

**Universidad de Nariño.**

**Ingeniería de Sistemas.**

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

**Presentado por:**

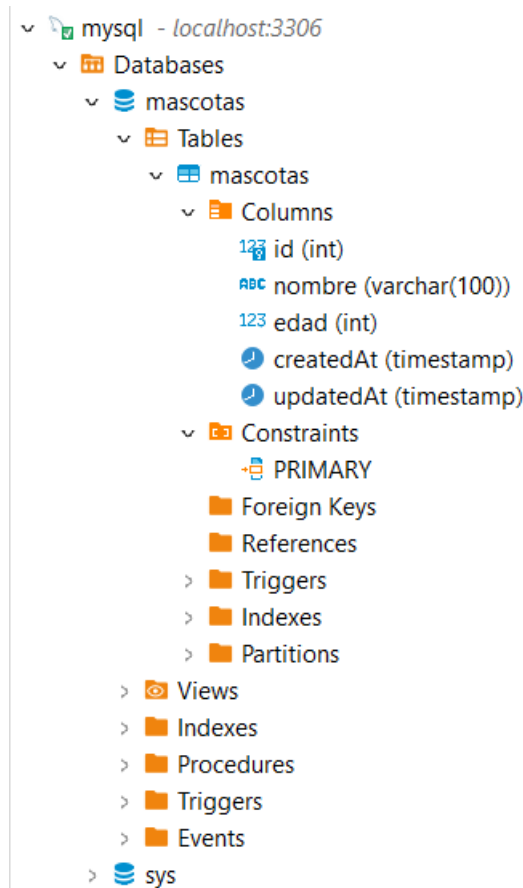
Darwin Estiven Diaz Espinosa

Código: 219036069

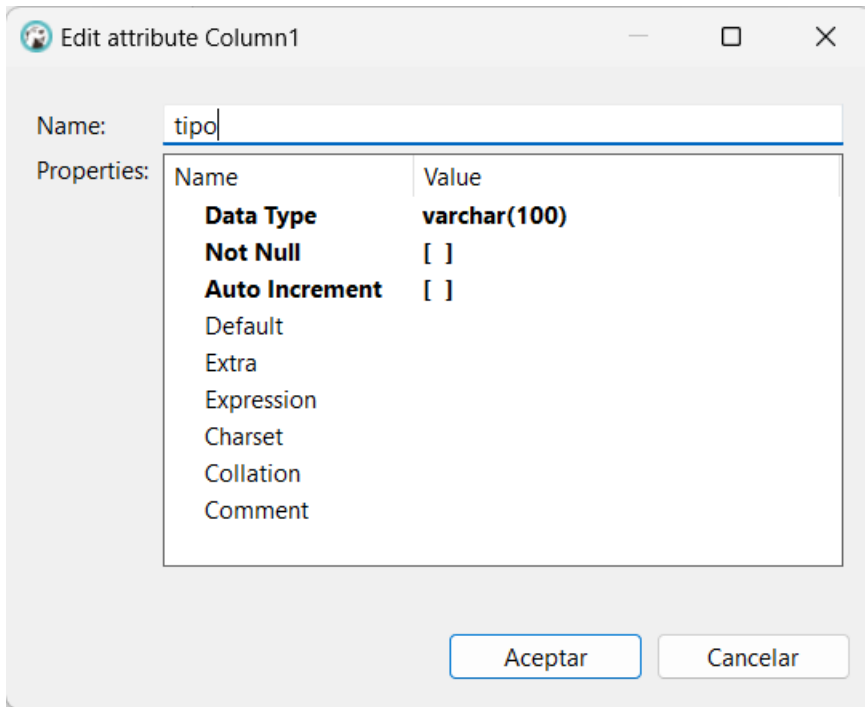
Celular: 3116282163

### **Taller Unidad 2 Backend**

Vamos a continuar trabajando con la base de datos que realizamos en clases, pero se van a agregar otra tabla y otros campos como se visualizará a continuación:



En la tabla mascotas agregamos el atributo “tipo” en el cual se podrá registrar si es perro o gato.

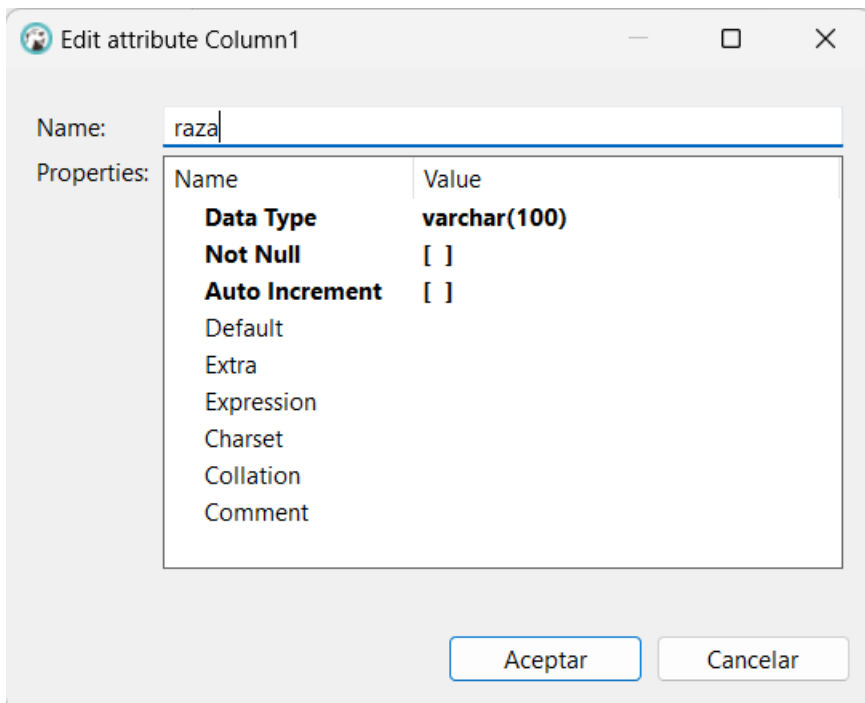


The screenshot shows a dialog box titled "Edit attribute Column1". The "Name:" field contains the text "tipo". Below it, the "Properties:" section contains a table with the following data:

Name	Value
<b>Data Type</b>	<b>varchar(100)</b>
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

At the bottom of the dialog box, there are two buttons: "Aceptar" and "Cancelar".

De igual manera agregamos el atributo “raza” de la mascota.

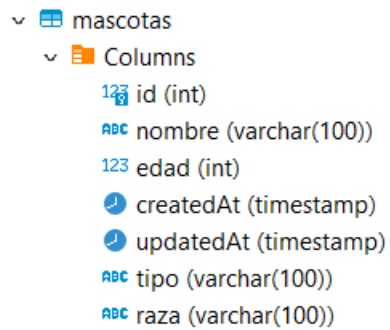


The screenshot shows a dialog box titled "Edit attribute Column1". The "Name:" field contains the text "raza". Below it, the "Properties:" section contains a table with the following data:

Name	Value
<b>Data Type</b>	<b>varchar(100)</b>
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

