Taller Unidad 2 Backend

ENTREGADO POR:

Sebastian Camilo Quenguan Culcha

ENTREGADO A:

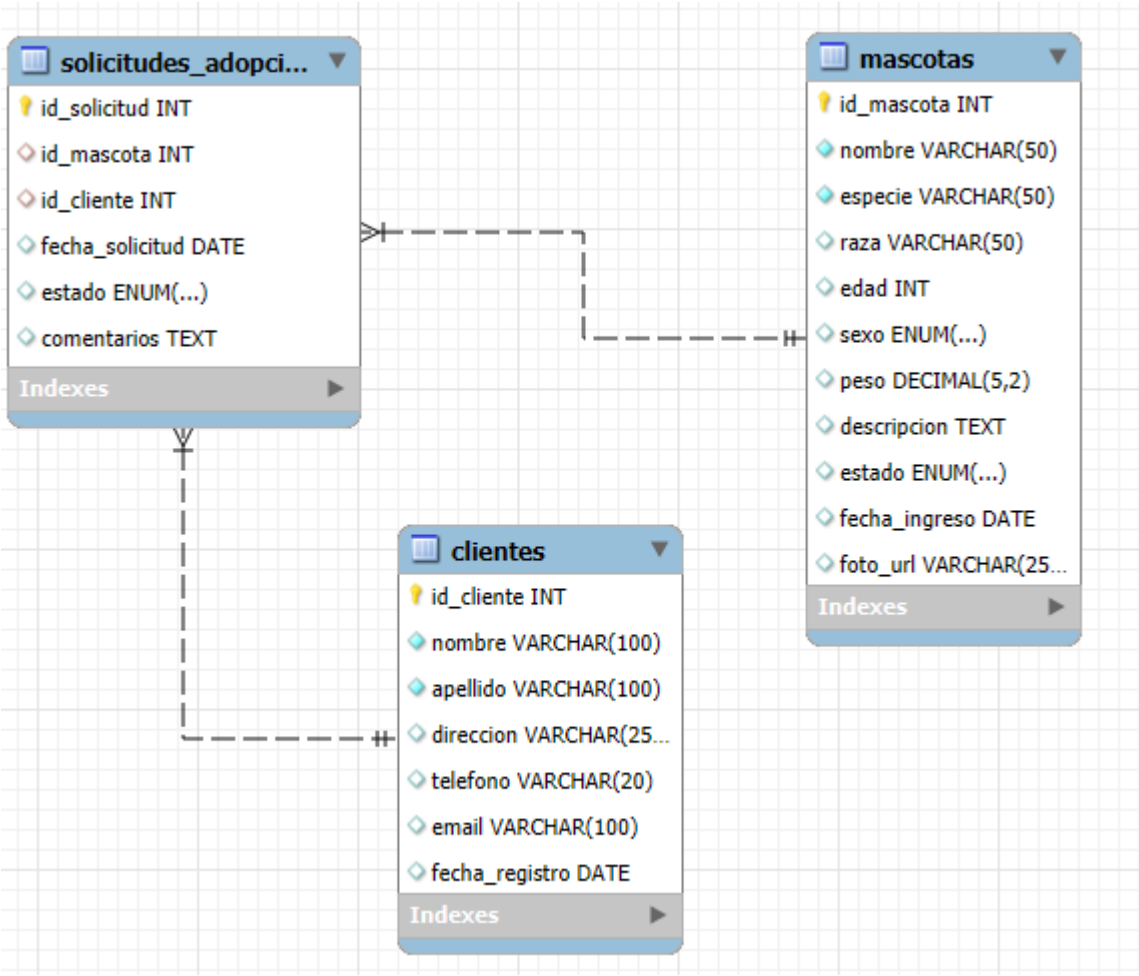
Vicente Aux Revelo

UNIVERSIDAD DE NARIÑO

DIPLOMADO

2024

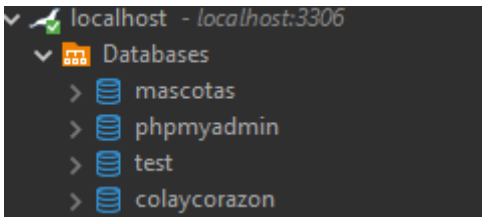
Crear una base de datos MYSQL/MARIADB que permita llevar el registro de una empresa de adopción de mascotas, debe soportar la administración de las mismas y la posibilidad de registrar solicitudes de adopción.



- Gestión clara de las mascotas: La tabla mascotas permite almacenar información detallada de cada mascota, incluyendo datos como nombre, especie, raza, edad, sexo, peso, y una descripción adicional. Además, se incluye el estado de disponibilidad ("Disponible" o "Adoptada"), lo cual es fundamental para llevar un control eficiente del proceso de adopción.
- Organización de clientes: La tabla clientes facilita la gestión de las personas interesadas en adoptar mascotas. Se almacenan datos personales como nombre,

apellido, dirección, teléfono y correo electrónico, lo que permite realizar un seguimiento efectivo de los adoptantes potenciales.

- Proceso estructurado de adopciones: La tabla solicitudes_adopcion estructura el proceso de adopción, registrando las solicitudes que los clientes hacen para adoptar mascotas. Cada solicitud está vinculada a una mascota y un adoptante, lo cual garantiza trazabilidad en las solicitudes. El campo estado define el progreso de la solicitud (Pendiente, Aprobada, Rechazada), y la fecha de solicitud permite un mejor control temporal del proceso.
- Integridad referencial: El uso de claves foráneas en la tabla solicitudes_adopcion, que referencian las tablas mascotas y clientes, garantiza la integridad referencial. Esto impide que se creen solicitudes de adopción para mascotas o clientes inexistentes, manteniendo la consistencia de los datos.
- Flexibilidad en la información: Las tablas están diseñadas para adaptarse a distintos casos de uso. Por ejemplo, la tabla de mascotas permite incluir una foto y descripciones de cada animal, mientras que en la tabla de solicitudes se pueden añadir comentarios específicos relacionados con el proceso de adopción.
- Escalabilidad y adaptabilidad: La estructura de la base de datos está diseñada de manera que es posible ampliarla sin dificultad en el futuro. Se podrían agregar nuevas tablas o modificar las existentes si el proceso de adopción evoluciona o se requieren nuevas funcionalidades.



