Backend Laboratorio

En este archivo se agrega el modelo, las rutas y cada controlador, así como las pruebas realizadas en Postman y MongoDB Compass.

1. Modelo

```
//10. definimos una constante para llamar a Mongoose
const mongoose = require ('mongoose');

//11. se crea el modelo, que debe ser igual al de DB
const productSchema = mongoose.Schema({
    id:{
        type: Number,
        required: true
    },
    productName:{
        type: String,
        required: true
    },
    productType:{
        type: String,
        required: true
    },
    provider:{
        type: String,
        required: true
    },
    price:{
        type: Number,
        required: true
    },
    quantity:{
        type: Number,
        required: true
    },
},{versionkey: false});

//11.1 exportar el modelo de la carpeta y el schema definido
module.exports = mongoose.model('Product', productSchema);
```

2. Rutas

```
//9. se debe llamar a express de nuevo
const express = require ('express');
const router = express.Router();
const productControllers = require('../controllers/productControllers');

//definimos rutas del CRUD (create, read, update, delete)
router.post('/addproduct', productControllers.addProducts);  //13.1
router.get('/searchproducts', productControllers.searchProducts);  //13.2
router.get('/:id', productControllers.searchByProduct);  //13.3
router.delete('/:id', productControllers.deleteByProduct);  //13.4
router.put('/:id', productControllers.updateProduct);  //13.4


//9. exportando el modulo
module.exports = router;

