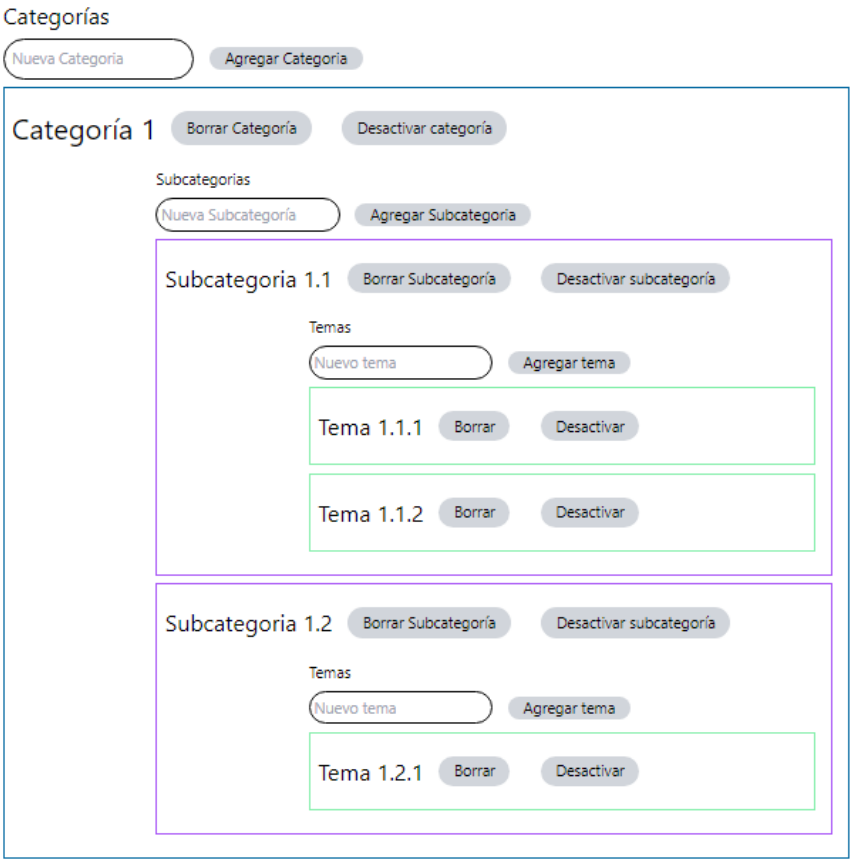
Prueba Técnica Desarrollador Junior para Sectorial S.A