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 (La empresa debe contar con un nombre), así como también las solicitudes de adopción. Se debe hacer uso

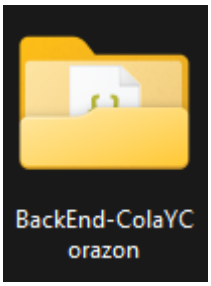
Requisitos:

- NodeJS instalado.

- ExpressJS como framework web.
- MySQL para la base de datos.
- ORM Sequelize para interactuar con la base de datos.
- MySQL2 para que Sequelize se comunique con MySQL.

Paso 1: Iniciar un proyecto NodeJS

Creamos una carpeta para el backEnd donde se realizará el proyecto



Entramos con la consola a esta carpeta y digitamos "**npm init -y**" se utiliza en Node.js para inicializar un nuevo proyecto y generar un archivo package.json con valores predeterminados

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> npm init -y
Wrote to C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon\package.json:

{
  "name": "backend-colaycorazon",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Paso 2: Instalar dependencias

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> npm install express mysql2 sequelize

added 98 packages, and audited 99 packages in 7s

15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon>
```

- ExpressJS: Framework para crear el servidor.
- MySQL2: Paquete que permite a Sequelize conectarse a MySQL.

- Sequelize: ORM para interactuar con la base de datos.

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> npm install nodemon -D

added 28 packages, and audited 127 packages in 1s

19 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Es una herramienta que automáticamente reinicia el servidor de Node.js cada vez que detecta cambios en los archivos del proyecto, lo que facilita el proceso de desarrollo, ya que no es necesario reiniciar el servidor manualmente

Configuramos el package.json

name:

"backend-colaycorazon"

El nombre del proyecto. En este caso, el backend parece estar relacionado con la empresa de adopción de mascotas llamada "Cola y Corazón".

type:

"module"

Define el tipo de módulos que se utilizan. Al especificar "module", se indica que el proyecto está utilizando módulos de ECMAScript (ESM) en lugar de los módulos comunes de Node.js (require), lo que permite el uso de import y export.

main:

"app.js"

Especifica el archivo de entrada principal del proyecto. Cuando otro archivo o aplicación ejecuta el paquete, este será el archivo ejecutado por defecto.

scripts:

Es un conjunto de comandos que puedes ejecutar usando npm run. Por ejemplo:

"start": "nodemon ./src/app.js"

Ejecuta nodemon para iniciar el servidor que se encuentra en ./src/app.js. nodemon reiniciará el servidor cada vez que se detecten cambios en el código.

También vemos que ya tenemos todas nuestras dependencias y sus versiones

Vamos a tener esta estructura para nuestro proyecto y vamos a comenzar creando **app.js**

Este archivo tiene como objetivo:

Configurar el servidor Express.

Manejar las rutas de diferentes entidades (mascotas, clientes, solicitudes).

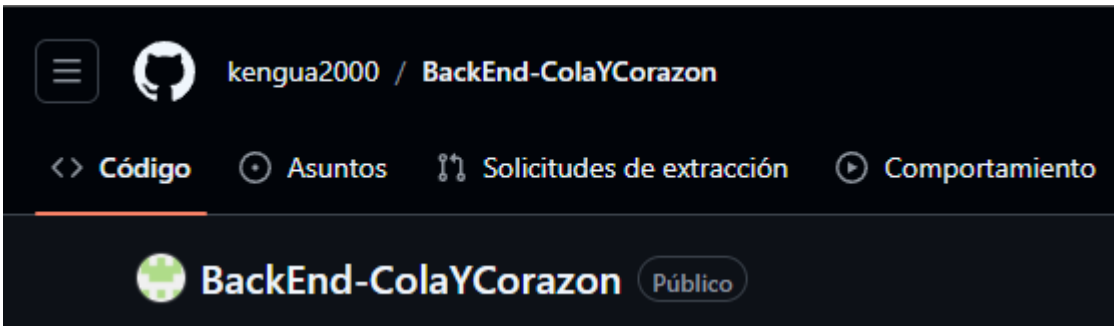
Conectar y sincronizar con la base de datos.

Escuchar peticiones en un puerto específico

Y mandamos nuestro primer commit

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git add .
warning: in the working copy of 'package-lock.json', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'package.json', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git commit -m "Primer commit inicio del proyecto(cola y corazón)"
>>
[main (root-commit) 1a35229] Primer commit inicio del proyecto(cola y corazón)
 3 files changed, 1528 insertions(+)
 create mode 100644 package-lock.json
 create mode 100644 package.json
 create mode 100644 src/app.js
```

