CONFIGURACION Y INSTALACION DE MYSQL USANDO SEQUELIZE

En su base de datos de software de visualización, agregar una nueva base de datos ,pero Asegúrese de que MySQL se está ejecutando, o no podrá conectarse a localhost.

Como fase inicial debemos instalar todos los paquetes necesarios para nuestra aplicación(teniendo en cuenta debe haber iniciado el proyecto con las instalaciones nombradas en la guía de express).

ejecutamos el siguiente comando:

npm install --save sequelize
npm install --save mysql2
npm install body-parser

Descripción de los paquetes:

MySQL: es un sistema de gestión de bases de datos relacionales de código abierto.
Sequelize: es un ORM para Nodejs que te permitirá agilizar bastante tus desarrollos que incluyen bases de datos relacionales como MySQL.
Como muchos otros ORM, Sequelize le permite mapear su base de datos relacional a objetos. Estos objetos tienen métodos y propiedades que le permiten evitar escribir sentencias SQL.
body-parser:Usualmente el cuerpo de una petición (payload), contiene información desde una petición tipo POST cuando un cliente desea crear una nueva entidad/registro o actualizar uno existente mediante PUT. Los desarrolladores quienes implementan servidores, requieren frecuentemente acceder a la información del cuerpo de dicha petición. El módulo npm body-parser permite realizar esta tarea. No es necesario programarla. Solo se requiere instalar body-parser y habilitar json() así como url-encode como middlewares para convertir datos a JSON

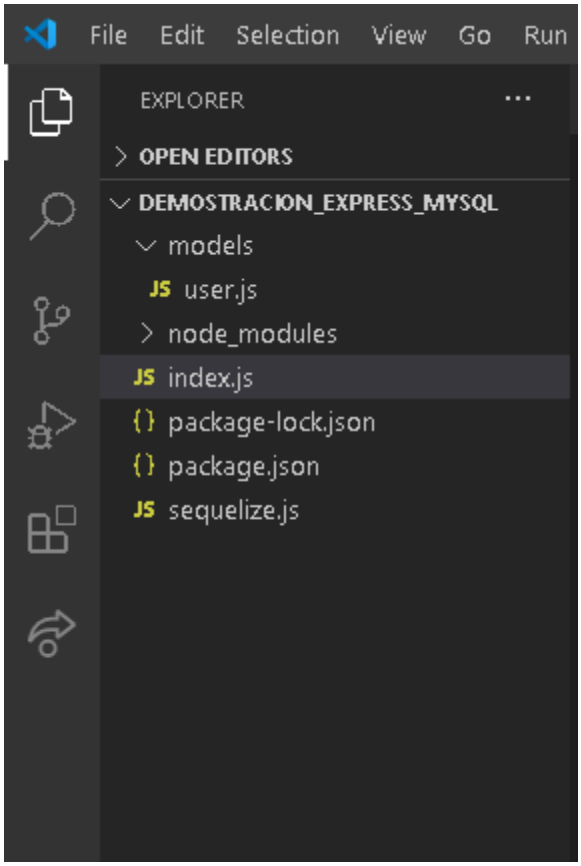
Configuración

para empezar con nuestra configuración de mysql con sequelize primero que todo debemos crear un proyecto ,daremos un repaso de cómo inicializar nuestro proyecto,abrimos vs code presionamos ctrl + n para abrir la consola integrada y ejecutamos los siguientes comandos:
mkdir my_app
cd my_app
arrastramos la carpeta que creamos a vs code y ejecutamos:
npm init -y
npm install express --save
y los comandos explicados anteriormente.
creación de base de datos

necesitaremos crear una base de datos,no explicaremos esta parte ya que en las sesiones anteriores se explicó .

archivos y estructura del proyecto

Estos son todos los archivos que necesitaremos. sequelize.js será el lugar para iniciar nuestro ORM y definir relaciones. Definiremos nuestros modelos en sus respectivos archivos y index.js será nuestra aplicación Express.
nuestro proyecto tendrá la siguiente estructura:



creamos un archivo sequelize.js en la raíz del proyecto y lo actualizamos con la siguiente línea de código

```
const Sequelize = require('sequelize')
const UserModel = require('./models/user')

const sequelize = new Sequelize('db_express', 'root', '', {
    host: 'localhost',
    dialect: 'mysql',
    pool: {
        max: 10,
```

```
        min: 0,
        acquire: 30000,
        idle: 10000
    }
})

const User = UserModel(sequelize, Sequelize)
sequelize.sync({ force: true })
    .then(() => {
        console.log(`Database & tables created!`)
    })

module.exports = {
    User,

}
```

Repasemos lo que está sucediendo aquí.

