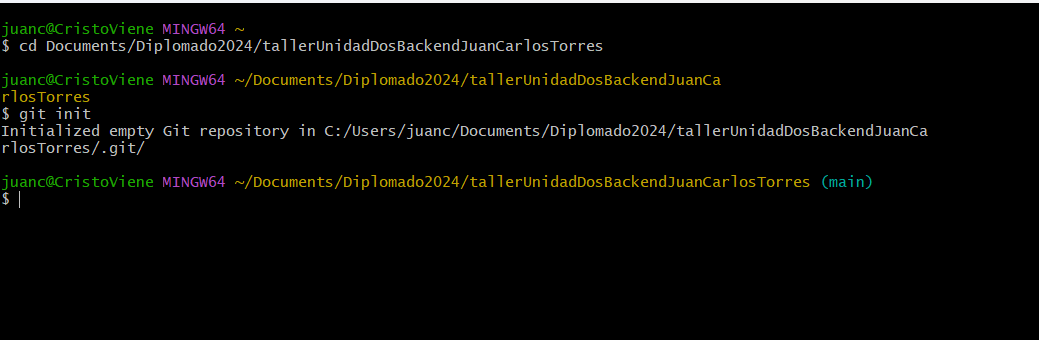
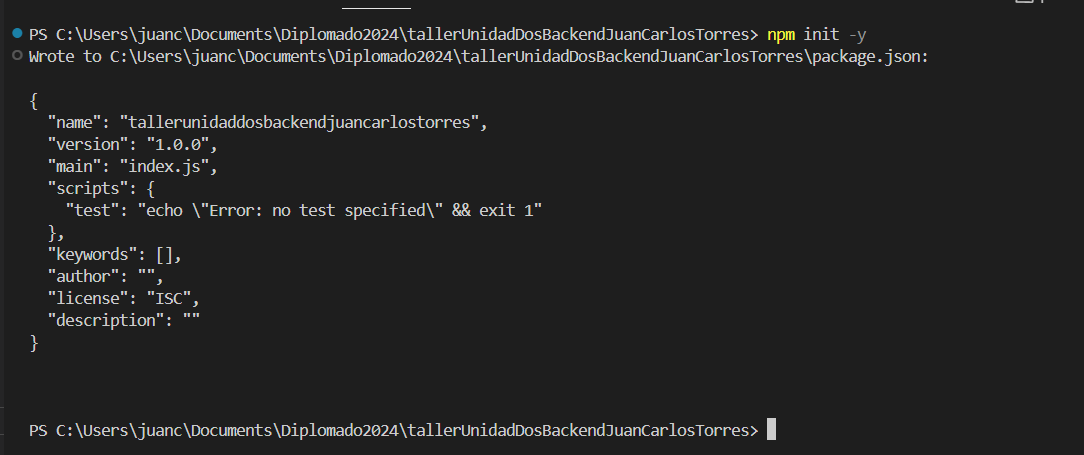
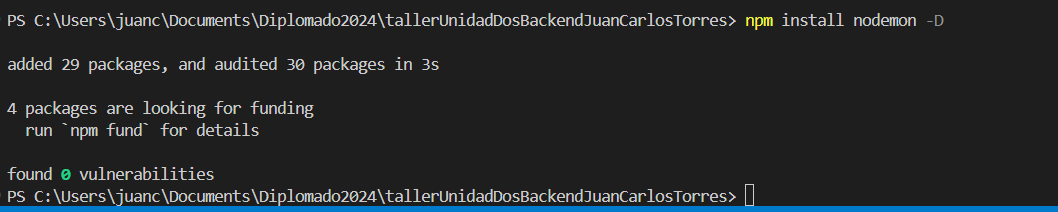
Entre a la carpeta tallerUnidadDosBackendJuanCarlosTorres para crear un repositorio para el seguimiento de versiones de mi proyecto



Desde visual Visual Studio entramos a la carpeta y escribimos el comando npm init -y para inicializar nuestro proyecto de backend



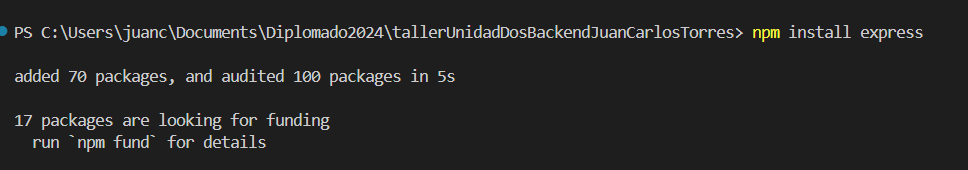
Escribimos el comando npm Install nodemon -D descarga un modulo para desarrollar el cual se encarga de reiniciar la aplicación de Node.js cada vez que detecta cambios en el código, esto evita que tenga que detener el servicio cada ves haga un cambio en mi codigo



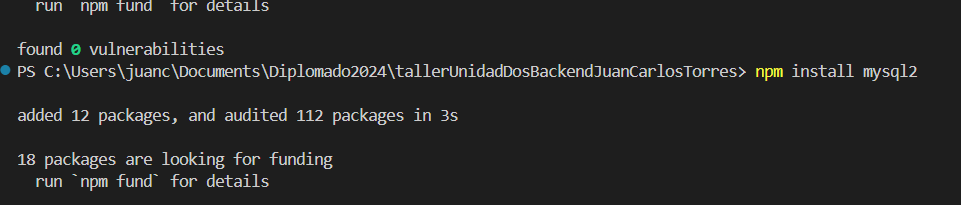
Cree una nueva caparte llamada src, es donde voy apilar todo mi código



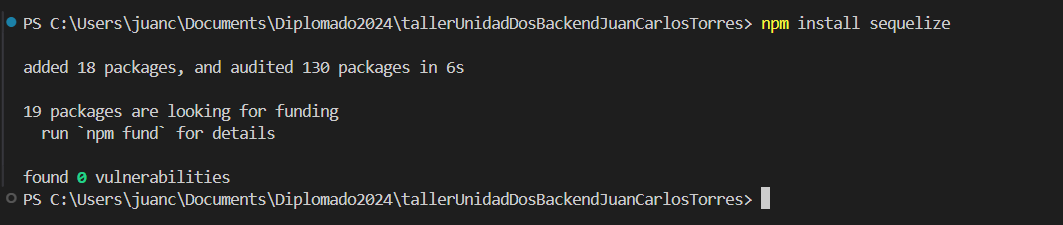
Instalamos express, que es el frameworks que me va permitir desarrollar el backend



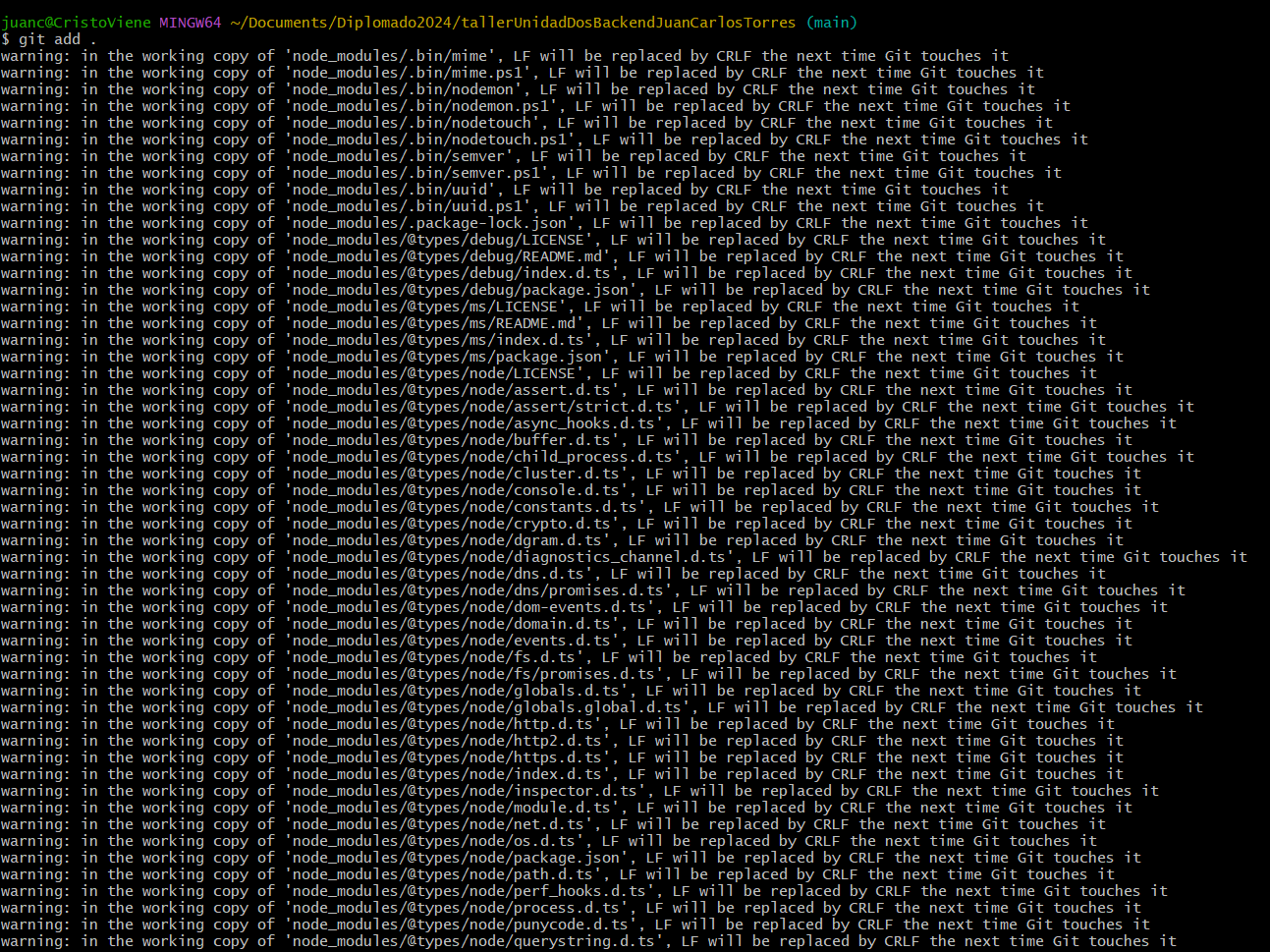
Instalamos mysql2 para que me permita conectarme base de datos mysql2