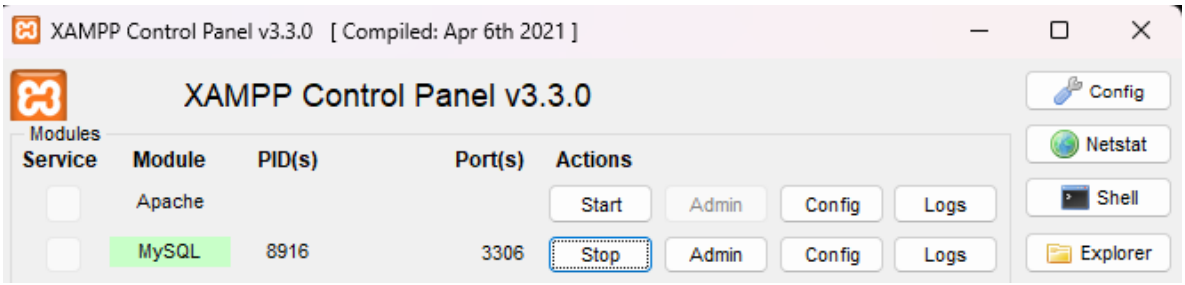
Y lo conectamos con github



y ahora implementamos la **conexion.js**

Que nos va conectar con la base de datos

Prendemos nuestro xampp



Y verificamos la conexión y mandamos otro commit

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> npm start

> backend-colaycorazon@1.0.0 start
> nodemon ./src/app.js

[nodemon] 3.1.5
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./src/app.js`
Executing (default): SELECT 1+1 AS result
```

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git add .
warning: in the working copy of 'src/database/conexion.js', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git commit -m "Establecida la conexion con mysql"
>>
[main e6035b9] Establecida la conexion con mysql
 2 files changed, 22 insertions(+), 2 deletions(-)
 create mode 100644 src/database/conexion.js
```

Ahora si vamos a desarrollar la parte lógica de Mascotas

Comenzando por **mascotaModelo.js**

Este archivo define el modelo de **Mascotas** para gestionar las mascotas en la base de datos. Los principales campos incluyen el nombre, especie, raza, edad, sexo, peso, estado de adopción, descripción, fecha de ingreso, y una URL para su foto. Este modelo es fundamental para manejar las operaciones relacionadas con las mascotas en la aplicación, como listar las mascotas disponibles para adopción o registrar nuevas mascotas

De ahí pasamos a **mascotasController.js**

El archivo tiene como propósito proporcionar la **lógica de negocio** para gestionar las mascotas, implementando las funcionalidades que permiten a la aplicación manejar las operaciones básicas con los registros de mascotas. Estas operaciones incluyen:

- Crear un nuevo registro de mascota.
- Buscar todas las mascotas.
- Buscar una mascota por su ID.
- Actualizar la información de una mascota.
- Eliminar una mascota.

De ahí vamos a **mascotasRouter.js**

Este archivo tiene el objetivo de crear un **enrutador (router)** para gestionar las peticiones relacionadas con las mascotas. Cada ruta está asociada a una operación específica (como crear, buscar, actualizar o eliminar una mascota) y utiliza las funciones definidas en el controlador mascotasController.js.

Y hacemos un commit

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git add .
warning: in the working copy of 'src/controladores/mascotasController.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/modelos/mascotaModelo.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/controladores/mascotasController.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/modelos/mascotaModelo.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/rutas/mascotasRouter.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/modelos/mascotaModelo.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/rutas/mascotasRouter.js', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'src/rutas/mascotasRouter.js', LF will be replaced by CRLF the next time Git touches it
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git commit -m "agregada modelos, rutas y controladores para mascotas"
>>
[main 143fbf9] agregada modelos, rutas y controladores para mascotas
 4 files changed, 282 insertions(+), 2 deletions(-)
 create mode 100644 src/controladores/mascotasController.js
```

Hacemos algo parecido para clientes

Objetivo del archivo clienteModelo.js:

El propósito de este archivo es definir el modelo **Clientes** utilizando **Sequelize**. Este modelo representa la tabla de los clientes en la base de datos, y establece las columnas y sus tipos de datos, así como las restricciones.

Objetivo del archivo clientesController.js:

Este archivo contiene las funciones controladoras que manejan las **operaciones CRUD** (Crear, Leer, Actualizar, Eliminar) para el modelo de Clientes.

Objetivo del archivo clientesRouter.js:

El propósito de este archivo es **definir las rutas** para manejar las solicitudes HTTP relacionadas con los clientes en la aplicación. Utiliza **Express** para crear y gestionar rutas, que conectan con las funciones controladoras definidas en clientesController.js.

Y hacemos un commit

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git add .
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git commit -m "agregada modelos, rutas y controladores para clientes"
 >>
[main 73014b7] agregada modelos, rutas y controladores para clientes
 4 files changed, 232 insertions(+), 1 deletion(-)
 create mode 100644 src/controladores/clientesController.js
 create mode 100644 src/modelos/clienteModelo.js
 create mode 100644 src/rutas/clientesRouter.js
```

Y por último para las solicitudes

Objetivo del archivo solicitudModelo.js:

El objetivo principal de este archivo es definir el modelo **SolicitudesAdopcion** utilizando **Sequelize**. Este modelo representa las solicitudes de adopción en la base de datos, estableciendo las relaciones entre las mascotas y los clientes que desean adoptarlas.

Objetivo del archivo solicitudesController.js:

Este archivo contiene las funciones que manejan las operaciones CRUD para las **Solicitudes de Adopción**, permitiendo crear, buscar, actualizar y eliminar solicitudes.

Objetivo del archivo solicitudesRouter.js:

El objetivo de este archivo es **definir las rutas** para manejar las solicitudes HTTP relacionadas con las **Solicitudes de Adopción**. Utiliza **Express** para establecer rutas que conectan con las funciones controladoras en solicitudesController.js.

Y hacemos un commit

```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git add .
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> git commit -m "agregada modelos, rutas y controladores para solicitudes"
 >>
[main 99568c1] agregada modelos, rutas y controladores para solicitudes
 4 files changed, 245 insertions(+)
 create mode 100644 src/controladores/solicitudesController.js
 create mode 100644 src/modelos/solicitudModelo.js
 create mode 100644 src/rutas/solicitudesRouter.js
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon>
```

Por parte de los verbos HTML usados son:

GET: Se utiliza para recuperar información de la base de datos sin alterar su estado.

