1 º DAM

Antonio Trujillo Jiménez

Carlos Javier Moreno Vilches

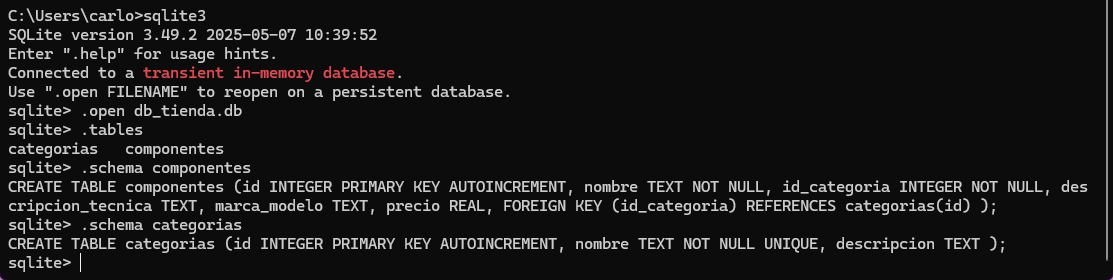
Mario López Sanmartín

José Márquez Gonzales

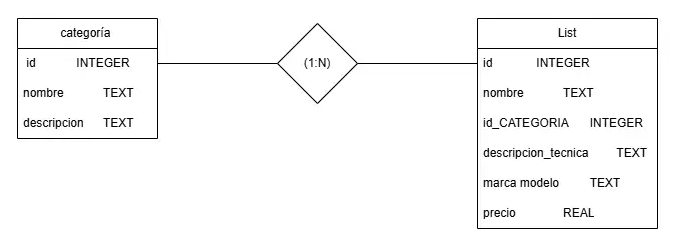
Proyecto de Backend de Componentes informáticos



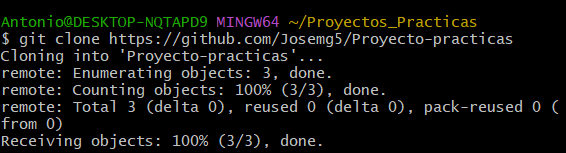
En este paso creamos la base de datos que lo realizara muestro compañero Carlos y las tablas que necesitamos para este proyecto y luego lo que haremos nuestro modelo entidad relacion



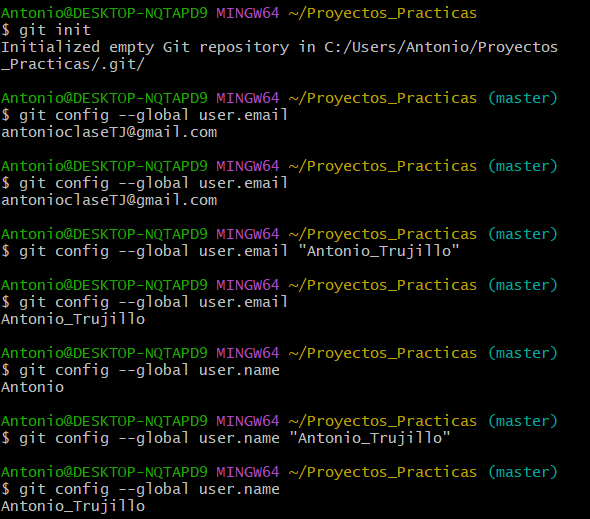
Aquí esta nuestro modelo entidad relaccion