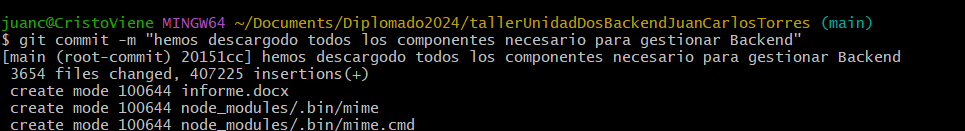
Instalamos sequelize porque este es un ORM el cual es un intermediario entre el desarrollador y la base de datos, evitando que este tenga que preocuparse por saber exactamente cuáles son las sentencias de mysql2 para gestionar la base de datos



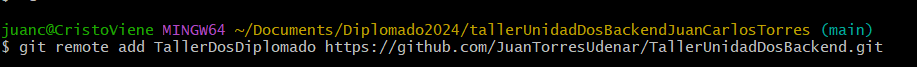
Me voy a mi consola Git y añado los cambios que tengo hasta el momento al área de preparación



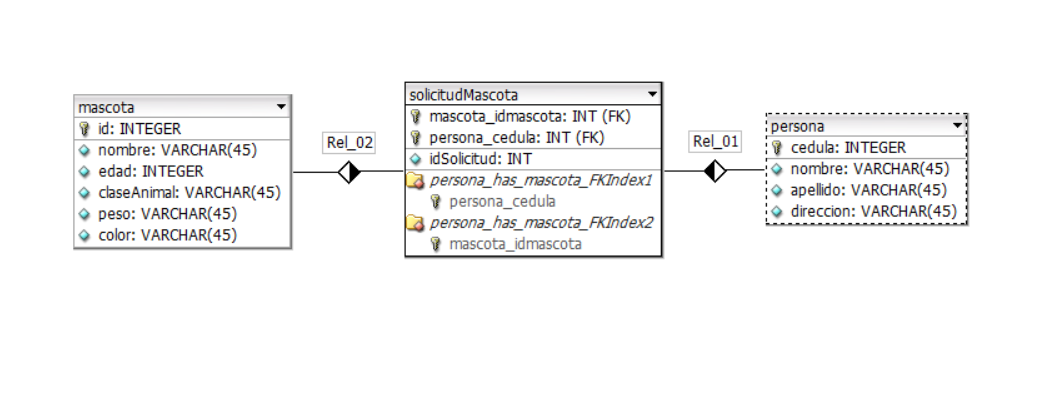
Hacemos un commit



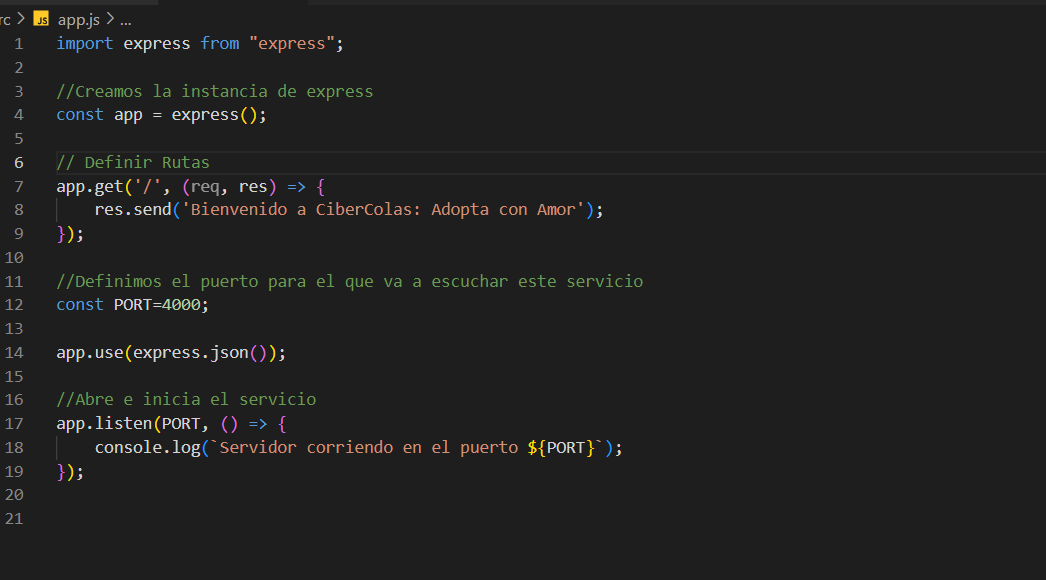
Sincronizamos el repositorio con el repositorio local



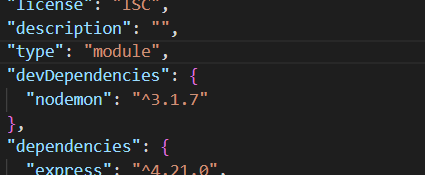
Hice un diagrama entidad relación para tener una idea clara de cuantas tablas hacer para mi base de datos, los atributos



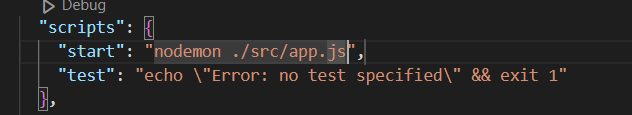
Empezamos a codificar nuestro archivo app.js



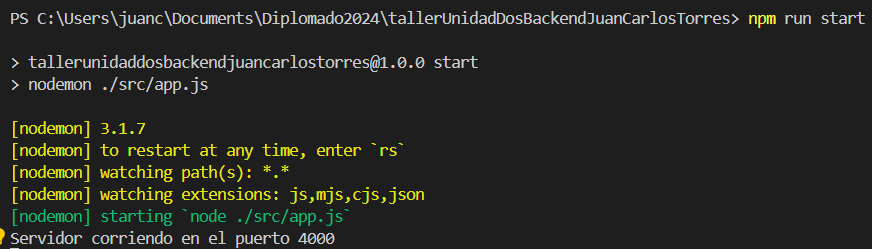
Se me había olvidado decirle que se debe comportar como un modulo



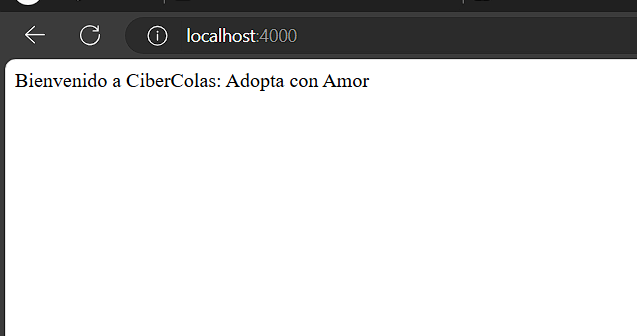
Me estaba olvidando de hacer un script que contiene la ruta donde esta el archivo app.js