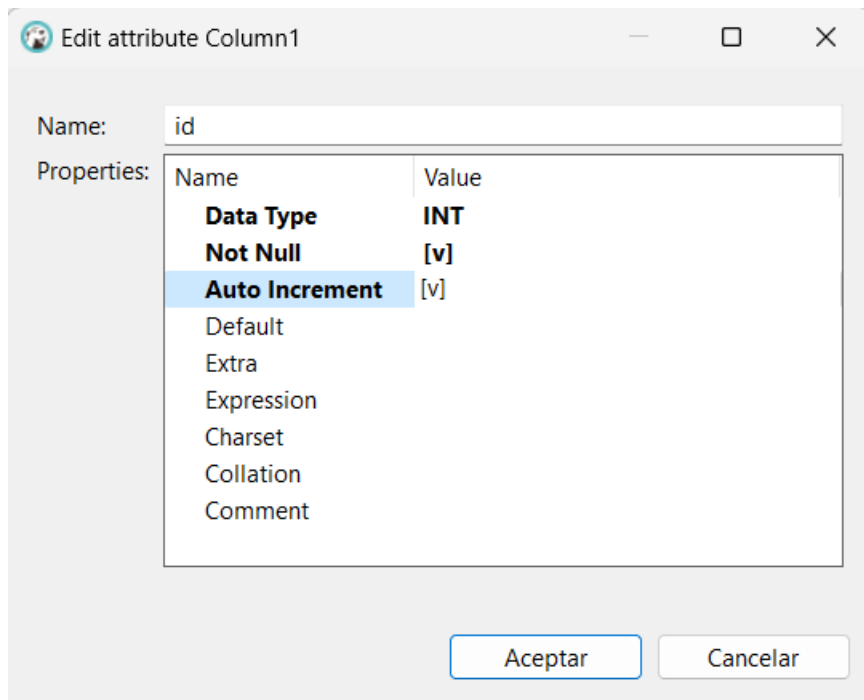
At the bottom of the dialog box, there are two buttons: "Aceptar" and "Cancelar".

Por lo tanto, la tabla “mascotas” quedaría así:



Creamos una nueva tabla llamada “adopciones” y agregamos los siguientes atributos:

id de adopcion



id\_mascota es la llave foranea

The screenshot shows a dialog box titled "Edit attribute Column1". The "Name:" field contains "id\_mascota". The "Properties:" section is a table with two columns: "Name" and "Value".

Name	Value
<b>Data Type</b>	INT
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

At the bottom of the dialog are two buttons: "Aceptar" and "Cancelar".

nombre\_solicitante: es el nombre de la persona que va a adoptar la mascota

The screenshot shows a dialog box titled "Edit attribute Column1". The "Name:" field contains "nombre\_solicitante". The "Properties:" section is a table with two columns: "Name" and "Value".

Name	Value
<b>Data Type</b>	varchar(100)
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

At the bottom of the dialog are two buttons: "Aceptar" and "Cancelar".

telefono de la persona que va a adoptar

Edit attribute Column1

Name: telefono

Properties:

Name	Value
<b>Data Type</b>	INT
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

Aceptar Cancelar

estado de adopción: pendiente, aprobada y rechazada

Edit attribute Column1

Name: estado

Properties:

Name	Value
<b>Data Type</b>	varchar(100)
<b>Not Null</b>	[ ]
<b>Auto Increment</b>	[ ]
Default	
Extra	
Expression	
Charset	
Collation	
Comment	

Aceptar Cancelar

Agregamos id como Clave primaria

Create constraint for table "adopcions"

Table: mascotas.adopcions

Name: adopcions\_pk

Type: PRIMARY KEY

Columns:

Column	#	Type
<input checked="" type="checkbox"/> 123 id	1	int
<input type="checkbox"/> 123 id_mascotas		int
<input type="checkbox"/> ABC nombre_solicitante		varchar(255)
<input type="checkbox"/> 123 telefono		int
<input type="checkbox"/> ABC estado		varchar(255)
<input type="checkbox"/> createdAt		datetime
<input type="checkbox"/> updatedAt		datetime

Clear All

AceptarCancelar

Agregamos id\_mascota como Clave foránea

Create foreign key | Create foreign key for table "..."

Table: mascotas.adopcions

Container: mascotas

Reference table:

adopcions	
mascotas	

Unique Key: PRIMARY (Primary Key)

Columns:

Column	Column Type	Ref Column	Ref Column Type
id_mascotas	int	id	int

On Delete: No Action On Update: No Action

Aceptar Cancelar

Así quedaría estructurada la tabla adopcions

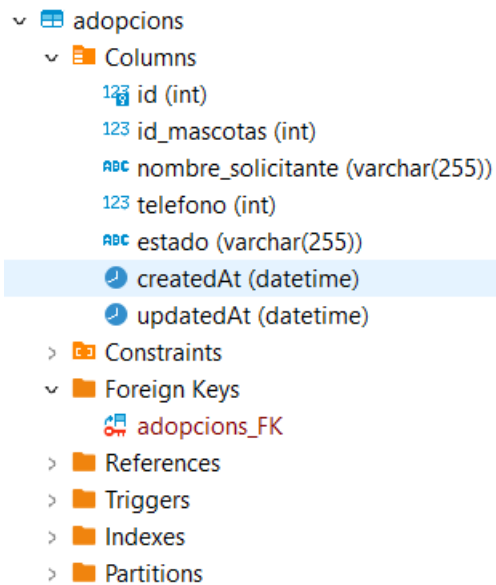
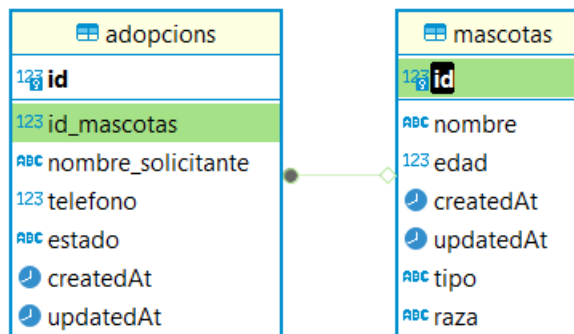


Diagrama entidad – relación de la base de datos “mascotas”





En el controlador mascotas hacemos las siguientes modificaciones de acuerdo con lo que ya se tenía de lo visto en clase

```
//Crear un recurso
const crear = (req,res)=>{
  if(!req.body.nombre){
    res.status(400).json({
      mensaje: "El nombre no puede estar vacio."
    });
    return;
  }
  const dataset={
    nombre: req.body.nombre,
    edad: req.body.edad,
    tipo: req.body.tipo,
    raza: req.body.raza
  };

  //Usar Sequelize para crear el recurso
  mascotas.create(dataset).then((resultado)=>{
    res.status(200).json({
      mensaje: "Registro creado correctamente"
    })
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `Error al crear el registro ::: ${err}`
    })
  })
};
```

Esos son los campos que se agrego a la tabla mascotas por lo tanto se debió agregarlos en el código para que haga el registro de todos los atributos.