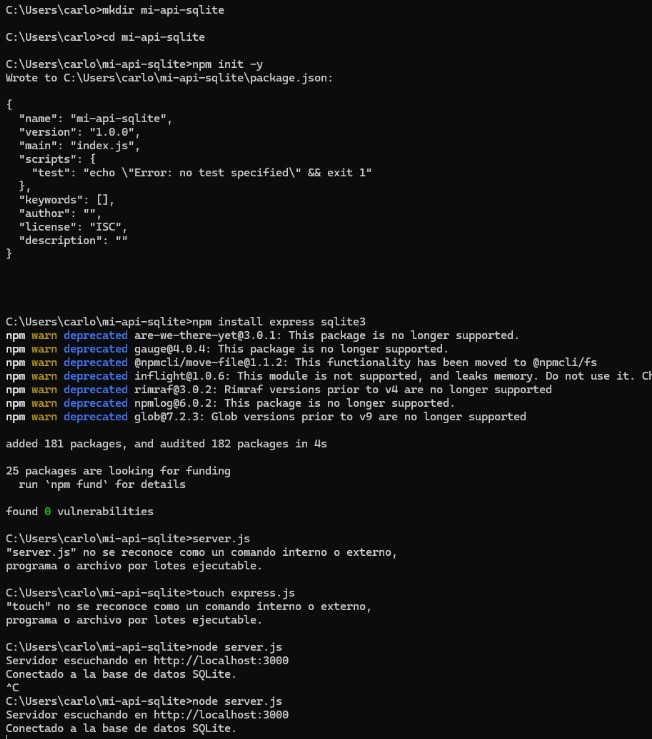


-Lo primero que hemos hecho ha sido clonar el repositorio para poder trabajar todos con el mismo repositorio

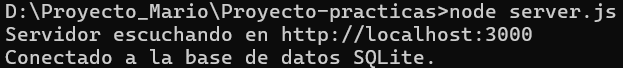


Lo que hicimos acontinuacion fue hacer que nuestros correo y usuario coincidan con nuestros nombre y apellidos como el ejemplo de la foto



Antes de añadir las APIS nuestros compañeros Carlos y Mario instalaron como se ve en esta imagen 

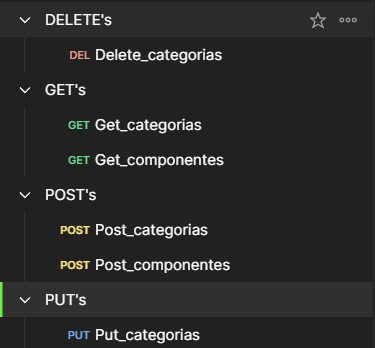
Para poder hacer que nuestro compañero Mario puede usar el server.js y programar junto a nuestro compañero Carlos los métodos de las APIS