Ejemplos: obtener una lista de solicitudes, buscar una solicitud específica por su ID, obtener información sobre una mascota o cliente.

Ruta: /buscarSolicitudesAdopcion, /buscarIdMascota/:id, /buscarClientes.

POST: Utilizado para crear un nuevo recurso. En este caso, se utiliza para crear nuevas solicitudes de adopción, clientes o mascotas.

Ejemplo: Al enviar los datos de un nuevo cliente o una nueva solicitud de adopción.

Ruta: /crearSolicitudAdopcion, /crearCliente.

PUT: Utilizado para actualizar un recurso existente por medio de su ID. En este caso, se usa para actualizar los detalles de una solicitud de adopción, cliente o mascota.

Ejemplo: Actualizar el estado de una solicitud de adopción o modificar la información de un cliente.

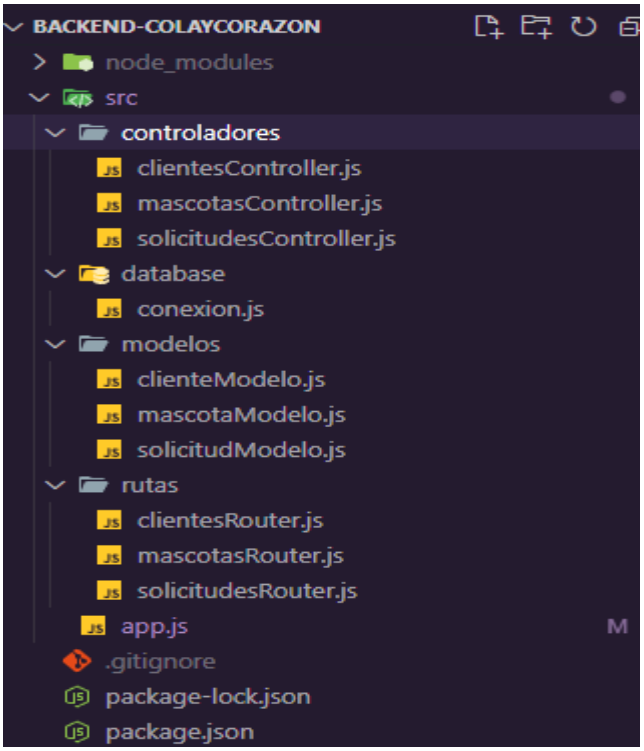
Ruta: /actualizarSolicitudAdopcion/:id, /actualizarCliente/:id.

DELETE: Se utiliza para eliminar un recurso existente en la base de datos por su ID. En este contexto, sirve para eliminar solicitudes, clientes o mascotas.

Ejemplo: Al eliminar un cliente o una solicitud de adopción de la base de datos.

Ruta: /eliminarSolicitudAdopcion/:id, /eliminarMascota/:id

Y así queda nuestro proyecto en Visual Studio Code



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.

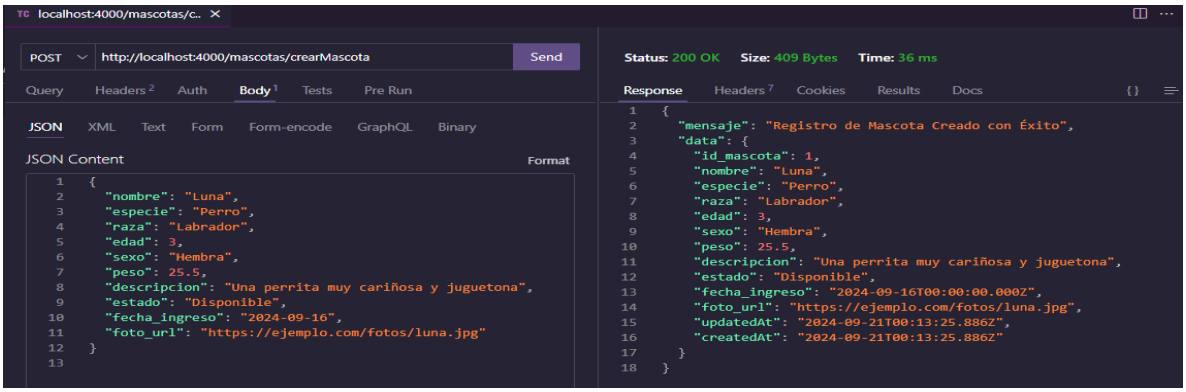
Iniciamos nuestro servicio

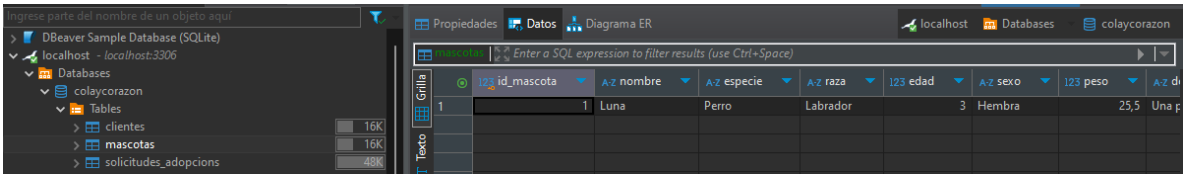
```
PS C:\Users\Kengua\Documents\Diplomado2024\BackEnd-ColaYCorazon> npm start

> backend-colaycorazon@1.0.0 start
> nodemon ./src/app.js

[nodemon] 3.1.5
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node ./src/app.js`
Executing (default): SELECT 1+1 AS result
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'mascotas' AND TABLE_SCHEMA = 'cola
orazon'
Conexion a Base de datos correcta
Executing (default): SHOW INDEX FROM `mascotas`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'clientes' AND TABLE_SCHEMA = 'cola
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'clientes' AND TABLE_SCHEMA = 'cola
orazon'
Executing (default): SHOW INDEX FROM `clientes`
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'solicitudes_adopcions' AND TABLE_S
EMA = 'colaycorazon'
Executing (default): SHOW INDEX FROM `solicitudes_adopcions`
Servidor Inicializado en el puerto 4000
Executing (default): INSERT INTO `mascotas` (`id_mascota`,`nombre`,`especie`,`raza`,`edad`,`sexo`,`peso`,`descripcion`,`estado`,`fecha_ingreso`,`foto_url`
createdAt`,`updatedAt`) VALUES (DEFAULT,?,?,?,?,?,?,?,?,?,?,?,?);
```