Para actualizar los registros también se modificó unas líneas de código que son las siguientes:

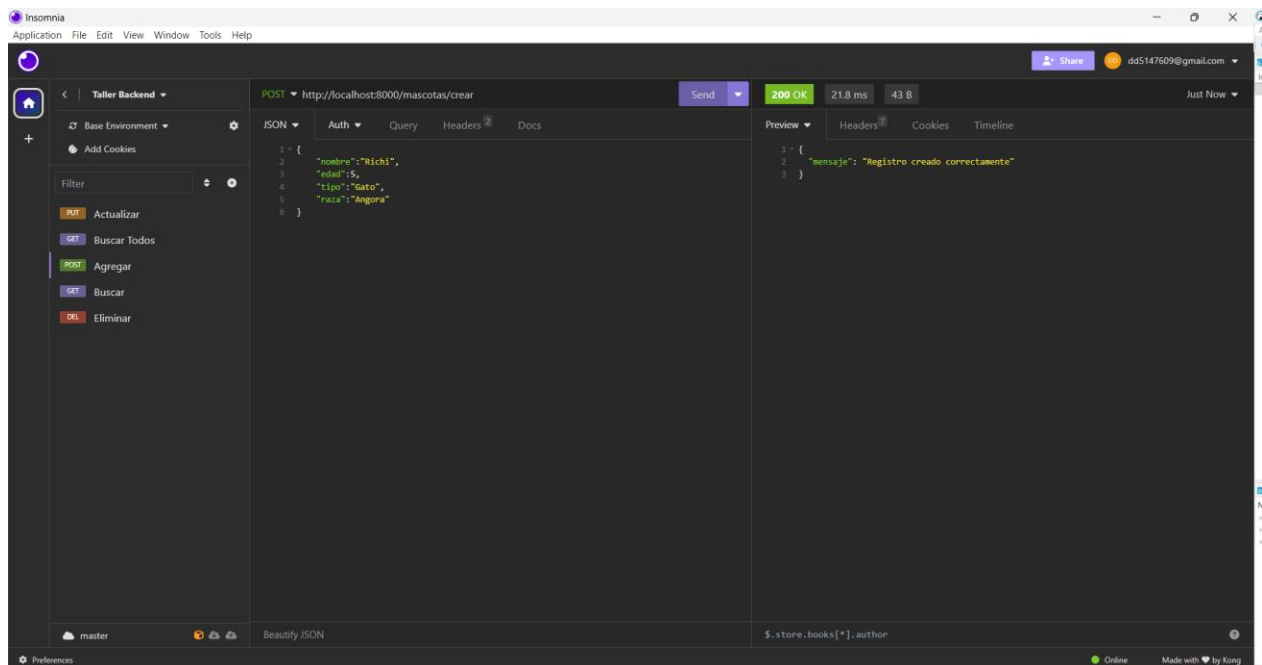
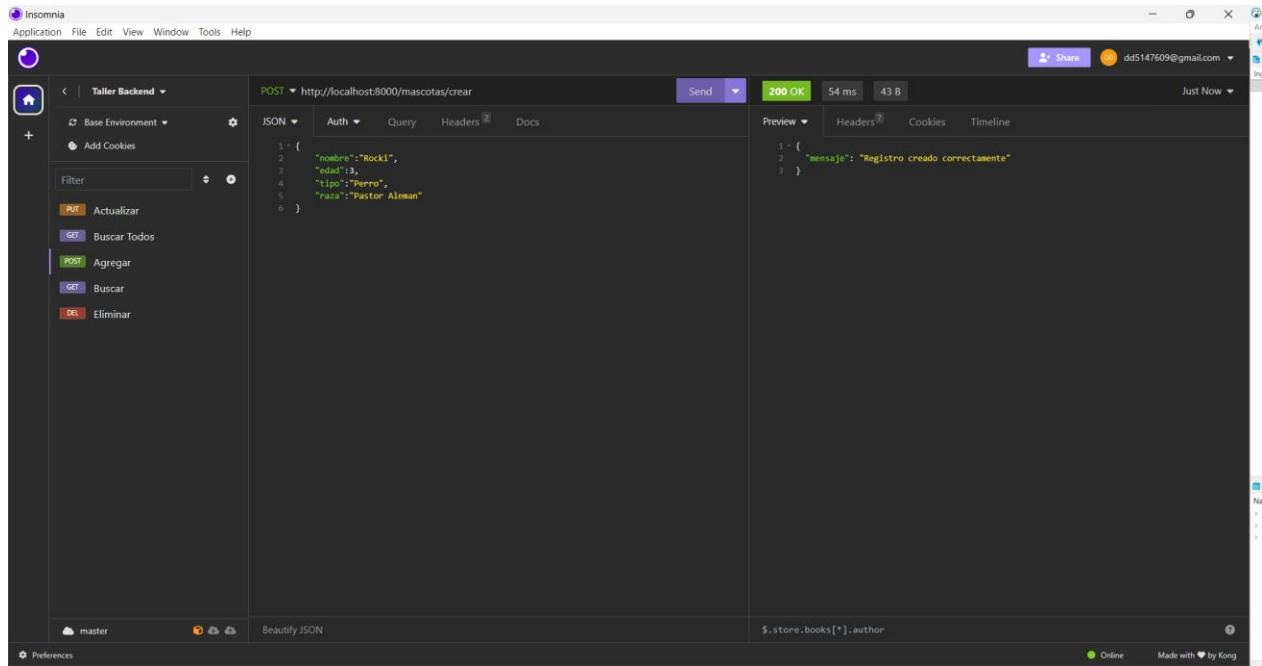
```
//Actualizar un recurso
const actualizar=(req,res)=>{
  const id= req.params.id;
  if(!req.body.nombre && !req.body.edad && !req.body.tipo && !req.body.raza){
    res.status(400).json({
      mensaje: `No se encontraron Datos para Actualizar`
    });
    return;
  }
  else{
    const nombre= req.body.nombre;
    const edad= req.body.edad;
    const tipo= req.body.tipo;
    const raza= req.body.raza;
    mascotas.update({nombre,edad,tipo,raza},{where:{id}})
      .then((resultado)=>{
        res.status(200).json({
          mensaje: `Registro Actualizado`
        });
      })
      .catch((err)=>{
        res.status(500).json({
          mensaje: `Error al actualizar Registro ::: ${err}`
        });
      })
  }
};
```

Se agregó el tipo y raza de la mascota

Lo demás queda tal y como se lo trabajó en clases.

A continuación, se visualizará las capturas de pantalla de las solicitudes realizadas a esta tabla con Imnsomia:

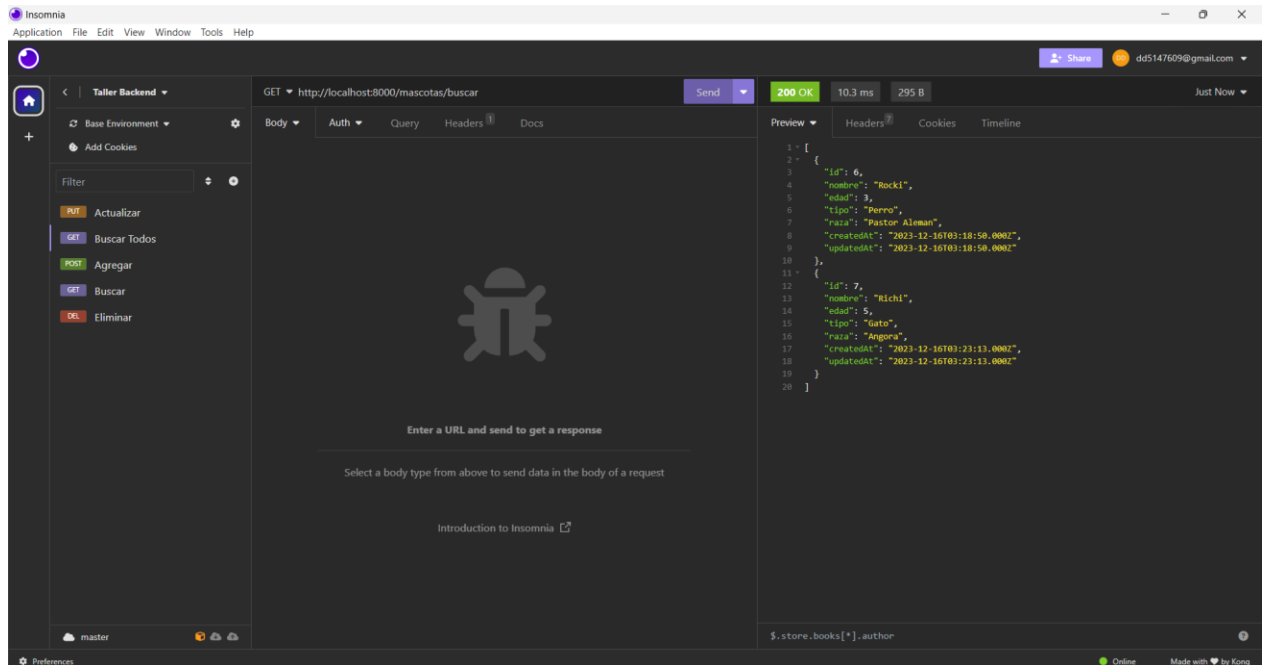
Aquí se agregan datos a la tabla mascotas:



Aquí se pueden ver los datos en la base de datos:

Propiedades Datos Diagrama ER							
mascotas Enter a SQL expression to filter results (use Ctrl+Space)							
Grilla	id	nombre	edad	createdAt	updatedAt	tipo	raza
1	6	Rocki	3	2023-12-15 22:18:50	2023-12-15 22:18:50	Perro	Pastor Aleman
2	7	Richi	5	2023-12-15 22:23:13	2023-12-15 22:23:13	Gato	Angora

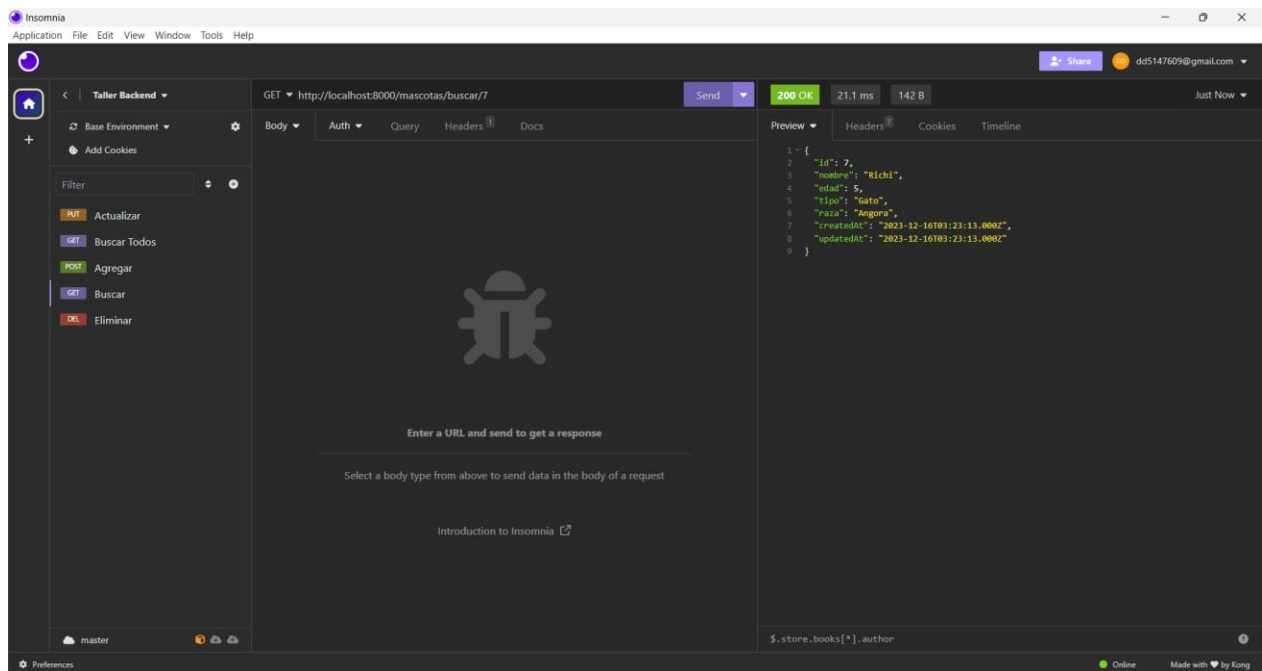
Aquí se visualiza la consulta de todos los datos ingresados:



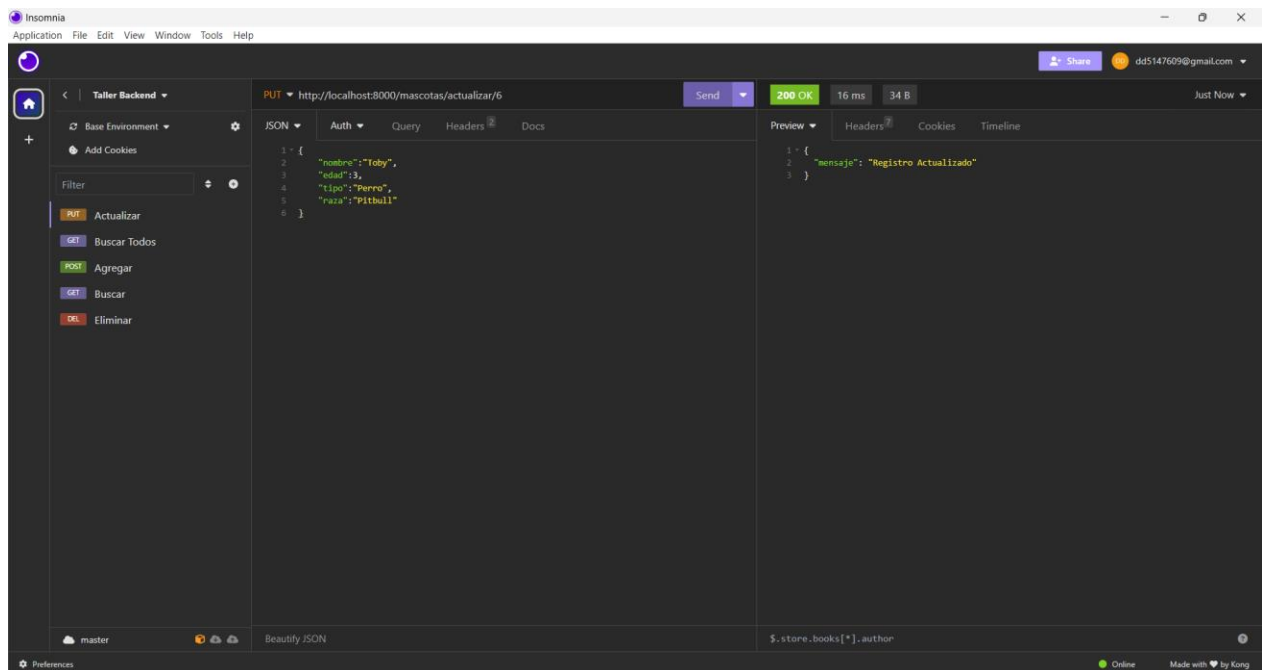
The screenshot shows the Insomnia application interface. On the left, there's a sidebar with a 'Filter' section and a list of actions: 'Actualizar', 'Buscar Todos', 'Agregar', 'Buscar', and 'Eliminar'. The main area displays a GET request to 'http://localhost:8000/mascotas/buscar'. The response is a JSON array with two objects, each representing a pet. The first object is for 'Rocki', a 3-year-old German Shepherd, and the second is for 'Richi', a 5-year-old Angora cat. The response status is '200 OK' with a response time of '10.3 ms' and a size of '295 B'.

```
1 [
2   {
3     "id": 6,
4     "nombre": "Rocki",
5     "edad": 3,
6     "tipo": "Perro",
7     "raza": "Pastor Aleman",
8     "createdAt": "2023-12-15T22:18:50.000Z",
9     "updatedAt": "2023-12-15T22:18:50.000Z"
10  },
11  {
12    "id": 7,
13    "nombre": "Richi",
14    "edad": 5,
15    "tipo": "Gato",
16    "raza": "Angora",
17    "createdAt": "2023-12-15T22:23:13.000Z",
18    "updatedAt": "2023-12-15T22:23:13.000Z"
19  }
20 ]
```