Por: Diego Medina Velásquez



Frontend

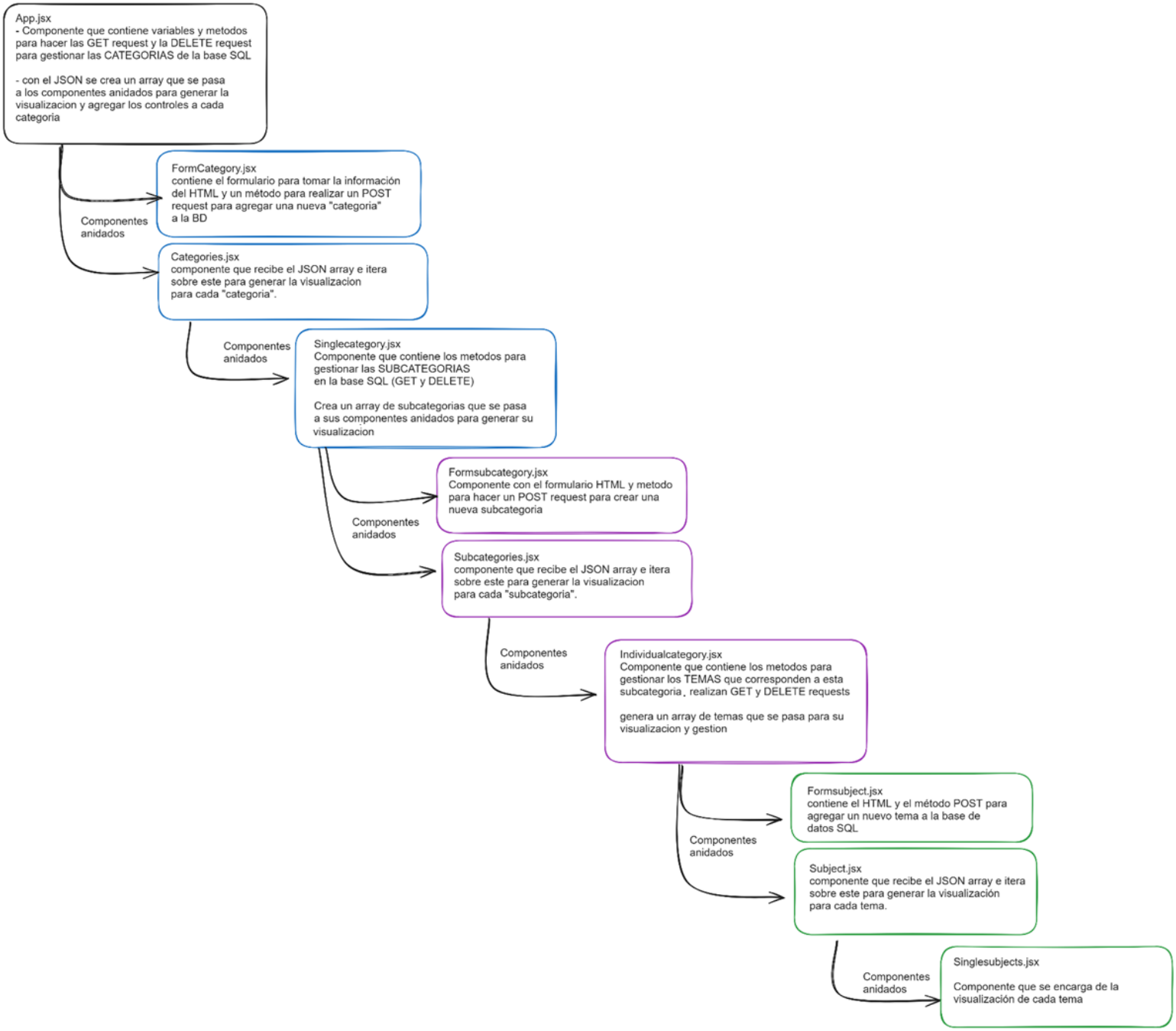
Para la realización del frontEnd se utilizaron las siguientes herramientas:

- React 18.2.0
- Tailwind 3.3.5 para la aplicación de estilos CSS
- Axios 1.6.2 para realizar HTTP request a la API que se encarga de gestionar la base de datos
- Nanoid para la numeración automática de componentes

El concepto de la app es crear componentes anidados que se encargan de gestionar las tablas que contienen la información de la base de datos, hacer las operaciones CRUD para obtener y publicar información y pasar objetos JSON a componentes de JSX para generar el HTML a visualizar.