Y con ayuda de **Thunder client** comenzamos a verificar nuestras operaciones y rutas

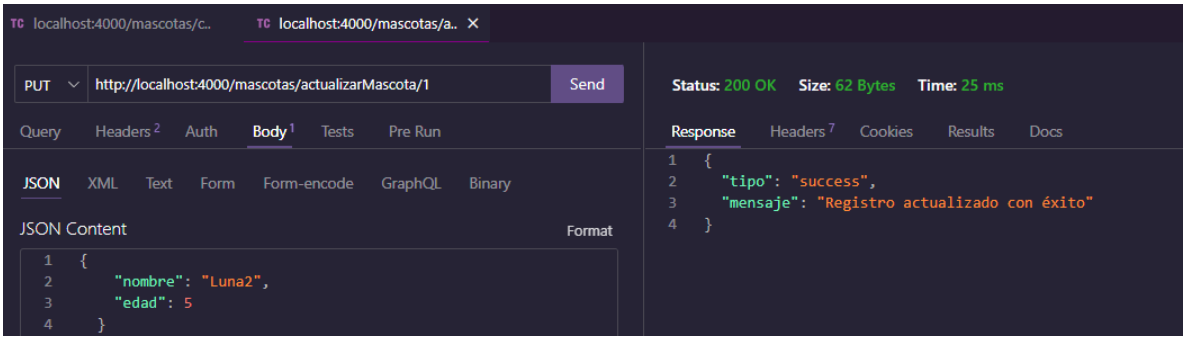
Funciones de mascotas

Crear

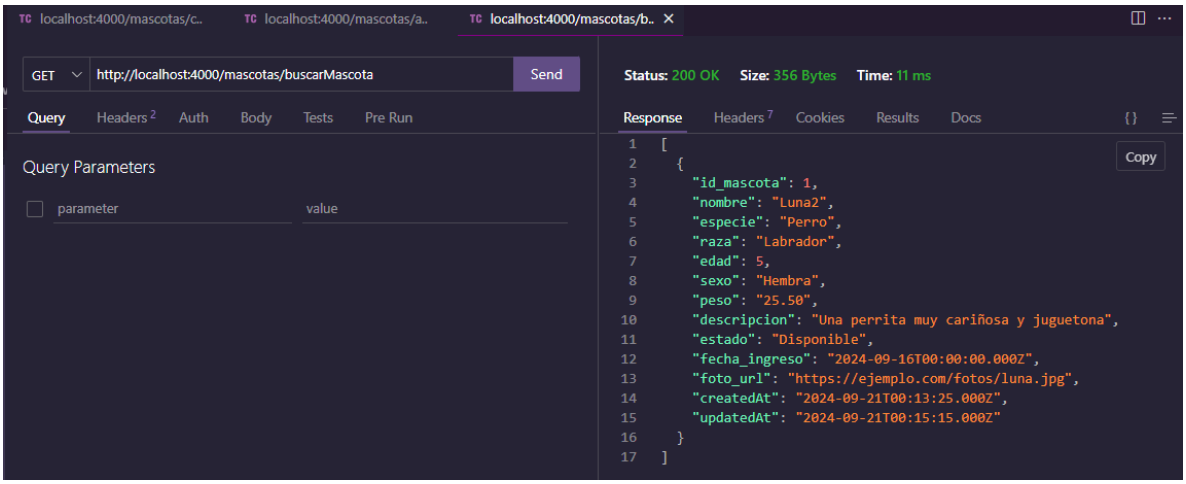




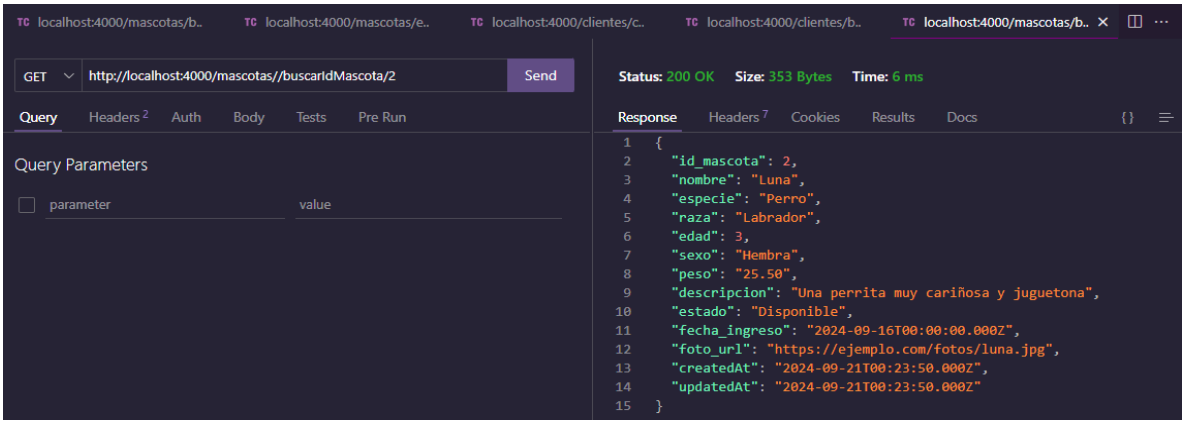
Actualizar



