=======================================================================

estrutura e primeiros passos

=======================================================================

app

|\_ config

| |\_ env.js

| |\_ db.config.js

|

|\_ servicos

| |\_servico.js

|

|\_models

| |\_cliente.model.js

|

|\_routes

| route.js

|

|\_server.js

**criar o package.json**

npm init -y

**instalar as dependências no projeto**

npm install cors sequelize mysql2 express

=======================================================================

db.config.js

=======================================================================

const env = require('./env.js');

const Sequelize = require('sequelize');

const sequelize = new Sequelize(env.database, env.username, env.password, {

host: env.host,

dialect: env.dialect,

operatorsAliases: false,

});

const db = {};

db.Sequelize = Sequelize;

db.sequelize = sequelize;

db.Cliente = require('../models/cliente.model.js')(sequelize, Sequelize);

module.exports = db;

=======================================================================

env.js

=======================================================================

const env = {

database: 'db\_backend',

username: 'nome do seu usuário',

password: 'senha do seu banco',

host: 'localhost',

dialect: 'mysql',

};

module.exports = env;

=======================================================================

servico.js

=======================================================================

const db = require('../config/db.config.js');

const Cliente = db.Cliente;

exports.createCliente = (req, res) => {

let cliente = {};

try{

cliente.nome = req.body.nome;

cliente.idade = req.body.idade;

cliente.email = req.body.email;

Cliente.create(cliente, {attributes: ['id', 'nome', 'idade', 'email']})

.then(result => {

res.status(200).json(result);

});

}catch(error){

res.status(500).json({

message: "Fail!",

error: error.message

});

}

}

exports.getCliente = (req, res) => {

const {id} = req.params

Cliente.findByPk(id, {attributes: ['id', 'nome', 'idade','email']})

.then(cliente => {

res.status(200).json(cliente);

})

.catch(error => {

console.log(error);

res.status(404).json({message: "Client not found!",error: error});

})

}

exports.clientes = (req, res) => {

try{

Cliente.findAll({attributes: ['id', 'nome', 'idade', 'email']})

.then(clientes => {

res.status(200).json(clientes);

})

}catch(error) {

console.log(error);

res.status(404).json({message: "Not Found!",error: error});

}

}

exports.deleteCliente = async (req, res) => {

try{

const {id} = req.params;

let cliente = await Cliente.findByPk(id);

if(!cliente){

res.status(404).json({

message: "Does Not exist a Customer with id = " + id,

error: "404",

});

} else {

await cliente.destroy();

res.status(200).json('cliente deleted sucesfull.');

}

} catch(error) {

res.status(500).json({

message: "Error -> Can NOT delete a customer with id = " + id,

error: error.message

});

}

}

exports.updateCliente = async (req, res) => {

const {id} = req.body;

try{

let cliente = await Cliente.findByPk(id);

if(!cliente){

res.status(404).json({

message: "Not Found for updating a customer with id = " + id,

error: "404"

});

} else {

let updatedObject = {

nome: req.body.nome,

idade: req.body.idade,

email: req.body.email,

}

let result = await Cliente.update(updatedObject,

{

returning: true,

where: {id: id},

attributes: ['id', 'nome', 'idade', 'email']

}

);

if(!result) {

res.status(500).json({

message: "Error -> Can not update a customer with id = " + id,

error: "Can NOT Updated",

});

}

res.status(200).json(result);

}

} catch(error){

res.status(500).json({

message: "Error -> Can not update a customer with id = " + id,

error: error.message

});

}

}

=======================================================================

cliente.model.js

=======================================================================

module.exports = (sequelize, Sequelize) => {

const Cliente = sequelize.define('clientes', {

id: {

type: Sequelize.INTEGER,

autoIncrement: true,

primaryKey: true

},

nome: {

type: Sequelize.STRING

},

idade: {

type: Sequelize.INTEGER

},

email: {

type: Sequelize.STRING

},

});

return Cliente;

}

=======================================================================

route.js

=======================================================================

let express = require('express');

let router = express.Router();

const clientes = require('../servicos/servico.js');

router.post('/cliente', clientes.createCliente);

router.get('/cliente/:id', clientes.getCliente);

router.get('/clientes', clientes.clientes);

router.put('/cliente', clientes.updateCliente);

router.delete('/cliente/:id', clientes.deleteCliente);

module.exports = router;

=======================================================================

server.js

=======================================================================

const express = require('express');

const app = express();

const db = require('./config/db.config.js');

const router = require('./routes/route.js');

const bodyParser = require('body-parser');

const Cliente = db.Cliente;

const cors = require('cors')

const corsOptions = {

origin: 'http://localhost:5173',

optionsSuccessStatus: 200

}

app.use(cors(corsOptions));

app.use(bodyParser.json());

app.use(express.static('resources'));

app.use('/api', router);

const server = app.listen(8080, function () {

let host = server.address().address

let port = server.address().port

console.log("App está rodando no endereço http://%s:%s", host, port);

});

db.sequelize.sync({force: true}).then(() => {

console.log('Apaga e recria a tabela usando { force: true }');

Cliente.sync().then(() => {

const clientes = [

{ nome: 'Pedro', idade: 23, email: 'pedro@email.com' },

{ nome: 'Sara' , idade: 31 , email: 'sara@email.com'},

{ nome: 'Emilly', idade: 18 , email: 'emilly@email.com'},

]

clientes.map(cliente =>{

Cliente.create(cliente);

});

})

});

=======================================================================

=======================================================================