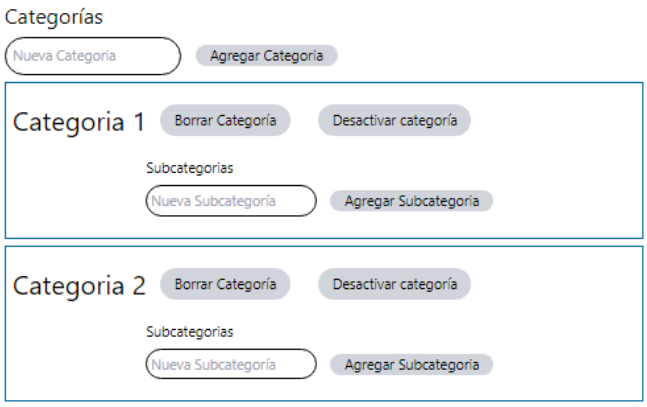
Estructura de App y componentes

La siguiente figura muestra un esquema de la estructura de los componentes

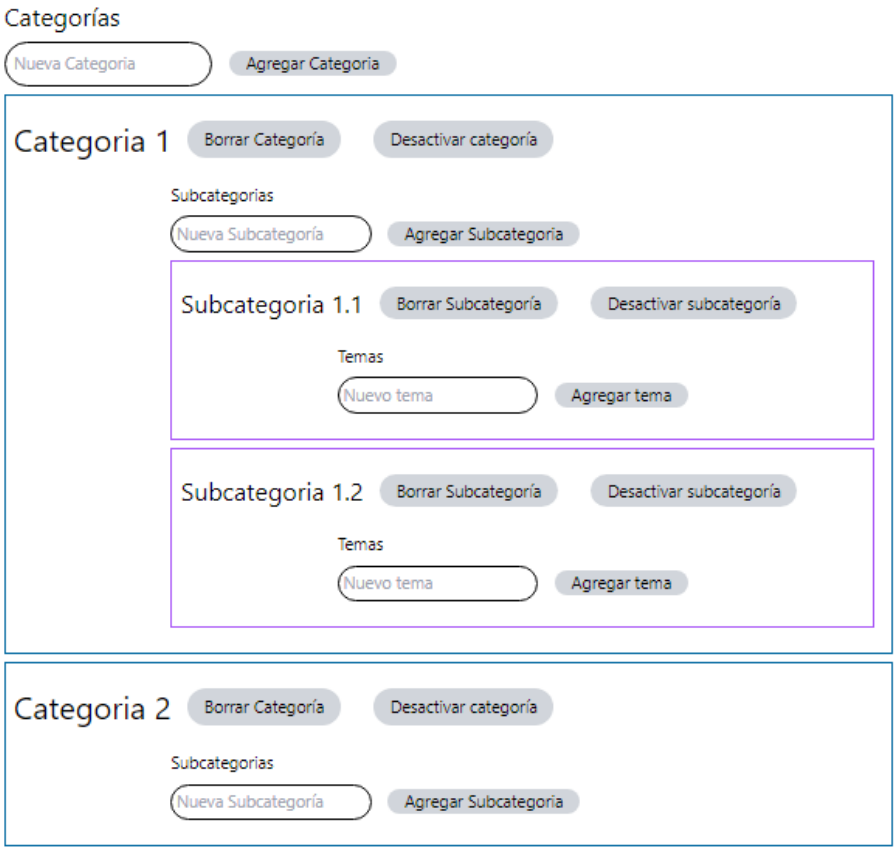


Funciones

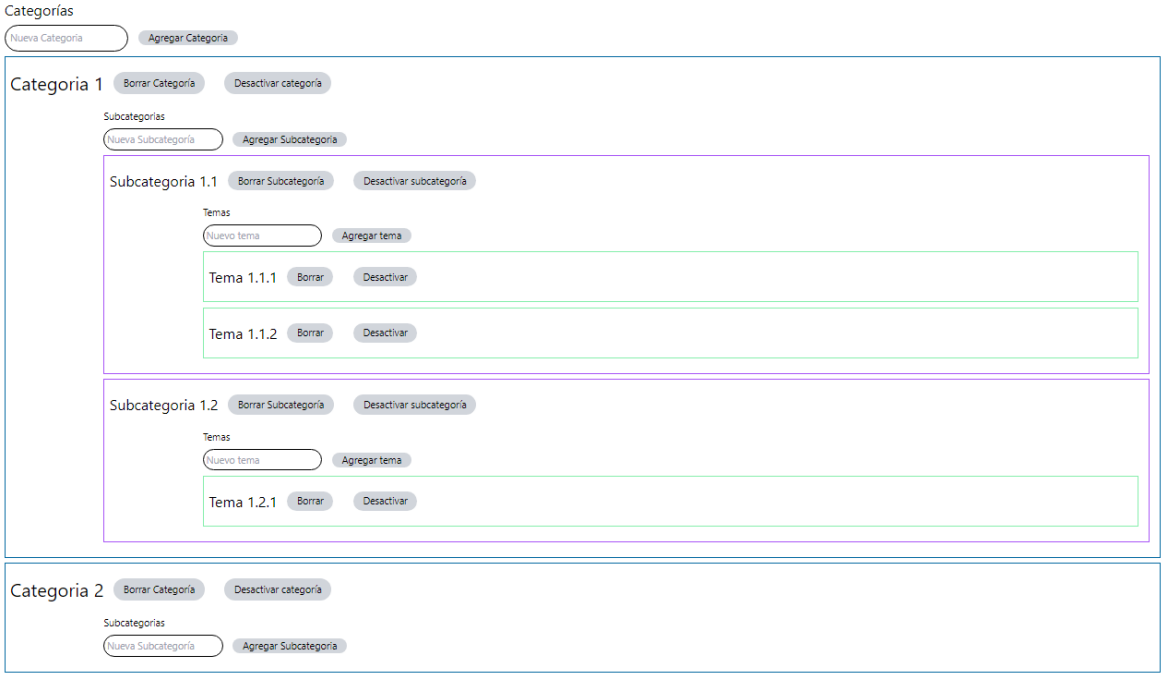
1. Categorías
   Se puede agregar categorías para generar una nueva estructura jerárquica. Cada categoría se gestionará independientemente y contiene sus propios controles para “borrar categoría” y de “desactivar categoría”