Aquí se visualiza el dato buscado por id:



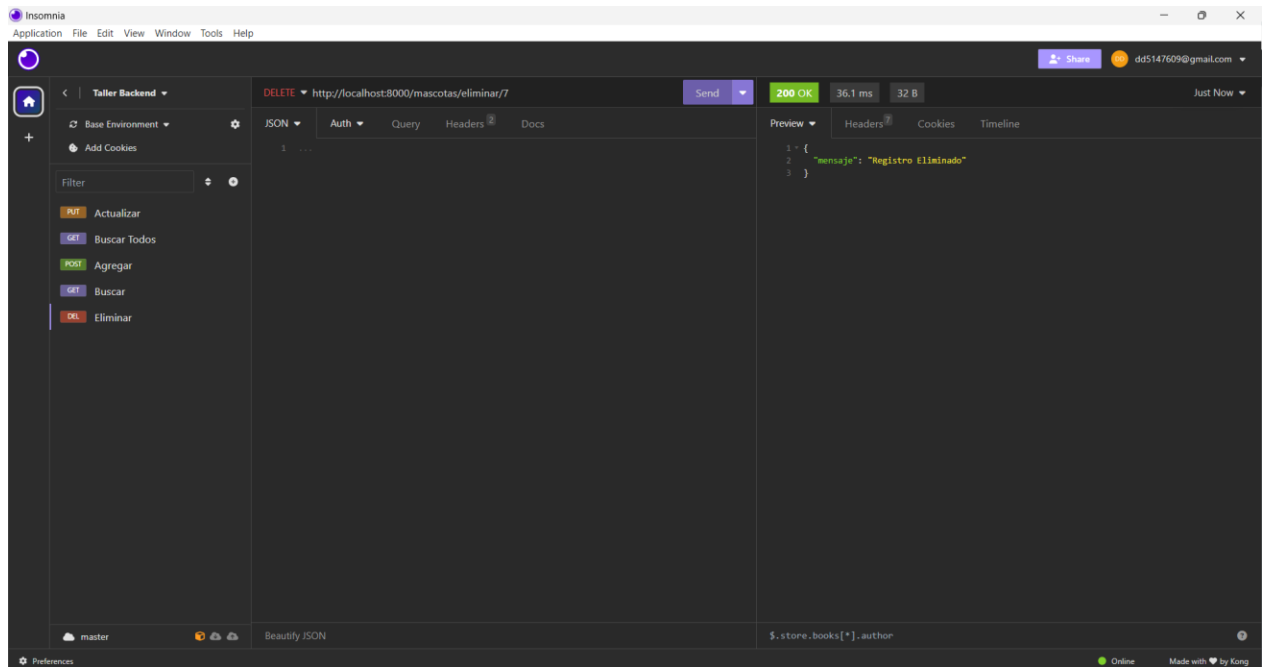
Aquí se muestra la consulta de actualizar:



Aquí se puede ver el dato con id=6 actualizado en la base de datos

mascotas		Enter a SQL expression to filter results (use Ctrl+Space)					
	id	nombre	edad	createdAt	updatedAt	tipo	raza
1	6	Toby	3	2023-12-15 22:18:50	2023-12-15 22:27:55	Perro	Pitbull
2	7	Richi	5	2023-12-15 22:23:13	2023-12-15 22:23:13	Gato	Angora

Aquí se observa la consulta eliminar y se elimina los datos con id=7:

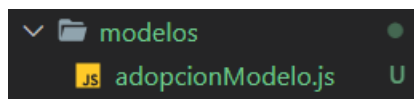


En la base de datos también se visualiza que ya no está el registro

mascotas <small>Enter a SQL expression to filter results (use Ctrl+Space)</small>								
	id	nombre	edad	createdAt	updatedAt	tipo	raza	
1	6	Toby	3	23-12-15 22:18:50	023-12-15 22:27:55	Perro	Pitbull	

Para la inserción de datos en la tabla adopcions se creó lo siguiente:

Creación del modelo:



El cual contiene lo siguiente:

```
modelos > Js adopcionModelo.js > adopcion
1  import Sequelize from "sequelize";
2  import {db} from "../database/conexion.js";
3
4  const adopcion = db.define("adopcions",{
5      id:{
6          type:Sequelize.INTEGER,
7          allowNull: false,
8          autoIncrement: true,
9          primaryKey: true
10     },
11     id_mascotas:{
12         type: Sequelize.INTEGER,
13         allowNull: true
14     },
15     nombre_solicitante:{
16         type: Sequelize.STRING,
17         allowNull: true
18     },
19     telefono:{
20         type: Sequelize.INTEGER,
21         allowNull: true
22     },
23     estado:{
24         type: Sequelize.STRING,
25         allowNull: true
26     }
27 });
28
29 export {adopcion}
```

**Creación del controlador:**

```
▼ controladores
  Js adopcionController.js U
```

El cual contiene lo siguiente:

Función crear:

```

//Crear
const crear = (req, res) => {
  // Verificar si el idMascota está presente y no es nulo
  if (!req.body.id_mascotas || !req.body.nombre_solicitante || !req.body.telefono) {
    return res.status(400).json({
      mensaje: "id_mascotas, nombre_solicitante y telefono son campos obligatorios y no pueden ser nulos."
    });
  }

  // Verificar si la mascota con el idMascota existe y está disponible
  mascotas.findByPk(req.body.id_mascotas)
    .then((mascota) => {
      if (!mascota) {
        return res.status(404).json({
          mensaje: `La mascota con el id ${req.body.id_mascotas} proporcionado no existe`
        });
      }

      // Crear un objeto dataset con los campos relevantes
      const dataset = {
        id_mascotas: req.body.id_mascotas,
        nombre_solicitante: req.body.nombre_solicitante,
        telefono: req.body.telefono,
        estado: req.body.estado
      };

      // Utilizar Sequelize para crear el recurso
      adoption.create(dataset)
        .then((resultado) => {
          res.status(200).json({
            mensaje: "Registro creado correctamente",
            resultado: resultado
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: `Error al crear el registro: ${err.message}`
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al buscar la mascota: ${err.message}`
      });
    });
};

```



Función para buscar los registros por id:

```
//Buscar recurso por ID
const buscarId = (req,res)=>{
  const id = req.params.id;
  if(id == null){
    res.status(203).json({
      mensaje: `El id no puede estar vacio`
    });
    return;
  }

  adopcion.findByIdPk(id).then((resultado)=>{
    res.status(200).json(resultado);
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `Registro no encontrado ::: ${err}`
    });
  });
}
```

Función para listar todos los registros:

```
//Buscar recurso todos
const buscar = (req,res)=>{
  adopcion.findAll().then((resultado)=>{
    res.status(200).json(resultado);
  }).catch((err)=>{
    res.status(500).json({
      mensaje: `No se encontraron Registros ::: ${err}`
    });
  });
};
```