Después de requerir nuestros modelos y dependencias, instamos Sequelize y definimos todos los detalles necesarios para conectarse a la base de datos. Su base de datos y credenciales específicas serán diferentes.

A continuación, creamos una instancia de nuestros modelos pasando una instancia de sequelize y una biblioteca a los archivos de modelo necesarios.

sequelize.sync()creará todas las tablas en la base de datos especificada. Si pasa {force: true}como parámetro a sync método, eliminará tablas en cada inicio y creará nuevas. No hace falta decir que esta es una opción viable solo para el desarrollo.

Por último, queremos exportar nuestros modelos para poder usarlos en cualquier otro lugar de nuestra aplicación.

Pasando a nuestros modelos:
crearemos un directorio llamado models en la raíz de nuestro proyecto ,en este estarán todos nuestros modelos y en este caso como ejemplo y demostración se creará el modelo user:
añadimos al directorio model el archivo user.js y se actualizará con el siguiente fragmento de código:

```
module.exports = (sequelize, type) => {
    return sequelize.define('user', {
```

```
        id: {
            type: type.INTEGER,
            primaryKey: true,
            autoIncrement: true
        },
        name: type.STRING,
        email: type.STRING
    })
}
```

Aquí es donde definimos campos, tipos y otra información relevante sobre una tabla y su definición. Estamos exportando una función que devolverá una instancia de modelo. Necesitamos pasar dos cosas a nuestros archivos de modelo.

Primero, la instancia de Sequelize y Sequelize sí mismo. Necesitamos el primero para crear (definir) nuestro modelo y el segundo para la definición de tipo. Sequelize tiene un montón de tipos disponibles como propiedades estáticas. en este caso se está usando STRING, pero casi cualquier otro tipo también está disponible.

En la parte de la API.

Queremos una aplicación simple que nos retorne todos los usuarios almacenados en nuestra bd y esta será nuestra ruta :

GET /API/users// obtener todos los usuarios

express.js,Agreguemos el texto estándar básico .en este ejemplo toda nuestra API estará en el archivo index,js, pero en un proyecto más grande, querrá organizar su código un poco mejor.

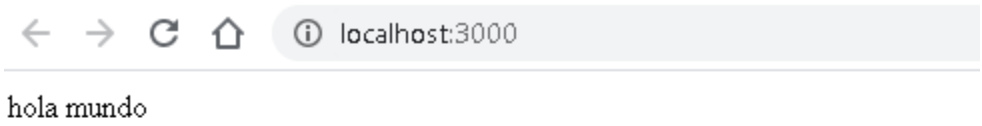
index.js

```
const { User } = require('./sequelize')
const express = require('express')
const bodyParser = require('body-parser')

const app = express()
app.use(bodyParser.json())

// API ENDPOINTS
app.get('/', function(req, res) {
    res.send('hola mundo');
```

```
});
const port = 3000
app.listen(port, () => {
    console.log(`Running on http://localhost:${port}`)
})
```

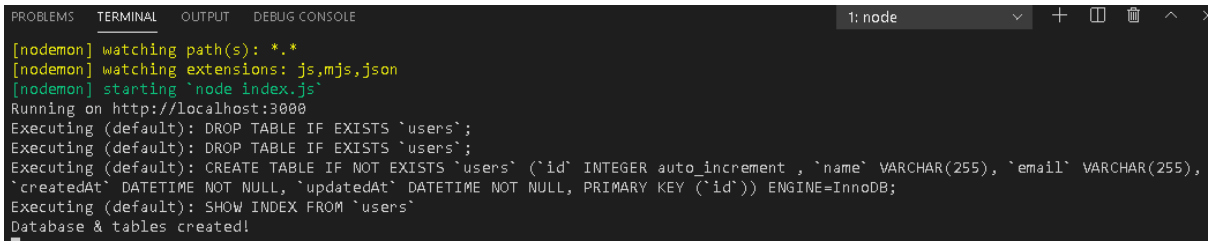
en este paso levantaremos nuestro servidor con el comando node index.js pero como recomendación y comodidad ,instalaremos el paquete de nodemon con el siguiente comando:
npm install -g nodemon

