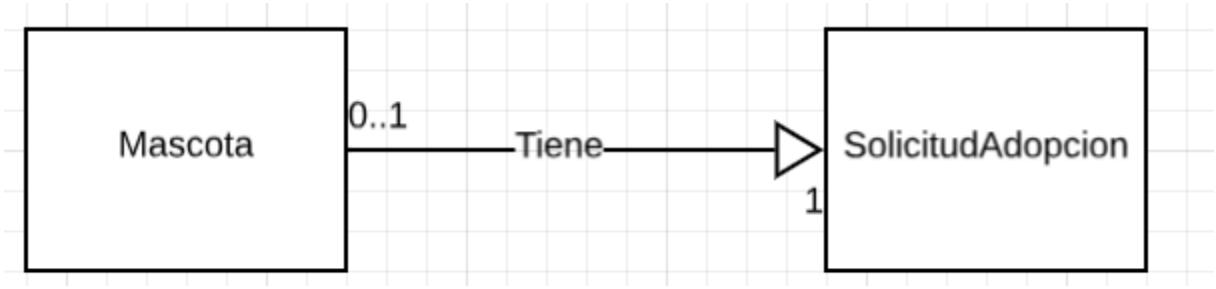
TALLER UNIDAD 2 BACKEND

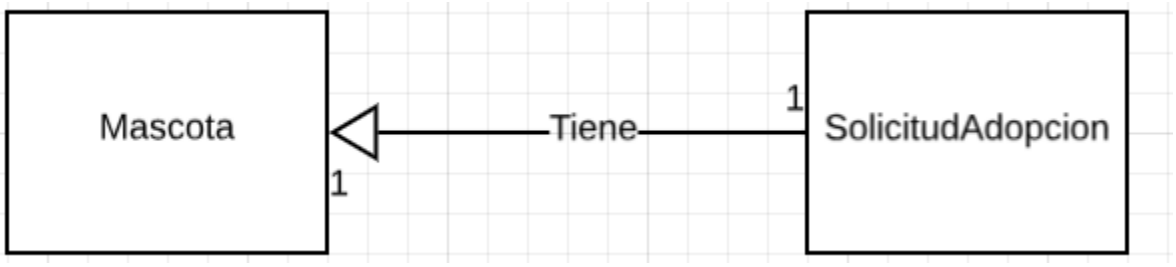
ÍNDICE

DEFINICIÓN DE LA BASE DE DATOS

Se plantea la base de datos de la siguiente forma:



En esta relación se entiende que la mascota puede tener o no una solicitud de adopción.



En la recíproca se asume que una solicitud de adopción tiene una mascota involucrada, los atributos son los siguientes:

Mascota		
Atributo	Tipo	Descripción
pk	integer	Llave primaria.
nombre	string	Este atributo hace referencia al nombre asignado a la mascota.
edad	integer	Este atributo almacena la edad en años del animal.
tipo_mascota	char	Este atributo puede contener el carácter **P** para referirse a que la mascota es un perro y el atributo **G** para indicar que es un gato.
estado	integer	Este dato almacena el estado de la mascota que retorna **1** si la mascota ya fue adoptada y **0** si la mascota aún no ha sido adoptada, **por defecto es 0.**

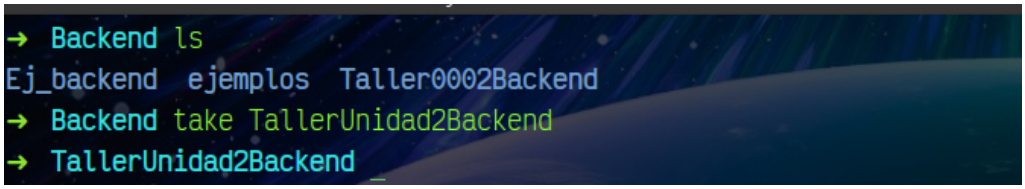
Solicitud Adopción		
Atributo	Tipo	Descripción
pk	integer	Llave Primaria.
mascota	integer	Llave foránea que apunta a una mascota.
adoptante	string	El nombre de la persona que está adoptando a la mascota.
estado	char	Este atributo puede contener el carácter **A**, cuando el proceso fue aceptado y la mascota fué adoptada, **P** para indicar que la adopción se

		encuentra en trámite y **R** para indicar que la adopción fue rechazada.
fecha_inicio	date	Este atributo almacena la fecha en la cual se recibió la solicitud de adopción.
fecha_fin	date	Este atributo indica el momento en el cual se Acepta o Rechaza la solicitud, esta columna puede estar vacía, ya que las solicitudes de adopción que se encuentran en trámite aún no habrán finalizado.

CREACIÓN DEL REPOSITORIO LOCAL

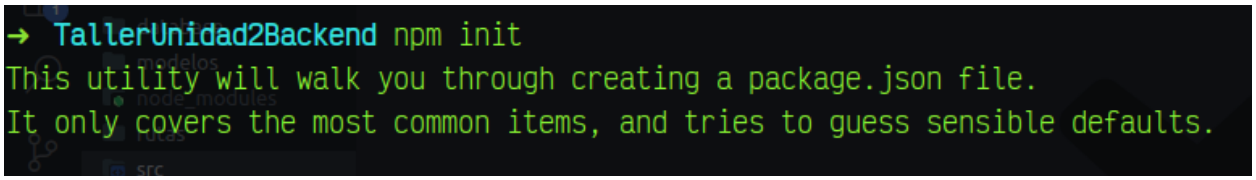
1. Usar el comando cd y ubicarse en una carpeta que considere adecuada.
2. Crear el repositorio local.

take TallerUnidad2Backend

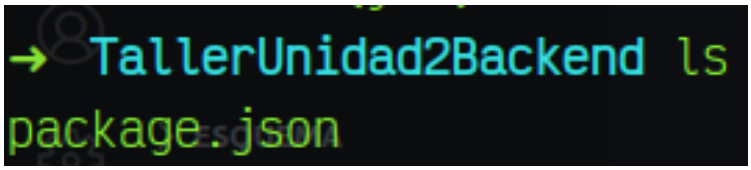


Proceso de Instalación y adecuamiento del repositorio local.
inicializar npm(node package manager)

npm init



Una vez diligenciadas las preguntas, se crea automáticamente el package.json



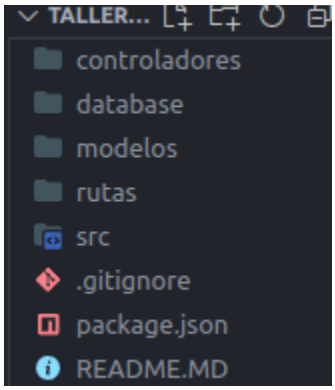
Ahora se recomienda usar el comando **code .** para abrir el vscode en esta carpeta, a continuación creamos la siguiente lista de carpetas en el directorio raíz:

1. controladores
2. database
3. modelos
4. rutas
5. src

Y los archivos:

1. .gitignore
2. README.MD

En este punto el directorio debería verse de la siguiente manera:



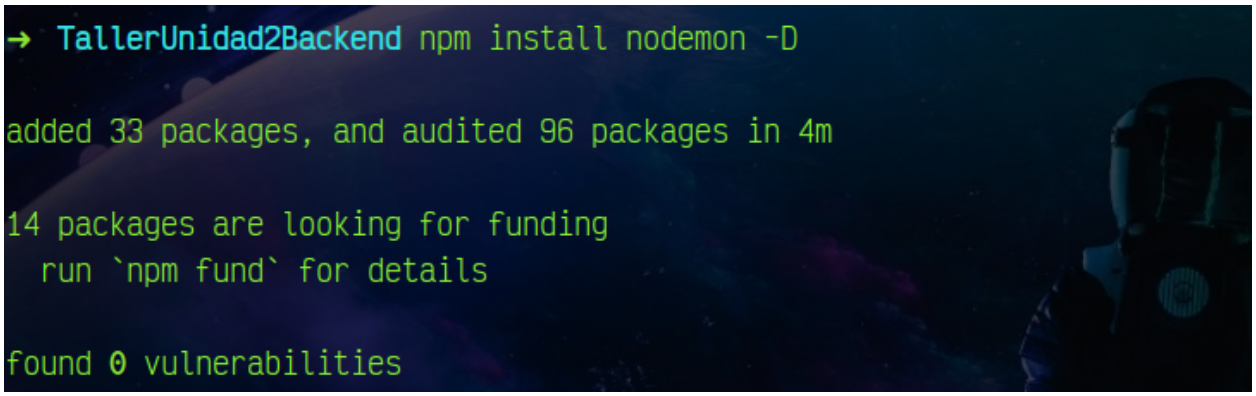
Instalar express js <- esto trae los node modules https://expressjs.com/
npm install express

una vez instalados los módulos de node, accedemos al archivo .gitignore y colocamos las siguiente líneas:

```
/node_modules
node_modules
node_modules/
```

Como paso final, antes de iniciar el trabajo sobre el repositorio local, se recomienda instalar nodemon como dependencia únicamente para desarrollo.

