**COMP229 – Web Application Development**

Assignment 2

# Dress Store Application – Node.js, Express REST APIs & MongoDB

Due Week #6 (……………………….) @ midnight

Value 15%

Dress Store – Node.js, Express REST APIs & MongoDB **Maximum Mark: 100**

**Overview**: Create the Node.js Express exports REST APIs that interacts with MongoDB Database using Mongoose ODM for a Dress Store application (Note: The Front-end of the application is not included in this Assignment).

# Instructions :

The Dress Store application:

1. Using MongoDB database, create:**(25 Marks):**
   1. A database by name DressStore**.**
   2. Create the following collections with their respective property. (5 Marks: Functionality).
2. **products**

name: string

description: string

price: number

published: Boolean

category: string

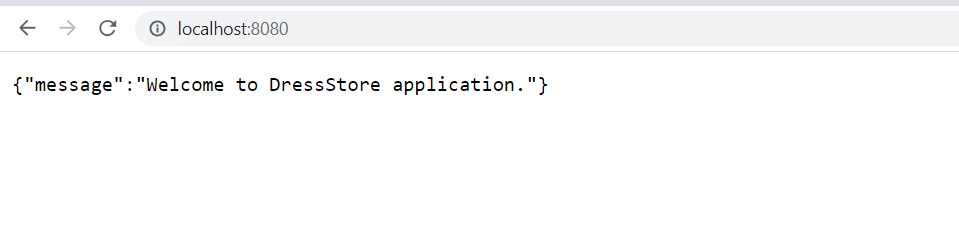
1. **categories**

name: string

the categories of products to be included are Men, Women, teens.

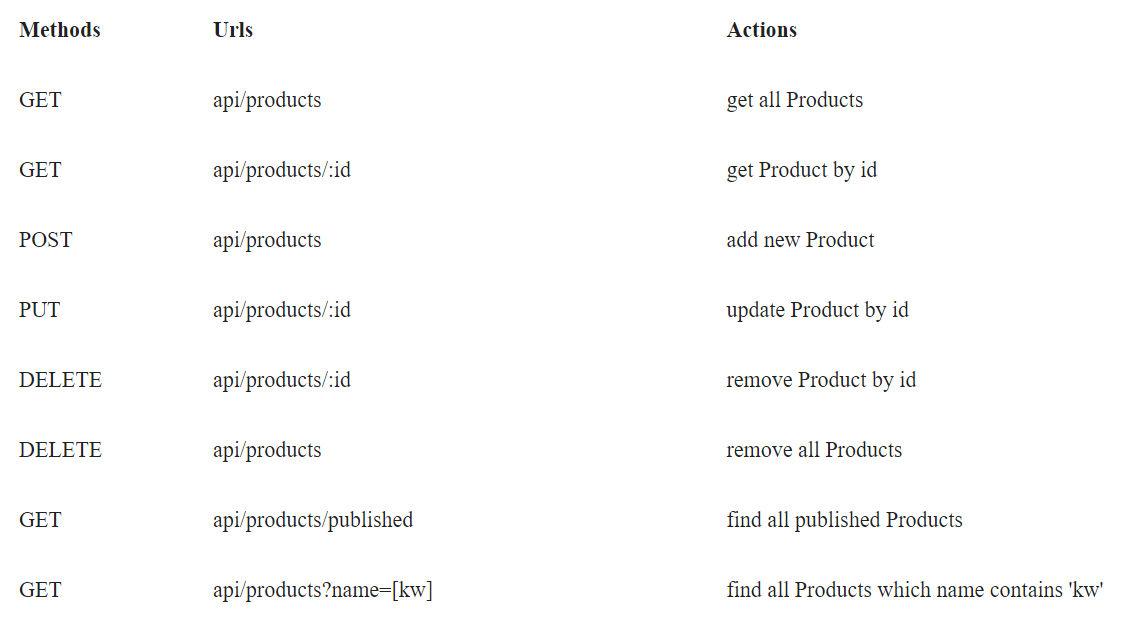
* 1. Obtain your connection string ( url or uri)
  2. Provide the screen snapshot of your MongoDB database showing the above steps from 1a – c.

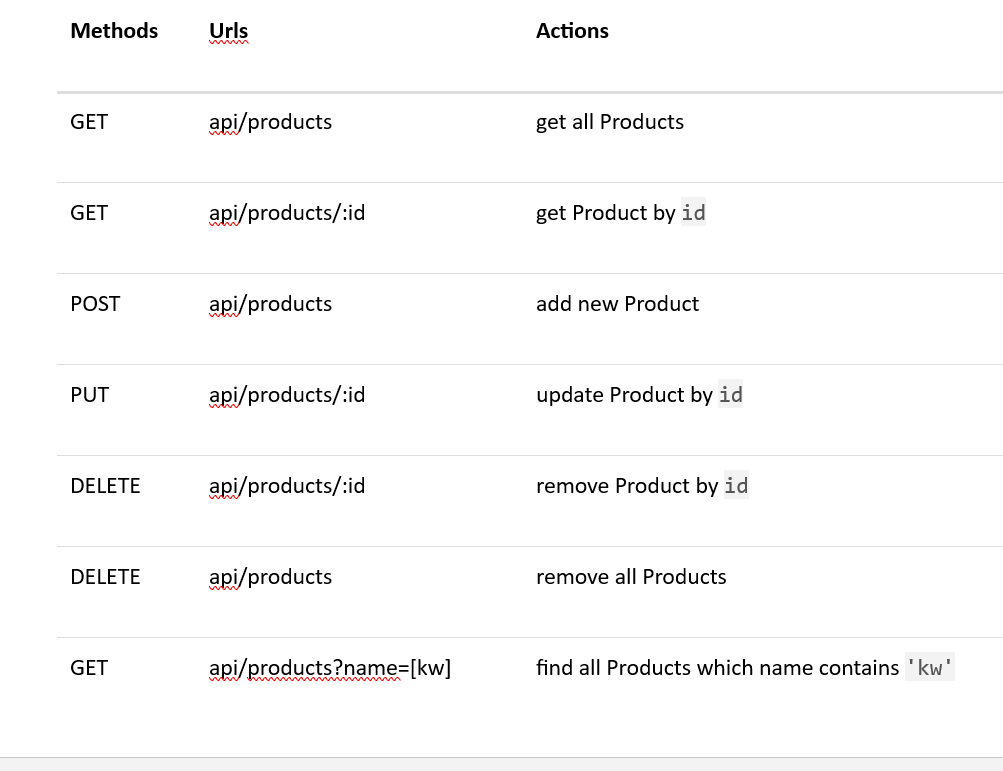
1. Using Visual studio code as the IDE: **(25 Marks)**
   1. create a node.js App for the DressStore by setting up the Express web server. Ensure to install all the necessary modules: express, mongoose, cors e.t.c.
   2. Run the app and provide a screen snapshot of it running in the browser as follows:



