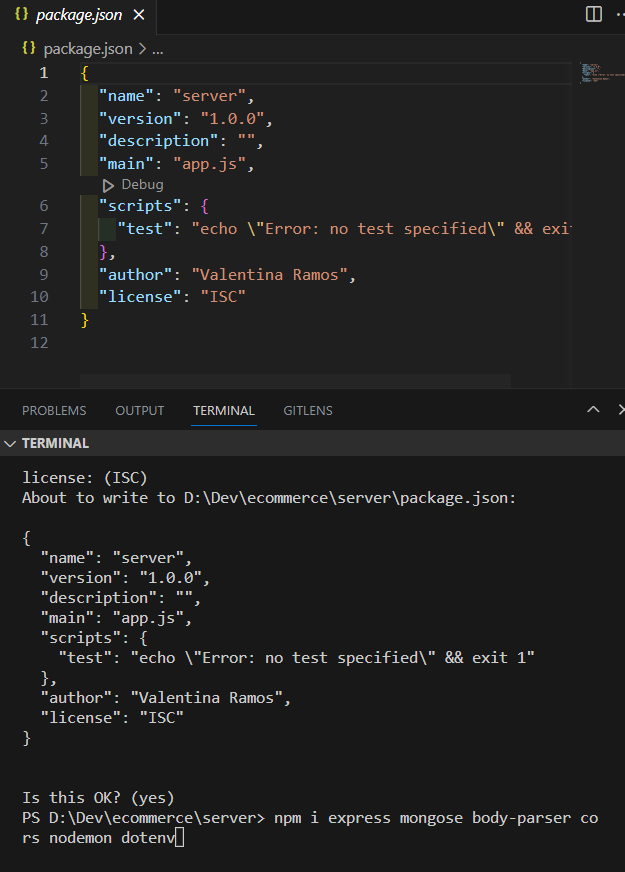
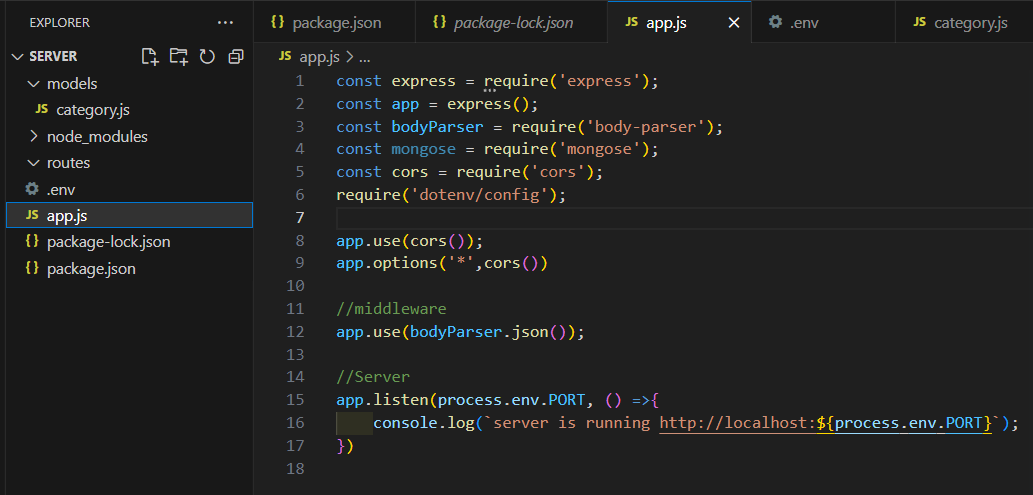
Elaboración del SignIn

BACK END

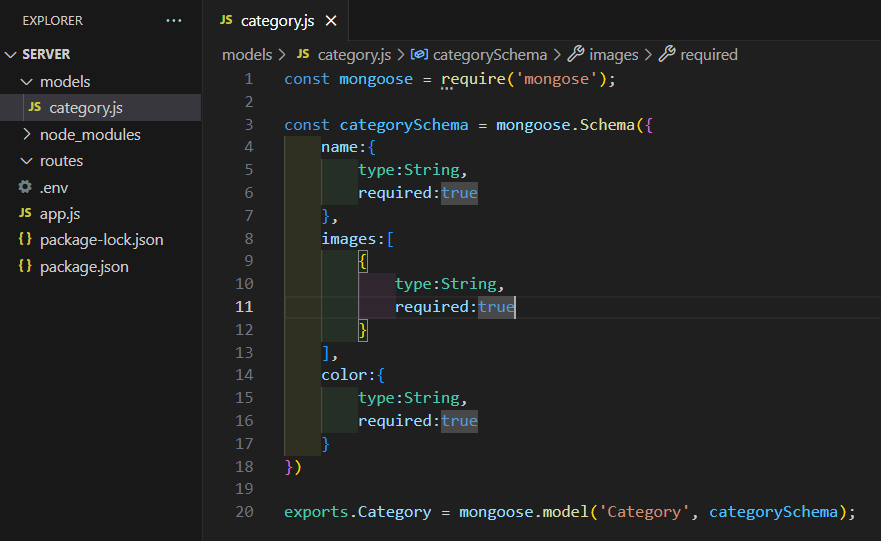
* Creación base de datos con Mongo DB



* Nodemon
* Creación y configuración del archivo app.js con express:



