Práctica MVC

SOLUCIONARIO

En este documento iremos reflexionando sobre lo realizado en la práctica

# Estructura básica de la aplicación

**Copia y pega el código de app.js**

var express = require('express');

const hbs = require('express-handlebars');

const path = require('path');

var app = express();

app.engine('hbs', hbs({

    extname: 'hbs',

    defaultLayout: 'layout',

    layoutsDir: path.join(\_\_dirname, './views/layouts/')

   }));

 app.set('view engine', 'hbs');

app.get('/', function (req, res) {

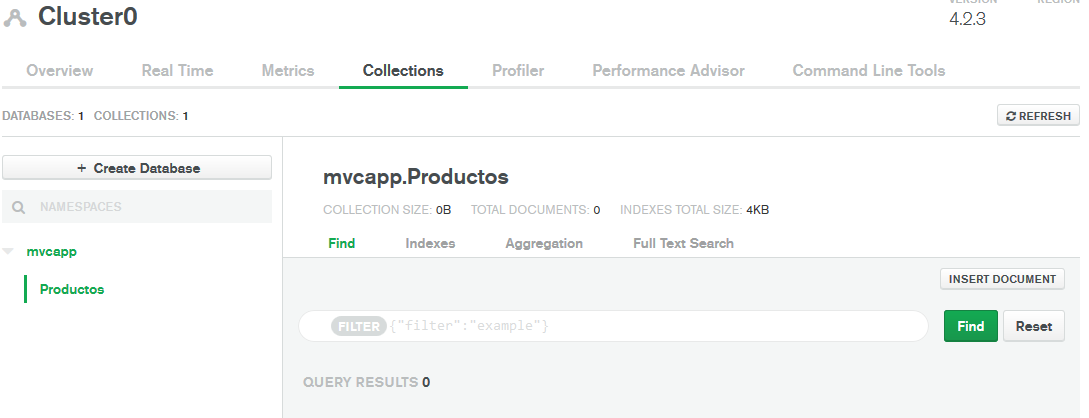
    res.render('home', { title: " aquí pon un mensaje "});

});

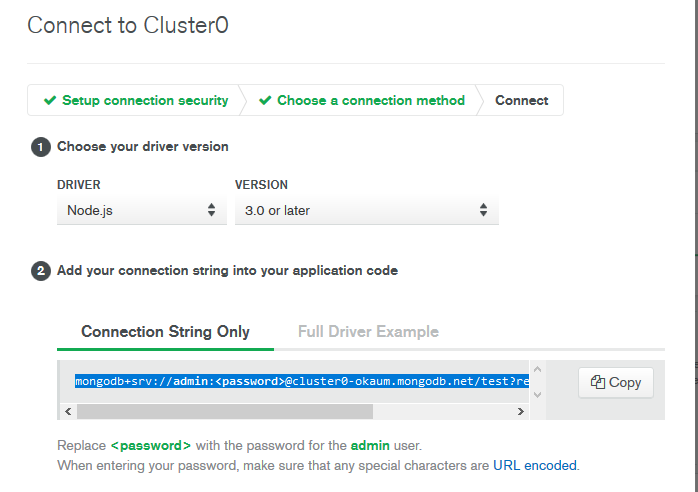
app.listen(3000);

# Conexión con la base de datos

**Muestra pantalla de tu base de datos en mongodb atlas**



**Muestra la ruta de conexión a tu base de datos**



# Configuración de variables de entorno

**Pegar código de package.json donde usas cross-env**

  "dev": "cross-env NODE\_ENV=development nodemon app.js"

**Pegar código de como usas config para conectarte a la base de datos**

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

     require('dotenv').config();

module.exports = {

    port: process.env.PORT || 3000,

    env: process.env.ENV || 'development',

    dbUser: process.env.DB\_USER,

    dbPassword: process.env.DB\_PASSWORD,

    dbName: process.env.DB\_NAME,

    dbCluster: process.env.DB\_CLUSTER,

    get dbUrl() {

        return `mongodb+srv://${this.dbUser}:${this.dbPassword}@${this.dbCluster}.mongodb.net/${this.dbName}?retryWrites=true&w=majority`