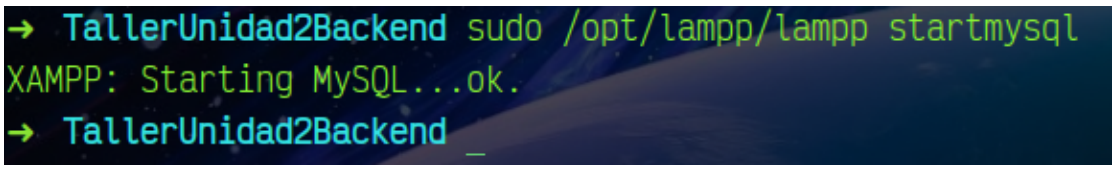
npm install nodemon -D



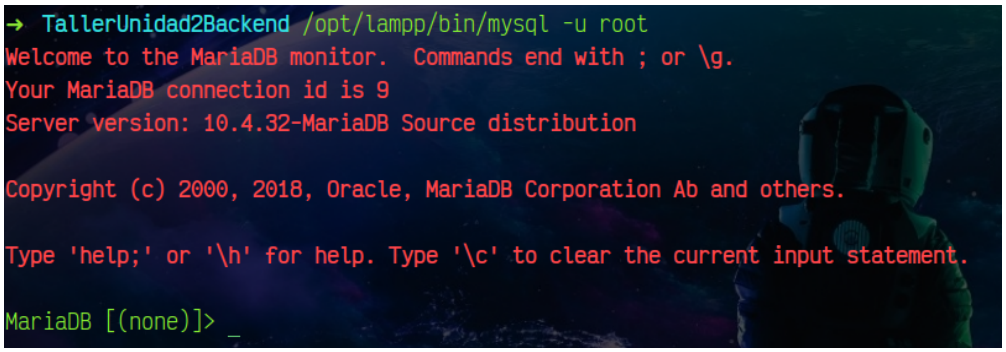
IMPLEMENTACIÓN DE LA BASE DE DATOS

A Continuación procedemos a crear la base de datos de la siguiente manera:

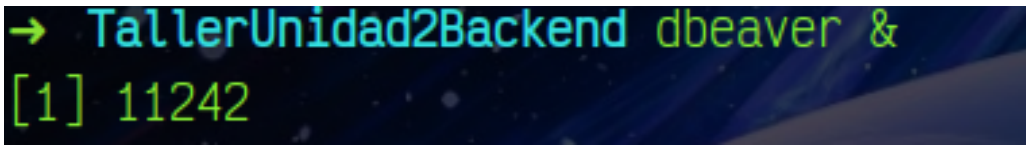
sudo /opt/lampp/lampp startmysql <- Inicializamos el servicio de MySQL



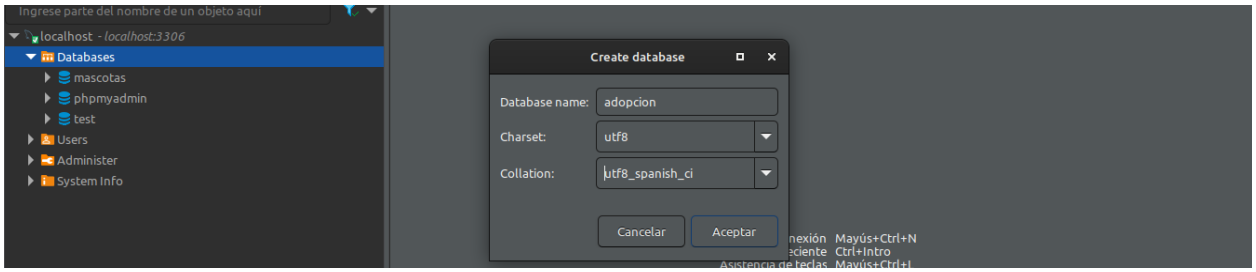
/opt/lampp/bin/mysql -u root <- Verificamos que el servicio se encuentre corriendo:



Abre otra pestaña de la consola y ejecuta dbeaver, de la siguiente manera.
dbeaver & <-Inicializamos Dbeaver:



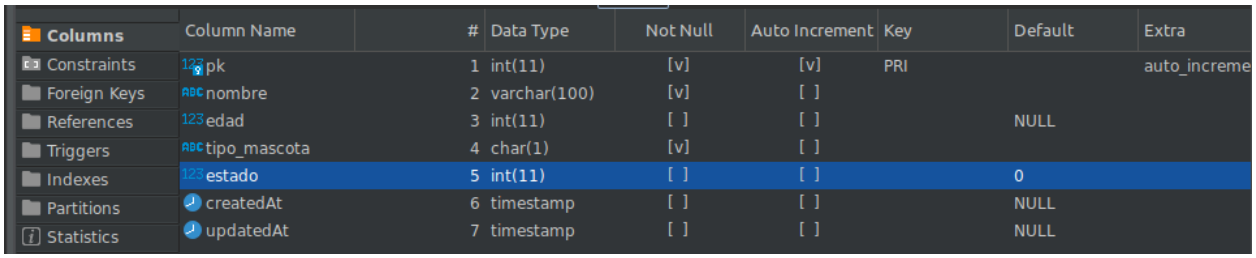
Se abrirá el programa y aquí procedemos a diseñar la base de datos, de la siguiente manera:
Clic derecho en databases <- nuevo database



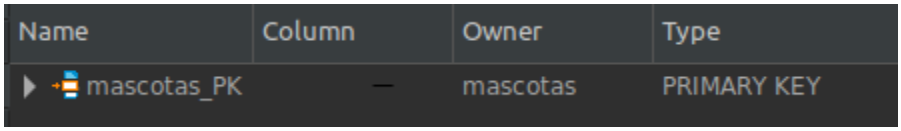
Establecemos la configuración y procedemos a pulsar en el botón aceptar.

Ahora procedemos a crear las tablas de la forma ya establecida previamente en este documento:

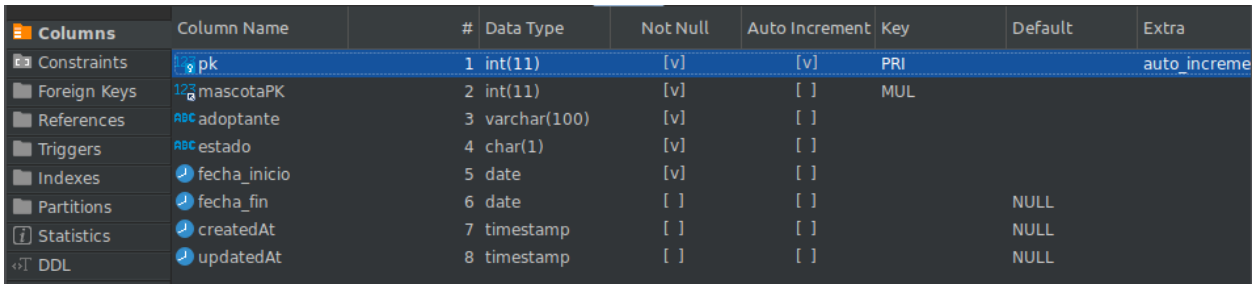
MASCOTAS



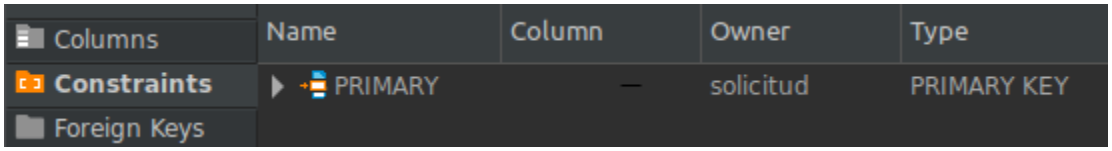
Téngase en cuenta añadir el constrain correspondiente a la llave primaria de la mascota:



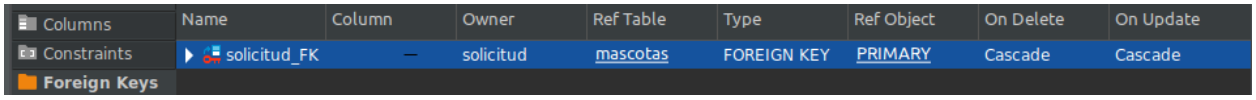
SOLICITUD DE ADOPCIÓN



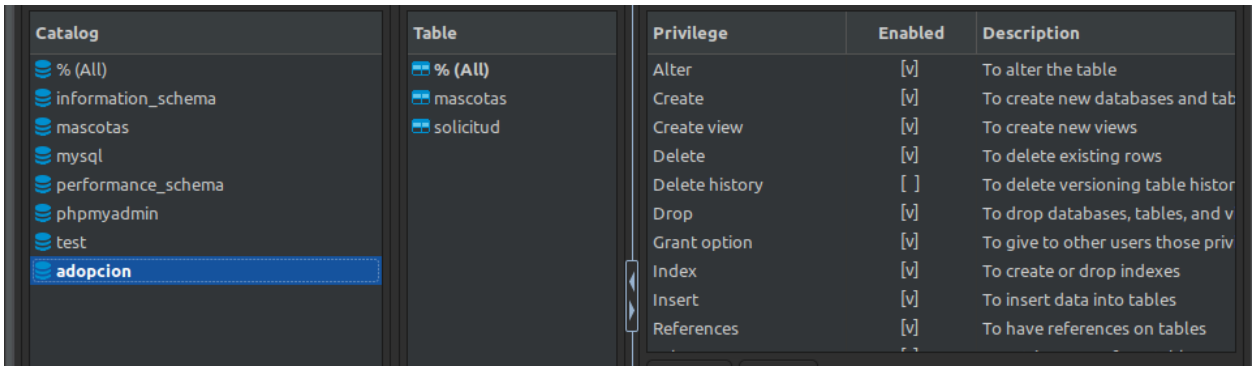
Los constraints son:



La llave foránea que apunta a la mascota de la solicitud:

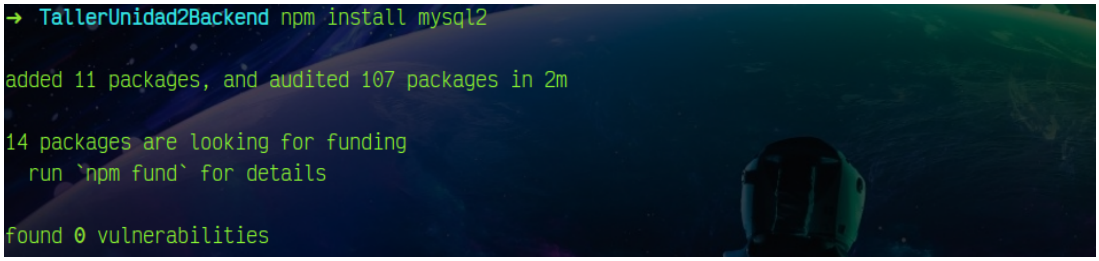


Ahora procedemos a crear el usuario y darle todos los permisos sobre las tablas mascotas y adopciones:

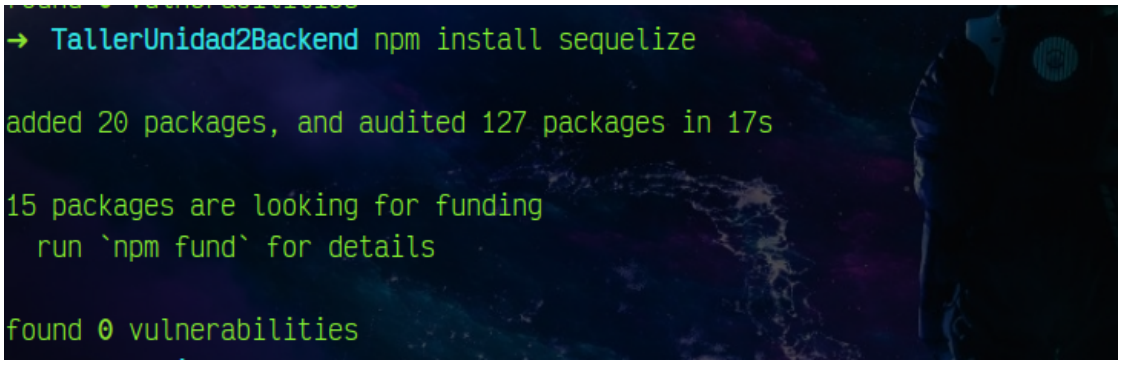


Ahora debemos instalar las dependencias que nos permitirán trabajar con bases de datos MySql:

npm install mysql2



npm install sequelize



Ahora, es momento de configurar el package.json

Añadimos el archivo principal de la aplicación, que en este caso será app.js y declaramos que la aplicación es es de tipo module:

```
 "main": "app.js",
 "type": "module",
```

Adicionalmente, adicionamos el nodemon al stack de script para que la app corra con el mismo:

```
"scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "start": "nodemon ./src/app.js"
 },
```

En esta instancia, el package.json debe verse similar a esto:

```
{
 "name": "tallerunidad2backend",
 "version": "1.0.0",
 "description": "",
 "main": "app.js",
 "type": "module",
 "scripts": {
   "test": "echo \"Error: no test specified\" && exit 1",
   "start": "nodemon ./src/app.js"
 },
 "author": "",
 "license": "ISC",
 "dependencies": {
   "express": "^4.18.2",
   "mysql2": "^3.6.5",
   "sequelize": "^6.35.2"
 },
 "devDependencies": {
   "nodemon": "^3.0.2"
 }
}
```

Es momento de crear el script que establecerá la conexión a la base de datos, para ello creamos el archivo **database/conexion.js,** que contendrá el siguiente código:

```
// importar la biblioteca Sequelize
import { Sequelize } from "sequelize";

//definir y exportar la constante que sostendrá la conexión a la base de datos
export const db = new Sequelize('adopcion', 'adopcion', 'mascotas2023', {
   dialect:'mysql',
   host:'localhost'
});
```

Ahora debemos crear los modelos correspondientes, de la siguiente manera:

MODELOS

MASCOTAS

Ubicación: **modelos/mascotaModelo.js**

```
//importar Sequelize
import { Sequelize } from "sequelize";
//importar la variable de conexion
import { db } from "../database/conexion.js";

//definicion del objeto que comunicara con la tabla
export const mascotas = db.define('mascotas',{
   //definicion de los atributos
   pk:{
       //tipo de dato
       type:Sequelize.INTEGER,
       //no se permite vacio
       allowNull:false,
       //es la llave primaria
       primaryKey:true,
       //es autoincrementado
       autoIncrement:true
   },
   nombre:{
```

```
        //tipo de dato
        type:Sequelize.STRING,
        //no se permite vacio
        allowNull:false
    },
    edad:{
        //tipo de dato
        type:Sequelize.INTEGER,
        //se permite vacio
        allowNull:true
    },
    tipo_mascota:{
        //tipo de dato
        type:Sequelize.CHAR,
        //no se permite vacio
        allowNull:false
    },
    estado:{
        //tipo de dato
        type:Sequelize.INTEGER,
        //no se permite vacio
        allowNull:true
    }
});
```

SOLICITUDES

Ubicación: **modelos/solicitudModelo.js**

```
//importar Sequelize
import { Sequelize } from "sequelize";
//importar la variable de conexion
import { db } from "../database/conexion.js";
//importar el modelo mascotas
import { mascotas } from "./mascotaModelo.js";

//definicion del objeto que comunicara con la tabla
const solicitudes = db.define('solicitudes',{
    //definicion de los atributos
    pk:{
```

```
        //tipo de dato
        type: Sequelize.INTEGER,
        //no se permite vacio
        allowNull: false,
        //es la llave primaria
        primaryKey: true,
        //es autoincrementado
        autoIncrement: true
    },
    mascotaPK:{
        //tipo de dato
        type: Sequelize.INTEGER,
        //no se permite vacio
        allowNull: false
    },
    adoptante:{
         //tipo de dato
         type: Sequelize.STRING,
         //no se permite vacio
         allowNull: false
    },
    estado:{
        //tipo de dato
        type: Sequelize.CHAR,
        //no se permite vacio
        allowNull: false
    },
    fecha_inicio:{
        //tipo de dato
        type: Sequelize.DATE,
        //no se permite vacio
        allowNull: false
    },
    fecha_fin:{
        //tipo de dato
        type: Sequelize.DATE,
        //no se permite vacio
        allowNull: true
    }
});
```

```
// definir la relación de solicitud con mascota
solicitudes.belongsTo(mascotas, { foreignKey: 'mascotaPK' });

export { solicitudes };
```

El siguiente paso es crear los controladores correspondientes.