```

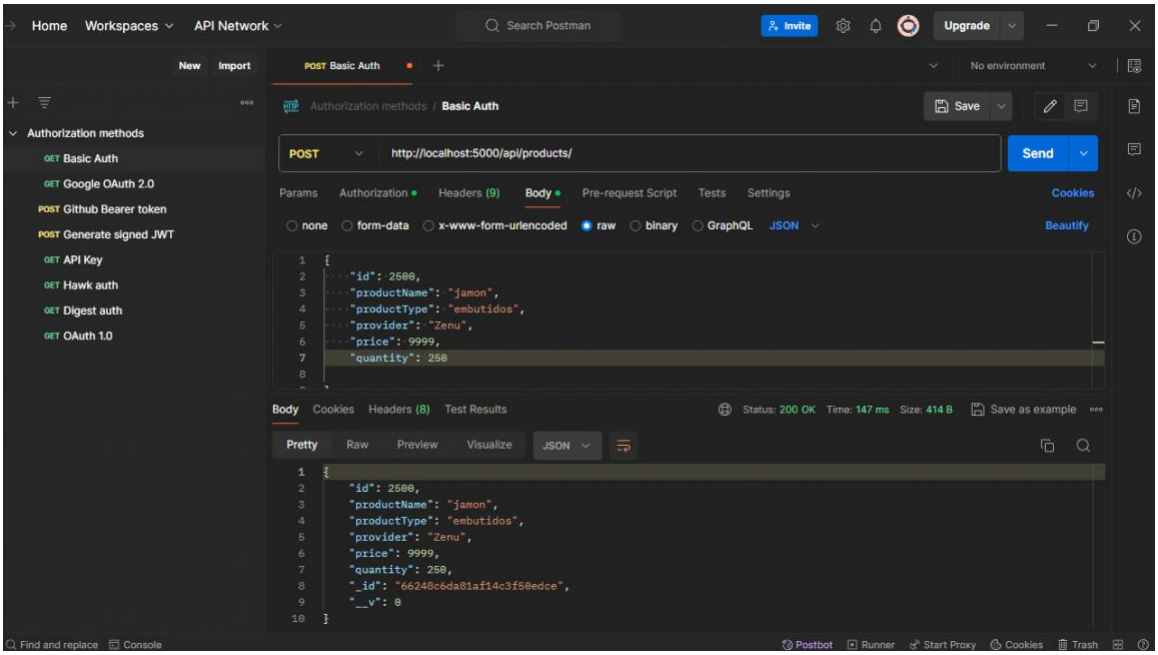
3. Controladores

3.1 Función para agregar productos

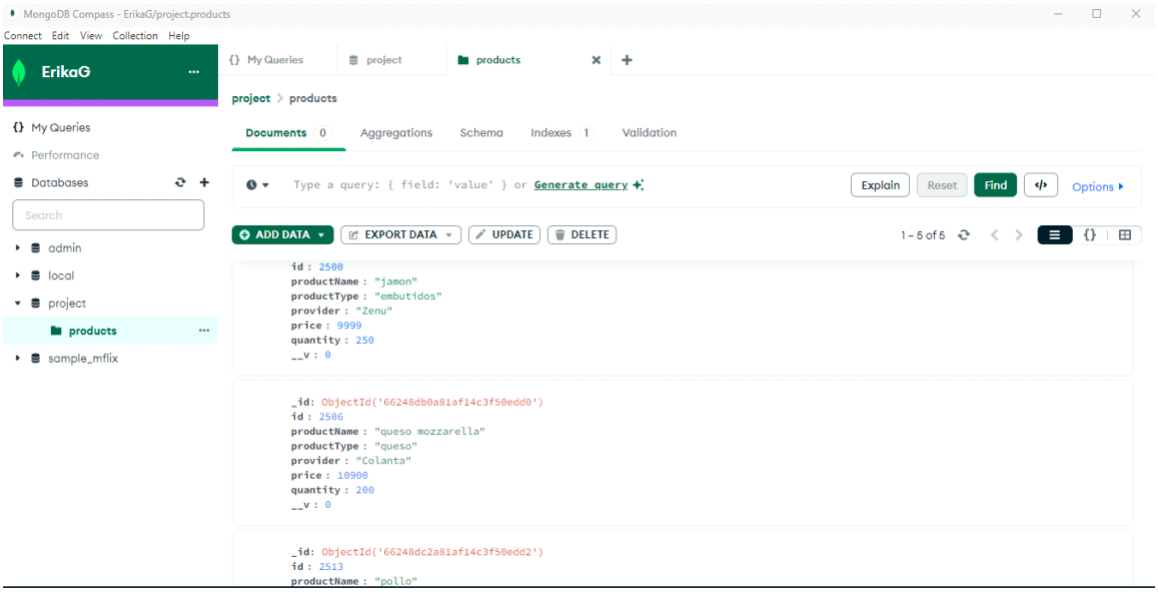
```
//13. se crea y exporta cada funcion asincrona para agregar desde postman
//13.1 Funcion para agregar
exports.addProducts= async(req, resp) =>{
    try {
        let products;
        products = new Product(req.body)
        await products.save();
        resp.send(products)


    } catch (error) {
        console.log(error)
        resp.status(500).send('There was an error when adding a new product')
    }
}
```

Prueba realizada con Postman donde los productos han sido agregados



Esta es la base de datos vista desde MongoDB Compass



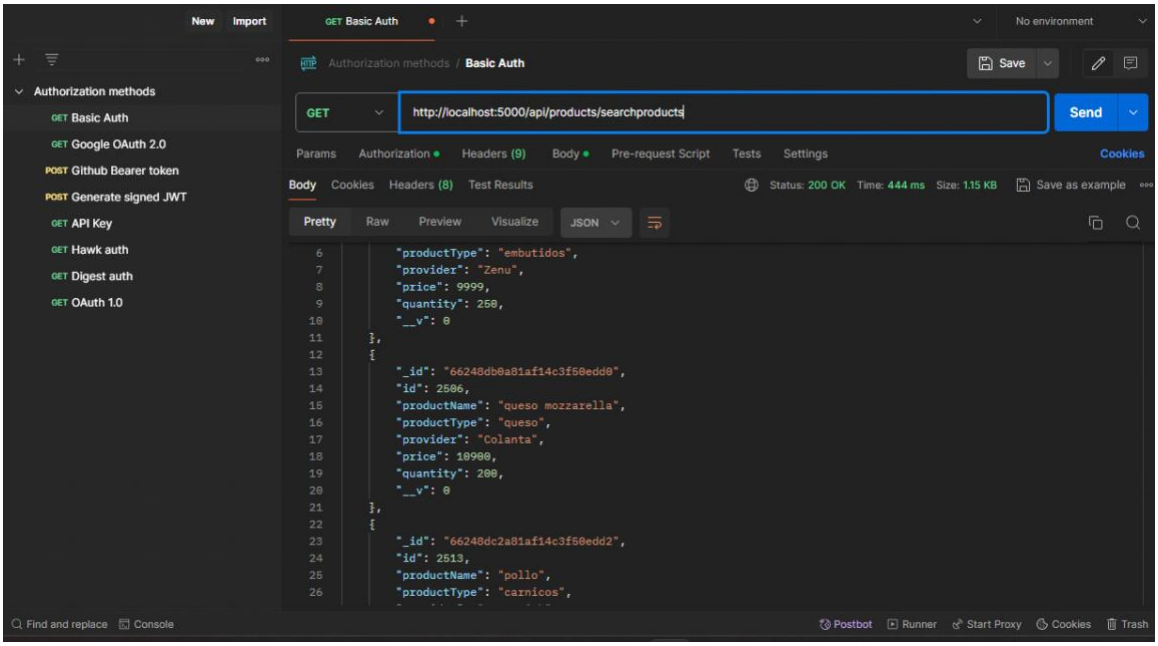
3.2 Función para buscar todos los productos

```
//13.2 Funcion para buscar
exports.searchProducts = async(req, resp) =>{
    try {

        let products = await Product.find();
        resp.json(products)

    } catch (error) {
        console.log(error)
        resp.status(500).send('There was an error when searching the products')
    }
}
```

