualizado"
71             })
72         }).catch((err)->{
73             res.status(500).json({
74                 mensaje: "error: ${err}"
75             })
76         })
77     }
78 }
79
80 //eliminar
81 const eliminar=(req,res)->{
82     const id = req.params.id;
83     if(id==null){
84         res.status(203).json({
85             mensaje: "id vacio"
86         });
87         return;
88     }
89     else{
90         adopcion.destroy({ where:{id} })
91         .then((resultado)->{
92             res.status(200).json({
93                 mensaje: "eliminado"
94             })
95         }).catch((err)->{
96             res.status(500).json({
97                 mensaje: "error: ${err}"
98             })
99         })
100     }
101 }
102
103 export { buscar, buscarId, crear, actualizar, eliminar}
```

```
mascoasController.js X
controladores > mascotasController.js > (0) buscar > catch() callback
44 }
45
46 mascotas.create(dataset).then((resultado)->{
47     res.status(200).json({
48         mensaje: "registrado"
49     })
50 }).catch((err)->{
51     res.status(500).json({
52         mensaje: "error: ${err}"
53     })
54 })
55
56 //actualizar
57 const actualizar=(req,res)->{
58     const id = req.params.id;
59     if(!req.body.nombre && !req.body.edad){
60         res.status(400).json({
61             mensaje: "no se encontraron datos para actualizar"
62         });
63         return;
64     }
65     else{
66         const nombre= req.body.nombre;
67         const edad= req.body.edad;
68         mascotas.update({nombre, edad}, {where:{id}}).then((resultado)->{
69             res.status(200).json({
70                 mensaje: "actualizado"
71             })
72         }).catch((err)->{
73             res.status(500).json({
74                 mensaje: "error: ${err}"
75             })
76         })
77     }
78 }
79
80 //eliminar
81 const eliminar=(req,res)->{
82     const id = req.params.id;
83     if(id==null){
84         res.status(203).json({
85             mensaje: "id vacio"
86         });
87         return;
88     }
89     else{
90         mascotas.destroy({ where:{id} })
91         .then((resultado)->{
92             res.status(200).json({
93                 mensaje: "eliminado"
94             })
95         }).catch((err)->{
96             res.status(500).json({
97                 mensaje: "error: ${err}"
98             })
99         })
100     }
101 }
102
103 export { crear, buscar, buscarId, actualizar, eliminar}
```

```
adopcionController.js X
controladores > adopcionController.js > ...
86 if(id==null){
87     res.status(203).json({
88         mensaje: "id vacio"
89     });
90     return;
91 }
92 else{
93     adopcion.destroy({ where:{id} })
94     .then((resultado)->{
95         res.status(200).json({
96             mensaje: "eliminado"
97         })
98     }).catch((err)->{
99         res.status(500).json({
100             mensaje: "error: ${err}"
101         })
102     })
103 }
104
105 export { buscar, buscarId, crear, actualizar, eliminar}
```