CONTROLADORES

MASCOTAS

Ubicación: **controladores/mascotaControlador.js**

```
//importar el modelo de mascotas
import {mascotas} from "../modelos/mascotaModelo.js";

//listar todas las mascotas
const listarMascotas = (req, res) => {
   mascotas.findAll().then((r) => {
       res.status(200).json(r);
   }).catch((e) => {
       res.status(500).json({mensaje: "No se ha podido encontrar ningun
registro"+e});
   });

   return;
}

//buscar una mascota por id
const buscarMascota = (req , res) => {
   const id = parseInt(req.params.id);

   if(id == null){
       res.status(400).json({mensaje: "El id no puede estar vacio"});
       return;
   }

   mascotas.findByPk(id).then((r) => {
       res.status(200).json(r);
   }).catch((e) => {
```

```
        res.status(500).json({mensaje: "No se ha podido encontrar el
registro"});
    });

    return;
}

//crear una mascota en la tabla
const crearMascota = (req, res) => {
    //los campos nombre, tipo mascota y estado son obligatorios
    if(!req.body.nombre){
        res.status(400).json({mensaje: "el campo nombre es requerido"});
        return;
    }
    if(!req.body.tipo_mascota){
        res.status(400).json({mensaje: "el tipo_mascota es requerido"});
        return;
    }

    //definicion del dataset
    const dataset = {
        nombre: req.body.nombre,
        edad: req.body.edad,
        tipo_mascota: req.body.tipo_mascota,
        estado: req.body.estado
    }

    //creacion de la mascota en la base de datos
    mascotas.create(dataset).then((r)=>{
        res.status(200).json({mensaje: "Mascota registrada con exito"});
    }).catch((e)=>{
        res.status(500).json({mensaje: "No se ha podido registrar la
mascota"+e});
    });

    return;
}

//eliminar una mascota por id
const eliminarMascota = (req , res) => {
```

```
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "El id no puede estar vacio"});
        return;
    }

    mascotas.destroy({
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({mensaje: "Mascota eliminada exitosamente"});
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido eliminar el
registro"});
    });

    return;
}

//actualizar una mascota por id
const actualizarMascota = (req , res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "El id no puede estar vacio"});
        return;
    }

    if(!req.body.nombre && !req.body.edad && !req.body.tipo_mascota &&
!req.body.estado){
        res.status(400).json({mensaje: "No se ha encontrado ningun dato
para actualizar"});
        return;
    }

    const nombre = req.body.nombre;
    const edad = req.body.edad;
    const tipo_mascota = req.body.tipo_mascota;
    const estado = req.body.estado;
```

```
    mascotas.update({
        nombre: nombre,
        edad:edad,
        tipo_mascota:tipo_mascota,
        estado:estado
    },{
        where:{pk: id} //recordemos que en la base de datos, el campo de la
llave primaria es pk, no id
    }).then((r) => {
        res.status(200).json({mensaje: "Mascota actualizada
exitosamente"});
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido actualizar el
registro"});
    });

    return;
}

export { crearMascota, listarMascotas, buscarMascota, eliminarMascota,
actualizarMascota };
```

SOLICITUDES

Ubicación: **controladores/solicitudControlador.js**

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

//listar todas las solicitudes
const listarSolicitudes = (req, res) => {
    solicitudes.findAll().then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido consultar las
solicitudes"});
    });
```

```
    return;
}

//buscar solicitud por id
const buscarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.findByPk(id).then((r) => {
        res.status(200).json(r);
    }).catch((e) => {
        res.status(500).json({mensaje: "No se ha podido encontrar el
registro"});
    });

    return;
}

//eliminar solicitud por id
const eliminarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    solicitudes.destroy({
        where:{ pk: id}
    }).then((r)=>{
        res.status(200).json({mensaje: "Registro eliminado con exito !"});
    }).catch((e) => {
```

```
        res.status(500).json({mensaje: "No se a podido remover el registro
de la base de datos"});
    });

    return;
}

//crear una solicitud
const crearSolicitud = (req, res) => {
    //validar que vengan los campos requeridos
    if(!req.body.mascotaPK){
        res.status(400).json({mensaje: "El campo de id (mascotaPK) de la
mascota es requerido"});
        return;
    }
    if(!req.body.adoptante){
        res.status(400).json({mensaje: "El nombre del adoptante (adoptante)
es requerido"});
        return;
    }
    if(!req.body.estado){
        res.status(400).json({mensaje: "El estado de la solicitud es
requerido"});
        return;
    }
    if(!req.body.fecha_inicio){
        res.status(400).json({mensaje: "La fecha de inicio es requerida"});
        return;
    }

    //todo el dataset
    const dataset = {
        mascotaPK: req.body.mascotaPK,
        adoptante: req.body.adoptante,
        estado: req.body.estado,
        fecha_inicio: req.body.fecha_inicio,
        fecha_fin: req.body.fecha_fin
    }

    //creacion de la solicitud en la base de datos
```

```
    solicitudes.create(dataset).then((r) => {
        res.status(200).json({mensaje: "Solicitud registrada con exito!"});
    }).catch((e) => {
        res.status(500).json({mensaje:" Error, no se ha podido crear el
registro en la base de datos: "+e});
    });

    return;
}

//buscar solicitud por id
const actualizarSolicitud = (req, res) => {
    const id = parseInt(req.params.id);

    if(id == null){
        res.status(400).json({mensaje: "Se requiere el id para poder buscar
el registro"});
        return;
    }

    if(!req.body.adoptante && !req.body.mascotaPK && !req.body.estado &&
!req.body.fecha_inicio && !req.body.fecha_fin){
        res.status(400).json({mensaje: "No se a detectado ningun campo para
actualizar"});
        return;
    }

    const mascotaPK = req.body.mascotaPK;
    const adoptante = req.body.adoptante;
    const estado = req.body.estado;
    const fecha_inicio = req.body.fecha_inicio;
    const fecha_fin = req.body.fecha_fin;

    solicitudes.update({
        mascotaPK:mascotaPK,
        adoptante:adoptante,
        estado: estado,
        fecha_inicio:fecha_inicio,
        fecha_fin: fecha_fin
    },{
```

```
        where: {pk:id}
    }).then((r) => {
        res.status(200).json({mensaje: "Solicitud actualizada
exitosamente"});
    }).catch((e) => {
        res.status(500).json({mensaje: "No se pudo alterar el registro en
la base de datos"});
    });

    return;
}

export { listarSolicitudes, crearSolicitud, buscarSolicitud,
eliminarSolicitud, actualizarSolicitud }
```

RUTAS

MASCOTAS

Ubicación: **rutas/mascotasRouter.js**

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

//crear la instancia de tipo router
const mascostasRouter = express.Router();

//rutas
//DE TIPO GET
mascostasRouter.get('/', (req, res)=> {
    listarMascotas(req, res);
});

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

//DE TIPO POST
mascostasRouter.post('/crear', (req, res)=>{
```

```
    crearMascota(req, res);
});

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

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

export { mascostasRouter };
```

SOLICITUDES

Ubicación: **rutas/solicitudesRouter.js**

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

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

//rutas
//Tipo GET
solicitudesRouter.get('/', (req, res) => {
    listarSolicitudes(req, res);
});
solicitudesRouter.get('/buscar/:id', (req, res) => {
    buscarSolicitud(req, res);
});

//tipo POST
solicitudesRouter.post('/crear', (req, res) => {
    crearSolicitud(req, res);
});
```

```
//tipo DELETE
solicitudesRouter.delete('/eliminar/:id', (req, res) => {
   eliminarSolicitud(req, res);
});

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

export { solicitudesRouter };
```

APP

Ahora creamos el archivo **src/app.js** que es el archivo principal de nuestra aplicación, para este caso añadiremos en el el siguiente código:

```
//importar express
import express from "express";
import { mascostasRouter } from "../rutas/mascotasRouter.js";
import { db } from "../database/conexion.js";
import { solicitudesRouter } from "../rutas/solicitudesRouter.js";

