Instalación y configuración

use MongoDBNodeJS

db.createUser({user: "redusers", pwd: "redusers", roles: [ "readWrite" ]})

npm install express –save

npm install mysql

npm install mongoose

var mongoose = require('mongoose');

var mysql = require('mysql');

// Prueba de conexion a MongoDB

mongoose.connect('mongodb://redusers:redusers@localhost:27017/MongoDBNodeJS', {

        useNewUrlParser: true,

        useUnifiedTopology: true

    },

    (err, res) => {

        if (err)

            console.log('La conexión a la base de datos MongoDB ha fallado.', err);

        else {

            console.log('La conexión a la base de datos MongoDB se estableció satisfactoriamente.');

        }

        mongoose.connection.close();

    });

// Prueba de conexion a MySQL

var connection = mysql.createConnection({

    connectionLimit: 50,

    host: 'localhost',

    user: 'redusers',

    password: 'redusers',

    database: `redusers\_schema`

});

connection.connect(function(err) {

    if (err) {

        console.log('La conexión a la base de datos MySQL ha fallado.', err);

        return;

    }

    console.log('La conexión a la base de datos MySQL se estableció satisfactoriamente.');

    connection.end();

});

Base de datos

var mongoose = require('mongoose');

function iniciar() {

    return mongoose.connect('mongodb://redusers:redusers@localhost:27017/MongoDBNodeJS', {

            useNewUrlParser: true,

            useUnifiedTopology: true

        },

        (err, res) => {

            if (err)

                console.error('Ocurrió un error al establecer la conexión a la base de datos MongoDB.', err);

            else {

                console.log('La conexión a la base de datos MongoDB se estableció satisfactoriamente.');

            }

        });

}

module.exports = { iniciar };

var mongoose = require('mongoose');

var schema = mongoose.Schema;

var telefonoSchema = new schema({

    telefono: String

});

var clienteSchema = new schema({

    nombre: String,

    apellido: String,

    edad: String,

    telefonos: [telefonoSchema]

});

module.exports = {

    cliModel: mongoose.model('Clientes', clienteSchema),

    telModel: mongoose.model('Telefonos', telefonoSchema)

}

const clientesCollection = require('../models/clienteSchema');

async function obtenerClientesAsync() {

    var query = clientesCollection.cliModel.find().lean();

    const result = await query.exec();

    const docsJSON = result.map((doc) => {

        doc.telefonos = doc.telefonos.map((tel) => {

            return tel.telefono;

        })

        return doc;

    });

    return docsJSON;

}

async function guardarClienteAsync(cliente) {

    let nuevoCliente = new clientesCollection.cliModel();

    nuevoCliente.nombre = cliente.nombre;

    nuevoCliente.apellido = cliente.apellido;

    nuevoCliente.edad = cliente.edad;

    // Crea los telefonos del cliente

    cliente.telefonos.map((telCli) => {

        let tel = new clientesCollection.telModel();

        tel.telefono = telCli.telefono;

        nuevoCliente.telefonos.push(tel);

    })

    return await nuevoCliente.save(null);

}

async function eliminarClienteAsync(idCliente) {

    return await clientesCollection.cliModel.find({ \_id: idCliente }).deleteOne();

}

module.exports = { obtenerClientesAsync, guardarClienteAsync, eliminarClienteAsync };

var mysql = require('mysql');

const config = {

    connectionLimit: 50,

    host: 'localhost',

    user: 'redusers',

    password: 'redusers',

    database: `redusers\_schema`

};

const poolMySQL = mysql.createPool(config);

function iniciar() {

    poolMySQL.on('error', (err) => {

        console.error('Ocurrió un error con la conexión a la base de datos MySQL', err);

    })

}

function getPool() {

    return poolMySQL;

}

module.exports = { iniciar, getPool }

const poolMySQL = require('./adPoolMySQL');

async function obtenerClientesAsync() {

    const consulta = `SELECT cli.nombre, cli.apellido, cli.edad,

                             GROUP\_CONCAT(clitel.telefono SEPARATOR '|') as telefonos

                      FROM Clientes cli

                      INNER JOIN ClientesTelefonos clitel ON cli.id = clitel.idCliente

                      GROUP BY cli.nombre, cli.apellido, cli.edad;`;

    return await new Promise((resolve, reject) => {

        poolMySQL.getPool().query(consulta, function(error, results, fields) {

            if (error) {

                return reject(error);

            }

            let clientesRet = results.map((cli) => {

                return {

                    nombre: cli.nombre,

                    apellido: cli.apellido,

                    edad: cli.edad,

                    telefonos: cli.telefonos.split('|').map((tel) => {

                        return { telefono: tel };

                    })

                };

            });

            console.log(clientesRet);

            return resolve(clientesRet);

        });

    });

}

async function guardarClienteAsync(cliente) {

    return await new Promise((resolve, reject) => {

        poolMySQL.getPool().getConnection(function(err, connection) {

            connection.beginTransaction(function(err) {

                if (err) { throw err; }

                connection.query('INSERT INTO clientes(nombre, apellido, edad) VALUES (?,?,?);', [cliente.nombre, cliente.apellido, cliente.edad],

                    function(error, resultsNuevoCli, fields) {

                        if (error) {

                            return connection.rollback(function() {

                                reject(error);