1. After creating the Express web server next: **(30 Marks)**
   1. Add the configuration for the MongoDB database.
   2. Create the product model with Mongoose.
   3. Write the controller.
   4. Define the routes for handling all CRUD operations.

Below is an overview of the REST APIs that will be exported:





1. a) Test the REST APIs using Postman, Thunder client or any tool you are familiar with. e.t.c.

b) Provide the screen snapshot of the test. **(20 Marks)**

**SUBMITTING YOUR WORK**

Your submission should include:

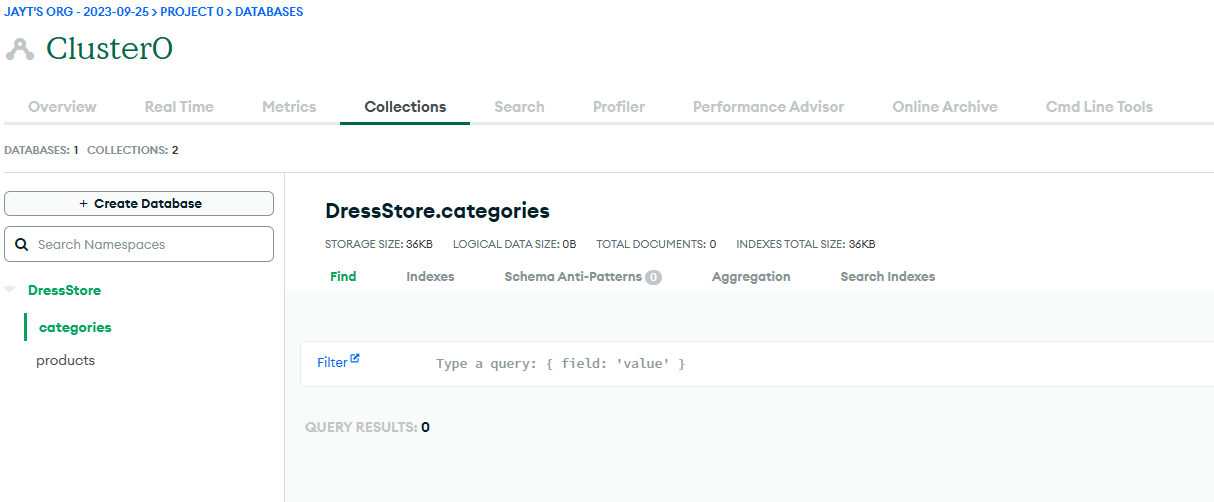
1. A zip archive of your Dress Store Project files
2. A link to GitHub
3. A word doc. Showing the snapshots

This assignment is weighted **15%** of your total mark for this course. Late submissions:

* + 20% deducted for each day late.

**Screenshots**

**1.a.**



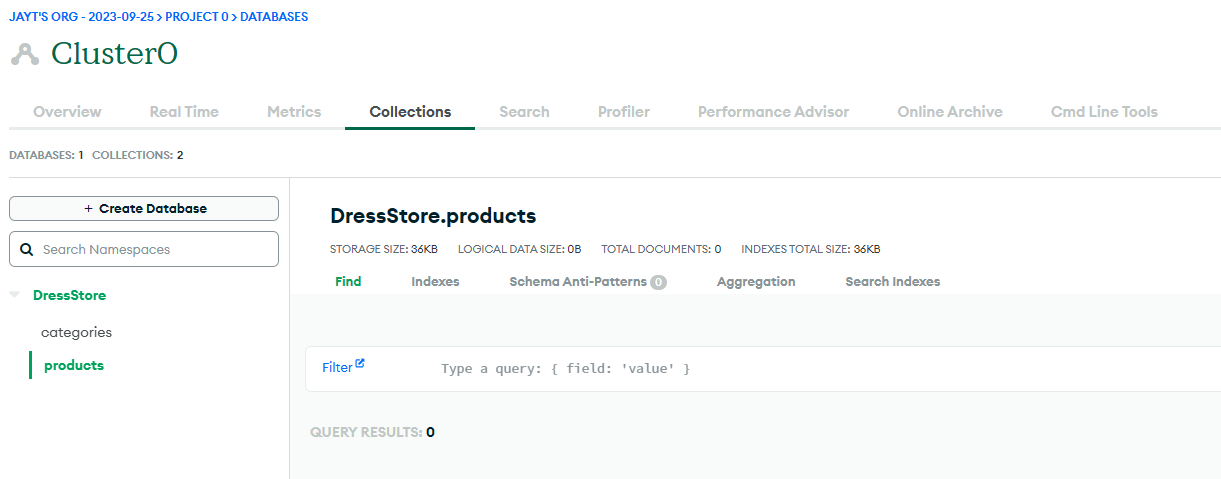
mongoose.connect('mongodb+srv://tuazonjayson23:MyPass1234.@cluster0.e5uc2oj.mongodb.net/DressStore', {

    useNewUrlParser: true,

    useUnifiedTopology: true,

});