    },

    //si se desea algo especifico

    production: {

        //aquí se pone la configuración de producción

    },

    development: {

        //aquí se pone la configuraciópn de development

    }

};

const mongoose = require('mongoose');

const URL =  require('../config/config.js').dbUrl;

console.log(URL);

mongoose.connect(URL, {

 useNewUrlParser: true,

 useCreateIndex: true,

 useUnifiedTopology: true

}).then(()=>{

 console.log("Connected to database");

}).catch((err)=>{

 console.log("Not connected to database", err);

});

module.exports = mongoose;

**Código de .env**

PORT=3000

ENV=development

DB\_USER=admin

DB\_PASSWORD=jBpsMtN009Wf7eDO

DB\_NAME=mvcapp

DB\_CLUSTER=cluster0

# Creación de rutas

**Muestra el código de tus rutas de ambos archivos**

const router = require('express').Router();

router.get('/', async (req,res)=>{

    res.send("favorites");

});

router.post('/:id/', async (req, res) => {

    res.send("favorite id");

   });

module.exports = router;

const router = require('express').Router();

router.get('/', async (req,res)=>{

    res.send("products");

});

router.get('/:id/', async (req, res) => {

    res.send("product id");

   });

module.exports = router;

**Muestra el código donde importas los routers en app.js**

const favoriteRouter = require('./routes/favorite.route');

const productRouter = require('./routes/product.route');

app.use('/favorite',favoriteRouter);

app.use('/product',productRouter);

# Creación de modelos

**Muestra el código de DB donde haces test de tu base de datos**

const mongoose = require('./mongodb-connection')

class DB{

    async query(query,projection={},options={}){

        console.log(query,projection,options);

        return await this.\_model.find(query,projection,options);

    }

    async queryOne(query,projection={},options={}){

        return await this.\_model.findOne(query,projection,options);

    }

    async update(query,dataObject){

        return await this.\_model.findOneAndUpdate(query,{$set:dataObject},{new:true});

    }

    async exists(query){

        return await this.\_model.exists(query);

    }

    async add(document){

        return await this.\_model.add(document);

    }

}//esquema de prueba (esto podráser borrado o comentado después)

class Test extends DB{

    constructor(){

        super();

        this.schema=new mongoose.Schema({name:{type:String,required:true}});

        this.\_model=mongoose.model('products',this.schema);

    }

    async getData(){

        return await this.\_model.find({});//COMPLETAR AQUÍ

    }

        //PRUEBA OTROSMÉTODOS PARA ACTUALIZAR, AÑADIR, EXISTE

}

let test=new Test();

(async()=>console.log(await test.getData()))()

module.exports=DB;

**Pega el código de tu modelo de Producto**

const mongoose=require('../db/mongodb-connection')

const DB=require('../db/DB');

class Product extends DB{

    constructor(){

        super();

        this.schema=newmongoose.Schema({

            uid:{type:Number,unique:true},

            name:{type:String,required:true},

            description:{type:String,required:true},

            imageUrl:{type:String},

            price:{type:Number,required:true},

            quantity:{type:Number,required:true}});

        this.\_model=mongoose.model('products',this.schema);

    }

    async getProducts(query,projection="",options={}){

        return await super.query(query,projection,options);

    }

    async getProductById(uid,projection="",options={}){

        return  await super.query({uid:uid},projection,options);

    }

    async exists(conditions){

        let doc= await super.exists(conditions);

        return doc;

    }

}

let product=new Product();

module.exports=product;