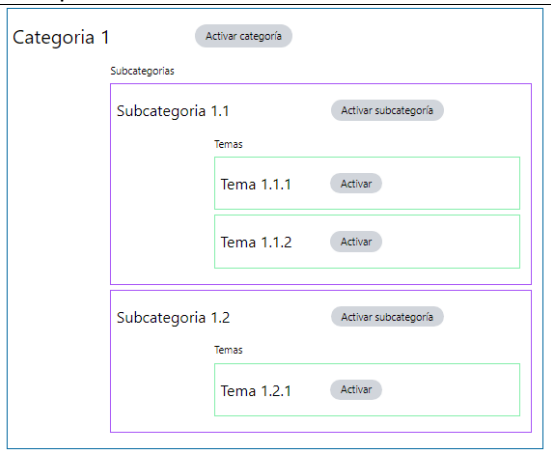
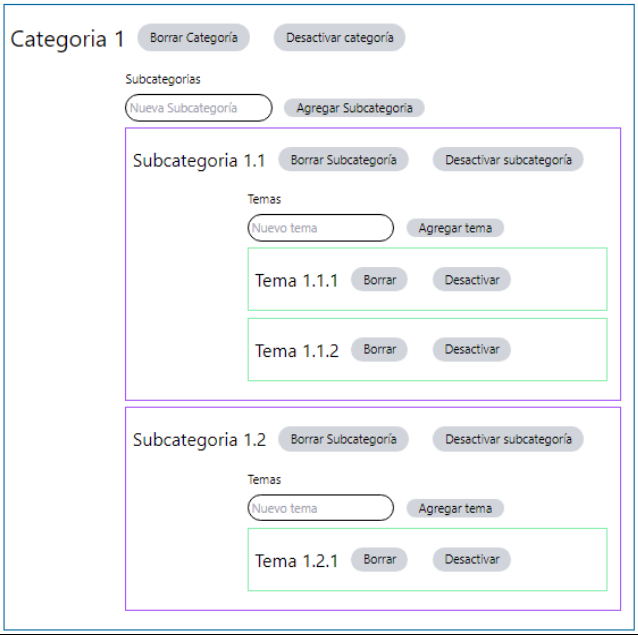
2. Subcategorías
   En cada categoría deseada puede agregar una subcategoría que quedará relacionada a la categoría correspondiente. La base de datos está construida usando el modelo Snowflake, de modo que cada subcategoría está relacionada a la categoría a la cual pertenece mediante “foreigns keys” que mantienen su asociación.
   También puede borrar y desactivar subcategorías usando los controles propios