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

// Middleware para procesar datos JSON en el cuerpo de las solicitudes
app.use(express.json());

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

//definir la constante que contendra el puerto por el cual correrá el
servidor
const PORT = 9000;

db.authenticate().then(()=>{
   console.log("La base de datos ha sido cargada con exito");
```

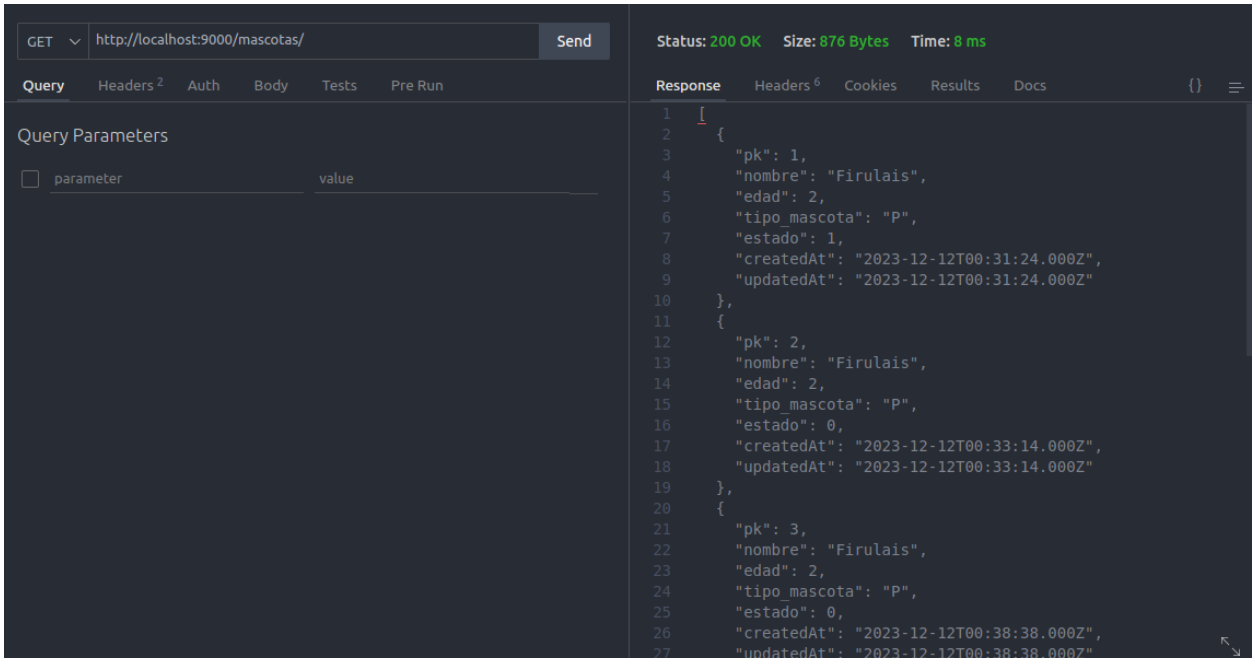
```
}).catch((r) => {
   console.log("Error al cargar la base de datos: "+e);
});

//definir las rutas
app.get("/", (req, res) => {res.send("Hola Desde Backend MySQL");});

//definir las rutas para mascotas
app.use('/mascotas', mascostasRouter);

//definir las rutas para solicitudes
app.use('/solicitudes', solicitudesRouter);

//si fue posible conectarse a la base de datos
db.sync().then(() => {
   // puerto de ecucha, recibe el numero de puerto y un función
   app.listen(PORT, ()=>{
       console.log(`Servidor inicializado en el puerto: ${PORT}`);
   });
}).catch((e) => {
   console.log("No se pudo sincronizar con la base de datos: "+e);
});
```

LISTADO DE RUTAS

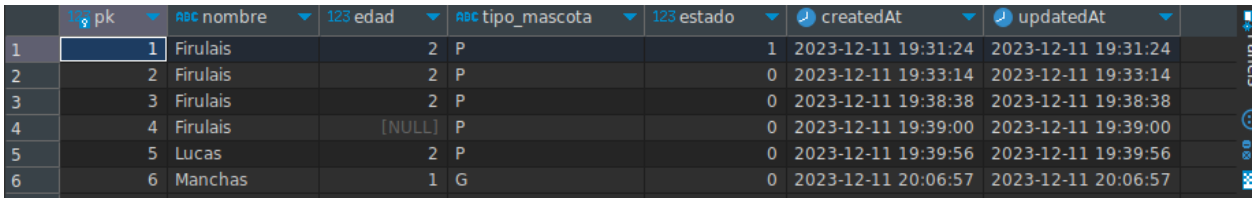
Se hace uso de Thunder Client para testear las rutas

MASCOTAS

Ruta: http://localhost:9000/mascotas/
Tipo: GET
Parámetros:
Body:
Resultado:



Base de datos:



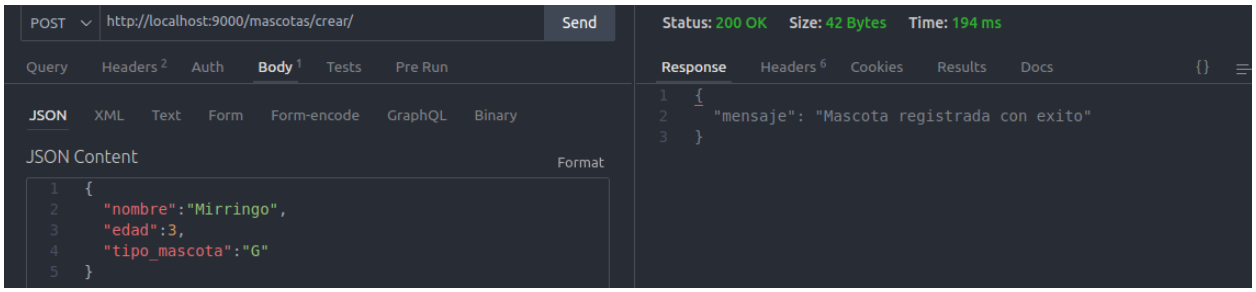
	pk	nombre	edad	tipo_mascota	estado	createdAt	updatedAt
1	1	Firulais	2	P	1	2023-12-11 19:31:24	2023-12-11 19:31:24
2	2	Firulais	2	P	0	2023-12-11 19:33:14	2023-12-11 19:33:14
3	3	Firulais	2	P	0	2023-12-11 19:38:38	2023-12-11 19:38:38
4	4	Firulais	[NULL]	P	0	2023-12-11 19:39:00	2023-12-11 19:39:00
5	5	Lucas	2	P	0	2023-12-11 19:39:56	2023-12-11 19:39:56
6	6	Manchas	1	G	0	2023-12-11 20:06:57	2023-12-11 20:06:57

Se puede observar que los resultados corresponden.

Ruta: http://localhost:9000/mascotas/crear/
Tipo: POST
Parámetros:
Body:

```
{
 "nombre":"Mirringo",
 "edad":3,
 "tipo_mascota":"G"
}
```

Resultado:

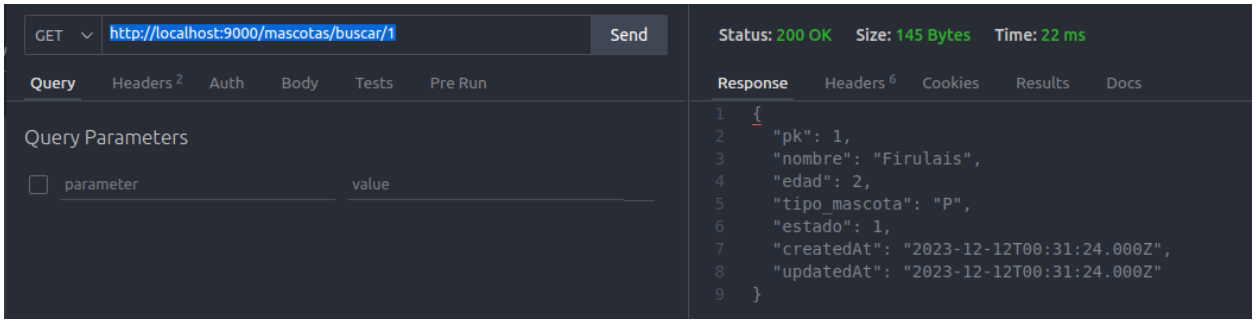


Base de datos:

	pk	nombre	edad	tipo_mascota	estado	createdAt	updatedAt
1	1	Firulais	2	P	1	2023-12-11 19:31:24	2023-12-11 19:31:24
2	2	Firulais	2	P	0	2023-12-11 19:33:14	2023-12-11 19:33:14
3	3	Firulais	2	P	0	2023-12-11 19:38:38	2023-12-11 19:38:38
4	4	Firulais	[NULL]	P	0	2023-12-11 19:39:00	2023-12-11 19:39:00
5	5	Lucas	2	P	0	2023-12-11 19:39:56	2023-12-11 19:39:56
6	6	Manchas	1	G	0	2023-12-11 20:06:57	2023-12-11 20:06:57
7	7	Mirringo	3	G	0	2023-12-11 20:22:09	2023-12-11 20:22:09

Podemos observar que la mascota se agregó con éxito.

Ruta: http://localhost:9000/mascotas/buscar/1
Tipo: GET
Parámetros: 1
Body:
Resultado:

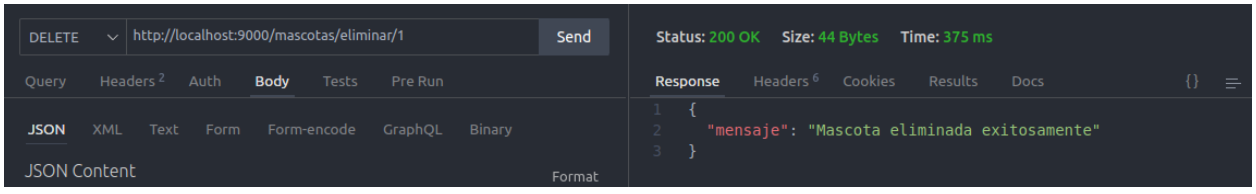


Base de datos:

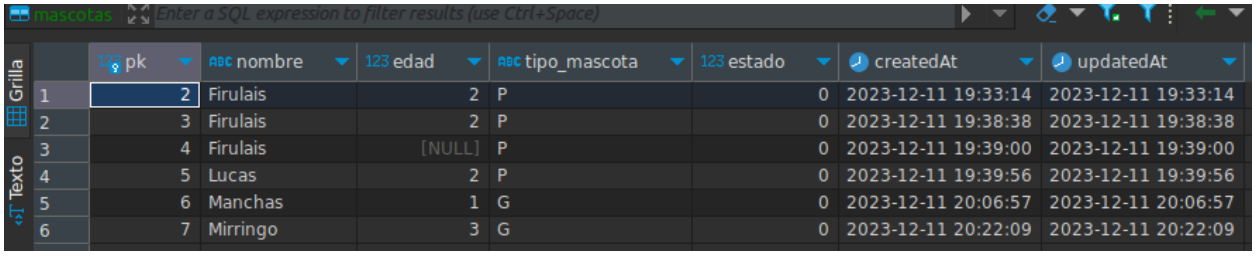
	pk	nombre	edad	tipo_mascota	estado	createdAt	updatedAt
1	1	Firulais	2	P	1	2023-12-11 19:31:24	2023-12-11 19:31:24
2	2	Firulais	2	P	0	2023-12-11 19:33:14	2023-12-11 19:33:14
3	3	Firulais	2	P	0	2023-12-11 19:38:38	2023-12-11 19:38:38
4	4	Firulais	[NULL]	P	0	2023-12-11 19:39:00	2023-12-11 19:39:00
5	5	Lucas	2	P	0	2023-12-11 19:39:56	2023-12-11 19:39:56
6	6	Manchas	1	G	0	2023-12-11 20:06:57	2023-12-11 20:06:57

Se puede observar que los resultados corresponden.

Ruta: http://localhost:9000/mascotas/eliminar/1
Tipo: DELETE
Parámetros: 1
Body:
Resultado:



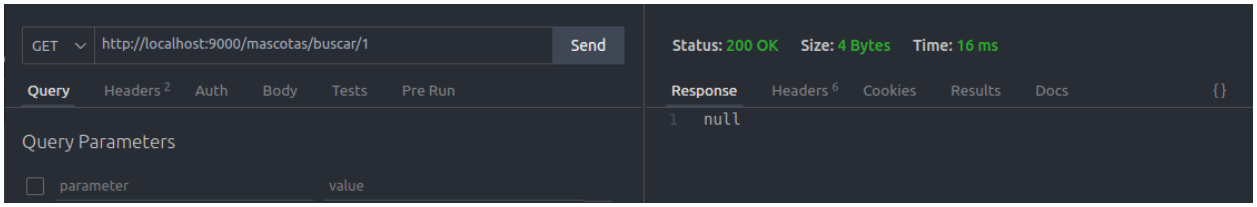
Base de datos:



	pk	nombre	edad	tipo_mascota	estado	createdAt	updatedAt
1	2	Firulais	2	P	0	2023-12-11 19:33:14	2023-12-11 19:33:14
2	3	Firulais	2	P	0	2023-12-11 19:38:38	2023-12-11 19:38:38
3	4	Firulais	[NULL]	P	0	2023-12-11 19:39:00	2023-12-11 19:39:00
4	5	Lucas	2	P	0	2023-12-11 19:39:56	2023-12-11 19:39:56
5	6	Manchas	1	G	0	2023-12-11 20:06:57	2023-12-11 20:06:57
6	7	Mirringo	3	G	0	2023-12-11 20:22:09	2023-12-11 20:22:09

Se puede observar que el elemento fue removido de la base de datos.

Si intentamos consultar nuevamente la mascota con pk = 1, nos arroja nulo:

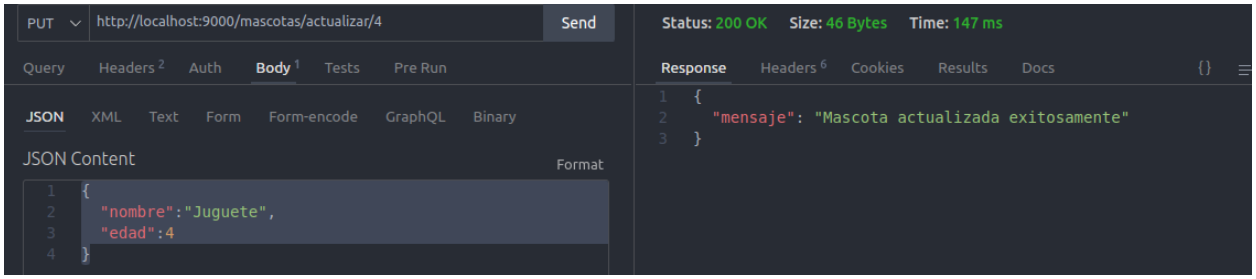


Esto se debe a que esta fue eliminada anteriormente por el método DELETE.

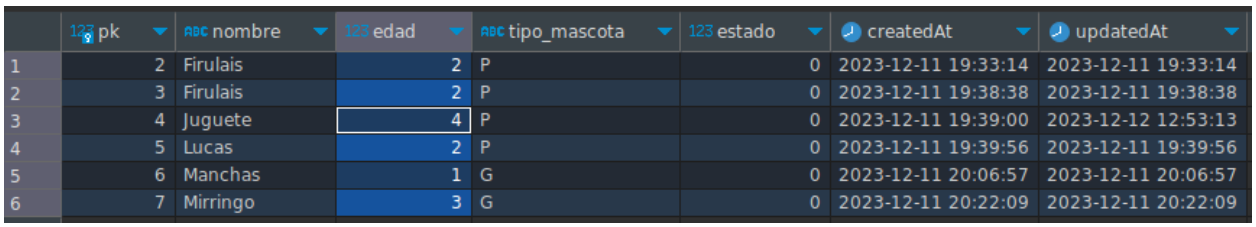
Ruta: http://localhost:9000/mascotas/actualizar/4
Tipo: PUT
Parámetros: 4
Body:

```
{
 "nombre":"Juguete",
 "edad":4
}
```

Resultado:



Base de datos:

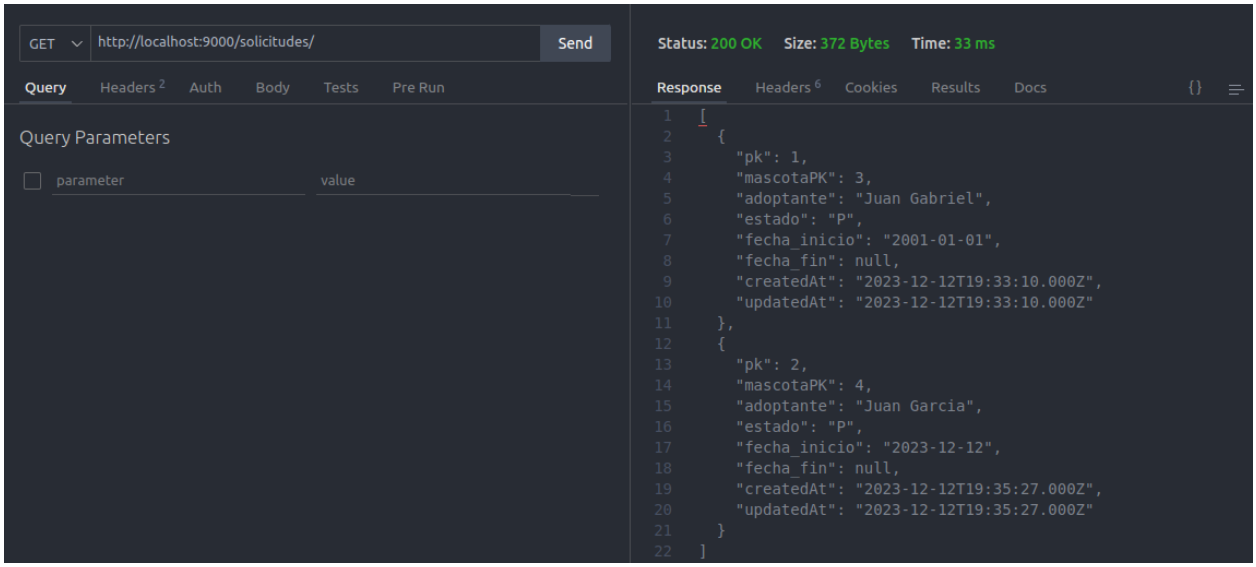


	pk	nombre	edad	tipo_mascota	estado	createdAt	updatedAt
1	2	Firulais	2	P	0	2023-12-11 19:33:14	2023-12-11 19:33:14
2	3	Firulais	2	P	0	2023-12-11 19:38:38	2023-12-11 19:38:38
3	4	Juguete	4	P	0	2023-12-11 19:39:00	2023-12-12 12:53:13
4	5	Lucas	2	P	0	2023-12-11 19:39:56	2023-12-11 19:39:56
5	6	Manchas	1	G	0	2023-12-11 20:06:57	2023-12-11 20:06:57
6	7	Mirringo	3	G	0	2023-12-11 20:22:09	2023-12-11 20:22:09

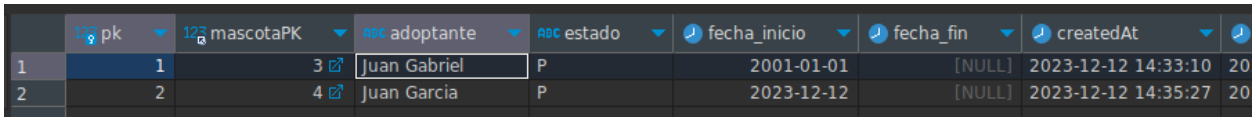
Vemos que el dato con pk = 4 actualiza sus datos de nombre y edad.

SOLICITUDES

Ruta: http://localhost:9000/solicitudes/
Tipo: GET
Parámetros:
Body:
Resultado:



Base de datos:



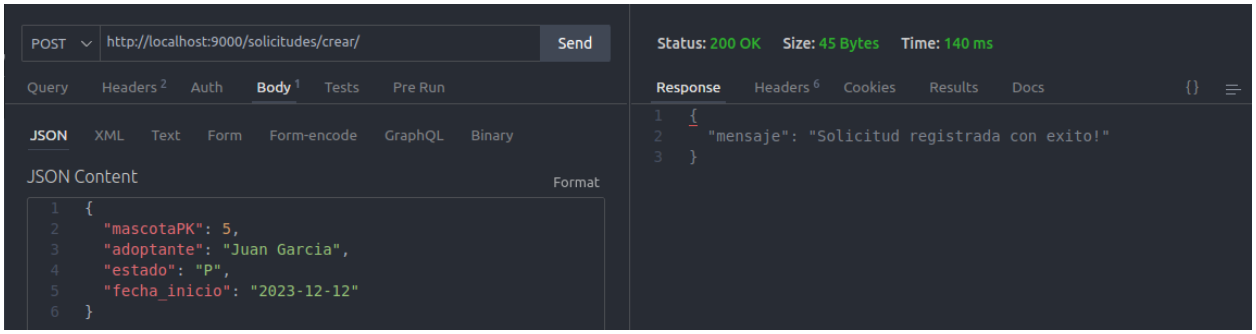
	pk	mascotaPK	adoptante	estado	fecha_inicio	fecha_fin	createdAt	
1	1	3	Juan Gabriel	P	2001-01-01	[NULL]	2023-12-12 14:33:10	20
2	2	4	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:35:27	20

Como se puede observar, los datos obtenidos en la consulta coinciden con los de la base de datos.

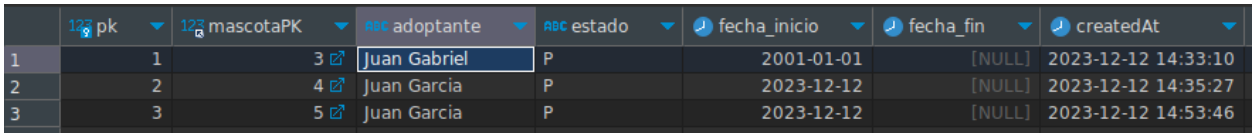
Ruta: http://localhost:9000/solicitudes/crear/
Tipo: POST
Parámetros:
Body:

```
{
  "mascotaPK": 4,
  "adoptante": "Juan Garcia",
  "estado": "P",
  "fecha_inicio": "2023-12-12"
}
```

Resultado:



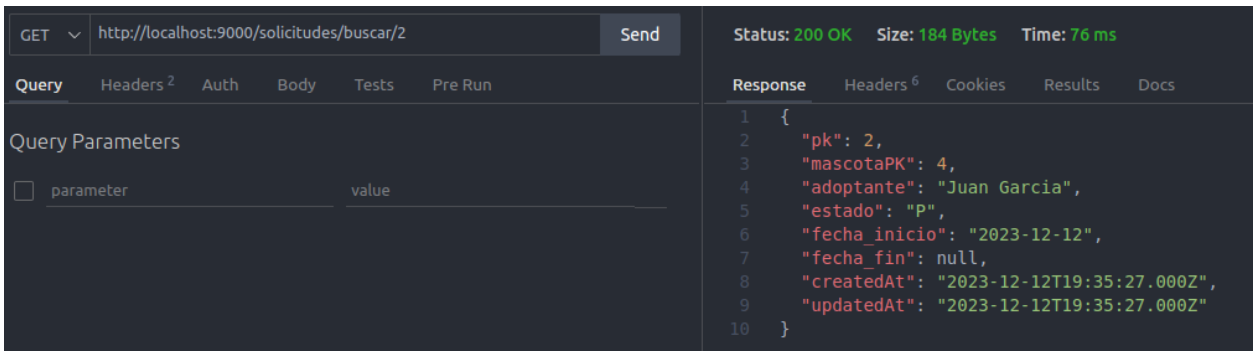
Base de datos:



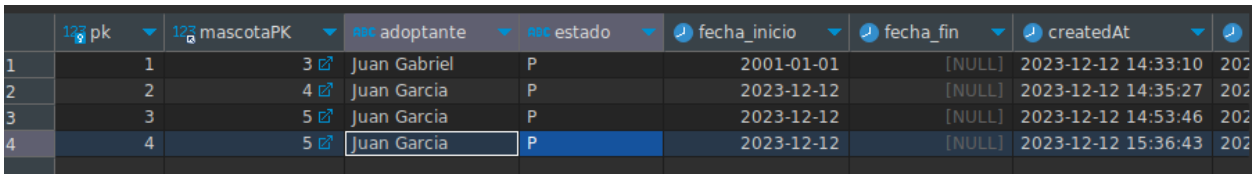
	pk	mascotaPK	adoptante	estado	fecha_inicio	fecha_fin	createdAt
1	1	3	Juan Gabriel	P	2001-01-01	[NULL]	2023-12-12 14:33:10
2	2	4	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:35:27
3	3	5	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:53:46

Podemos observar que el registro se ingresa correctamente.

Ruta: http://localhost:9000/solicitudes/buscar/2
Tipo: GET
Parámetros: 2
Body:
Resultado:



Bases de Datos:

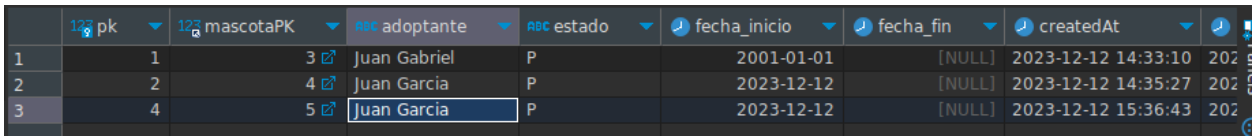


	pk	mascotaPK	adoptante	estado	fecha_inicio	fecha_fin	createdAt	
1	1	3	Juan Gabriel	P	2001-01-01	[NULL]	2023-12-12 14:33:10	202
2	2	4	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:35:27	202
3	3	5	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:53:46	202
4	4	5	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 15:36:43	202

Como podemos observar, los datos coinciden con los registros presentes en la base de datos.

Ruta:http://localhost:9000/solicitudes/eliminar/3
Tipo: DELETE
Parámetros: 3
Body:
Resultado:

Base de Datos:



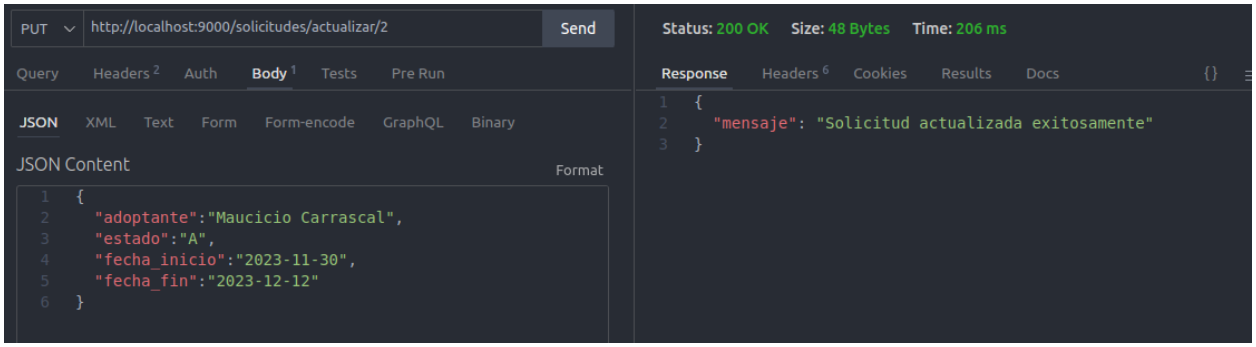
	pk	mascotaPK	adoptante	estado	fecha_inicio	fecha_fin	createdAt
1	1	3	Juan Gabriel	P	2001-01-01	[NULL]	2023-12-12 14:33:10
2	2	4	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 14:35:27
3	4	5	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 15:36:43

podemos observar que el elemento con pk = 3 fue removido de la base de datos, lo que indica que el método está trabajando correctamente.

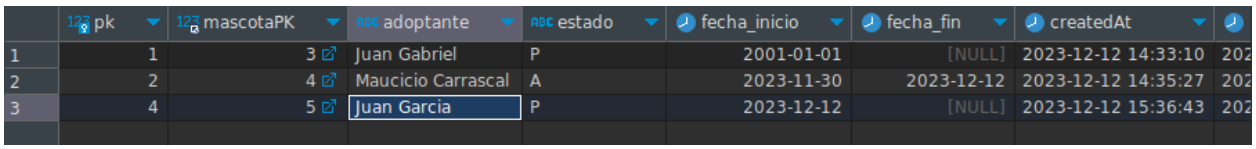
Ruta:http://localhost:9000/solicitudes/actualizar/2
Tipo: PUT
Parámetros: 2
Body:

```
{
 "adoptante":"Maucicio Carrascal",
 "estado":"A",
 "fecha_inicio":"2023-11-30",
 "fecha_fin":"2023-12-12"
}
```

Resultado:



Base de datos:

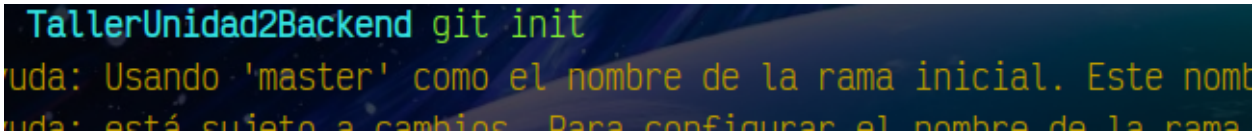


	pk	mascotaPK	adoptante	estado	fecha_inicio	fecha_fin	createdAt
1	1	3	Juan Gabriel	P	2001-01-01	[NULL]	2023-12-12 14:33:10
2	2	4	Maucicio Carrascal	A	2023-11-30	2023-12-12	2023-12-12 14:35:27
3	4	5	Juan Garcia	P	2023-12-12	[NULL]	2023-12-12 15:36:43

Como se puede observar, se actualizaron los registros correspondientes en la base de datos.

VERSIONAMIENTO Y SUBIDA A GITHUB

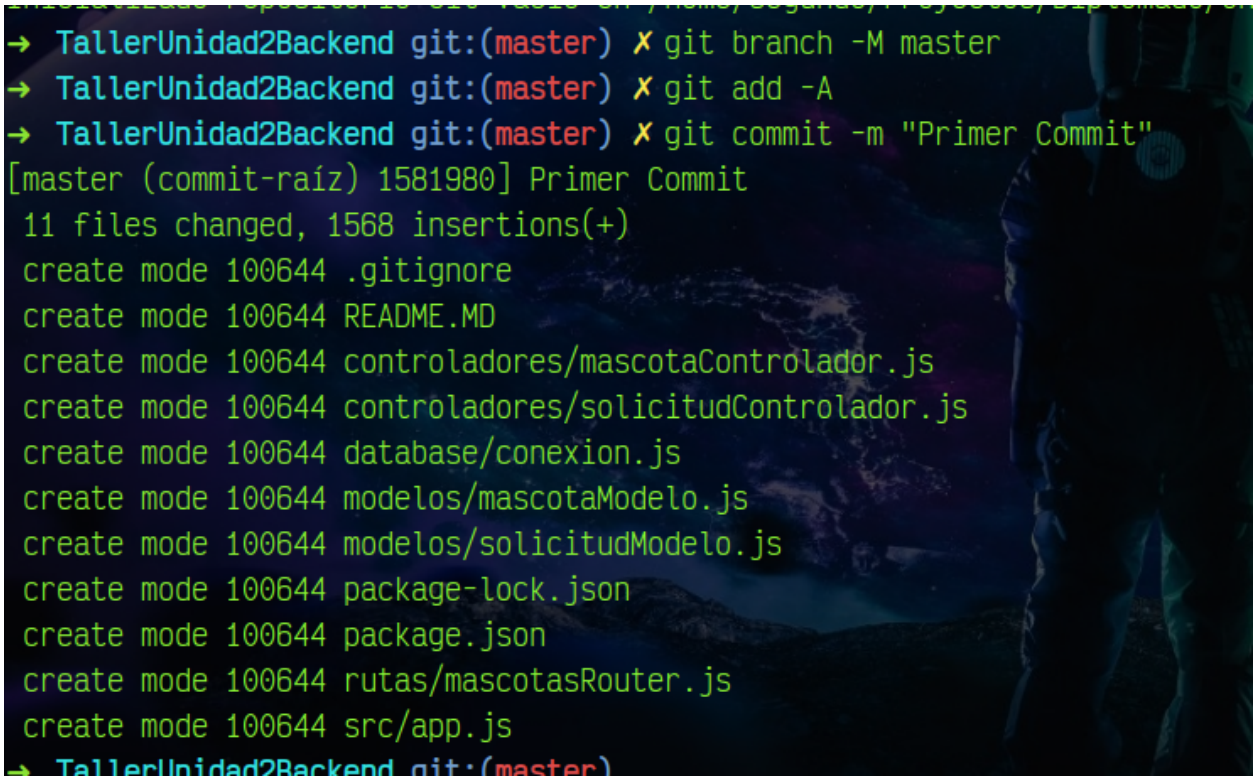
git init



git branch -M master

git add -A

git commit -m “mensaje”



git remote add origin url_repositorio

git push origin master