Pasos para utilizar la api

Antes de comenzar es necesario tener instaladas estas dependencias

npm i express -Con esto se crea la api rest

npm i dotenv -Para las configuraciones de variables de entorno

npm i cors -No me quedo claro para que sirve pero instalalo

npm i sequelize mysql2 -Esto para la base de datos

npm i -g nodemon -Ejecuta el javascript y si hay algun cambio lo vuelve a ejecutar

Además de typescript de forma global, mira el archivo index.ts o index.js para mas información.

Asegúrate de encender la api con el comando nodemon dist/index.js, para poder ejecutar este comando la consola se debe encontrar en la carpeta “Back”

```
PS C:\Users\ruben\Documents\GitHub\Angular\Scoring\Back> nodemon dist/index.js
```

Al encenderla deberías ver este mensaje

```
Aplicacion corriendo en el puerto 3000
Executing (default): SELECT 1+1 AS result
Base de datos conectada

 New version of nodemon available!
 Current Version: 3.1.4
 Latest Version: 3.1.10
```

Aquí lo que mas te interesa es el puerto en el que corre, en este ejemplo para hacer una llamada a la api usaras como base la url http://localhost:3000/, compruébala en tu navegador

```
{"msg":"Api Workingaaaaa"}
```

De momento hay 8 endpoints, dos por cada tabla existente, si crees que puedas necesitar algún endpoint que no se encuentre en la api mándale mensaje al Javier

```
router.post("/createalternativa", createalternativa); //añade alternativas
router.get("/getalternativa/:id_proyecto", getalternativas); //lista las alternativas de un proyecto
router.post("/createcriterio", createcriterio); //añade criterios
router.get("/getcriterio/:id_proyecto", getcriterios);  //lista los criterios de un proyecto
router.post("/createproyecto", createproyecto); //añade proyectos
router.get("/getproyecto/:id_proyecto", getproyectos); //lista los proyectos de un usuario
router.post("/createsatisfaccion", createsatisfaccion); //añade satisfaccion
router.get("/getsatisfaccion/:id_proyecto", getsatisfaccion); //lista la satisfaccion de un proyecto
```

Cada endpoint se llama utilizando la url http://localhost:3000/api/nombredelendpoint}

Para cada tabla hay declarado un modelo en la carpeta models, en estos modelos se representan los campos de cada tabla

```
const Alternativamodel = db.define(
  "alternativa",
  {
    //en db.define se pone el nombre de la tabla al que pertenece este modelo
    id_alternativa: {
      type: DataTypes.INTEGER,
      primaryKey: true,
      autoIncrement: true,
    },
    id_proyecto: {
      type: DataTypes.INTEGER,
    },
    nombre: {
      type: DataTypes.STRING,
    },
  },
  {
    freezeTableName: true,
    timestamps: false,
  }
);
export default Alternativamodel;
```

Desde el front debes mandar los datos en formato json, este json se forma con una interfaz que hagas desde el front, y es importante que los parámetros de esa interfaz sean los mismos que están declarados en el modelo de la tabla correspondiente, los modelos están en la carpeta models del back y puedes consultarlos ahí.

Endpoint para crear alternativas

Para llamar este endpoint usas esta url http://localhost:3000/api/createalternativa

Y te pedirá un body en la petición, este body consta de un json con los siguientes parámetros

{

Id_alternariva:numeroenteroconid,

id_proyecto: numeroenteroconid,

nombre:”cadenadetextoconnombre”

}

Endpoint para recibir alternativas

Para llamar este endpoint usas esta url
http://localhost:3000/api/getalternativa/iddeñlaalternativa

En este endpoint al momento de usar la url el ultimo dato debe ser la id del proyecto al que se desean acceder las alternativas, al llamar al endpoint este te responderá con un json con los datos de la base de datos, algo como esto, como mencioné puedes hacer una interfaz con esta misma estructura en el front

A partir de aquí ire mas al grano ya que todos los demás endpoints tienen exactamente el mismo funcionamiento que las dos anteriores

Endpoint createcriterio:

url: http://localhost:3000/api/createcriterio

Estructura

{

id_criterio:numeroconlaiddelcriterio,

id_proyecto: numeroenteroconid,

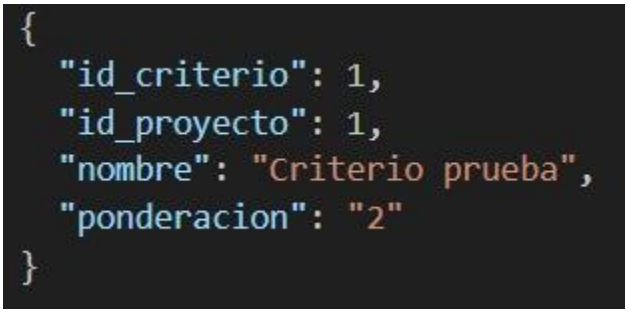
nombre:”cadenadetextoconnombre”,

ponderacion:”cadenaconlaponderacion”

}

Endpoint getcriterio:

http://localhost:3000/api/getalternativa/iddelcriterio



Endpoint createproyecto:

url: http://localhost:3000/api/createproyecto

Estructura

{

id_proyecto:numeroconlaiddelproyecto,

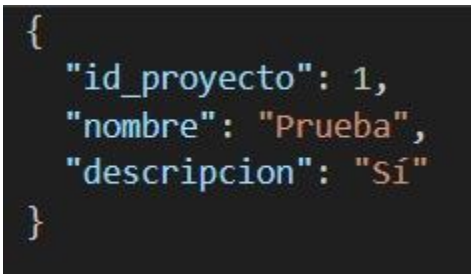
nombre: “cadena con el nombre del proyecto”,

nombre: ”cadena de texto con la descripción”

}

Endpoint getproyecto:

http://localhost:3000/api/getproyecto:iddelproyecto



Endpoint createsatisfaccion:

http://localhost:3000/api/createsatisfaccion

{

id_proyecto:numeroconlaiddelproyecto,

id_criterio:numeroconlaiddelcriterio

id_alternativa:numeroconlaiddelaalternativa

satisfacción:numeroconlasatisfaccion

}

Endpoint getproyecto:

http://localhost:3000/api/getsatisfaccion/id_delproyecto