Función para actualizar registros:

```
//Actualizar un recurso
const actualizar = (req, res) => {
  const id = req.params.id;

  // Verificar si existen datos para actualizar
  if (!req.body.id_mascotas && !req.body.nombre_solicitante && !req.body.telefono && !req.body.estado) {
    res.status(400).json({
      mensaje: `No se encontraron datos para actualizar`
    });
    return;
  } else {
    const id_mascotas = req.body.id_mascotas;
    const nombre_solicitante = req.body.nombre_solicitante;
    const telefono = req.body.telefono;
    const estado = req.body.estado;

    // Validar existencia de la clave foránea (id_mascota)
    mascotas.findByPk(id_mascotas) // Reemplaza "Mascota" con el nombre real de tu modelo de mascota
      .then((mascota) => {
        if (!mascota) {
          res.status(400).json({
            mensaje: `La mascota con id ${id_mascotas} no existe. No se puede actualizar el registro.`
          });
        } else {
          // La mascota existe, realizar la actualización
          adopcion.update({ id_mascotas, nombre_solicitante, telefono, estado }, { where: { id } })
            .then((resultado) => {
              res.status(200).json({
                mensaje: `Registro actualizado`
              });
            })
            .catch((err) => {
              res.status(500).json({
                mensaje: `Error al actualizar registro: ${err}`
              });
            });
        }
      })
      .catch((err) => {
        res.status(500).json({
          mensaje: `Error al verificar la existencia de la mascota: ${err}`
        });
      });
  }
};
```

Función para eliminar:

```
//Eliminar registros
const eliminar = (req, res) => {
  const id = req.params.id;

  if (id == null) {
    res.status(203).json({
      mensaje: `El id no puede estar vacío`
    });
    return;
  }

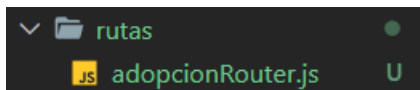
  // Verificar si el registro de solicitud existe antes de intentar eliminar
  adopcion.findByIdPk(id)
    .then((registroExistente) => {
      if (!registroExistente) {
        return res.status(404).json({
          mensaje: "Registro no encontrado"
        });
      }

      // El registro existe, ahora procedemos con la eliminación
      adopcion.destroy({ where: { id } })
        .then((resultado) => {
          res.status(200).json({
            mensaje: `Registro Eliminado`
          });
        })
        .catch((err) => {
          res.status(500).json({
            mensaje: `Error al eliminar Registro ::: ${err}`
          });
        });
    })
    .catch((err) => {
      res.status(500).json({
        mensaje: `Error al verificar la existencia del registro de solicitud ::: ${err}`
      });
    });
};
```

Una nueva ruta en el archivo app.js:

```
app.use("/adopcion",routerAdopcion);
```

El archivo adopcionRouter.js en la carpeta rutas



Este archivo contiene lo siguiente:

```

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

```

Importación de módulos y controladores:

- Se importa el módulo express para crear la aplicación.
- Se importan las funciones crear, buscarId, buscar, actualizar, y eliminar desde el controlador adopcionController.js.

Creación de un enrutador (routerAdopcion):

- Se crea un enrutador utilizando express.Router() para manejar las diferentes rutas relacionadas con adopciones.

Definición de rutas:

- Ruta raíz ("/"): Retorna un mensaje de bienvenida cuando se accede a la ruta principal.
- Ruta para crear adopciones ("/crear"): Configura una ruta POST que llama a la función crear del controlador cuando se accede a /crear.
- Ruta para buscar adopción por ID ("/buscar/:id"): Configura una ruta GET que llama a la función buscarId del controlador cuando se accede a /buscar/:id. El parámetro :id es un marcador de posición para el ID de la adopción que se busca.

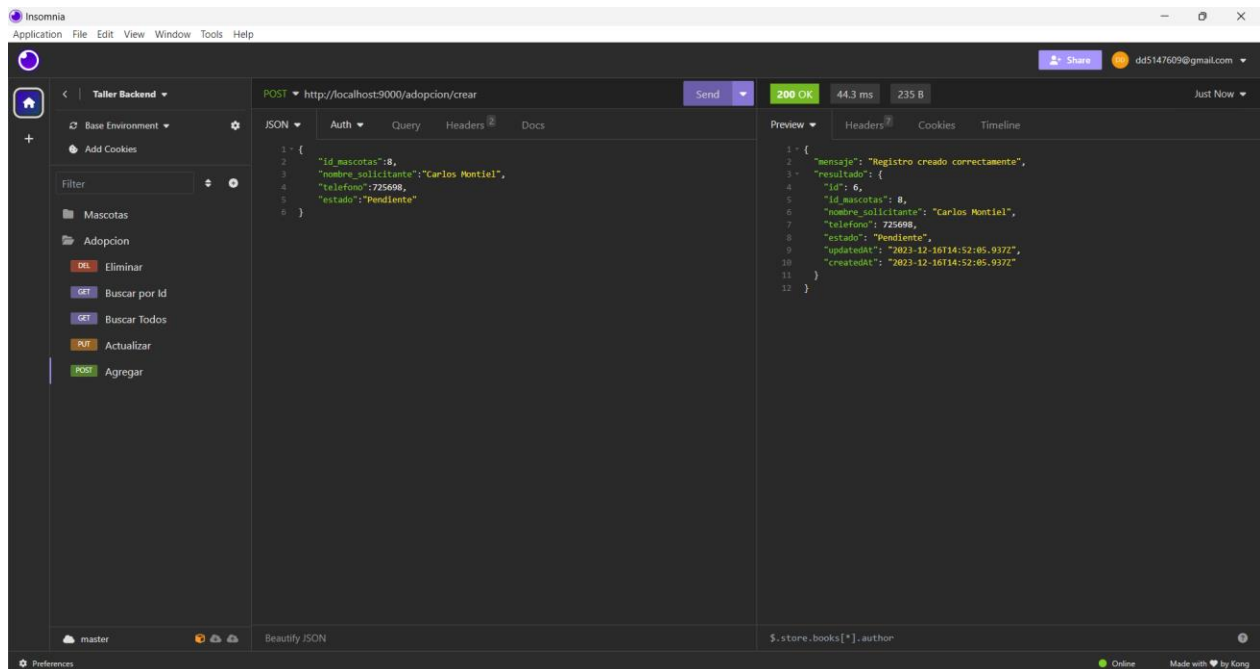
- Ruta para buscar todas las adopciones ("/buscar"): Configura una ruta GET que llama a la función buscar del controlador cuando se accede a /buscar.
- Ruta para actualizar adopción por ID ("/actualizar/:id"): Configura una ruta PUT que llama a la función actualizar del controlador cuando se accede a /actualizar/:id. Al igual que en la ruta de búsqueda por ID, :id es un marcador de posición para el ID de la adopción que se actualizará.
- Ruta para eliminar adopción por ID ("/eliminar/:id"): Configura una ruta DELETE que llama a la función eliminar del controlador cuando se accede a /eliminar/:id. El marcador de posición :id representa el ID de la adopción que se eliminará.

Exportación del enrutador (routerAdopcion):

- Se exporta el enrutador para que pueda ser utilizado en otros archivos de la aplicación.