**1.b.**



const productSchema = new mongoose.Schema({

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

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

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

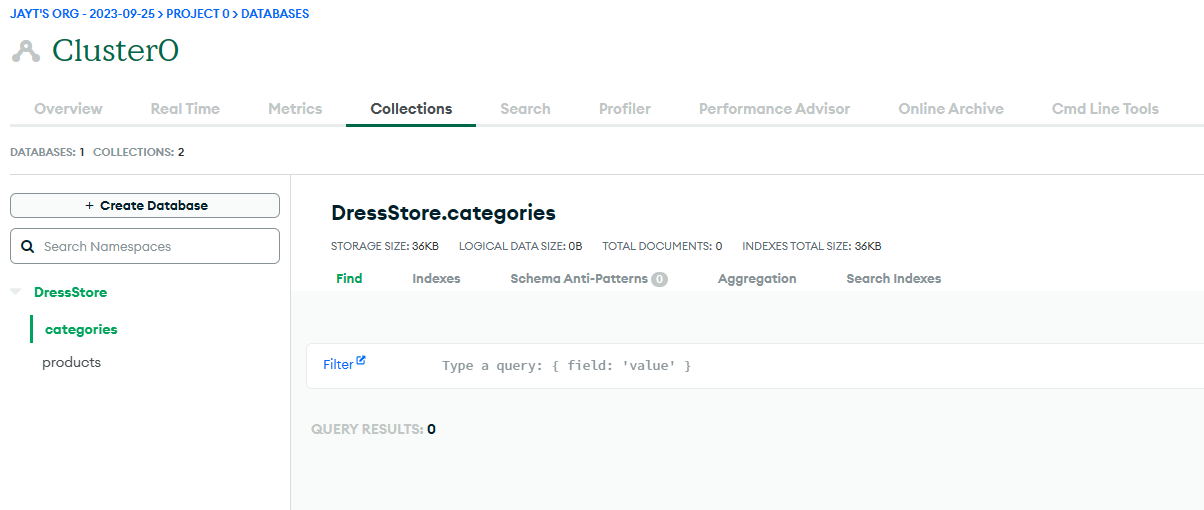
    published: { type: Boolean, required: true },

    category: { type: String, required: true }

});

const Product = mongoose.model('Product', productSchema);

**1.c.**

****

const categorySchema = new mongoose.Schema({

    name: { type: String, required: true }

});

const Category = mongoose.model('Category', categorySchema);

**2.a.**

const express = require('express');

const mongoose = require('mongoose');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(express.json());

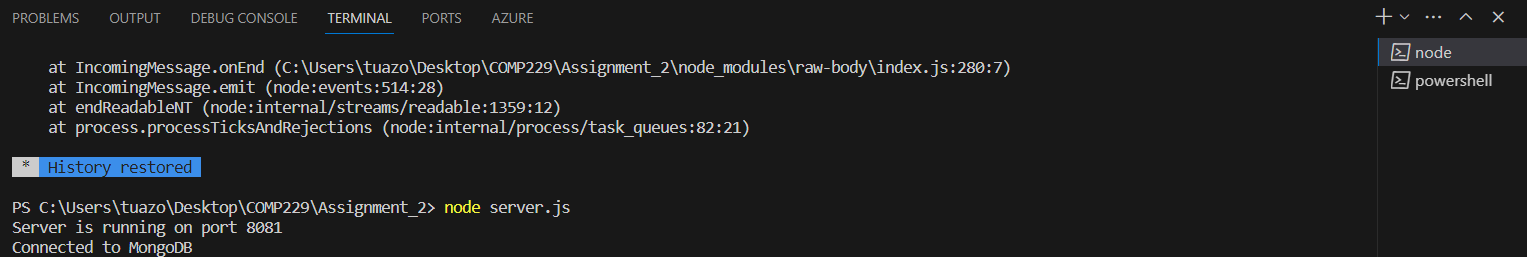
const port = 8081;

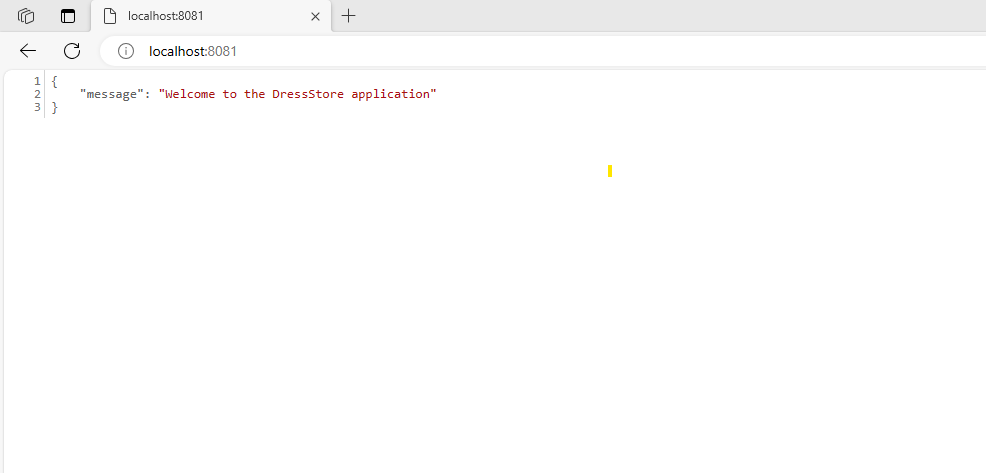
app.listen(port, () => {

    console.log(`Server is running on port ${port}`);

});

**2.b.**

****

****

**3.a.**

const express = require('express');

const { Product, Category } = require('./model/model');

const mongoose = require('mongoose');

const cors = require('cors');

const app = express();

app.use(cors());

app.use(express.json());

const {

    getAllProducts,

    getProductById,

    addProduct,

    updateProduct,

    deleteProduct,

    deleteAllProducts,

    findProductsByName,

    getAllCategories,

    getCategoryById,

    addCategory,

    updateCategory,

    deleteCategory,

    deleteAllCategories

} = require('./controller/controller');

app.get('/api/products', getAllProducts);

app.get('/api/products/:id', getProductById);

app.post('/api/products', addProduct);

app.put('/api/products/:id', updateProduct);

app.delete('/api/products/:id', deleteProduct);

app.delete('/api/products', deleteAllProducts);

app.get('/api/products', findProductsByName);

app.get('/api/categories', getAllCategories);

app.get('/api/categories/:id', getCategoryById);

app.post('/api/categories', addCategory);

app.put('/api/categories/:id', updateCategory);

app.delete('/api/categories/:id', deleteCategory);

app.delete('/api/categories', deleteAllCategories);

app.get('/', (req, res) => {

    res.json({ message: "Welcome to the DressStore application" });

