**API BUILDING:(Cheet Sheet)(changing frequently)**

1.Understand the problem statement and what are the Features we need to build.(structure what are the apis (routes) and controllers we need to do and what are the models involve in those apis mentioned)

2.We need to define the database connection using Sequelize module and create a folder where we used to store various configurations related details.And install all the dependencies

- npm install express,- npm install mysql2,- npm install bcryptjs,- npm install sequelize,-npm install jsonwebtoken,- npm install dotenv,-npm install body-parser

**Configs folder**= db.config.js , secret.config.js, server.config.js

db.config.js(database configuration)

module.exports={

    HOST: 'localhost',

    DB:'ecommerce',

    USER: 'root',

    PASSWORD: '3612',

    dialect: 'mysql',

    pool:{

        max:5,//max threads so only 5 connections can be established at a time

        min:0,

        acquire:30000,//maximum a connection establishment will try to establish within 30000 milli seconds

        idle:1000//connection if its idle for 1000 milli seconds without making any request then the connections establishment will be released

    }

}

secret.config.js (we have secret data using which we generate jwt token)

module.exports={

    secret:"SriSarvesh Data"

}

server.config.js

if(process.env.NODE\_ENV !== 'production')

{

    require('dotenv').config();//SO THE CONTENT IN THE .env file is read as environmental variables;=

}

module.exports ={

PORT: process.env.PORT }

Models folder=index.js(which has all the files that import overall contents of the models like schema ,and does building of relations ship between various schemas)

db connection code (inside index.js) with respect to config we wrote

const Sequelize=require('sequelize');

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

//db connection establishment

const sequelize=new Sequelize(config.DB,config.USER,config.PASSWORD,{

    host:config.HOST,

    dialect:config.dialect,

    pool:{

        max:config.pool.max,

        min:config.pool.min,

        acquire:config.pool.acquire,

        idle:config.pool.idle

    }

});

Even inside the same file we can write about some relationships that exist between vartious schema

One to many: model1.hasMany(model2);

Many to many:

//one to many relationship between model1 to model2

model1.belongsToMany(model2,{

  through:"bridge\_table",//The bridge table name that will be created( for many to many relationship existing purpose)

  foreignKey:"model1\_prim\_key",//this model1’s primary key acts as foreign key in that bridge table

  otherKey:"model2\_prim\_key"

});

//one to many relationship between model1 to model2

model2.belongsToMany(model1,{

through:"model2\_prim\_key",

    foreignKey:"model1\_prim\_key",

    otherKey:"role\_id"

});

3.First we need to find what are the various Entities involved here(in the question description) and create models/schema based upon the Entity(If that particular entity has certain attributes mentioned or we define the default attributes like id,name..etc)

Schema/model creation template:

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

    const modelname=sequelize.define('table\_name\_of\_model',{//table name will be categories as a plural(as its resource) and sequelize does this

        Coloumname1:{

            type:Sequelize.INTEGER,

            primaryKey:true,

            autoIncrement:true

        },

        Columname2:{

            type:Sequelize.STRING,

            allowNull:false

        }

        ,

        Columname3:{

            type:Sequelize.STRING

        }

    });

    return modelname;

}

After defining various schemas/entities we need to go to server.js and do database syncronization that is (initializing all the schemas,relationships)everything related to the sequelize(db connection) that we defined {force:true} drop table content when ever server res

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

    console.log("table/schema is created");

    //inside this we can write any functions that perform default creation of some values to the models we created so as we pass in json the values of the rows we need to create we can create array and do model.bulkcreate() also

}).catch(err => console.log(err));

4.And as per the question we need to see what are the various api’s we need to implement and we need to write the controller function that satisfies each api call.

5.And while defining the controllers and models we need to define the index.js that has all the function and properties that overall exports from that particular folder.

6. Sequlize cheetsheet:

<https://dev.to/projectescape/the-comprehensive-sequelize-cheatsheet-3m1m#order>

7.And we need to write various functions inside each controller functions that involve various sequelize functions.

8.And for each api url(which connects to the controller function) we need write separate routes .Categorize routes file based on the models they take effect on.