-Creación de la carpeta models, creación y configuración del archivo category.js

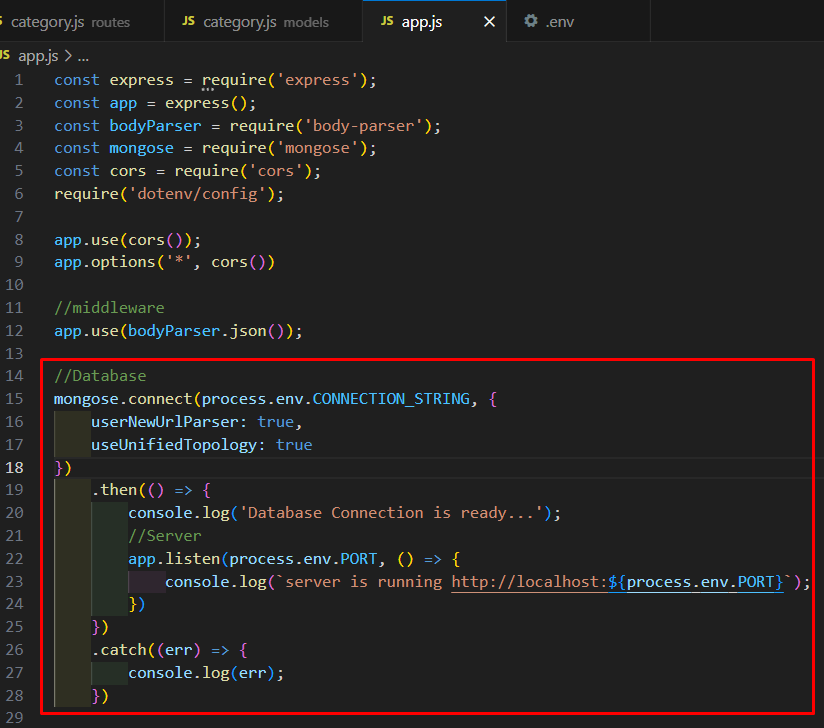


-Conexión con mongo db

mongodb+srv://dbValenRR:<db\_password>@cluster0.t8njf.mongodb.net/

mongodb+srv://dbValenRR:<db\_password>@cluster0.t8njf.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

Una vez crea la conexión con la web de mongo, se incorpora la base de datos al archivo app.js

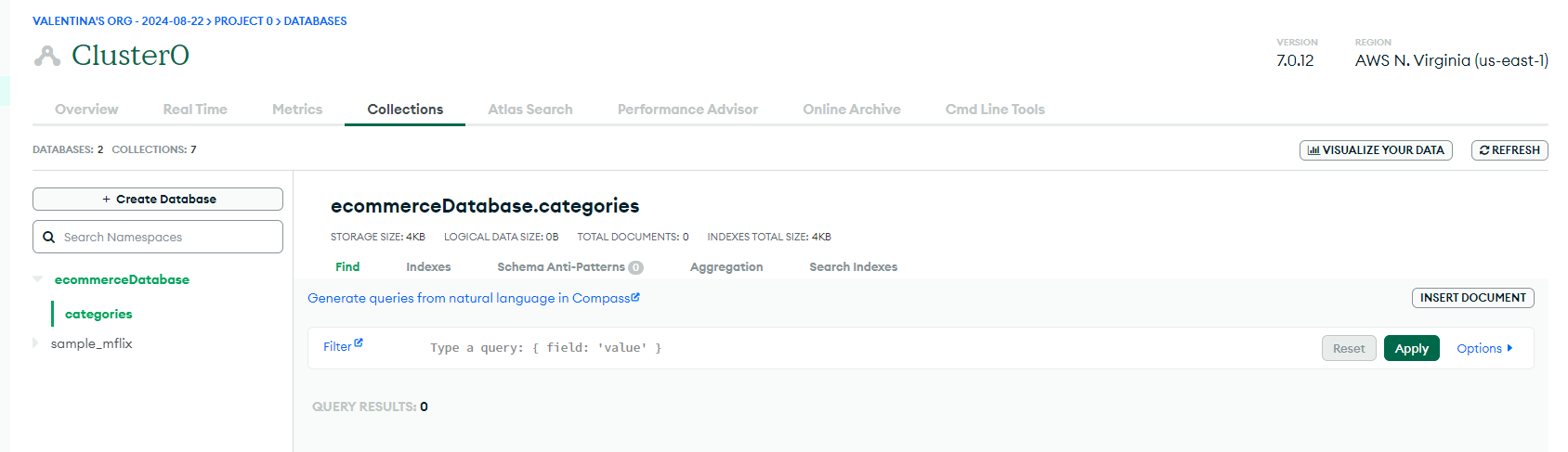


Y la codificación de las rutas en app.js:

//Routes

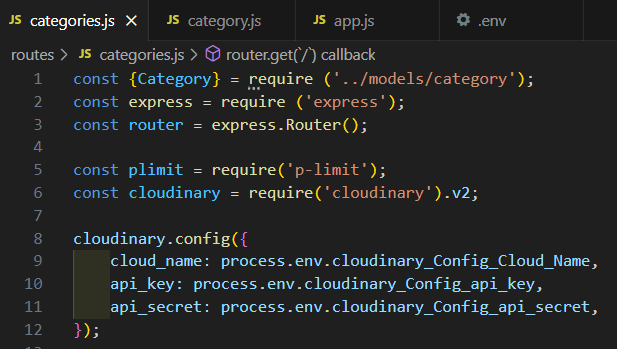
const categoryRoutes = require('./routes/categories');

Creación de la base de datos:



Petición post:

Creación de cuenta en cloudinary y configuración en la carpeta routes de la api, archivo categories.js



//Aca hacen una pruebas en postman que no se bien de donde sacan los datos.

Petición get:

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

    const category = await Category.findById(req.params.id);

    if(!category){

        res.status(500).json({message: 'The category with the given ID was not found.'})

    }

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

})

Petición delete:

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

    const deletedUser = await Category.findByIdAndRemove(req.params.id);

    if(!deletedUser){

        res.status(404).json({

            messahe: 'Category not find!',

            success: false

        })

    }

    res.status(200).json({

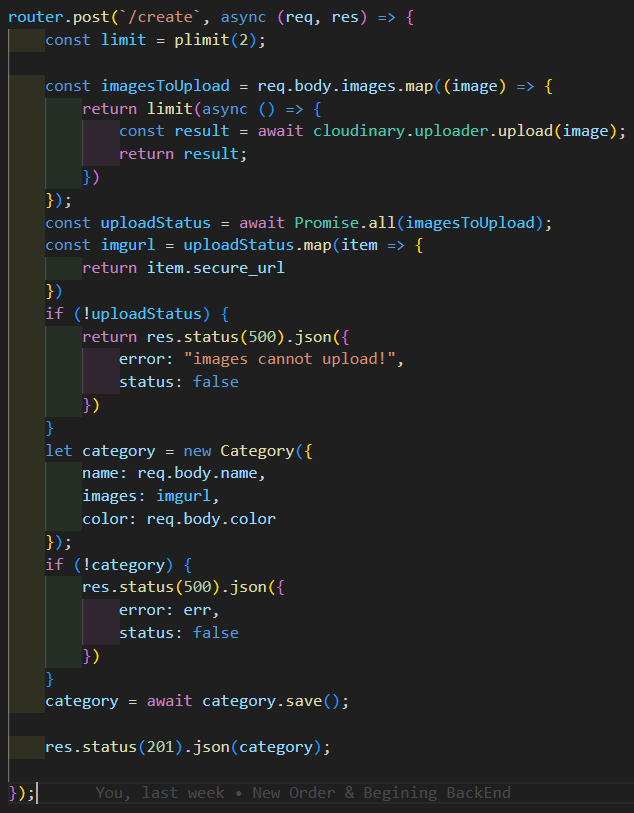
        success: true,

        message: 'Category Deleted'

    })

});

Petición Post:



Petición PUT:

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

    const category = await Category.findByIdAndUpdate(

        req.params.id,

        {

            name:req.body.name,

            icon:req.body.icon,

            color:req.body.color

        },

        {new:true}

    )

    if(!category){

        return res.status(500).json({

            message:'Category cannot be updated!',

            success: false

        })

    }

})

Configuración para obtener y modificar las categorías, desde el archivo category.js:

userSchema.virtual('id').get(function(){

    return this.\_id\_toHexString();

});

userSchema.set('toJSON',{

    virtuals: true,

});

* Creación del archivo products.js dentro de la carpeta models:

Se utiliza Mongo para obtener todos los datos de cada producto:

const mongose = require("mongose");

const productSchema = mongose.Schema({

    name: {

        type: String,

        required: true,

    },

    description:{

        type: String,

        required: true

    },

    brand:{

        type: String,

        required: true

    },

    price:{

        type: Number,

        default: 0

    },

    category:{

        type:mongose.Schema.Types.ObjectId,

        ref:'Category',

        required: true

    },

    countInStock:{

        type: Number,

        required: true

    },

    rating: {

        type: Number,

        default: 0,

    },

    numReviews:{

        type: Number,

        default: 0,

    },

    isFeatured: {

        type: Boolean,

        default: false,

    },

    dateCreated: {

        type: Date,

        default: Date.now,

    },

})

exports.Product = mongose.model('Product', productSchema);