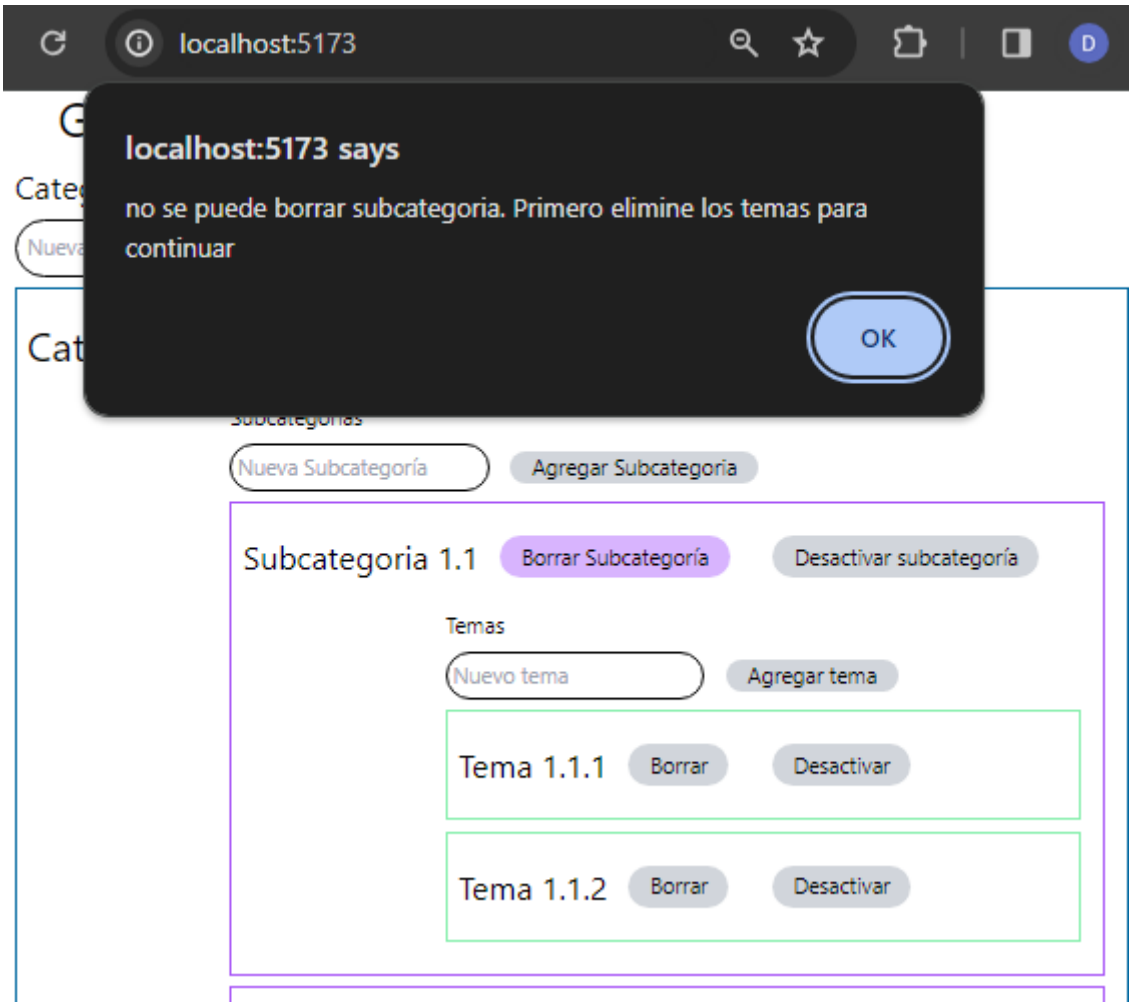
3. Temas
   De manera similar puede agregar temas a cada subcategoría, manteniendo estructuras jerárquicas independientes. También cuenta con controles para “inactivar tema” y “borrar tema”



4. Control de “desactivar”
   Cuando presiona, desaparece el formulario de ingreso de nuevos campos y también la posibilidad de borrar las subestructuras anidadas.