});

mongoose.connect('mongodb+srv://tuazonjayson23:MyPass1234.@cluster0.e5uc2oj.mongodb.net/DressStore', {

    useNewUrlParser: true,

    useUnifiedTopology: true,

});

const db = mongoose.connection;

db.on('error', console.error.bind(console, 'MongoDB connection error:'));

db.once('open', () => {

    console.log('Connected to MongoDB');

});

const port = 8081;

app.listen(port, () => {

    console.log(`Server is running on port ${port}`);

});

**3.b.**

const mongoose = require('mongoose');

const productSchema = new mongoose.Schema({

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

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

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

    published: { type: Boolean, required: true },

    category: { type: String, required: true }

});

const Product = mongoose.model('Product', productSchema);

const categorySchema = new mongoose.Schema({

    name: { type: String, required: true }

});

const Category = mongoose.model('Category', categorySchema);

module.exports = { Product, Category };

**3.c.**

const {Product} = require('../model/model');

const {Category} = require('../model/model');

// Get all products

const getAllProducts = async (req, res) => {

    try {

        const products = await Product.find();

        res.json(products);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Get product by id

const getProductById = async (req, res) => {

    try {

        const product = await Product.findById(req.params.id);

        if (!product) {

            return res.status(404).json({ message: "Product not found" });

        }

        res.json(product);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Add a new product

const addProduct = async (req, res) => {

    const { name, description, price, published, category } = req.body;

    try {

        const product = new Product({

            name,

            description,

            price,

            published,

            category

        });

        const savedProduct = await product.save();

        res.status(201).json(savedProduct);

    } catch (err) {

        res.status(400).json({ error: err.message });

    }

};

// Update a product by id

const updateProduct = async (req, res) => {

    const { name, description, price, published, category } = req.body;

    try {

        const updatedProduct = await Product.findByIdAndUpdate(

            req.params.id,

            {

                name,

                description,

                price,

                published,

                category

            },

            { new: true }

        );

        if (!updatedProduct) {

            return res.status(404).json({ message: "Product not found" });

        }

        res.json(updatedProduct);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Delete a product by id

const deleteProduct = async (req, res) => {

    try {

        const deletedProduct = await Product.findByIdAndRemove(req.params.id);

        if (!deletedProduct) {

            return res.status(404).json({ message: "Product not found" });

        }

        res.json({ message: "Product deleted" });

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Delete all products

const deleteAllProducts = async (req, res) => {

    try {

        await Product.deleteMany({});

        res.json({ message: "All products deleted" });

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Find products by name

const findProductsByName = async (req, res) => {

    const keyword = req.query.name;

    try {

        const products = await Product.find({ name: { $regex: keyword, $options: 'i' } });

        res.json(products);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

const getAllCategories = async (req, res) => {

    try {

        const categories = await Category.find();

        res.json(categories);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Get category by ID

const getCategoryById = async (req, res) => {

    try {

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

        if (!category) {

            return res.status(404).json({ message: "Category not found" });

        }

        res.json(category);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Add a new category

const addCategory = async (req, res) => {

    const { name } = req.body; // Assuming category only has a name

    try {

        const category = new Category({

            name

        });

        const savedCategory = await category.save();

        res.status(201).json(savedCategory);

    } catch (err) {

        res.status(400).json({ error: err.message });

    }

};

// Update a category by ID

const updateCategory = async (req, res) => {

    const { name } = req.body;

    try {

        const updatedCategory = await Category.findByIdAndUpdate(

            req.params.id,

            {

                name

            },

            { new: true }

        );

        if (!updatedCategory) {

            return res.status(404).json({ message: "Category not found" });

        }

        res.json(updatedCategory);

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Delete a category by ID

const deleteCategory = async (req, res) => {

    try {

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

        if (!deletedCategory) {

            return res.status(404).json({ message: "Category not found" });

        }

        res.json({ message: "Category deleted" });

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

// Delete all categories

const deleteAllCategories = async (req, res) => {

    try {

        await Category.deleteMany({});

        res.json({ message: "All categories deleted" });

    } catch (err) {

        res.status(500).json({ error: err.message });

    }

};

module.exports = {

    getAllProducts,

    getProductById,

    addProduct,

    updateProduct,

    deleteProduct,

    deleteAllProducts,

    findProductsByName,

    getAllCategories,

    getCategoryById,

    addCategory,

    updateCategory,

    deleteCategory,

    deleteAllCategories,

};

**3.d.**

app.get('/api/products', getAllProducts);

app.get('/api/products/:id', getProductById);

app.post('/api/products', addProduct);

app.put('/api/products/:id', updateProduct);

app.delete('/api/products/:id', deleteProduct);

app.delete('/api/products', deleteAllProducts);

app.get('/api/products', findProductsByName);

app.get('/api/categories', getAllCategories);

app.get('/api/categories/:id', getCategoryById);

app.post('/api/categories', addCategory);