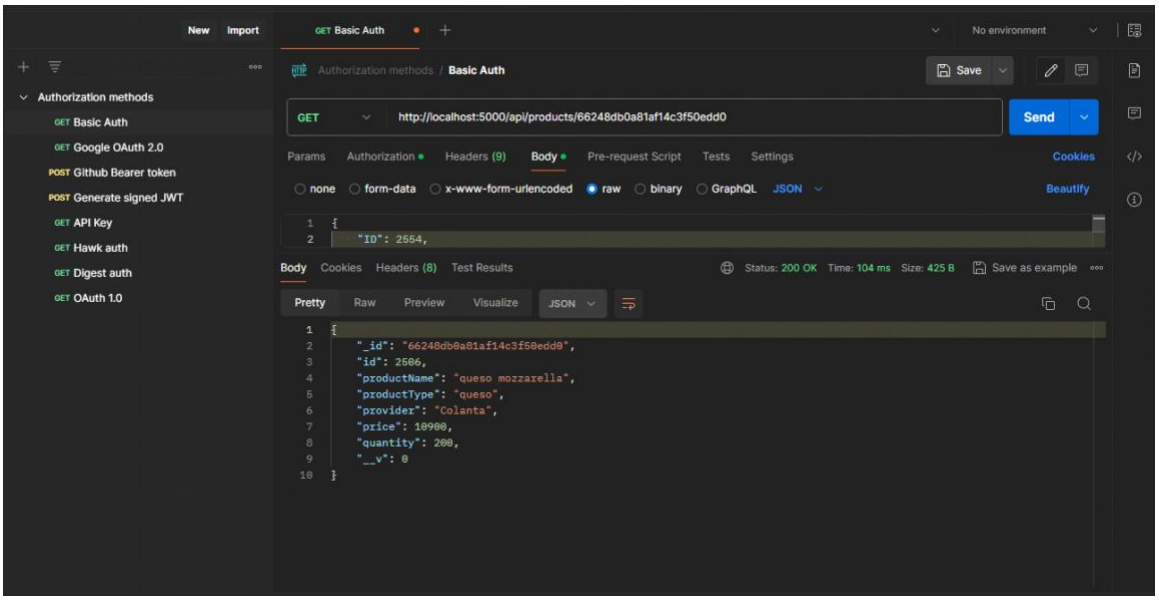
Prueba hecha con Postman donde se buscan productos