Campos desactivados	Campos activados
	

5. Control de “borrar”
   Permite eliminar cualquier nivel, siempre y cuando no tenga entradas anidadas en ella. Ej.: la subcategoría 1.1 no se puede eliminar ya que contiene el “tema 1.1.1” y el “tema 1.1.2”



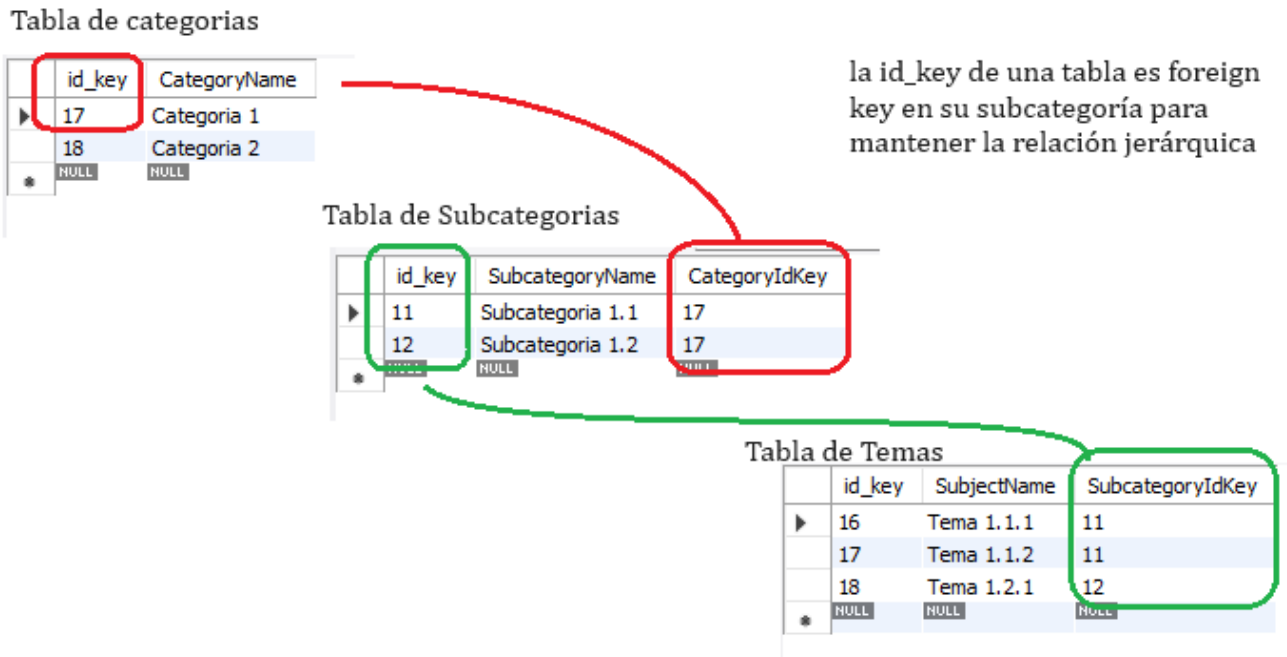
Backend

Para la realización de la API que controla la base de datos se utilizaron las siguientes herramientas:

1. Express 4.18.2 para la creación de un servidor usando Node.js
2. Mysql2 3.6.5 para gestionar una base de datos SQL
3. Sequelize 6.35.1 para generar SQL queries, que faciliten la creación de controladores que realicen las funciones CRUD para gestionar la base de datos
4. Dotenv para almacenar la información sensible de la base de datos, como nombre y contraseña
5. Cors para permitir el acceso del Frontend operando en otro puerto