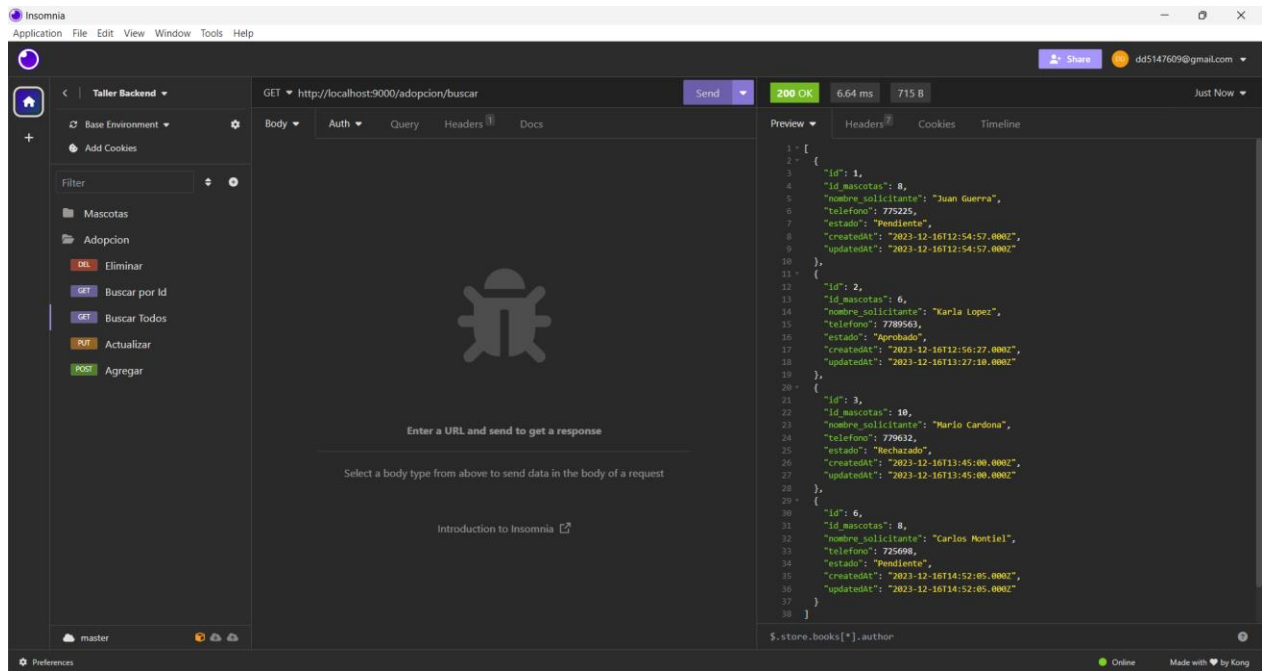
**A continuación, se realizará las diferentes consultas en Insomnia:**

Agregar:



adopciones   Enter a SQL expression to filter results (use Ctrl+Space)								
	123 id	123 id_mascotas	ABC nombre_solicitante	123 telefono	ABC estado	🕒 createdAt	🕒 updatedAt	
1	1	8	Juan Guerra	775.225	Pendiente	2023-12-16 12:54:57	2023-12-16 12:54:57	
2	2	6	Karla Lopez	7.789.563	Aprobado	2023-12-16 12:56:27	2023-12-16 13:27:10	
3	3	10	Mario Cardona	779.632	Rechazado	2023-12-16 13:45:00	2023-12-16 13:45:00	
4	6	8	Carlos Montiel	725.698	Pendiente	2023-12-16 14:52:05	2023-12-16 14:52:05	

Buscar todos los registros:



Insomnia Application | File | Edit | View | Window | Tools | Help

GET `http://localhost:9000/adopcion/buscar` **Send** **200 OK** 6.64 ms 715 B Just Now

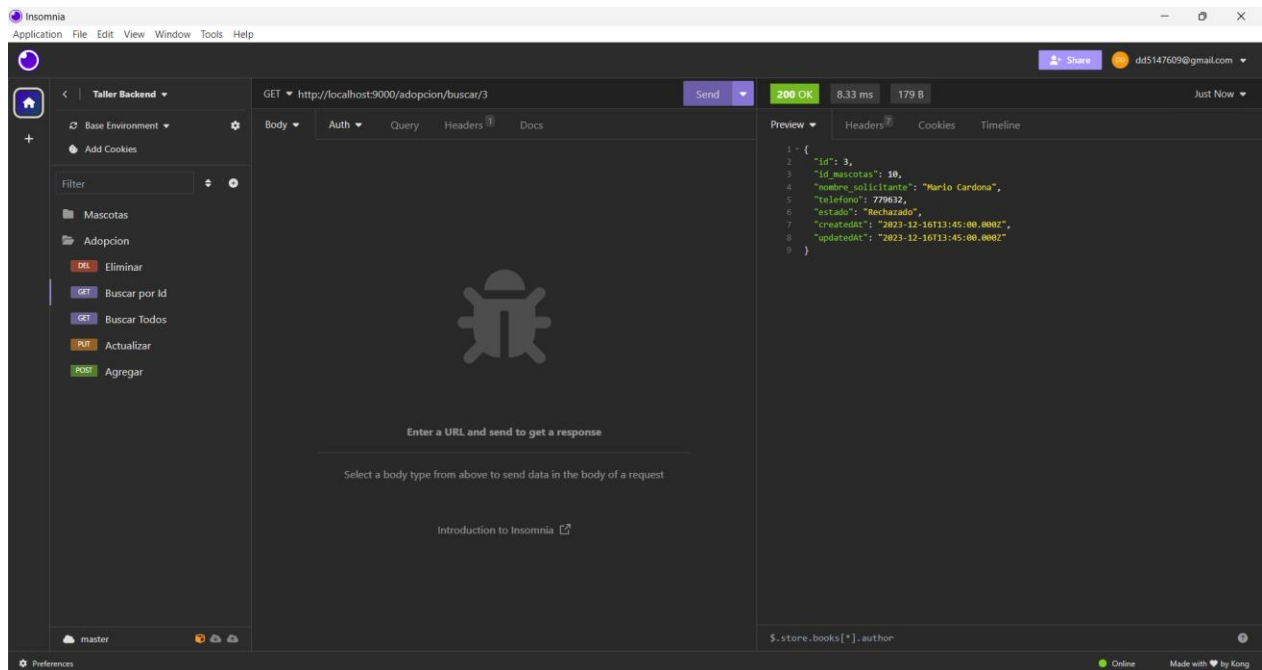
Body | Auth | Query | Headers | Docs

Preview

```
1 [
2   {
3     "id": 1,
4     "id_mascotas": 8,
5     "nombre_solicitante": "Juan Guerra",
6     "telefono": 775225,
7     "estado": "Pendiente",
8     "createdAt": "2023-12-18T12:54:57.000Z",
9     "updatedAt": "2023-12-18T12:54:57.000Z"
10  },
11  {
12    "id": 2,
13    "id_mascotas": 6,
14    "nombre_solicitante": "Karla Lopez",
15    "telefono": 7789563,
16    "estado": "Aprobado",
17    "createdAt": "2023-12-18T12:56:27.000Z",
18    "updatedAt": "2023-12-18T12:27:18.000Z"
19  },
20  {
21    "id": 3,
22    "id_mascotas": 10,
23    "nombre_solicitante": "Mario Cardona",
24    "telefono": 779632,
25    "estado": "Rechazado",
26    "createdAt": "2023-12-18T13:45:00.000Z",
27    "updatedAt": "2023-12-18T13:45:00.000Z"
28  },
29  {
30    "id": 6,
31    "id_mascotas": 8,
32    "nombre_solicitante": "Carlos Montiel",
33    "telefono": 725698,
34    "estado": "Pendiente",
35    "createdAt": "2023-12-18T14:52:05.000Z",
36    "updatedAt": "2023-12-18T14:52:05.000Z"
37  }
38 ]
```

\$.store.books[\*].author

Buscar por id:



Insomnia Application | File | Edit | View | Window | Tools | Help

GET `http://localhost:9000/adopcion/buscar/3` **Send** **200 OK** 8.33 ms 179 B Just Now