```
mascoasController.js X
controladores > mascotasController.js > (0) buscar > catch() callback
86 const id = req.params.id;
87 if(id==null){
88     res.status(203).json({
89         mensaje: "id vacio"
90     });
91     return;
92 }
93 else{
94     mascotas.destroy({ where:{id} })
95     .then((resultado)->{
96         res.status(200).json({
97             mensaje: "eliminado"
98         })
99     }).catch((err)->{
100         res.status(500).json({
101             mensaje: "error: ${err}"
102         })
103     })
104 }
105
106 export { crear, buscar, buscarId, actualizar, eliminar}
```

f. creamos los archivos router, donde se especifica las rutas para el crud



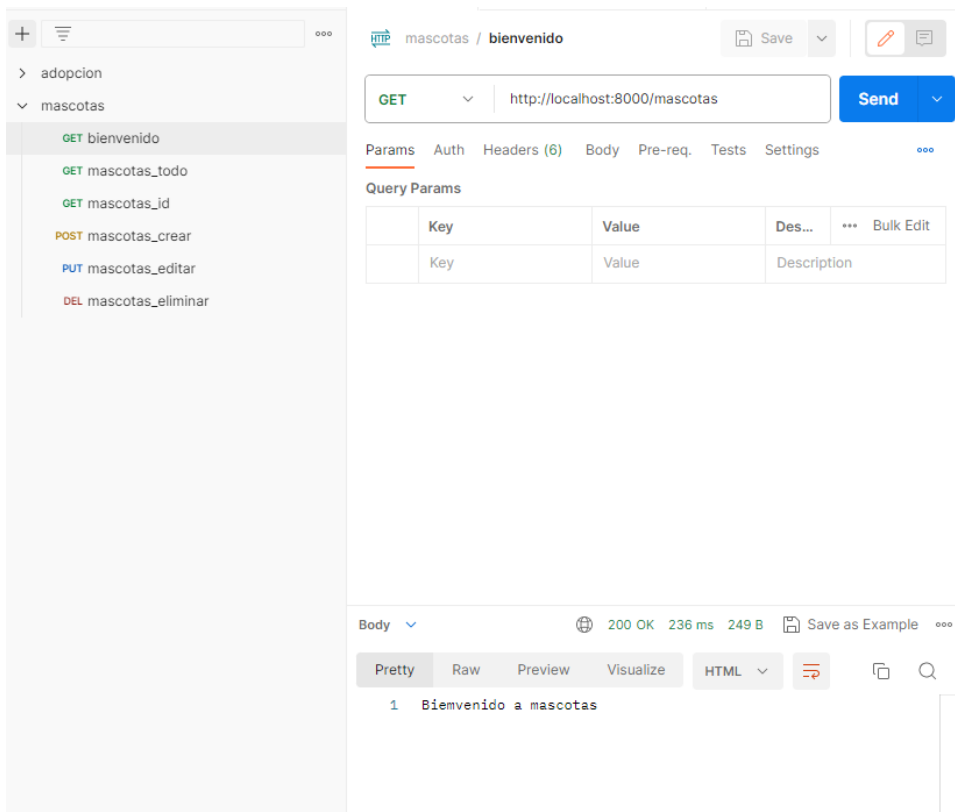
The image shows two side-by-side code editors. The left editor is named 'adopcionRouter.js' and the right is 'mascotasRouter.js'. Both files define Express.js routers with routes for a CRUD application. The 'adopcionRouter.js' file includes routes for a general welcome message, searching for adoptions, and creating, updating, and deleting adoptions. The 'mascotasRouter.js' file includes routes for a general welcome message, searching for mascots, and creating, updating, and deleting mascots. Both files use the 'express' module and the 'express.Router()' function to create the routers.

```
adopcionRouter.js
1 import express from "express";
2 import {buscar, buscarId, crear, actualizar, eliminar} from ...
3
4 const routerAdopcion = express.Router();
5
6 routerAdopcion.get("/", (req, res)=>{
7   res.send("Bienvenido a adopcion");
8 })
9
10 routerAdopcion.get("/buscar", (req, res)=>{
11   buscar(req,res)
12 })
13
14 routerAdopcion.get("/buscar/:id", (req, res)=>{
15   buscarId(req,res)
16 })
17
18 routerAdopcion.post("/crear", (req, res)=>{
19   crear(req,res)
20 })
21
22 routerAdopcion.put("/actualizar/:id", (req, res)=>{
23   actualizar(req,res);
24 })
25
26 routerAdopcion.delete("/eliminar/:id", (req, res)=>{
27   eliminar(req, res);
28 })
29
30
31 export {routerAdopcion}
```

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

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.

a. verificación para mascotas con postman



+

...

> adopcion

> mascotas

GET bienvenido

GET mascotas_todo

GET mascotas_id

POST mascotas_crear

PUT mascotas_editar

DEL mascotas_eliminar

mascoas / mascotas_todo

Save

GET

http://localhost:8000/mascotas/buscar

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

...

Query Params

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Body

200 OK 390 ms 574 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 2,
3    "nombre": "Toby",
4    "edad": 1,
5    "createdAt": "2023-12-20",
6    "updatedAt": "2023-12-22"
7  },
8  {
9    "id": 3,
10   "nombre": "Toby",
11   "edad": 2,
12   "createdAt": "2023-12-22",
13   "updatedAt": "2023-12-22"
14 },
15 {
16   "id": 4,
```

+

...

> adopcion

> mascotas

GET bienvenido

GET mascotas_todo

GET mascotas_id

POST mascotas_crear

PUT mascotas_editar

DEL mascotas_eliminar

mascoas / mascotas_todo

Save

GET

http://localhost:8000/mascotas/buscar/2

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

...