Este comando es para correr el Scripts el cual se llama start, con esto el sistema esta pendiente de los cambios que yo haga y me los actualiza en caliente, evitando de que este bajando y subiendo el servicio a cada rato



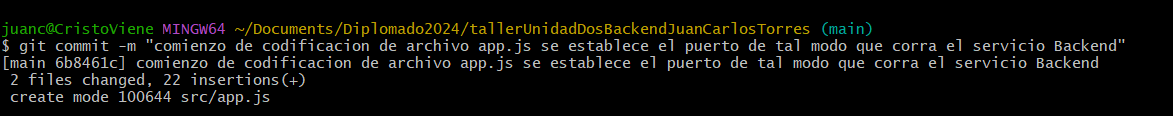
Podemos visualizar el mensaje de bienvenida y el nombre de la empresa la cual se encarga en dar en adopción mascotas



Añado los cambios que tengo hasta el momento al área de preparación



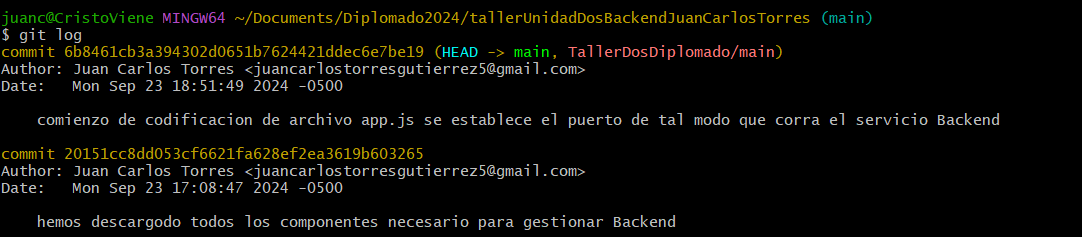
Hacemos un commit



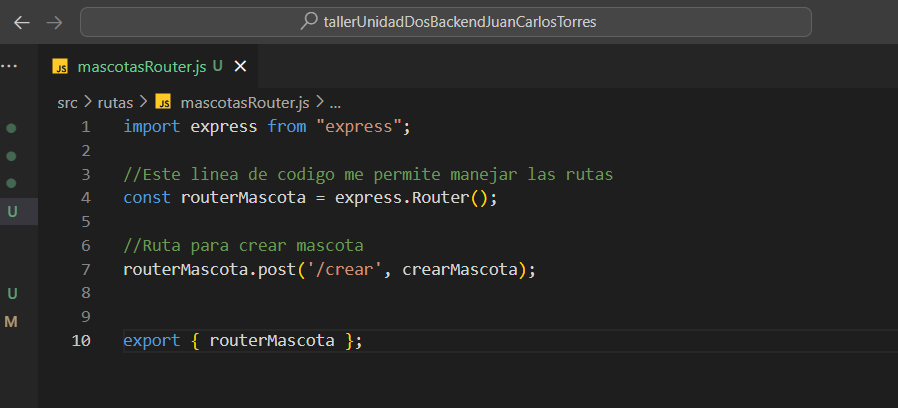
Subimos el avance



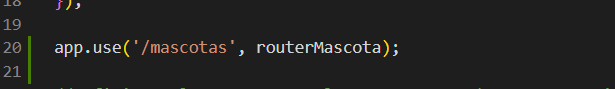
Evidenciamos los commit que tenemos hasta el momento de nuestro proyecto



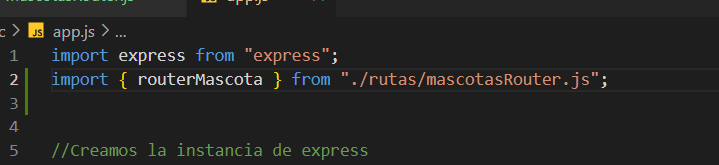
Creamos una carpeta dentro de la carpeta src la cual se llama rutas, y dentro de ruta un archivo llamado mascotasRouter.js, en ese archivo establecemos la primera ruta para crear una mascota



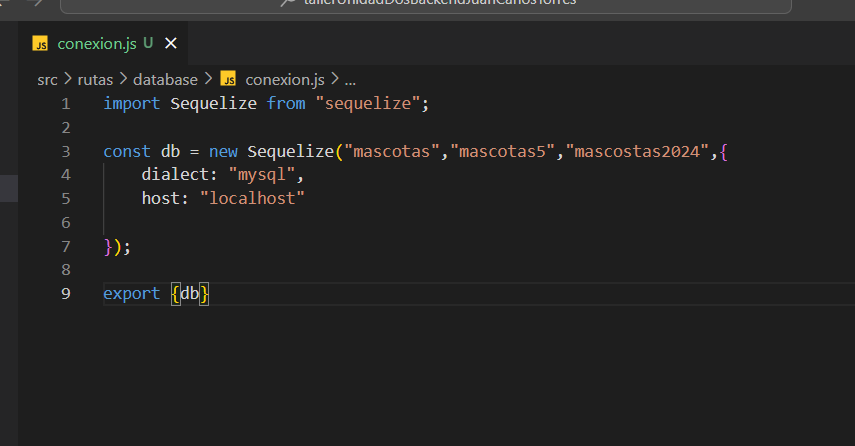
En el archivo app.js añadimos esta línea de código que me permite llamar, usar o implementar la ruta que ya establecí



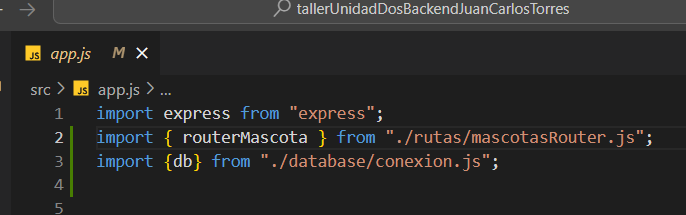
E importamos routerMascota en el archivo app.js



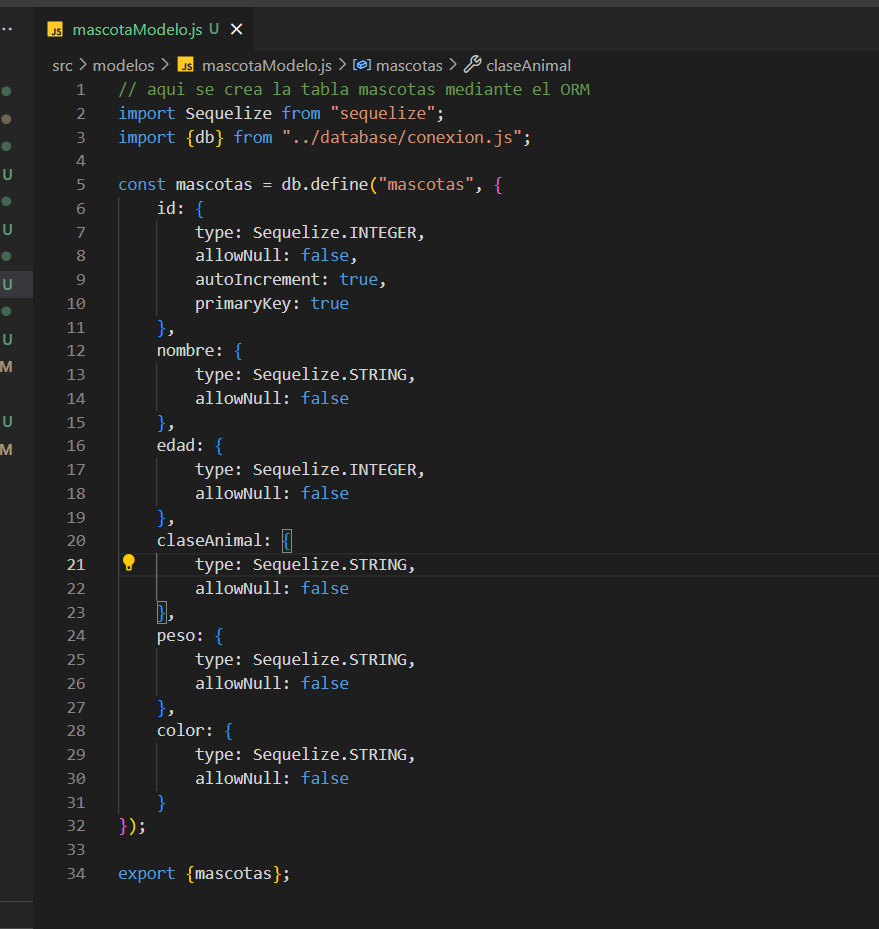
Dentro de src se creo una carpeta que sellama database, y dentro de database un archivo que se llama conexión.js, donde instanciamos db la cual contiene el nombre del la base de datos, el usuario y la contraseña. También se especifica que se está usando MySQL como sistema de gestión de bases de datos y que la base datos está alojada en el mismo servidor donde se ejecuta la aplicación..