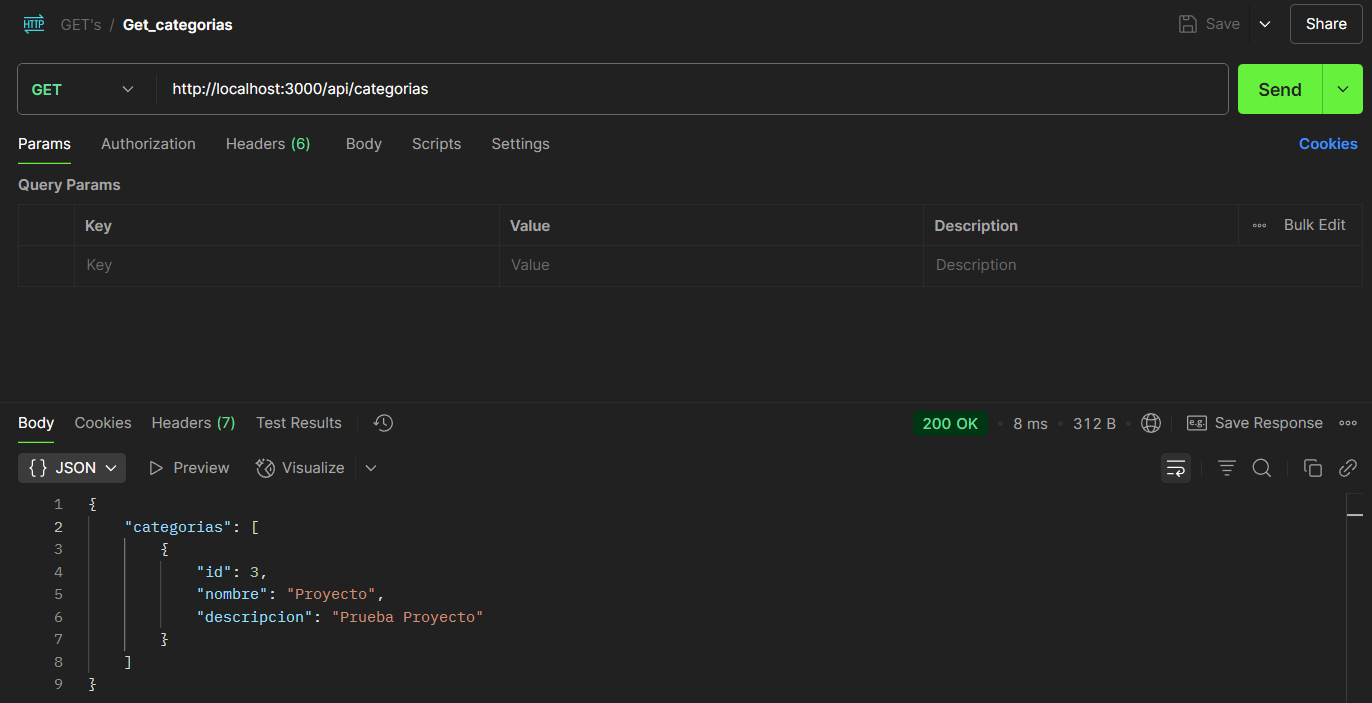


* **GET**

Nuestros compañeros Carlos y Mario se crearon los Get de categorías y componentes de la manera que se especifican en las imagines.

Nuestro Get está centrado en obtener información de las categorías o componentes de nuestra tienda





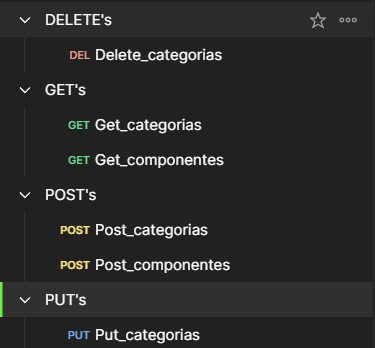
Aquí se muestra la información

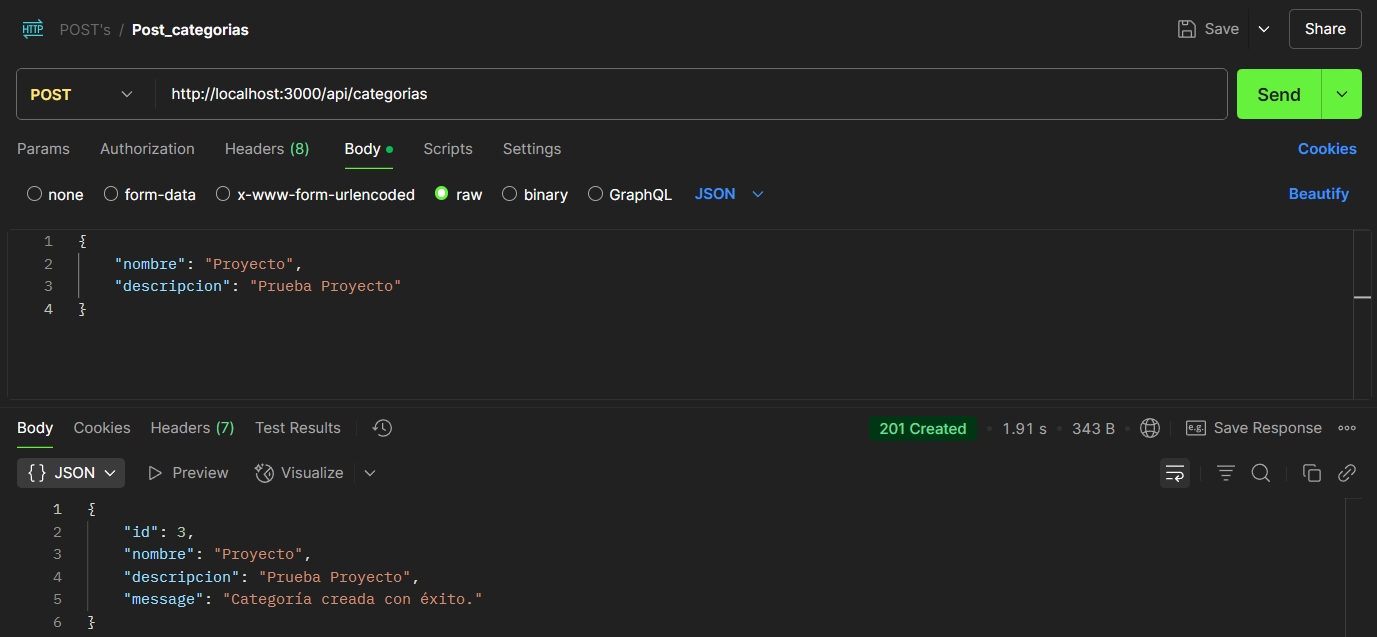
-Los mensajes mostrados con el nombre “message” son los avisos que aparece mediante el archivo “server.js”. Esto se aplica en todos los metodos