                            });

                        }

                        // Guarda los telefonos del cliente

                        const idCliente = resultsNuevoCli.insertId;

                        telefonos = cliente.telefonos;

                        for (let i = 0; i < telefonos.length; i++) {

                            const telefono = telefonos[i];

                            telefono.idCliente = idCliente;

                            connection.query('INSERT INTO clientestelefonos SET ?', telefono,

                                function(error, results, fields) {

                                    if (error) {

                                        return connection.rollback(function() {

                                            reject(error);

                                        });

                                    }

                                });

                        }

                        connection.commit(function(err) {

                            if (err) {

                                return connection.rollback(function() {

                                    reject(err);

                                });

                            }

                            resolve(resultsNuevoCli);

                        });

                    });

            }); // Fin de begintransaction

        });

    }); // Fin de Promise

}

async function eliminarClienteAsync(idCliente) {

    return await new Promise((resolve, reject) => {

        poolMySQL.getPool().getConnection(function(err, connection) {

            connection.beginTransaction(function(err) {

                if (err) { throw err; }

                // Elimina sus telefonos

                connection.query('DELETE FROM clientestelefonos WHERE idCliente = ?;', idCliente,

                    function(error, results, fields) {

                        if (error) {

                            return connection.rollback(function() {

                                reject(error);

                            });

                        }

                        // Elimina el cliente

                        connection.query('DELETE FROM clientes WHERE id = ?', idCliente,

                            function(error, results, fields) {

                                if (error) {

                                    return connection.rollback(function() {

                                        reject(error);

                                    });

                                }

                            });

                        connection.commit(function(err) {

                            if (err) {

                                return connection.rollback(function() {

                                    reject(err);

                                });

                            }

                            resolve(`El ${idCliente} cliente fue eliminado`);

                        });

                    });

            }); // Fin de begintransaction

        });

    }); // Fin de Promise

}

module.exports = { obtenerClientesAsync, guardarClienteAsync, eliminarClienteAsync };

Exponer las operaciones mediante Express

npm install --save-dev nodemon

npm install cors --save

npm install helmet --save

npm install morgan –save

npm install --save rotating-file-stream

var express = require('express');

var adMongoDB = require('../database/adMongoDbClientes');

var adMySQL = require('../database/adMySQLClientes');

const msgError = "Se debe definir el motor de base de datos";

const MONGODB = 1,

    MYSQL = 2;

var router = express.Router();

router.get('/', async function(req, res, next) {

    let clientes = null;

    switch (parseInt(req.headers.motor)) {

        case MONGODB:

            clientes = await adMongoDB.obtenerClientesAsync();

            break;

        case MYSQL:

            clientes = await adMySQL.obtenerClientesAsync();

            break;

        default:

            res.status(400).send(msgError);

            return;

    }

    res.setHeader('Content-Type', 'application/json');

    res.send(JSON.stringify(clientes, null, 4));

});

router.post('/add', async function(req, res, next) {

    let cliente = req.body,

        cliAdd = null;

    switch (parseInt(req.headers.motor)) {

        case MONGODB:

            cliAdd = await adMongoDB.guardarClienteAsync(cliente);

            break;

        case MYSQL:

            cliAdd = await adMySQL.guardarClienteAsync(cliente);

            break;

        default:

            res.status(400).send(msgError);

            return;

    }

    res.setHeader('Content-Type', 'application/json');

    res.send(JSON.stringify(cliAdd, null, 4));

});

router.delete('/del/:idCliente', async function(req, res, next) {

    let idCliente = req.params.idCliente,

        clientDel;

    switch (parseInt(req.headers.motor)) {

        case MONGODB:

            clientDel = await adMongoDB.eliminarClienteAsync(idCliente);

            break;

        case MYSQL:

            clientDel = await adMySQL.eliminarClienteAsync(idCliente);

            break;

        default:

            res.status(400).send(msgError);

            return;

    }

    res.setHeader('Content-Type', 'application/json');

    res.send(JSON.stringify(clientDel, null, 4));

});

module.exports = router;

var express = require('express');

var helmet = require('helmet');

var cors = require('cors');

var morgan = require('morgan');

const rfs = require("rotating-file-stream");

var path = require('path')

const adPoolMongoDb = require('./database/adPoolMongoDB');

const adPoolMySQL = require('./database/adPoolMySQL');

const port = 3000;

var app = express();

app.use(cors({

    origin: `http://localhost:${port}`

}));

app.use(helmet());

//Crea el logger

const streamLog = rfs.createStream("log.txt", {

    size: "10M",

    interval: "1d",

    compress: "gzip",

    path: path.join(\_\_dirname, 'log')

});

app.use(morgan('combined', { stream: streamLog }))

//Parser de datos recibidos

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

var apiClientesRouter = require('./controllers/apiClientes');

app.use('/api/clientes', apiClientesRouter);

app.listen(port, function() {

    console.log(`El sitio de APIs inició correctamente en el puerto ${port}.`);

    adPoolMongoDb.iniciar();

    adPoolMySQL.iniciar();

});

Prueba de la API con PostMan

{

"nombre": "Roberto",

"apellido": "Garcia",

"edad": 57,

"telefonos": [{

"telefono": "+5499997777"

},{

"telefono": "+5444447777"

}]

}