Nodemon es una utilidad que monitorea los cambios en el código fuente que se está desarrollando y automáticamente reinicia el servidor. Es una herramienta muy útil para el desarrollo de aplicaciones en nodejs.
Ahora en lugar de node index.js ejecutaremos nodemon index,js
damos control + click encima del enlace http://localhost:3000 que sale en nuestra consola y no abrirá el navegador ,como lo muestra la siguiente imagen:



hola mundo

Y en nuestra consola veremos la información de la creación de nuestras tablas en la bases de datos como a continuación



como se dijo arriba queremos obtener los usuarios de nuestra tabla user, primeros insertamos algunos usuarios por medio de nuestra interfaz gráfica que ya fueron explicadas en sesiones anteriores ,luego actualizamos nuestro archivo index.js insertando una nueva ruta que es:

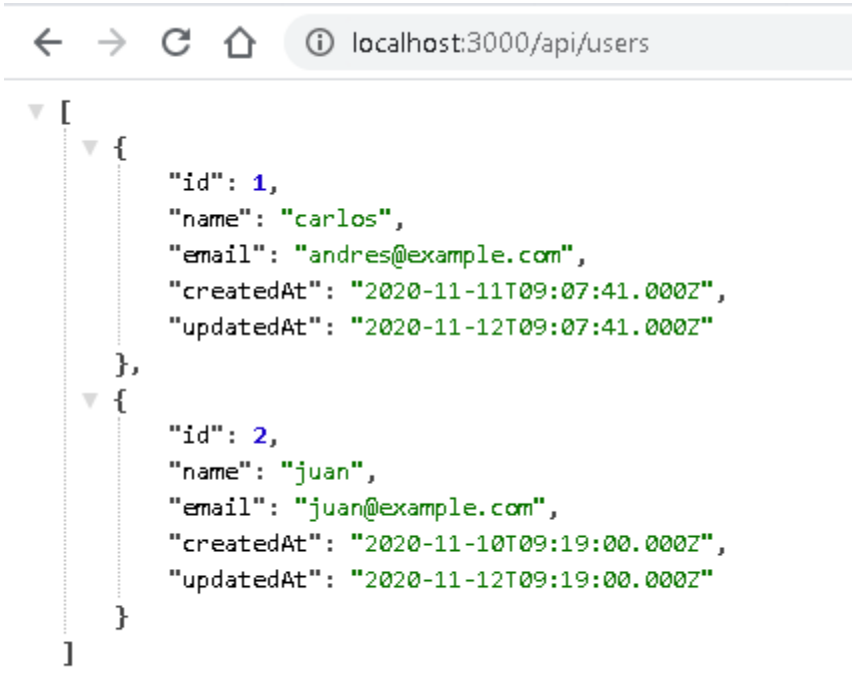
```
app.get('/', function(req, res) {
    User.findAll().then(users => res.json(users))
});
```

Puede leer la tabla completa de la base de datos con el findAll método, pero :Sequelize proporciona varios métodos para ayudarlo a consultar datos en su base de datos.

nuestro archivo index,js tendría el siguiente aspecto:

```
const { User } = require('./sequelize')
const express = require('express')
const bodyParser = require('body-parser')

const app = express()
app.use(bodyParser.json())

// API ENDPOINTS
app.get('/api/users', (req, res) => {
    User.findAll().then(users => res.json(users))
})

app.get('/', function(req, res) {
    User.findAll().then(users => res.json(users))
});
const port = 3000
app.listen(port, () => {
    console.log(`Running on http://localhost:${port}`)
})
```

ponemos la ruta en nuestro navegador http://localhost:3000/api/users y obtenemos la lista de estudiantes en formato json como se referenció


localhost:3000/api/users
[
{
"id": 1,
"name": "carlos",
"email": "andres@example.com",
"createdAt": "2020-11-11T09:07:41.000Z",
"updatedAt": "2020-11-12T09:07:41.000Z"
},
{
"id": 2,
"name": "juan",
"email": "juan@example.com",
"createdAt": "2020-11-10T09:19:00.000Z",
"updatedAt": "2020-11-12T09:19:00.000Z"
}
]