* **POST**

De la misma manera los compañeros Mario y Carlos nos creamos los métodos Post de categoría y componentes.

Nuestro Post se centra en crear nueva información sobre categorías o componentes.



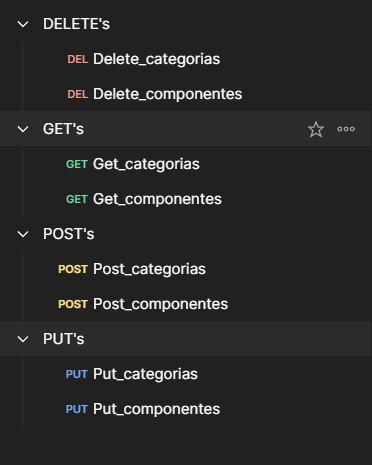


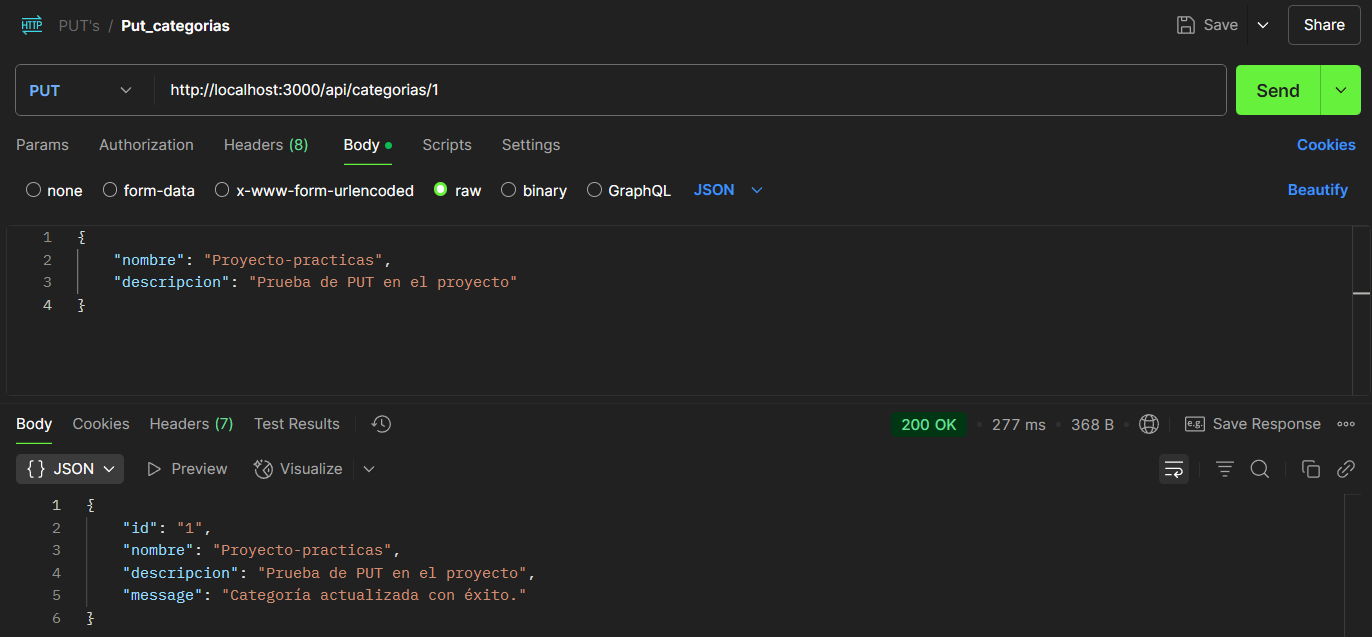
Aquí se muestra la información creada a través del método Post

* **PUT**

De la misma manera los compañeros Mario y Carlos nos creamos los métodos Put de categoría y componentes

Nuestro Put se centra en modificar la información ya creada sobre categorías o componentes.