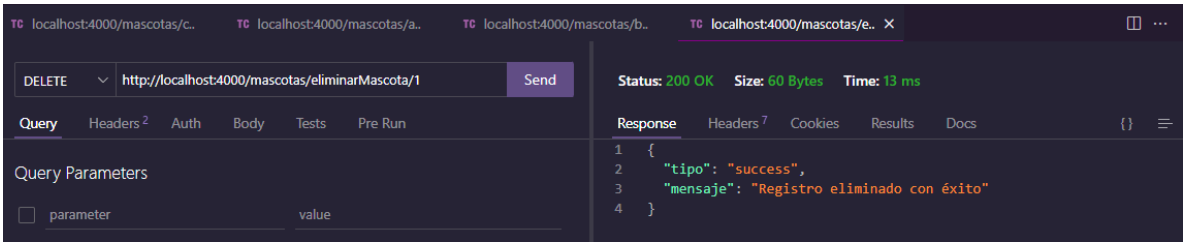
Buscar



Buscar por ID

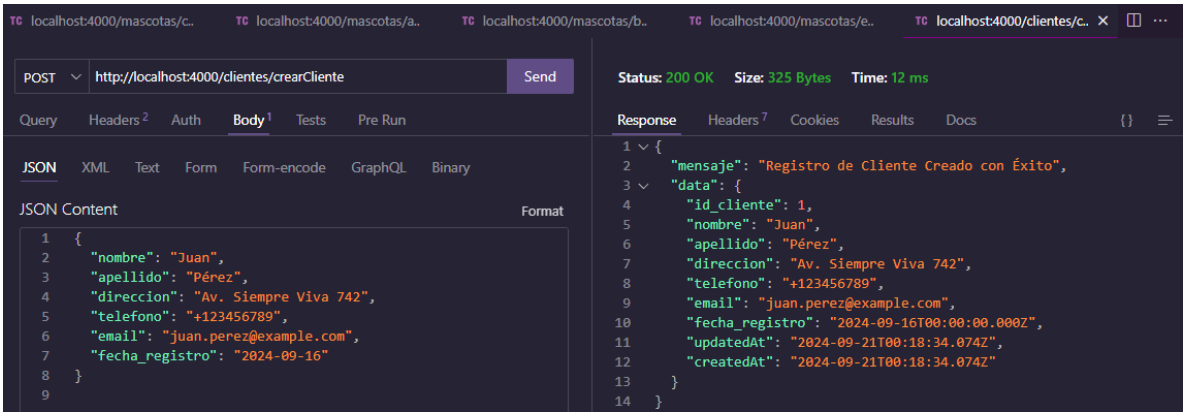


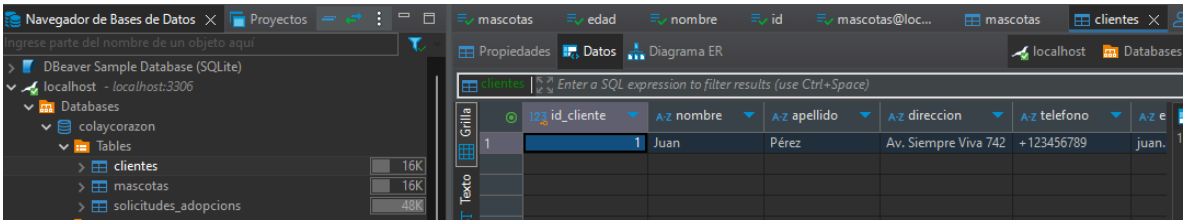
Eliminar



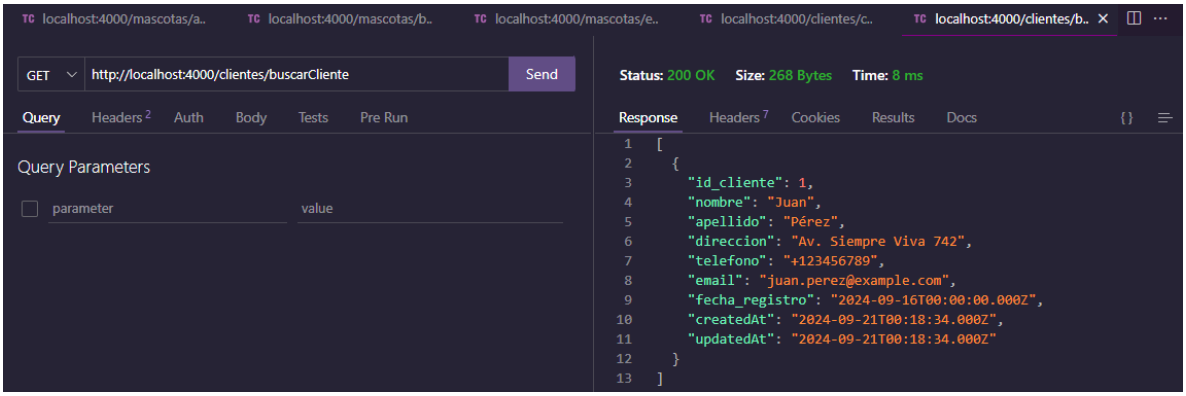
Funciones de clientes

Crear

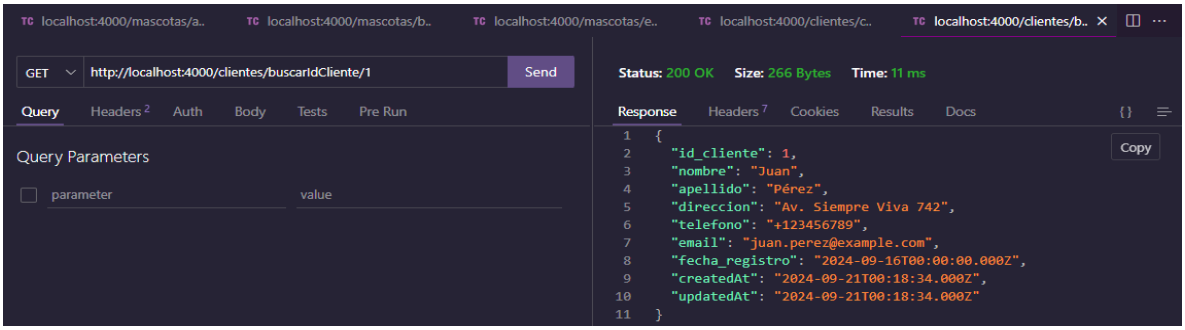