# Creación de controladores

**Muestra el código del controlador completo**

const product=require('../models/Product');

class ProductController{

    async read(req,res){

        //revisar si tiene quer y params

        let {query}=req//buscar por nombre o uid

        let {options}=req//página o limit

        let {projection} =req; //quéquiero ver de la información

        let docs=await product.getProducts(query,projection,options);

        let prods=JSON.parse(JSON.stringify(docs));

        console.log(prods);

        res.json(prods);

        //res.render('home',{productos:prods })

    }

    async readById(req,res){

        //revisar si tiene quer y params

        let {uid}=req//buscar por nombre o uid

        let {options}=req//página o limit

        let {projection} =req; //quéquiero ver de la información

        let docs=await product.getProductById(uid,projection,options);

        let prods=JSON.parse(JSON.stringify(docs));

        console.log(prods);

        res.json(prods);

        //res.render('home',{productos:prods })

    }

}

const productController = new ProductController();

module.exports=productController;

**Muestra un screenshot del json que regresa**



# Crear vista

**Muestra el archivo hbs**

{{#if values}}

<div class="container">

    {{#each values as |v|}}

        <div class="media">

            <img class="mr-3" src="{{v.imageUrl}}" alt="Generic placeholder image">

            <div class="media-body">

                <p>{{v.\_id}}</p>

                <p>{{v.uid}}</p>

                <p>{{v.name}}</p>

                <p>{{v.description}}</p>

                <p>{{v.imageUrl}}</p>

                <p>{{v.price}}</p>

                <p>{{v.quantity:}}</p>

            </div>

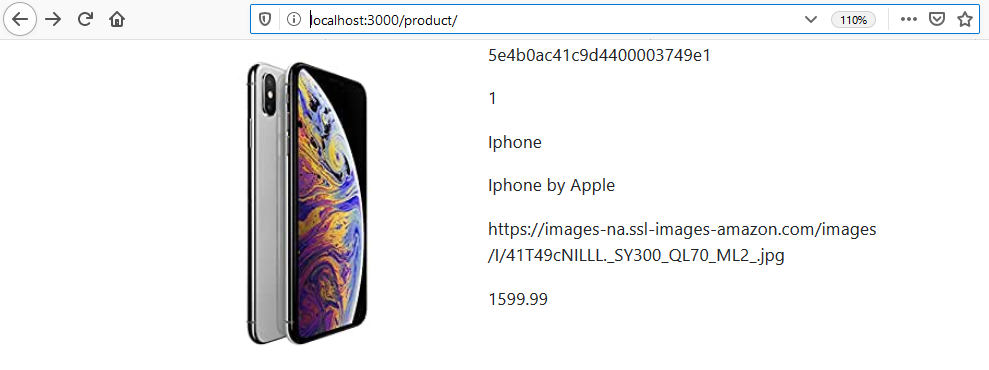
        </div>

    {{/each}}

</div>

{{/if}}

**Muestra screenshot de los productos**



# Creación de middlewares

**Muestra código de middlewares que validen los query params o los route params (existencia del id)**

const auth = async(req, res, next) => {

    try {

        let {id}=req.params

        if (typeof id === 'undefined') {

            throw { errorType: 'Unauthorized', code: 401, message: 'No tienes permisos para ver estos recursos' };

        }

        console.log(id)

        next()

    } catch (error) {

        let code = 500;

        if (error.code) code = error.code

        res.status(code);

        res.send(error);

    }

}

module.exports = auth

const auth = require('../middleware/auth');

router.get('/:id/',auth, productController.readById);

# Completar para favoritos

**Muestra los códigos relacionados**

# Conclusiones

En esta practica pude aprende buenas practicas, al saber como es que se abstrae los metodods basicos en mongodb para poder ser empleador en todos los esquemas. De igual manera pude aprender a usar controladores y middlewares con el fin de hacer una app responsiba. Me gusto el poder hacer la estructura basica del proyecto desde 0 ya que así puedo enteneder mejor el funcionamiento y la arquitectura.