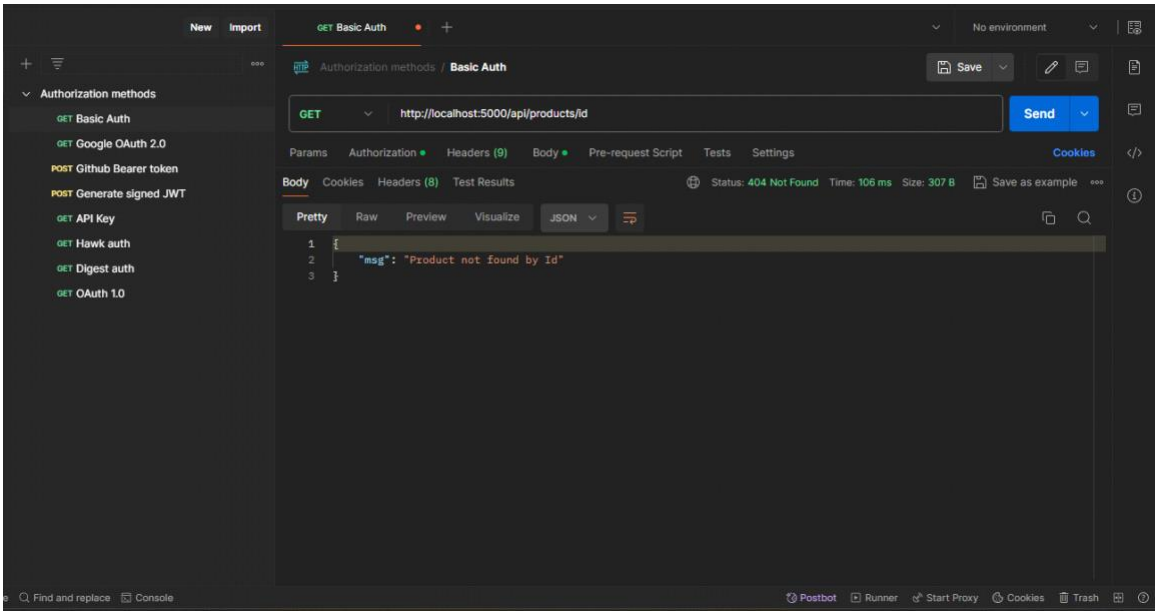
3.3 Función para filtrar por ID

```
//13.3 funcion para filtar por un solo producto
exports.searchByProduct = async(req, resp) =>{
    try {
        let product = await Product.findById(req.params.id);
        if (!product){
            resp.status(404).send({msg:'Product not found by Id'})
            return
        }
        resp.send(product);
    } catch (error) {
        console.log(error)
        resp.status(500).send('There was an error when searching that product')
    }
}
```

Prueba realizada con Postman donde el producto ha sido encontrado



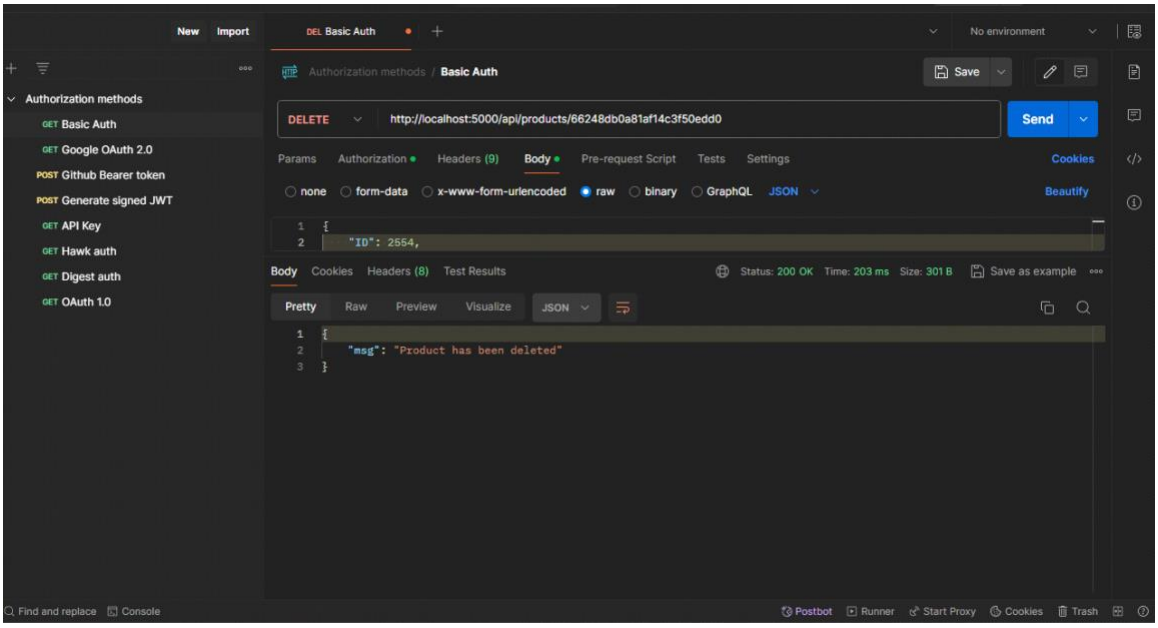
Prueba realizada con Postman donde el producto no ha sido encontrado