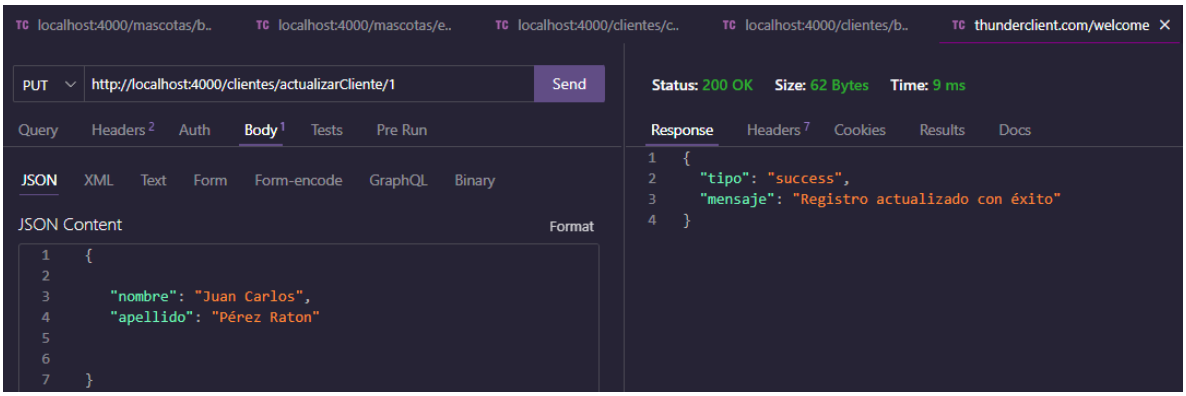
Buscar

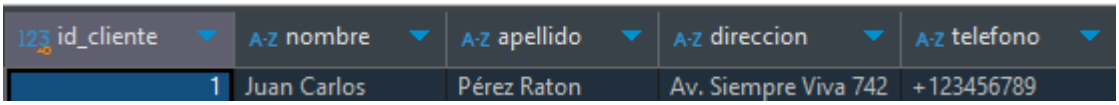


Buscar por ID

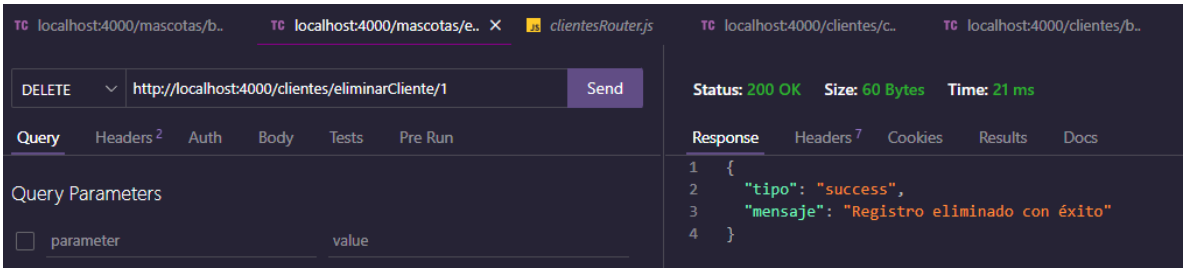


Actualizar



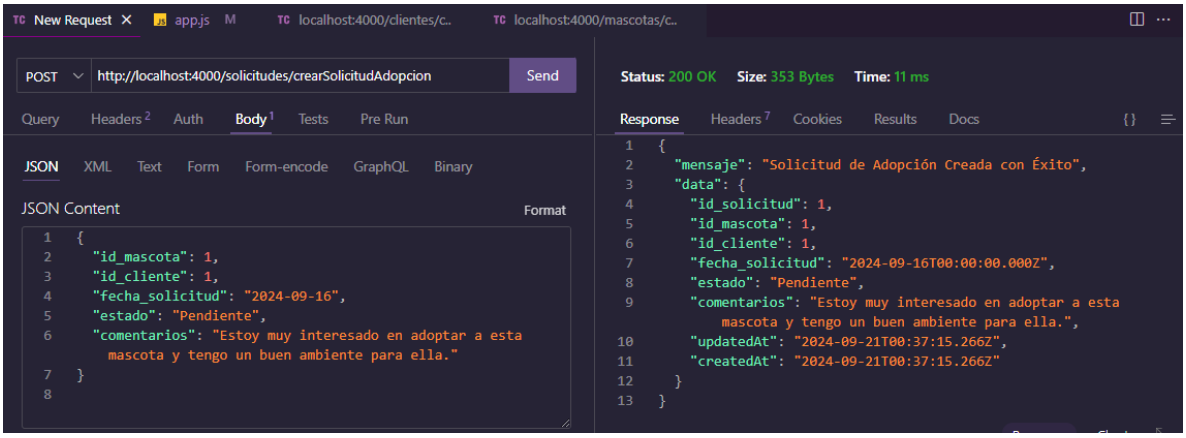


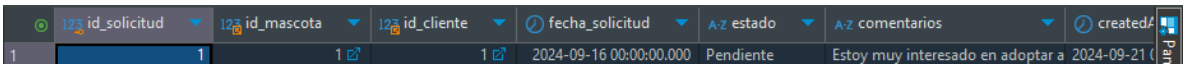