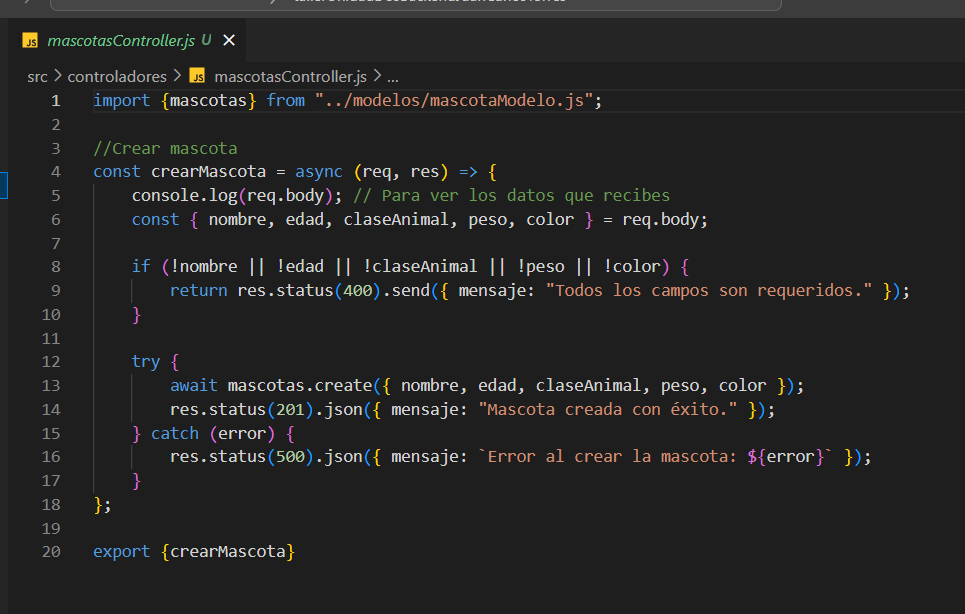
Se importa db desde el archivo app.js

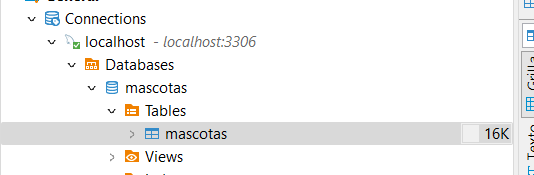


Dentron de src, se creo una carpeta llamada modelo, y dentro de modelo un archivo llamado mascotaModelo.js. en este archivo se define una tabla en la base de datos para almacenar información sobre mascotas, incluyendo su id, nombre, edad, clase de animal, peso y color.

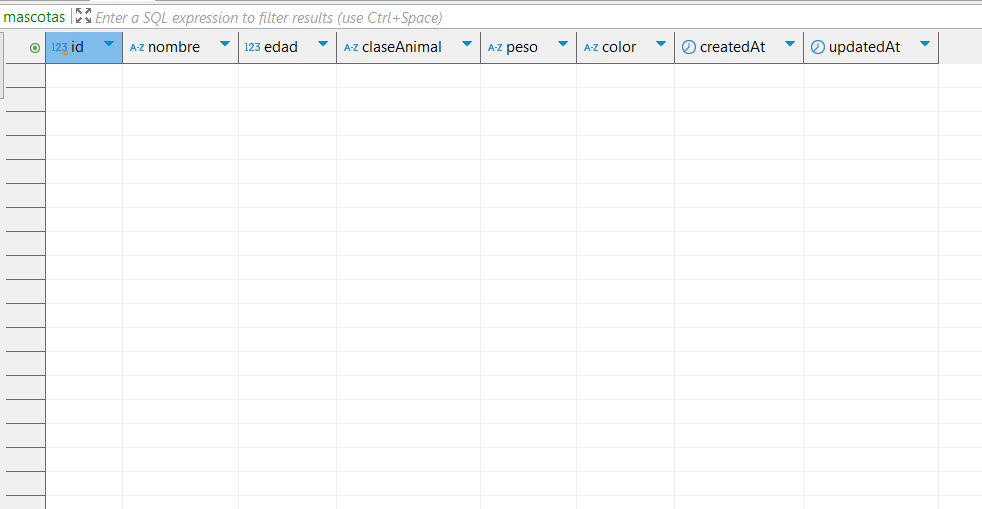


Dentro de src creamos otra carpeta que se llama controller, dentro de controller se creo el archivo mascotasController.js. En este archico se define crearMascota el cual permite recibir datos sobre una nueva mascota a través de una solicitud HTTP, valida que todos los campos requeridos estén presentes y, si es así, intenta crear un nuevo registro en la base de datos.

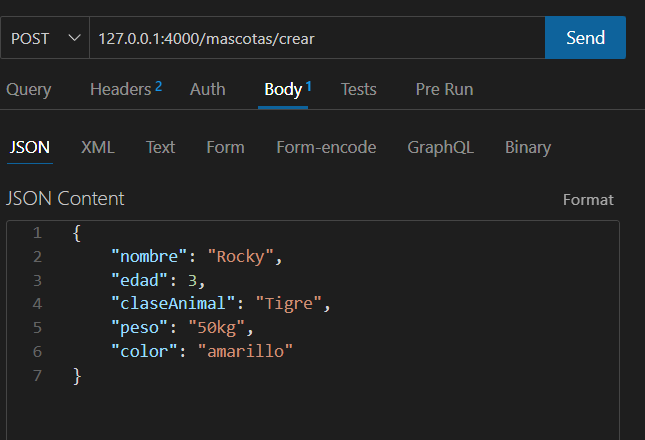


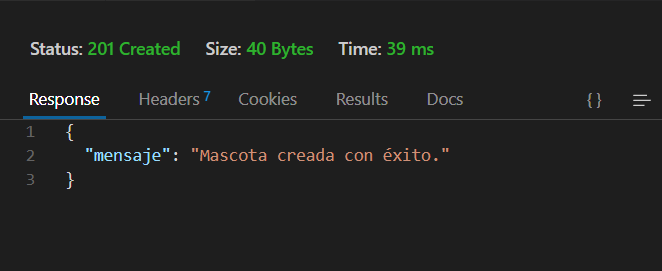
Podemos visualizar que ya se creo la tabla en la base de datos  


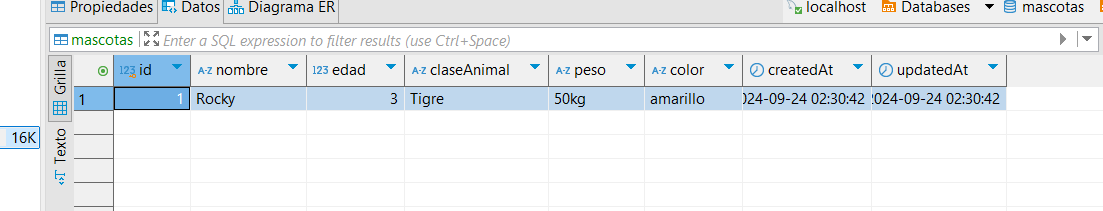
Miramos los atributos de la tabla



Enviamos una solicitud mediante POST para crear una mascota



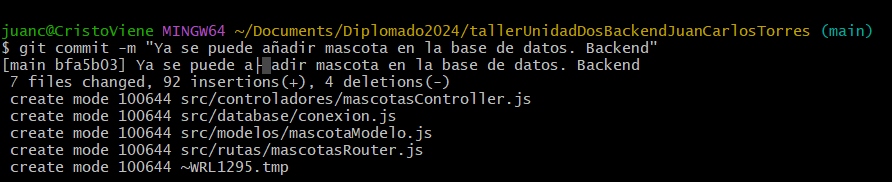




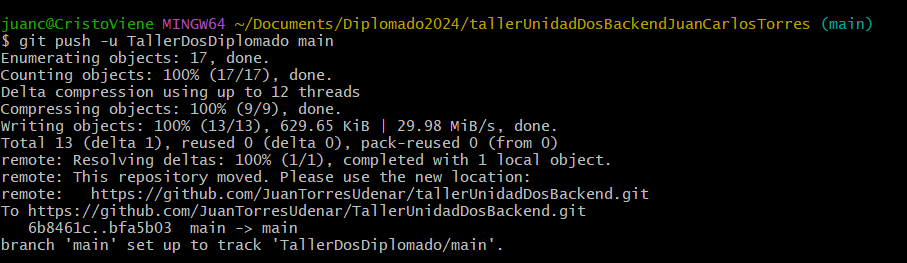
Añado los cambios que tengo hasta el momento al área de preparación