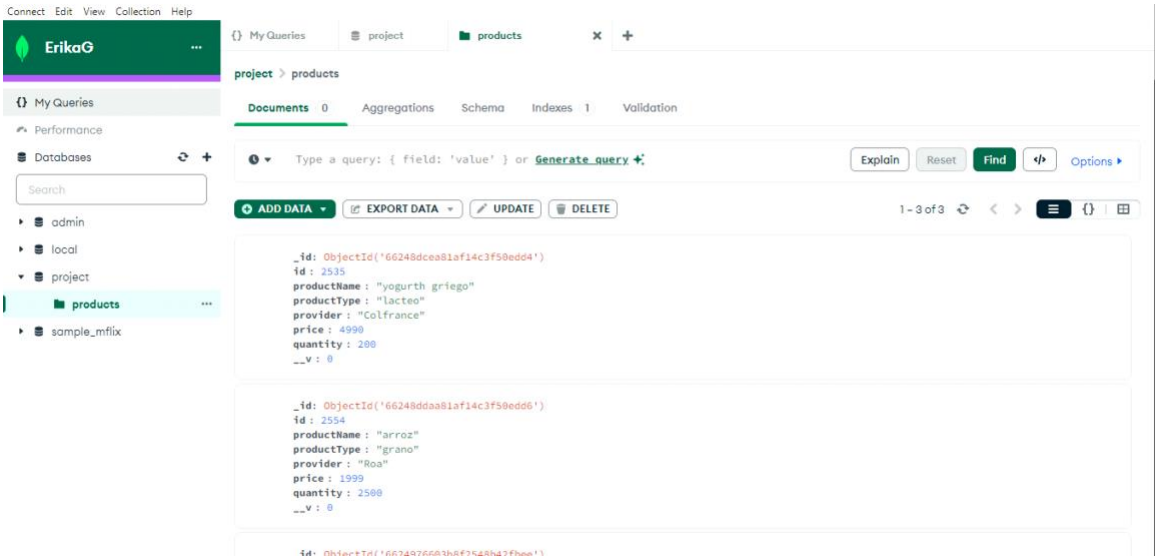
3.4 Función para eliminiar un producto

```
//13.4 funcion para eliminar producto
exports.deleteByProduct = async(req,resp)=>{
    try {
        let product = await Product.findById(req.params.id);
        if(!product){
            resp.status(404).json({msg: 'Product does not exist'})
            return
        }
        await Product.findOneAndDelete({_id: req.params.id});
        resp.json({msg: 'Product has been deleted'})
        return
    } catch (error) {
        console.log(error)
        resp.status(500).send('There was an error trying to delete that product')
    }
}
```

Prueba realizada en Postman donde se evidencia el producto eliminado



MongoDB Compass donde se muestran los productos eliminados



3.5 Función para actualizar/modificar un producto existente

```
//13.5 funcion para modificar/actualizar producto (de esta forma se ve el cambio en Postman)
exports.updateProduct = async(req,resp)=>{
    try {
        const {id, productName, productType, provider, price, quantity} = req.body
        let product = await Product.findById(req.params.id)
        if(!product){
            resp.status(404).json({msg: 'Product not found'})
            return
        }else{
            product.id = id;
            product.productName =productName;
            product.productType = productType;
            product.provider = provider;
            product.price = price;
            product.quantity = quantity;

            product = await Product.findOneAndUpdate({_id: req.params.id}, product, {new: true});
            resp.json(product);
        }
    } catch (error) {
        console.log(error)
        resp.status(500).send('Unable to update a product')
        return
    }
}
```

Prueba en postman con el producto ya modificado