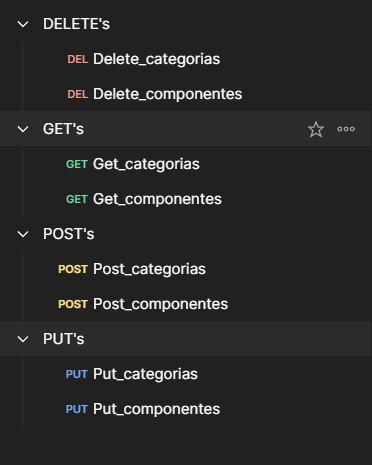


Aquí se muestra cómo se modifica la información ya creada

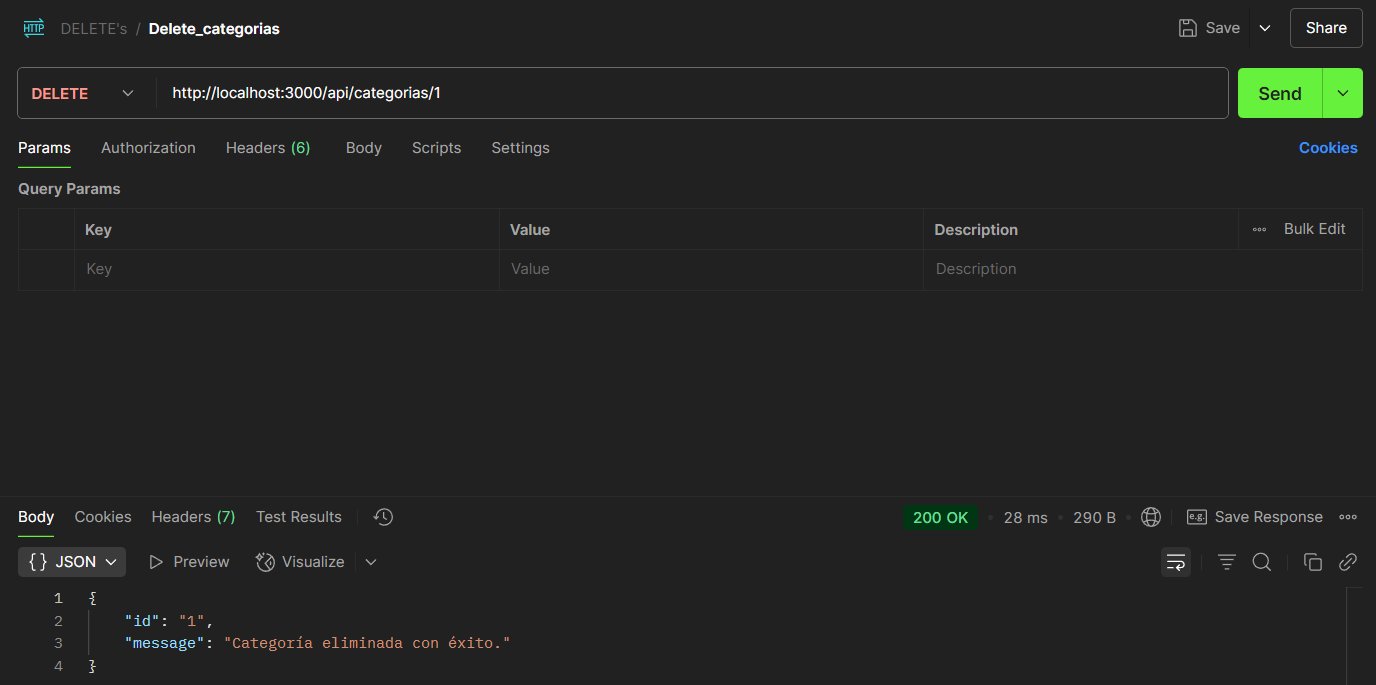
Para poder saber que vamos a modificar necesitamos insertar al final del enlace http “/id\_categoria o id\_componentes”.