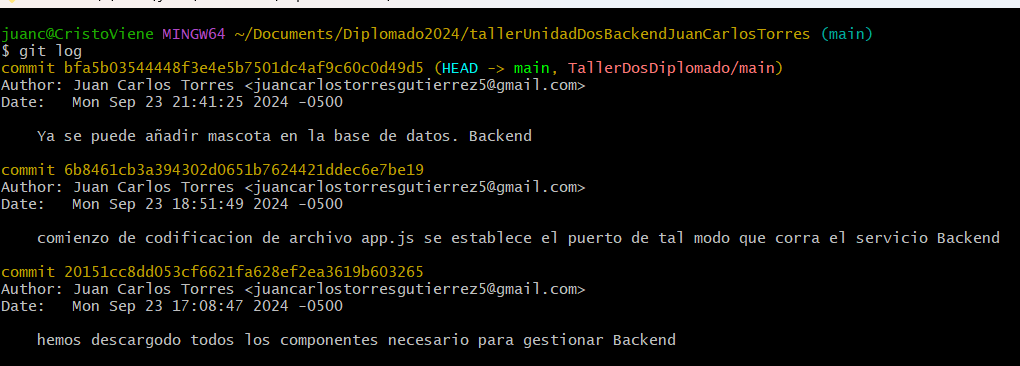
Hacemos un commit



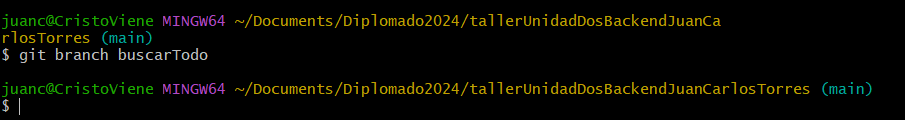
Subimos el avance



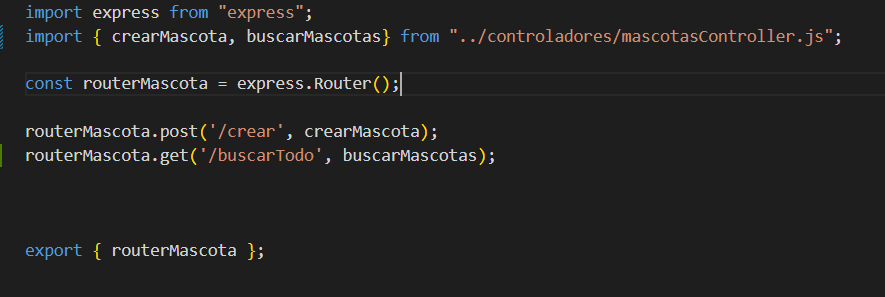
Se evidencia los commit que se lleva hasta el momento



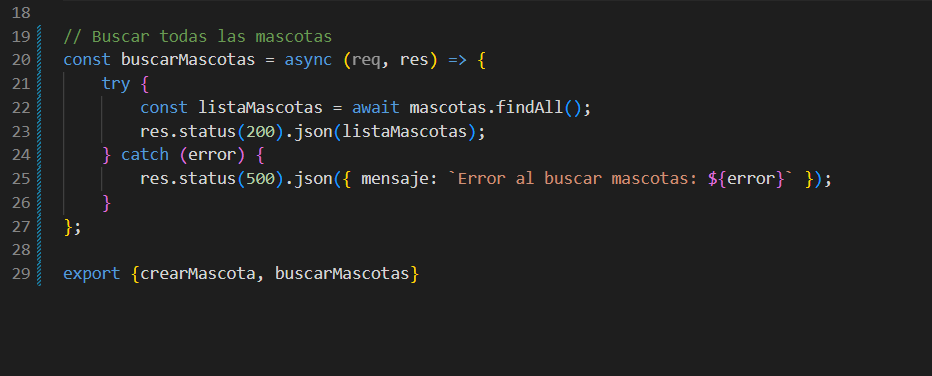
Se crea una rama llamada buscarTodo, para desarrollar en esa rama la función buscarTodo



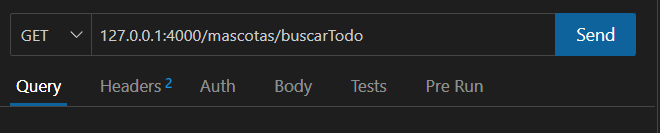
En el archivo mascotasRouter.js codificamos la ruta buscarMascotas

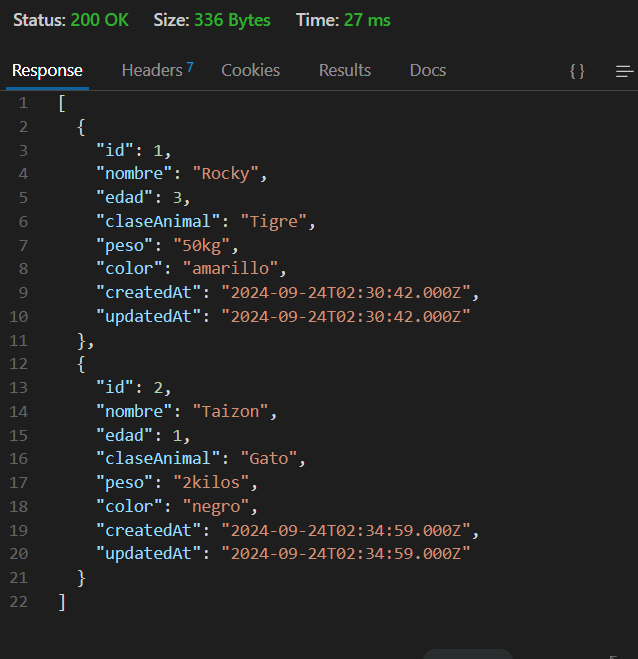


Se procedió a codificar la funcionalidad buscar a todas las mascotas dentro del archivo pertinente el cual es mascotasController.js

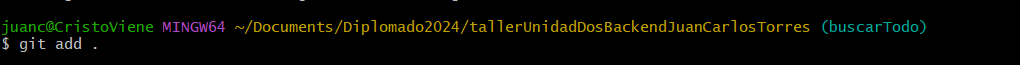


Se envió una solicitud mediante GET para crear buscar a todas las mascotas que existen en el momento





Añado los cambios que tengo hasta el momento al área de preparación



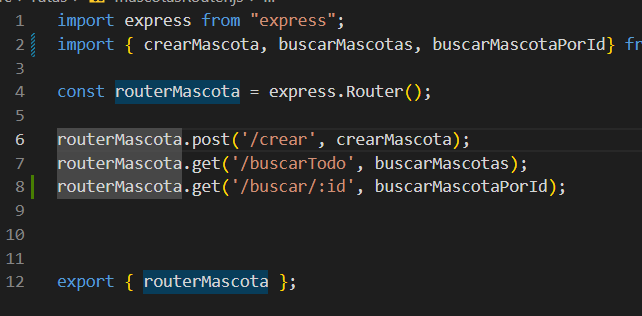
Hice un merge de los cambios realizados en la rama diseno a la rama main

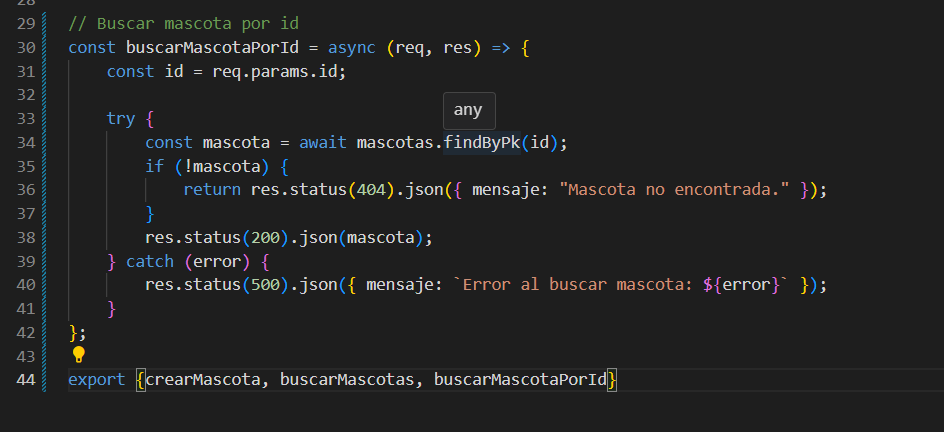


Se crea una rama llamada buscarId, para desarrollar en esa rama la función buscar por id