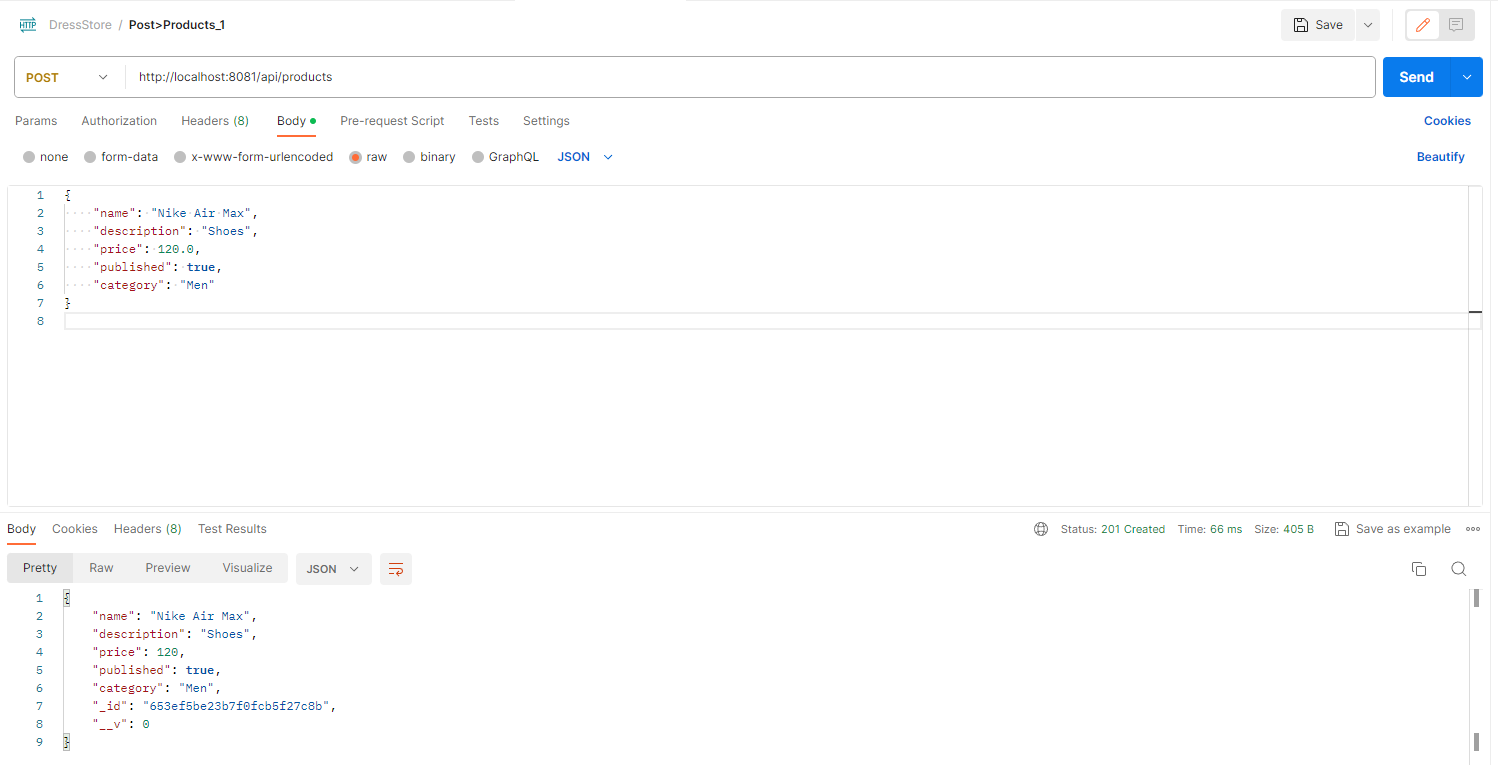
app.put('/api/categories/:id', updateCategory);

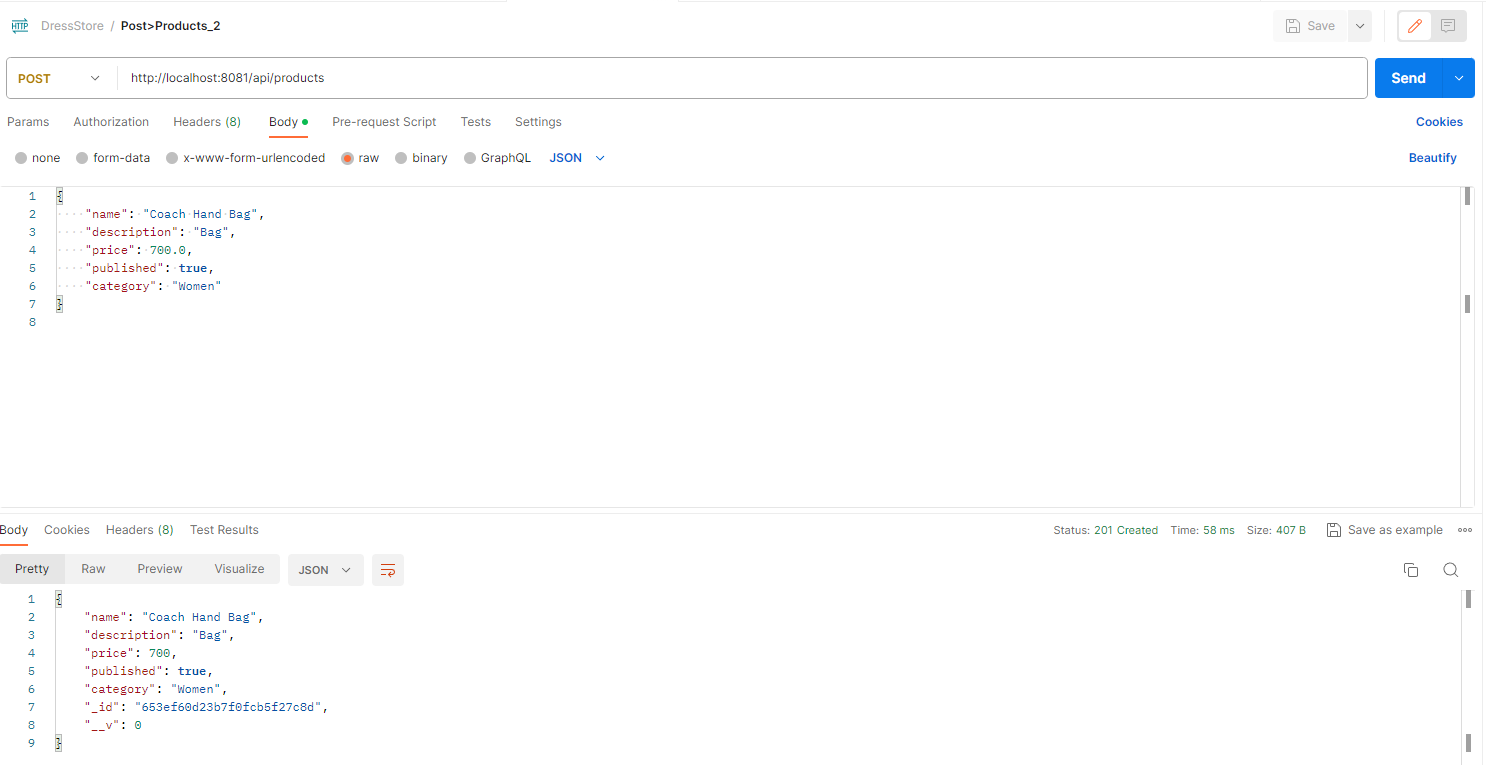
app.delete('/api/categories/:id', deleteCategory);

app.delete('/api/categories', deleteAllCategories);