Query Params

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Body

200 OK 84 ms 318 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 2,
3    "nombre": "Toby",
4    "edad": 1,
5    "createdAt": "2023-12-20",
6    "updatedAt": "2023-12-22"
7  }
```


Postman interface showing a DELETE request to `mascotas/eliminar/3`. The response body is a JSON object: `{\"mensaje\": \"eliminado\"}`.

Left sidebar (API Explorer):

- adopcion
 - GET bienvenido
 - GET mascotas_todo
 - GET mascotas_id
 - POST mascotas_crear
 - PUT mascotas_editar
 - DEL mascotas_eliminar

Request details:

- Method: DELETE
- URL: `http://localhost:8000/mascotas/eliminar/3`
- Params: Auth, Headers (6), Body, Pre-req., Tests, Settings
- Query Params table:

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Response details:

- Status: 200 OK, 32 ms, 258 B
- Body: Pretty, Raw, Preview, Visualize, JSON, Copy, Search
- Response body (JSON):

```
1 {
2   "mensaje": "eliminado"
3 }
```

b. verificación para adopcion con postman

Postman interface showing a GET request to `adopcion`. The response body is HTML: `<p>Bienvenido a adopcion</p>`.

Left sidebar (API Explorer):

- adopcion
 - GET bienvenido
 - GET adopcion_todo
 - GET adopcion_id
 - POST adopcion_crear
 - PUT adopcion_editar
 - DEL adopcion_eliminar
- mascotas

Request details:

- Method: GET
- URL: `http://localhost:8000/adopcion`
- Params: Auth, Headers (6), Body, Pre-req., Tests, Settings
- Query Params table:

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Response details:

- Status: 200 OK, 20 ms, 249 B
- Body: Pretty, Raw, Preview, Visualize, HTML, Copy, Search
- Response body (HTML):

```
1 <p>Bienvenido a adopcion</p>
```

+

...

adopcion

GET bienvenido

GET adopcion_todo

GET adopcion_id

POST adopcion_crear

PUT adopcion_editar

DEL adopcion_eliminar

> mascotas

adopcion / adopcion_todo

Save

GET

http://localhost:8000/adopcion/buscar

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

...

Query Params

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Body

200 OK 145 ms 723 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   {
3     "id": 2,
4     "id_mascota": 2,
5     "id_estado_adopcion": 1,
6     "createdAt": "2023-12-20",
7     "updatedAt": "2023-12-20"
8   },
9   {
10    "id": 3,
11    "id_mascota": 2,
12    "id_estado_adopcion": 1,
13    "createdAt": "2023-12-22",
14    "updatedAt": "2023-12-22"
15  },
16  {
17    "id": 4,
18    "id_mascota": 2
```

+

...

adopcion

GET bienvenido

GET adopcion_todo

GET adopcion_id

POST adopcion_crear

PUT adopcion_editar

DEL adopcion_eliminar

> mascotas

adopcion / adopcion_id

Save

GET

http://localhost:8000/adopcion/buscar/2

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

...

Query Params

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Body

200 OK 55 ms 331 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   {
3     "id": 2,
4     "id_mascota": 2,
5     "id_estado_adopcion": 1,
6     "createdAt": "2023-12-20",
7     "updatedAt": "2023-12-20"
```

adopcion / **adopcion_crear** Save Send

POST http://localhost:8000/adopcion/crear

Params Auth Headers (8) **Body** Pre-req. Tests Settings

raw **JSON** Beautify

```
1 {
2   ... "id_mascota": 2,
3   ... "id_estado_adopcion": 1
4 }
```

Body 200 OK 43 ms 259 B Save as Example

Pretty Raw Preview Visualize **JSON** Beautify

```
1 {
2   "mensaje": "registrado"
3 }
```

adopcion / **adopcion_editar** Save Send

PUT http://localhost:8000/adopcion/actualizar/1

Params Auth Headers (8) **Body** Pre-req. Tests Settings

raw **JSON** Beautify

```
1 {
2   ... "id_mascota": 4
3 }
```

Body 200 OK 21 ms 260 B Save as Example

Pretty Raw Preview Visualize **JSON** Beautify

```
1 {
2   "mensaje": "actualizado"
3 }
```

+

...

adopcion

GET

bienvenido

GET

adopcion_todo

GET

adopcion_id

POST

adopcion_crear

PUT

adopcion_editar

DEL

adopcion_eliminar

>

mascotas

adopcion / adopcion_eliminar

Save

DELETE

http://localhost:8000/adopcion/eliminar/1

Send

Params

Auth

Headers (6)

Body

Pre-req.

Tests

Settings

...

Query Params

	Key	Value	Des...	...	Bulk Edit
	Key	Value	Description		

Body

200 OK

14 ms

258 B

Save as Example

...

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"mensaje": "eliminado"