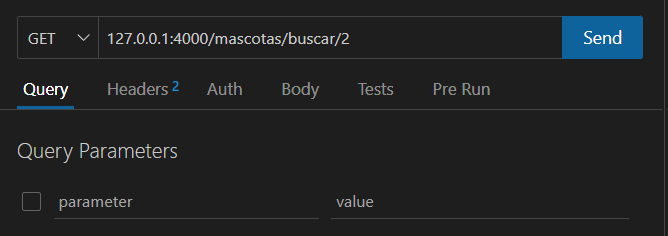


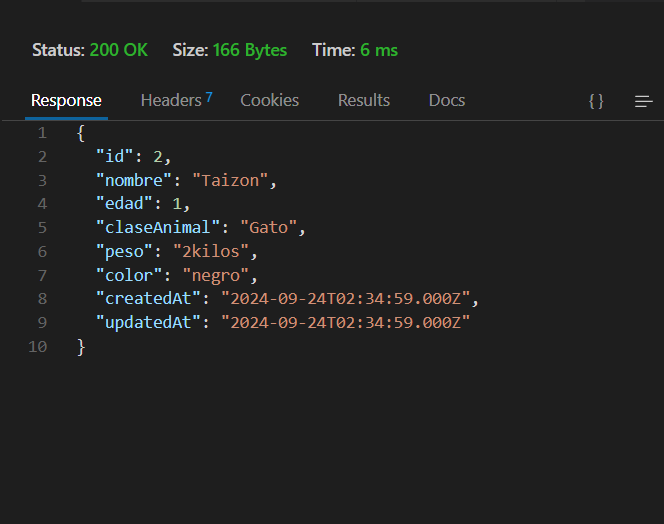
En el archivo mascotasRouter.js codificamos la ruta buscar por id



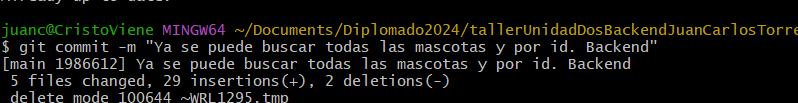
Se procedió a codificar la funcionalidad buscar mascota por id dentro del archivo pertinente el cual es mascotasController.js

Se envió una solicitud mediante GET para buscar mascota por id

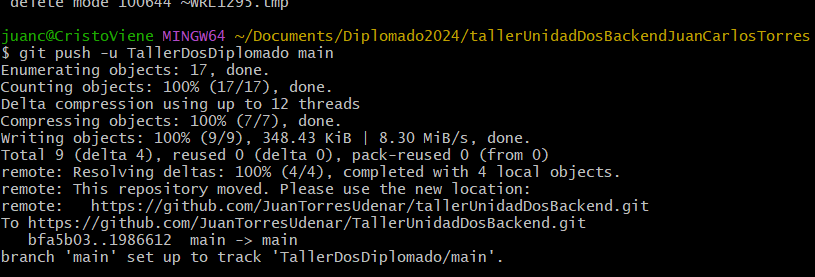




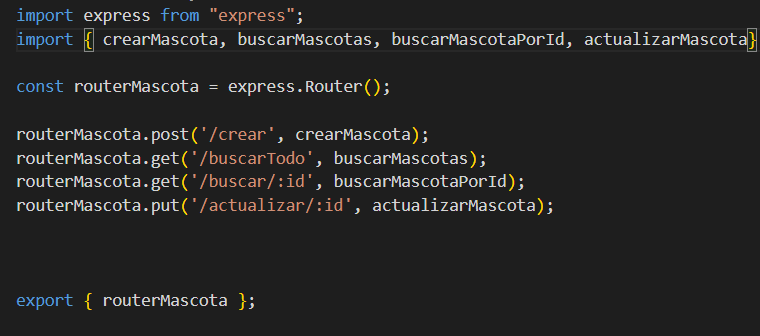
Hacemos un commit



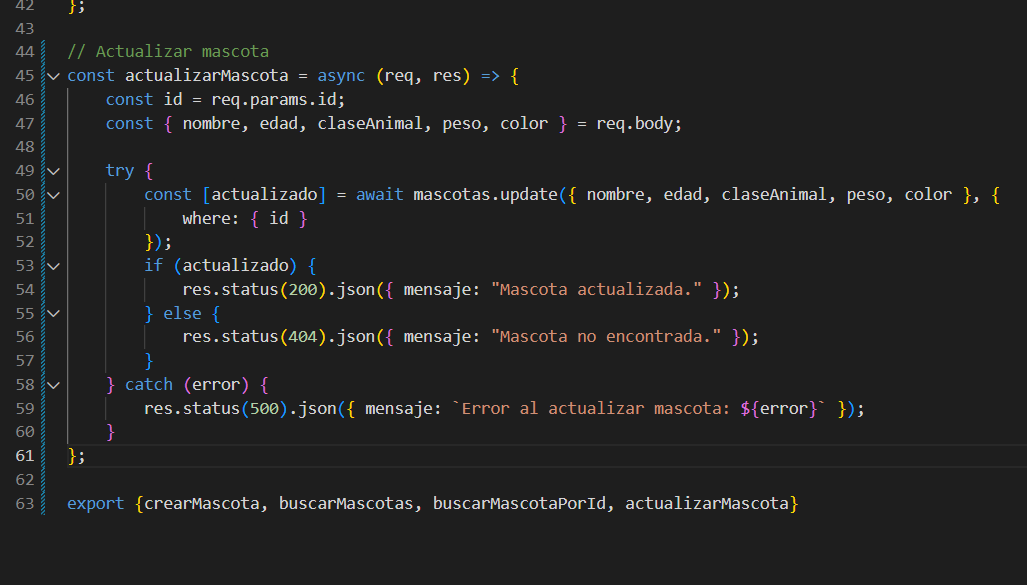
Subimos cambios



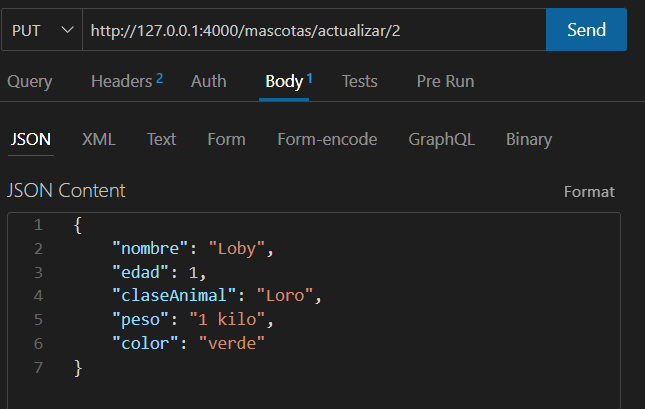
En el archivo mascotasRouter.js codificamos la ruta actualizar

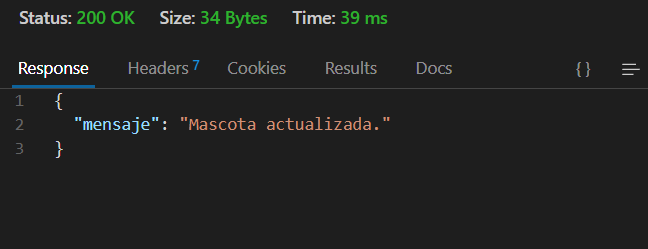


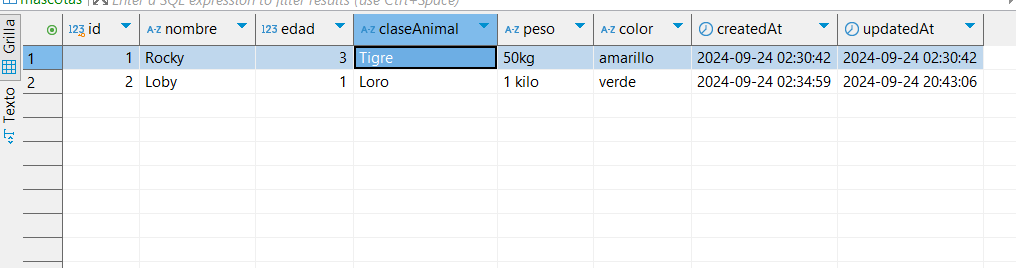
Se procedió a codificar la funcionalidad actualizar mascota dentro del archivo pertinente el cual es mascotasController.js



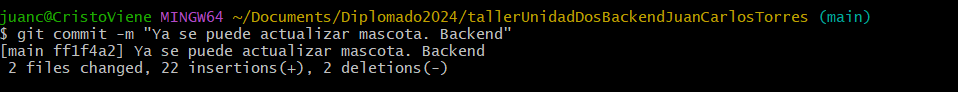
Se envió una solicitud mediante PUT para actualizar mascota