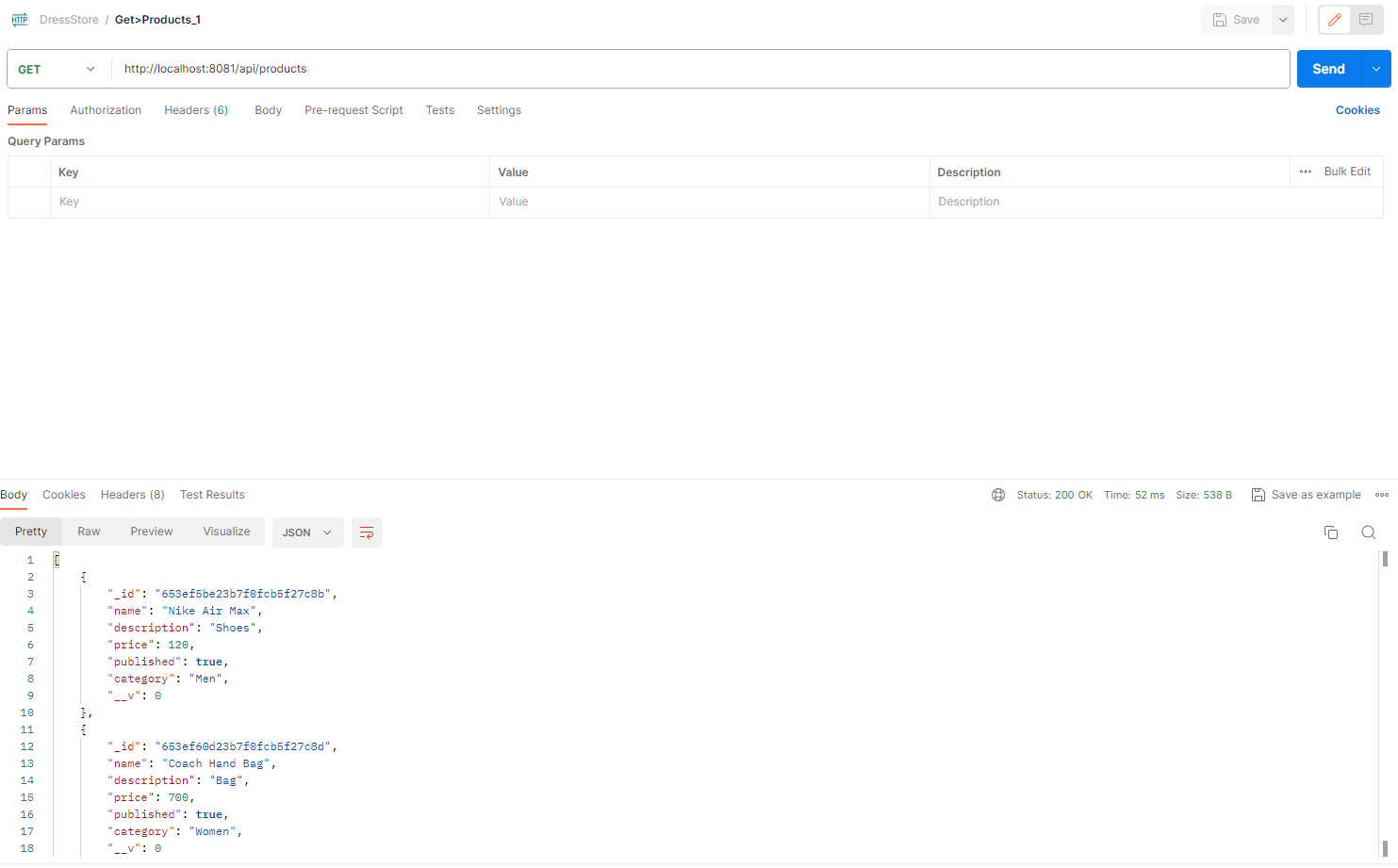
**4.**

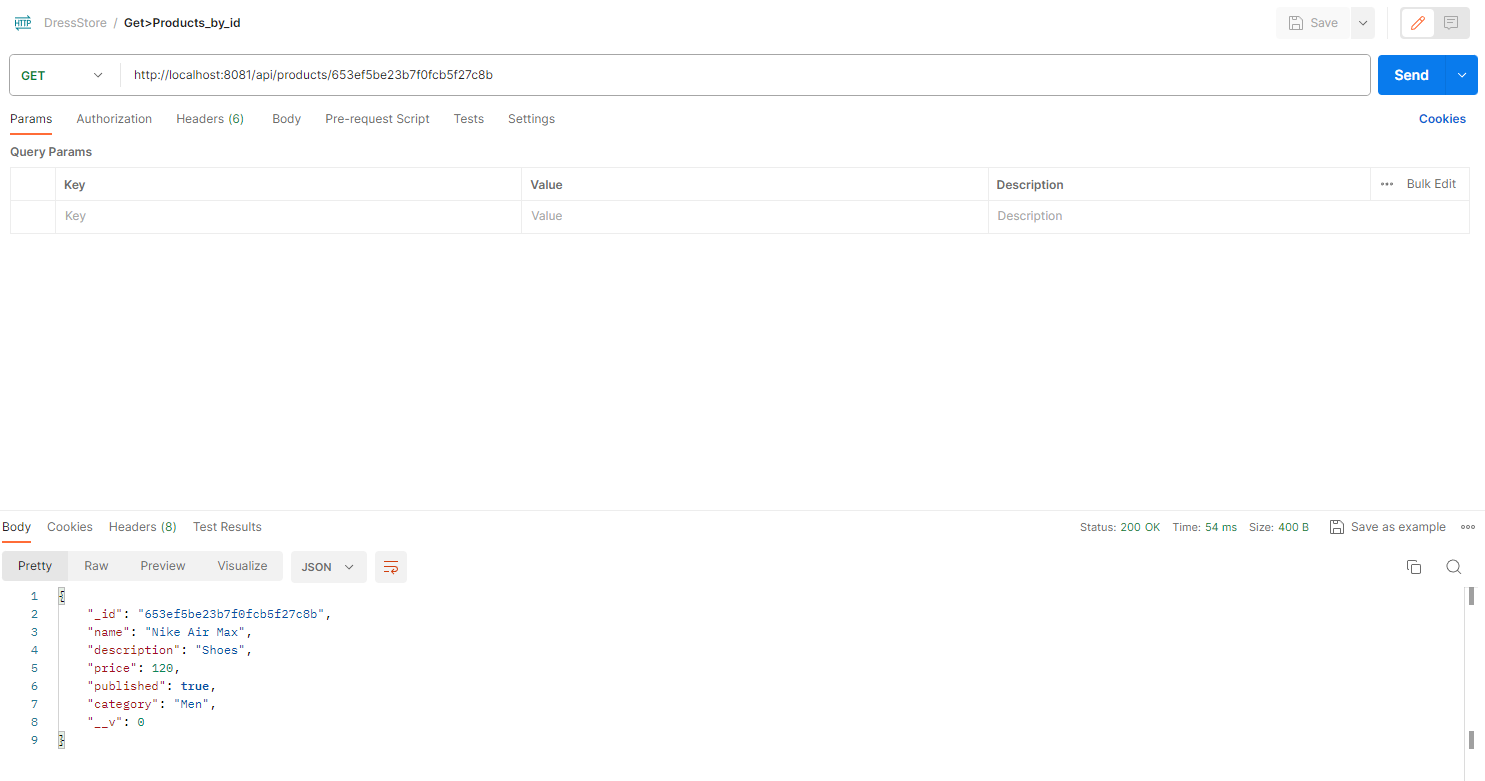