Base de Datos SQL

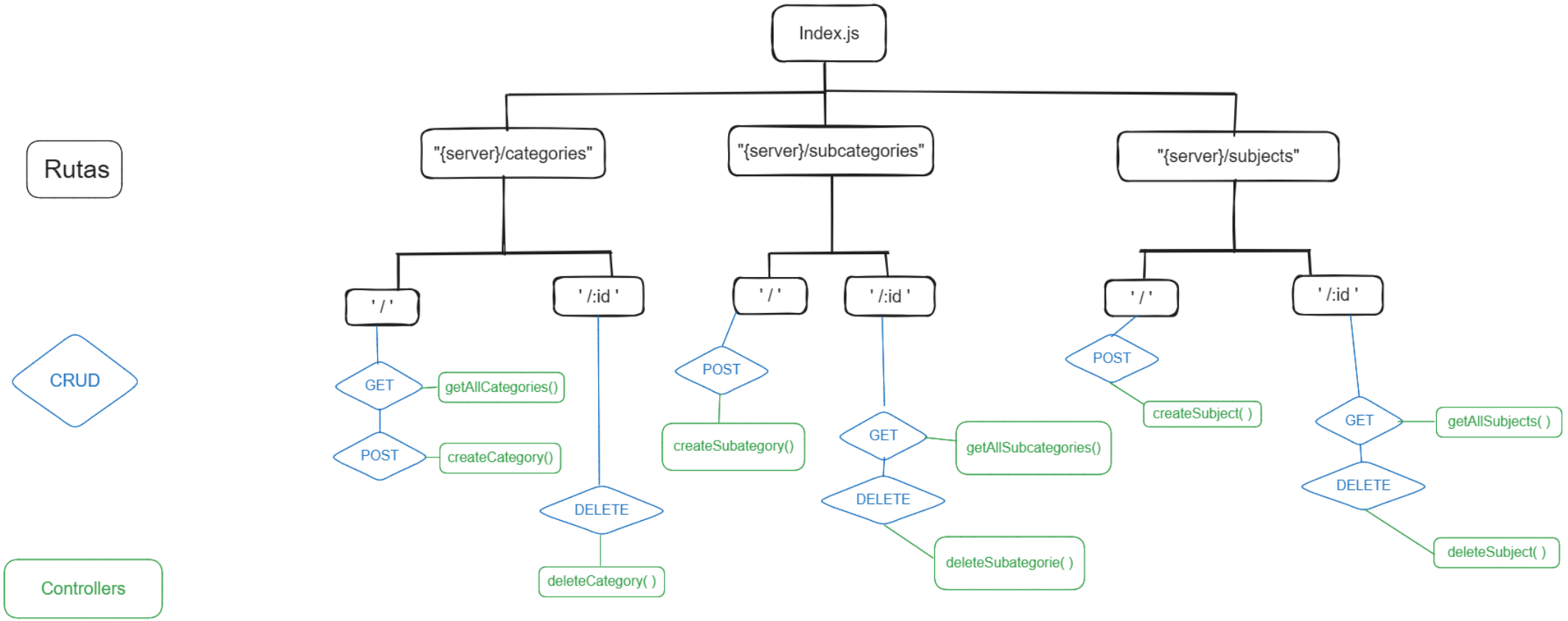
Como la información tiene una estructura jerárquica, se usa una base de datos configurada de manera snowflake, donde las relaciones de las tablas se definen con “foreign keys” como se muestra en la figura.



Cada entrada nueva de “Subcategoría” o de “Tema”, lleva el Id_key del nivel superior a sí misma, de modo que se pueda seguir la relación. Ej. el tema 16 pertenece a la subcategoría 11 que a su vez pertenece a la categoría 17

Servidor Express y rutas

El siguiente diagrama muestra el diseño de la API que se encargará de la gestión de la base de datos


Rutas
CRUD
Controllers
Index.js
"{server}/categories"
"{server}/subcategories"
"{server}/subjects"
' / '
' /:id '
' / '
' /:id '
' / '
' /:id '
GET
getAllCategories()
POST
createCategory()
DELETE
deleteCategory( )
POST
createSubatego ry()
GET
getAllSubcategories()
DELETE
deleteSubategorie( )
POST
createSubject( )
GET
getAllSubjects( )
DELETE
deleteSubject( )