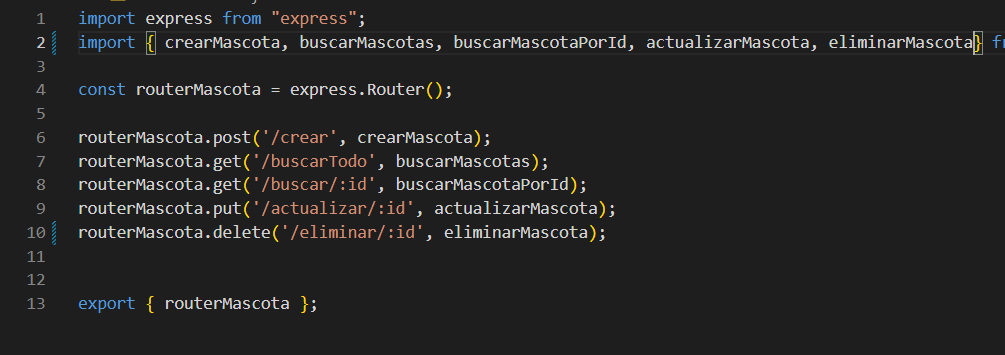




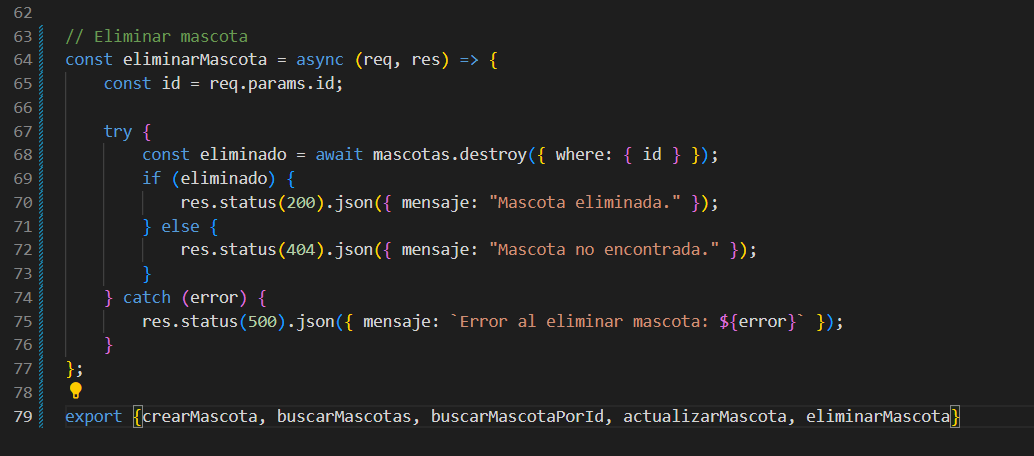
Hice un commit



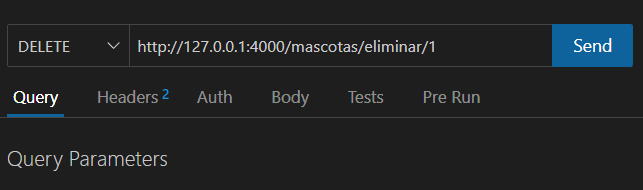
En el archivo mascotasRouter.js codificamos la ruta eliminar mascota

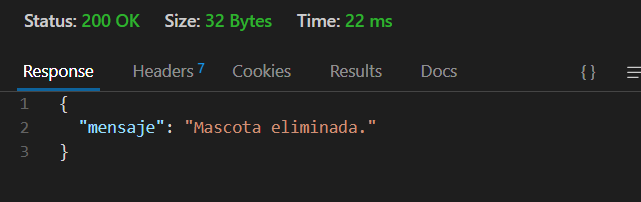


Se procedió a codificar la funcionalidad eliminar mascota dentro del archivo pertinente el cual es mascotasController.js

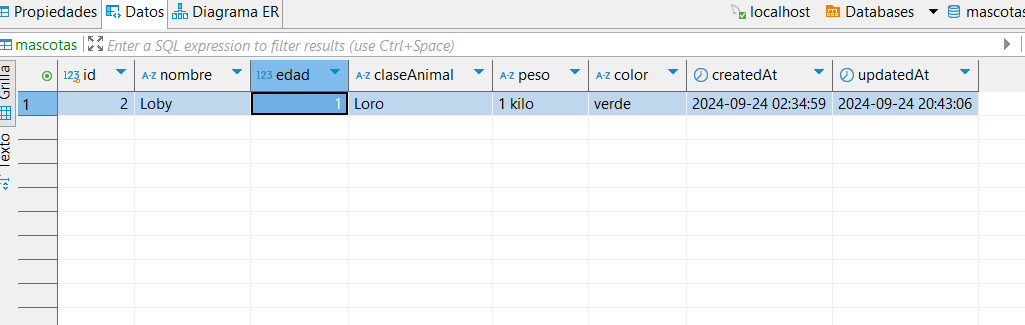


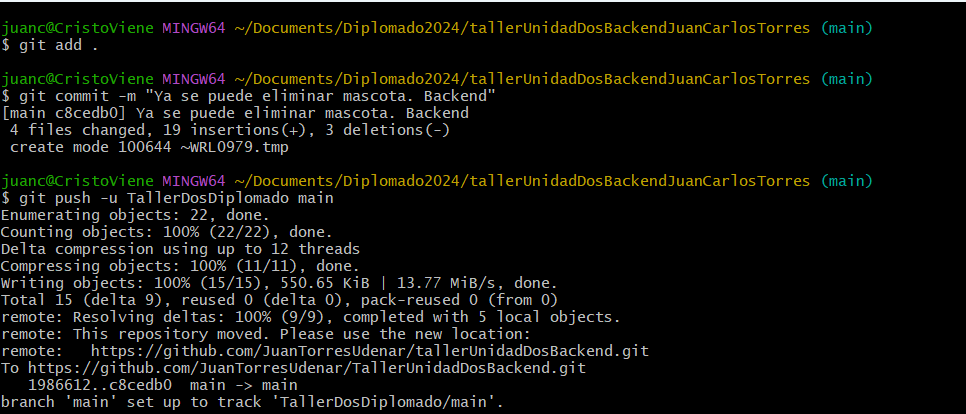
Se envió una solicitud mediante PUT para eliminar mascota