**POST**

****

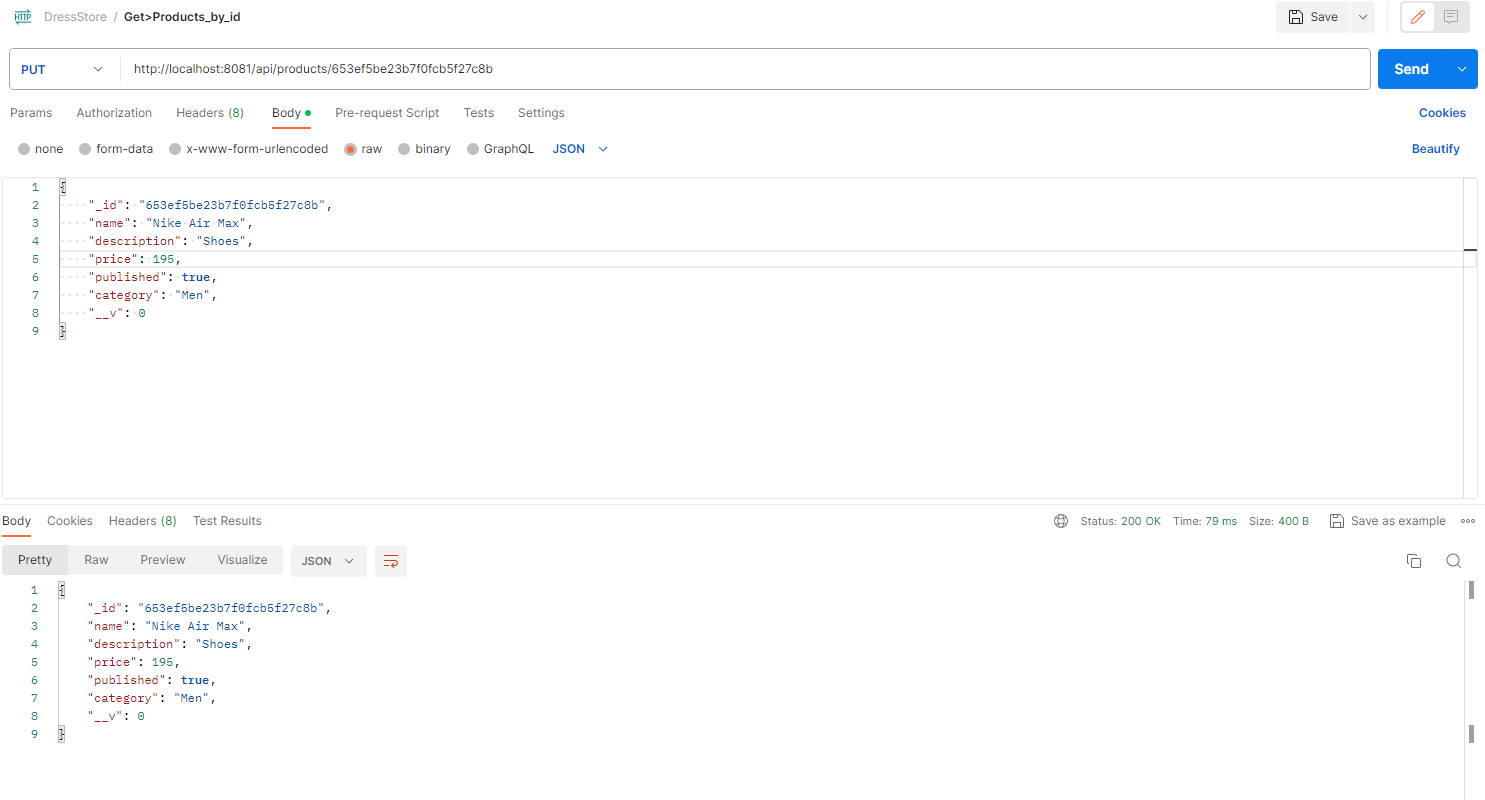
****

**GET**

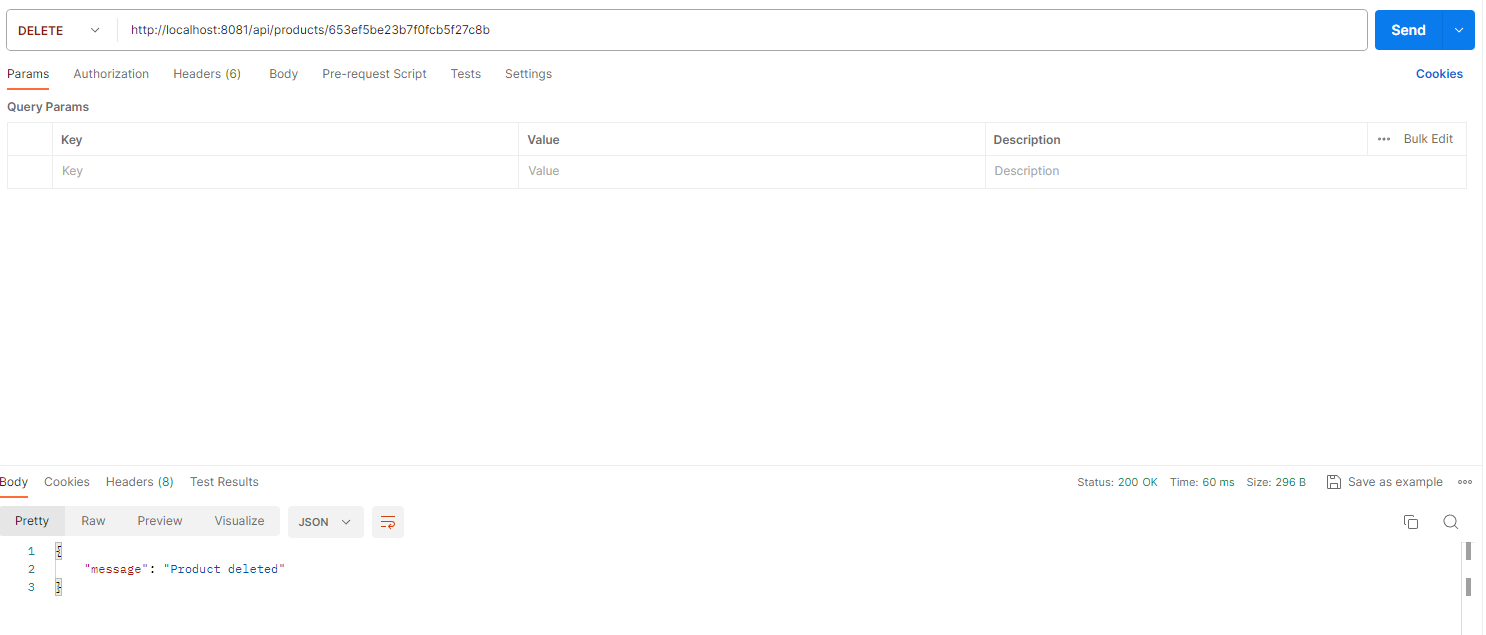