Some common Syntax:

1. For creating or post data into the database:

    const data={

        name:req.body.name,

        description:req.body.description

    }

    ModelName.create(data).then((catData)=>{

        res.status(201).send(catData);

    }).catch((err)=>{

        res.status(500).send(err.message);

    })

1. For getting all the datas from a table

ModelName.findAll({where:{condition}).then((categories)=>{

        res.status(200).send(categories);

    })

    .catch((err)=>{

        res.status(500).send(err);

    })

1. Finding a particular row or data using primary key(single data)(:id)req.params.id

 const categoryId=req.params.id;

    ModelName.findByPk(categoryId).then((category)=>{

        res.status(200).send(category);

    })

    .catch(err=>{

        res.status(500).send(err.message);

    })

1. Controller for getting based on the name (passed in the query parameter ?name=””

//for getting category based on name (query params)

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

    const categoryName=req.query.name;

    let promise;

    if(categoryName)

    {

         //if valid string is entered in between name query parameter

        promise=ModelName.findAll({

            where: {name: categoryName}

           });

    }

    else{

        //if name query params is not passed or if empty string is passed in the name query params

        promise=ModelName.findAll();

    }

  promise.then((categories)=>{

        res.status(200).send(categories);

    })

    .catch((err)=>{

        res.status(500).send(err.message);

    })

}

1. For deleting the based on condition

    const CategoryId =req.params.id;

    ModelName.destroy({

        where:{

            id:CategoryId

        }

    }).then(()=>{res.send("Data is deleted successfully")}).catch(()=>{res.send("Error")})

1. For Updating the data based on the condition (after updating we need to return the data (based on the id) after the updation of data)

 const category = {

        id: req.params.id,

        name: req.body.name,

        description: req.body.description

    }

    ModelName.update(category,

        {

            where: {id: category.id},

            returning:true

        }

    ).then(()=>{

        //here we need to get the content from another findByPk method where we pass id

        ModelName.findByPk(category.id).then((category)=>{res.status(201).send(category)}).catch((err)=>{res.status(500).send(err.message)});

}).catch((err)=>{res.status(500).send(err.message)});

1. Route file ex:a.route.js

module.exports =(app)=>{

app.post("/ecom/api/v1/resource",[middlewares(optional)],Controller.method1);

    app.post("/ecom/api/v1/v1/resourse",[middlewares(optional)],Controller.method);

}

1. we can store the password(req.body.password) while signup in a encrypted way by (password str ,length of the hashed string)

const bcrypt=require('bcryptjs');

bcrypt.hashSync(req.body.password,8);

This will syncronously hash the password

1. when there is many to many relationship set between model1 and model2 we set the values of the common property that relates both like:(ex)

**Model.setModel2s(arrayofobj).then().catch()**

User has multiple roles like(customer,admin)

And each role out of(customer,admin) there can be many users

   user.setRoles(roles).then(()=>{ //roles is array (multivalued in m to m relationship)

                    res.status(200).send("Registration completed");

                })

.catch((err) => {res.status(500).send(err.message)});

Bridge table: consider role\_id=1(for customer) and role\_id=2(for admin)

Which means we can say that user\_id 1 is assigned with role\_id 1 and 2 and also role\_id 1 has user\_id 1 and role\_id 2 has user\_id 1 …(when same user is alloted with both roles customer and admin)

User\_id role\_id

1 1

1 2

1. While defining schema when a particular attribute has primary key then that alone can represent the whole object so for example

Ex:1 is the primary key value which represent single role object ..

 user.setRoles([1]).then(()=>{//we are passing array of roles object in setRoles

              res.status(200).send("Registration successfully");

          }).catch(err=>{res.status(500).send(err.message)});

1. When a model1 is set many to many relationship with the other model2 and to get what are the values that are set from model 2 to model 1 is

user.getRoles().then().catch();

1. When the password is passed from the req.body while signin we need to check/compare with the req.body password and hashed value of the password from the user object .we do that by

bcrypt.compareSync(req.body.password,user.password);

1. And when ever a user successfully signed in we need to generate a token that will be sended to the user back as a response

Jwt.sign({data\_from\_user\_obj:value},secret data,{expiresIn:300}

Var token=jwt.sign({id:user.id},secretKey.secret,{expiresIn:300});

1. When we try to authorize a user based on token we do it by

//token //callback

jwt.verify(token,config.secret,(err,decodedKey) => {

        if(err) //decodedKey is a object contains data using

        { which we generate token

            //if any error occurs we try to send the message as invalid token when we try to validate

            res.status(401).send({message:"Invalid token"});

            return;

        }

        //decodedKey has info of the user(particular infor using which we created token)

        req.userId=decodedKey.id;

        next();

    });