* **DELETE**

De la misma manera los compañeros Mario y Carlos nos creamos los métodos Delete de categoría y componentes



Nuestro Delete se centra en borrar la información ya creada sobre categorías o componentes.

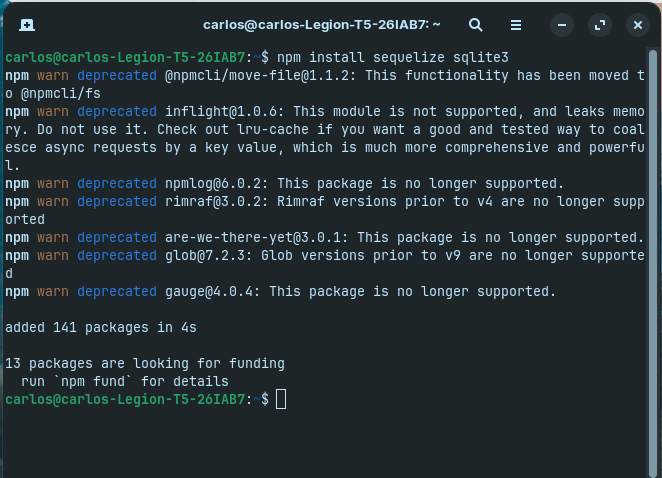


Aquí se muestra cómo se elimina la información

Para poder saber que vamos a borrar necesitamos insertar al final del enlace http “/id\_categoria o id\_componentes”.

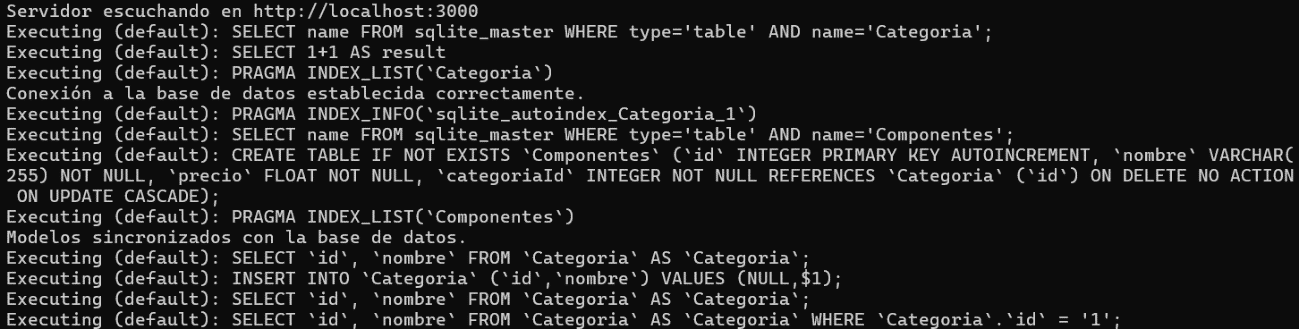
Despues de crear las APIS lo que hicimos fue hacer un mapeo Objeto-Relacional (ORM). Y actualizar las version de servidor pasando a la version 2.0

Con Sequelize vamos a realizar este mapeo. Crearemos archivos como categoria.js y componente.js que definen la estructura de nuestras tablas, estos modelos nos permiten interactuar con los datos de una manera más intuitiva.



Lo siguiente que haremos sera la sincronización de modelos utilizamos la función sequelize.sync() para que sequelize examine nuestros modelos y cree o actualice automáticamente las tablas.

El siguiente que hicimos es la abstracción de consultas en nuestros controladores controladoresCategoria.js y controladoresComponentes.js, en lugar de escribir consultas SQL directamente como SELECT \* FROM categorias.



Utilizamos los métodos que Sequelize proporciona a nuestros modelos como Categoria.findAll(), Componente.create(), Categoria.findByPk(), etc.

Por ultimo lo que haremos gestión de Relaciones definimos las relaciones entre nuestras tablas por ejemplo, que un Componente pertenece a una Categoria utilizando los métodos de asociación de Sequelize belongsTo, hasMany.

Esto facilita la consulta de datos relacionados entre las tablas.

Todos estos nuevos metodos se realizan automaticamentes cuando se inicia el server.js