Eliminar



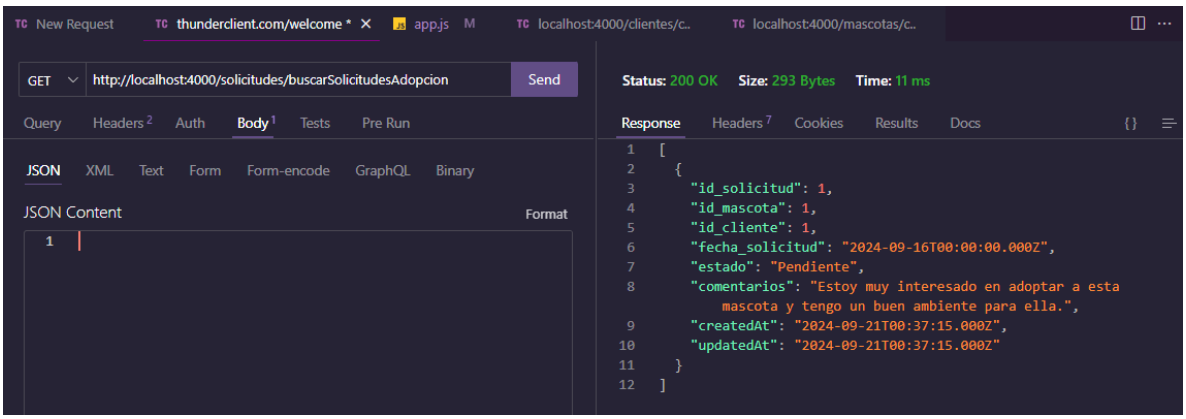
Funciones de solicitudes

Crear

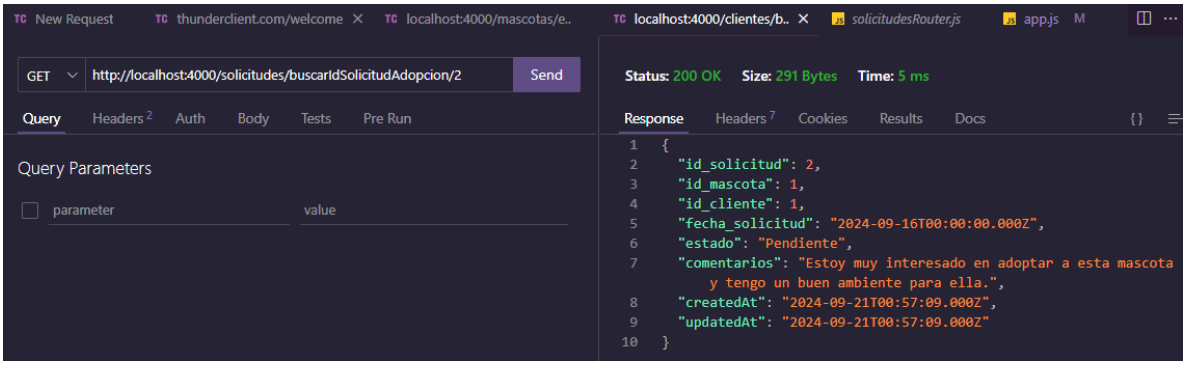




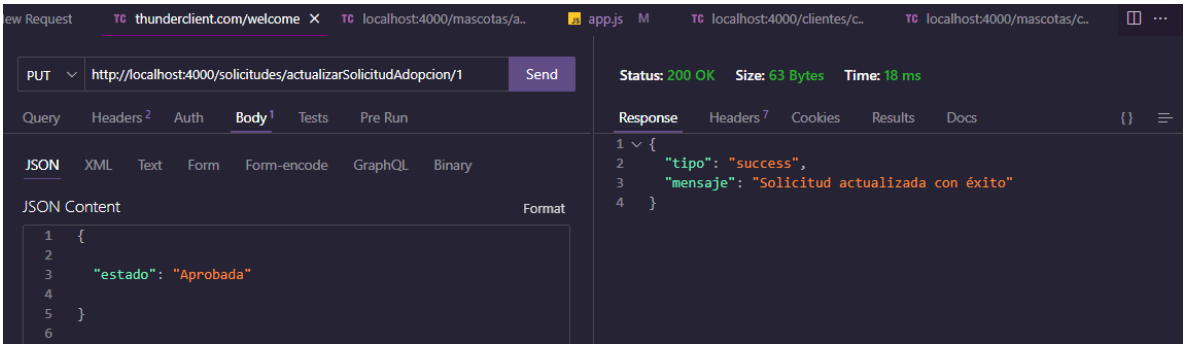
Buscar

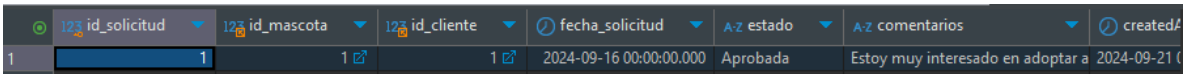


Buscar por ID



Actualizar





Eliminar