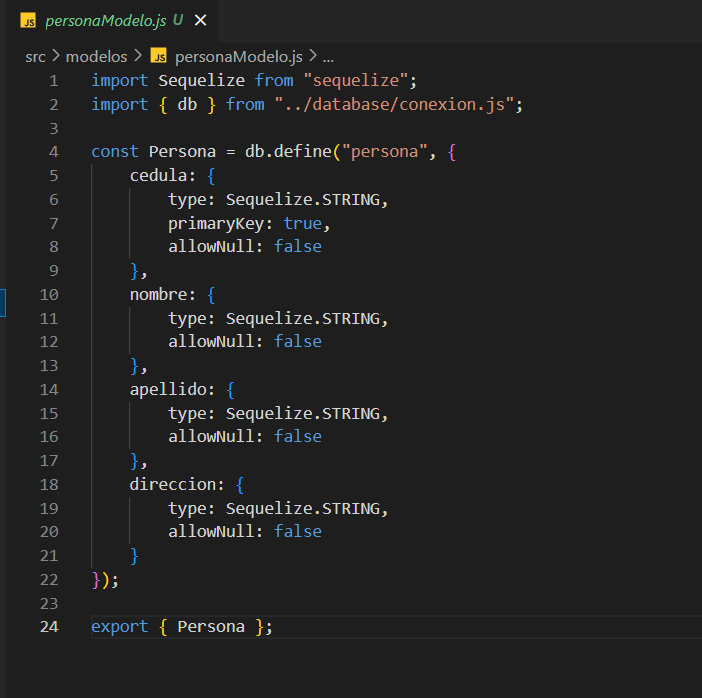


Se elimino con éxito, habían 2 mascota, ahora solo hay una

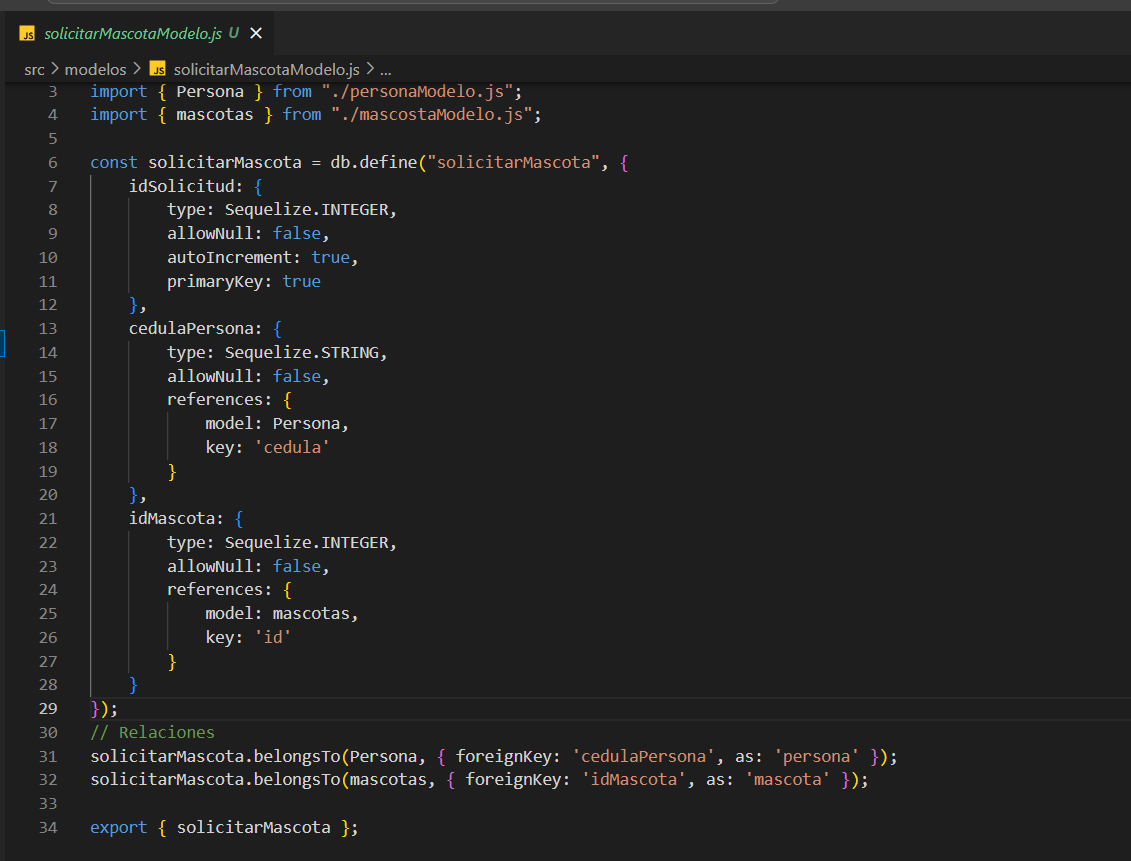




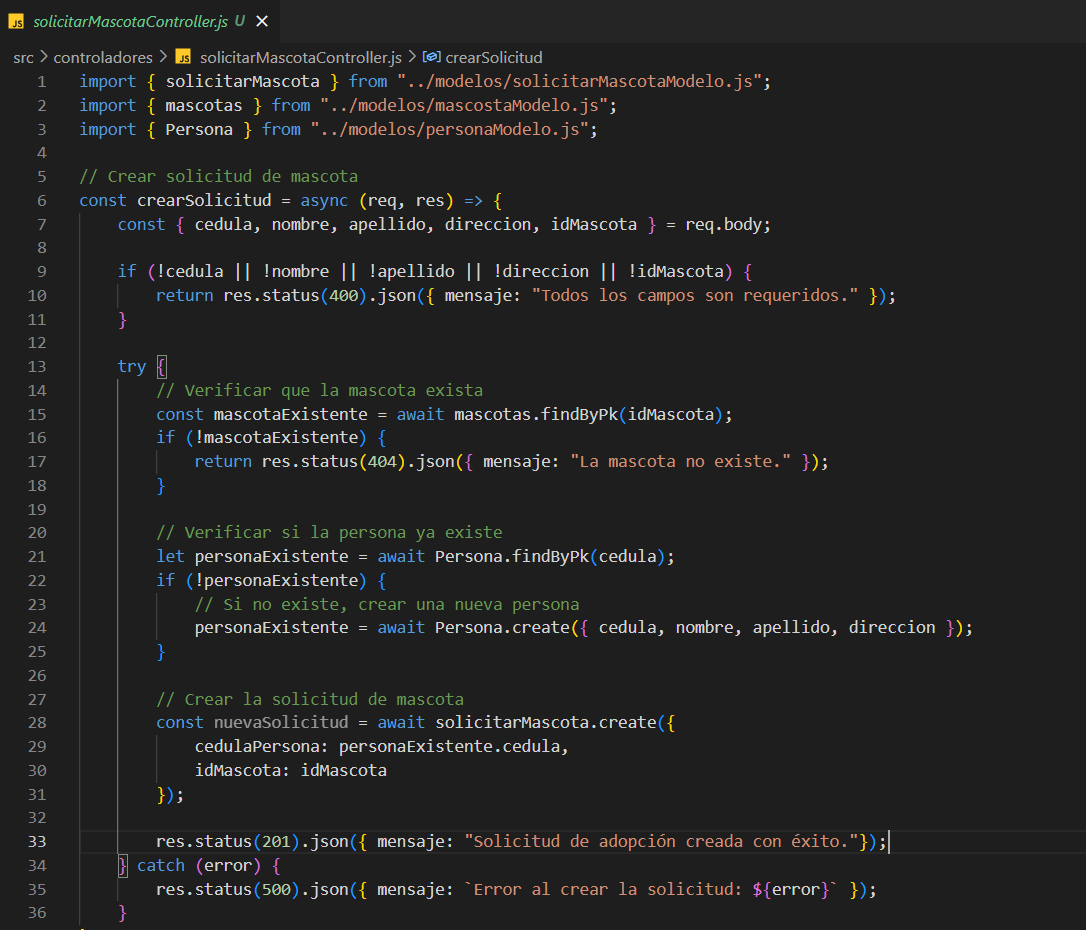
Dentro de la carpeta modelos se creo la carpeta modeloPesrona.js, en este archivo se define una tabla llamada persona en la base de datos para almacenar información sobre la persona quien solicitad la adopción de la mascota, la tabla tiene la siguientes columnas cedula, nombre, apellido y direccion.



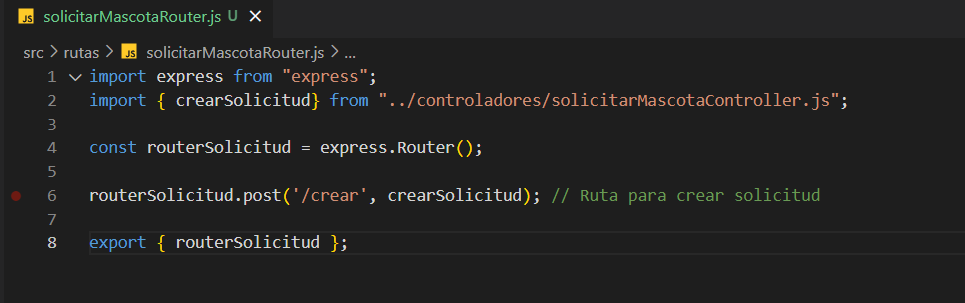
Dentro de la carpeta modelos se crea la carpeta solicitarMascotaModelo.js, en este código se define una tabla solicitarMascota que registra las solicitudes de adopción de mascotas, donde cada solicitud está relacionada con una persona (identificada por su cédula) y una mascota (identificada por su ID). Las relaciones de la tabla permiten que se asocie una solicitud con una persona y una mascota en la base de datos



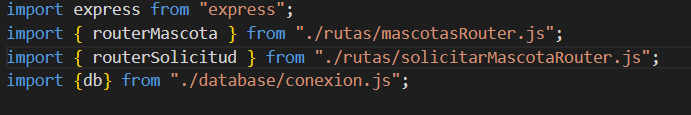
dentro de la carpeta controladores se creó el archivo solicitarMascotasController.js. En este archivo se codifico una función crearSolicidtdud de adopción de mascotas, validando que todos los campos requeridos estén presentes, verificando la existencia de la mascota y la persona en la base de datos, y creando nuevos registros si es necesario. Si todo es correcto, se guarda la solicitud de adopción y se envía una respuesta de éxito.



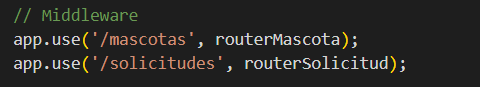
Dentro de la carpeta ruta creamos un archivo solicitarMascotaRouter.js en la cual se codifico la ruta crearSolicitud



En el archivo app.js importamos routerSolicitud



Y llamamos la rutas que hay en routerSolicitud



Ya tenemos 2 tablas más, personas y solicitarmascota