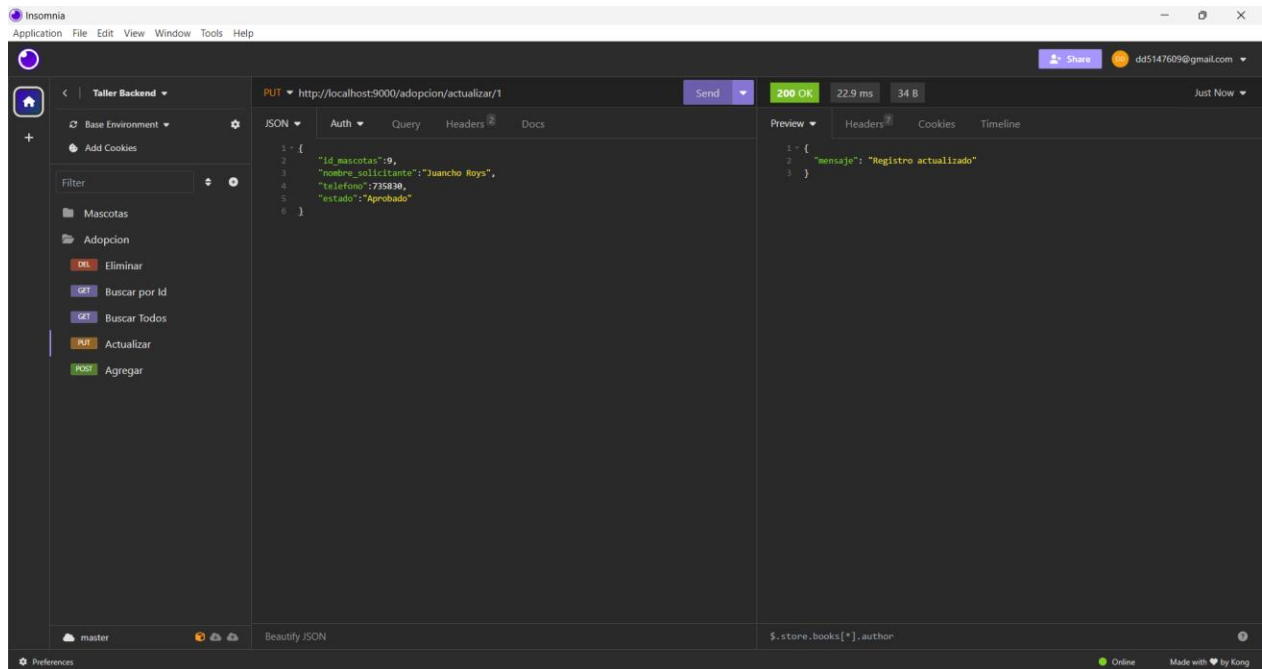
Body | Auth | Query | Headers | Docs

Preview

```
1 {
2   "id": 3,
3   "id_mascotas": 10,
4   "nombre_solicitante": "Mario Cardona",
5   "telefono": 779632,
6   "estado": "Rechazado",
7   "createdAt": "2023-12-18T13:45:00.000Z",
8   "updatedAt": "2023-12-18T13:45:00.000Z"
9 }
```

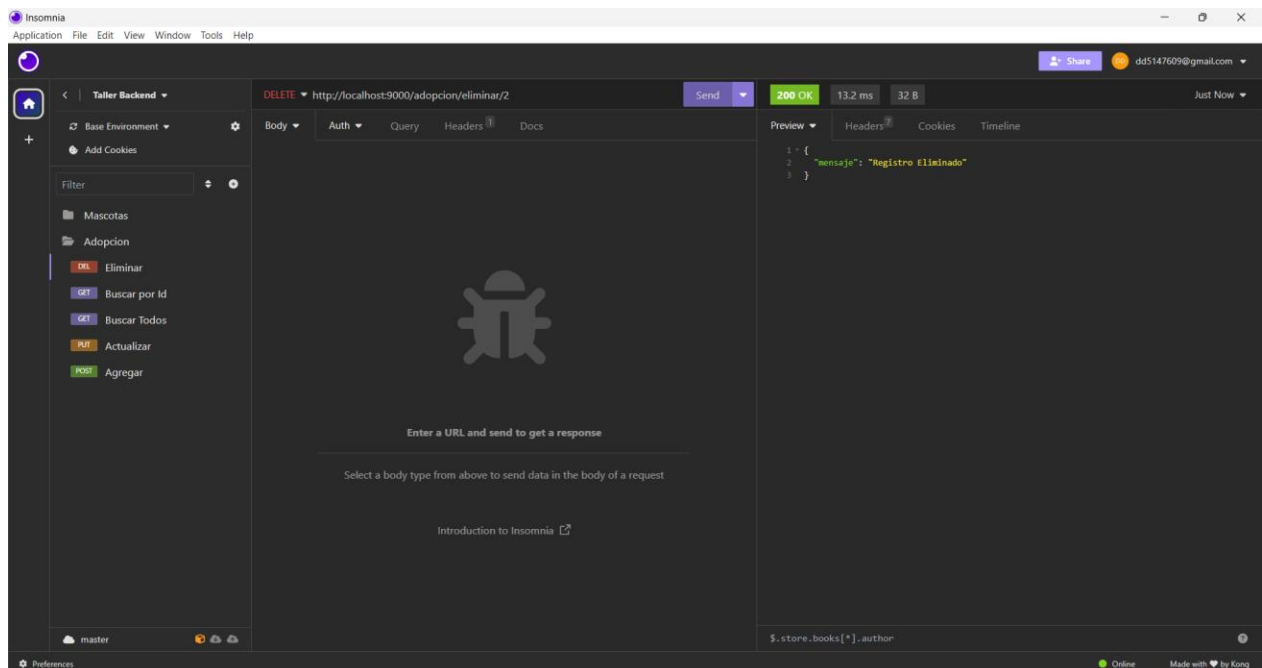
\$.store.books[\*].author

## Actualizar:



adopcions <small>Enter a SQL expression to filter results (use Ctrl+Space)</small>									
	id	id_mascotas	nombre_solicitante	telefono	estado	createdAt	updatedAt		
1	1	9	Juancho Roys	735.830	Aprobado	2023-12-16 12:54:57	2023-12-16 15:04:06		
2	2	6	Karla Lopez	7.789.563	Aprobado	2023-12-16 12:56:27	2023-12-16 13:27:10		
3	3	10	Mario Cardona	779.632	Rechazado	2023-12-16 13:45:00	2023-12-16 13:45:00		
4	6	8	Carlos Montiel	725.698	Pendiente	2023-12-16 14:52:05	2023-12-16 14:52:05		

## Eliminar:



adopciones <small>Enter a SQL expression to filter results (use Ctrl+Space)</small>								
	id	id_mascotas	nombre_solicitante	telefono	estado	createdAt	updatedAt	
1	1	9	Juancho Roys	735.830	Aprobado	23-12-16 12:54:57	23-12-16 15:04:06	
2	3	10	Mario Cardona	779.632	Rechazado	23-12-16 13:45:00	23-12-16 13:45:00	
3	6	8	Carlos Montiel	725.698	Pendiente	23-12-16 14:52:05	23-12-16 14:52:05	

## Conclusiones

En este proyecto, se ha desarrollado un sistema de gestión de adopciones de mascotas mediante una aplicación backend construida con Node.js, Express.js y Sequelize para interactuar con una base de datos MySQL. El código implementa operaciones CRUD para las entidades de "mascotas" y "adopciones", validando la existencia de mascotas antes de realizar adopciones. La aplicación utiliza rutas definidas en Express y se organiza en módulos para mejorar la estructura y mantenimiento del código. Además, se ha incorporado el manejo de errores y se han implementado prácticas de seguridad, como la verificación de la conexión a la base de datos y la validación de datos de entrada.