****

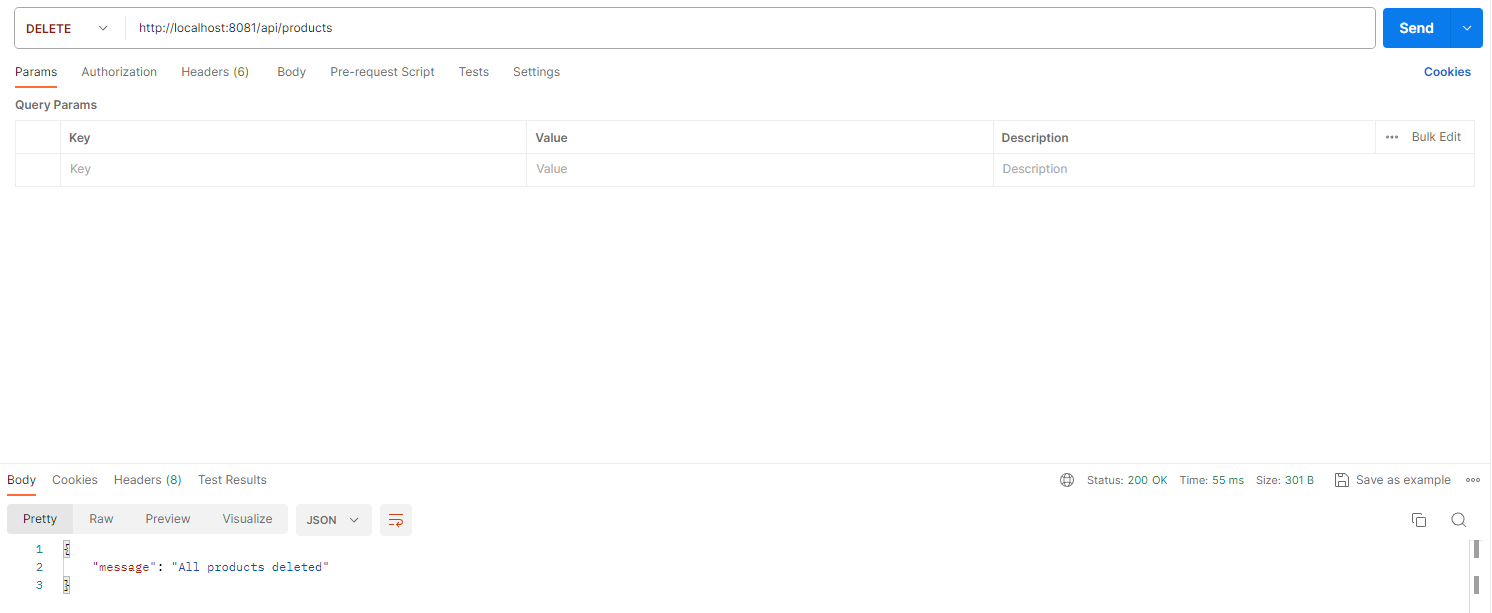
****

**PUT**

****

**DELETE**

****

****