**Assignment #2**

**Online Market Application**

**Node.js, Express REST APIs & MongoDB**

**CRUD**

Course: COMP 229 "Web Application Development"

Professor: Ronaldo Felipe

Student: Elena Ogneva

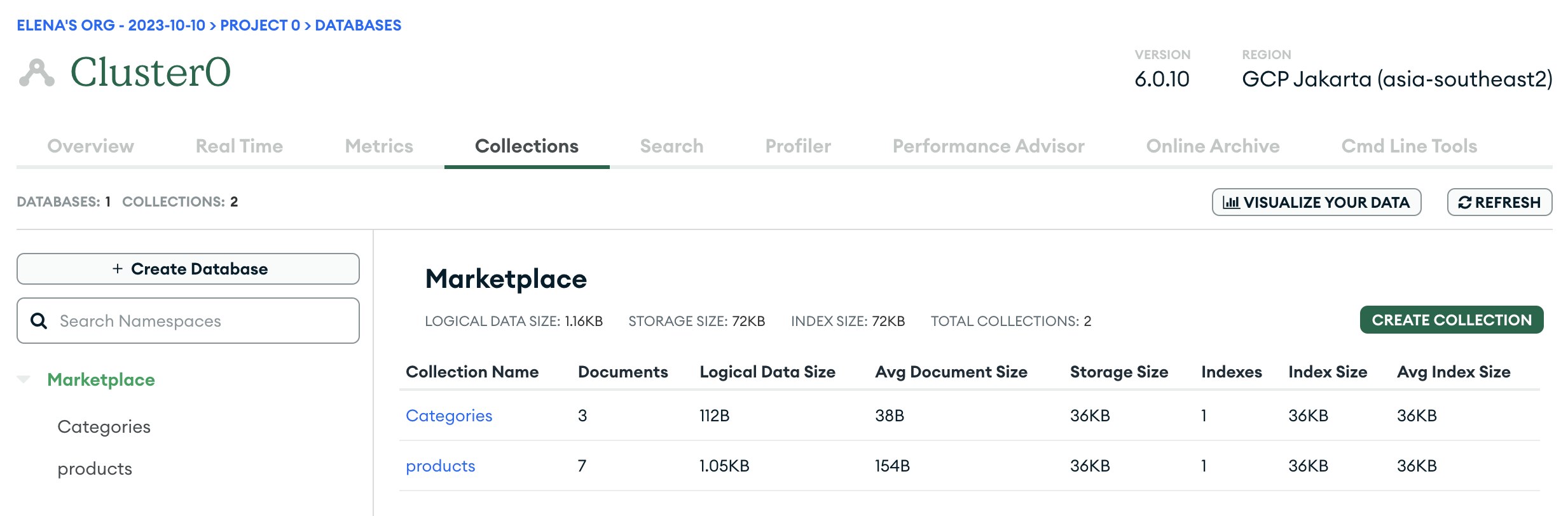
**Centennial College**

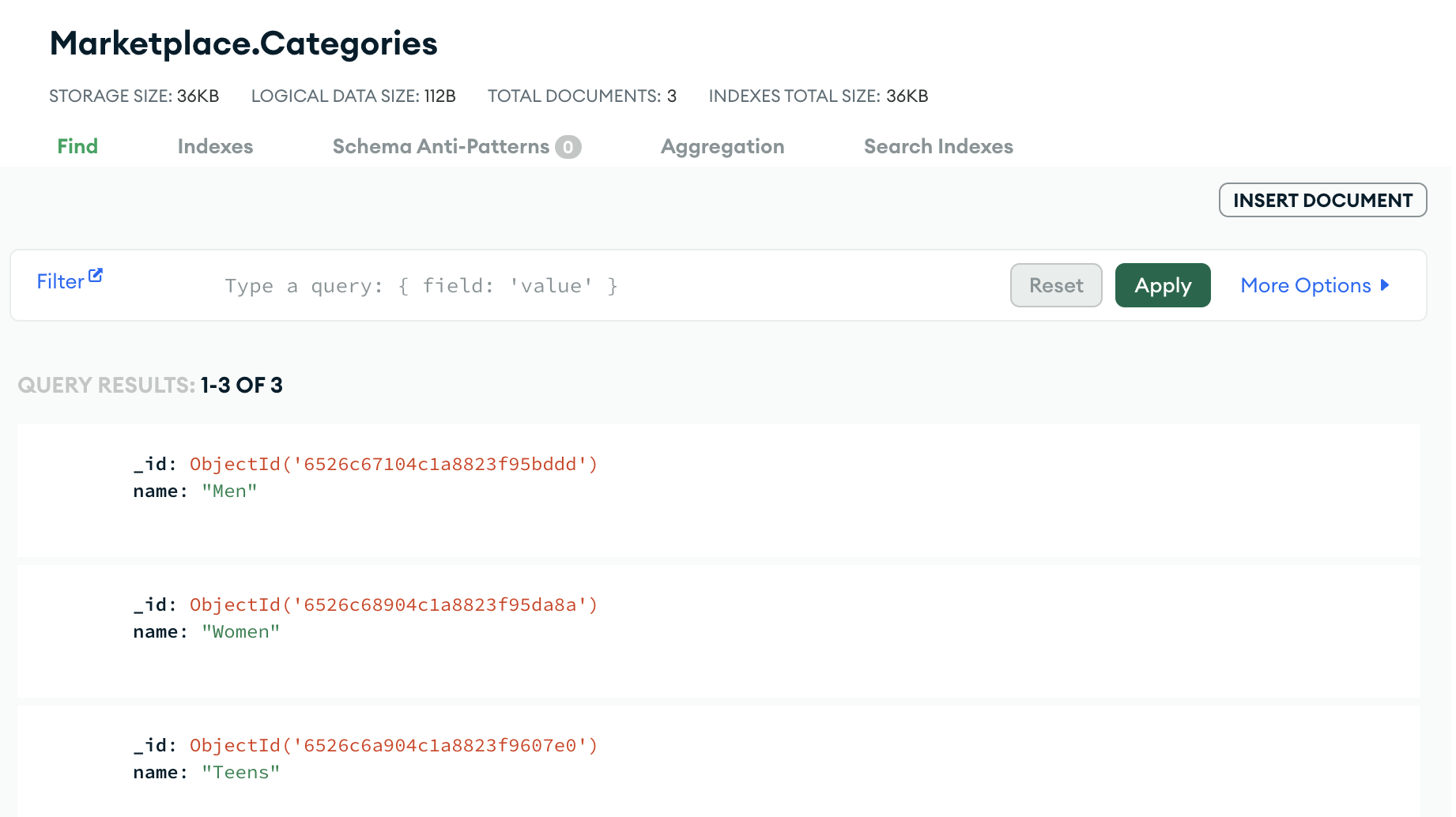
**2023**

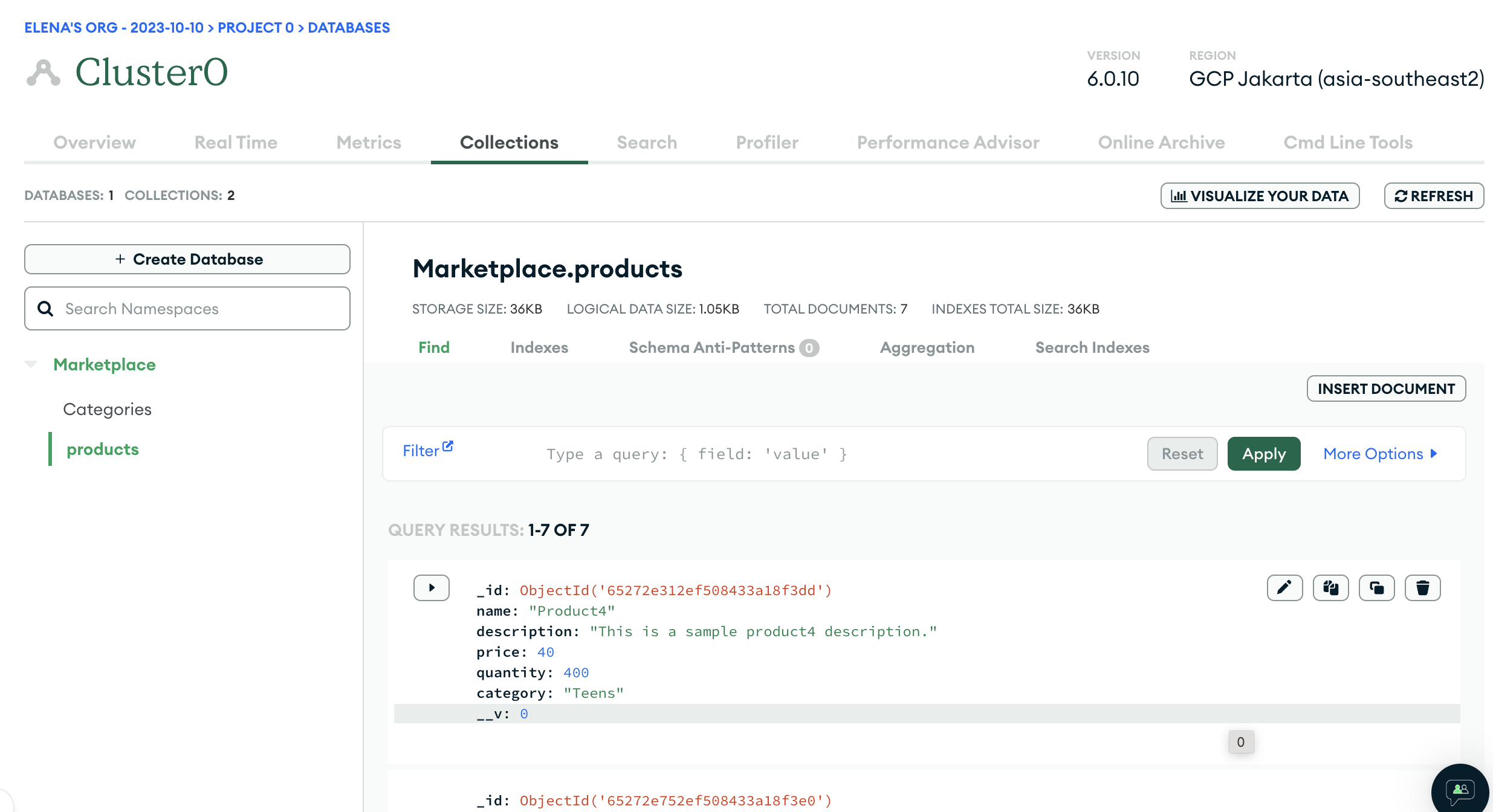
# Repository

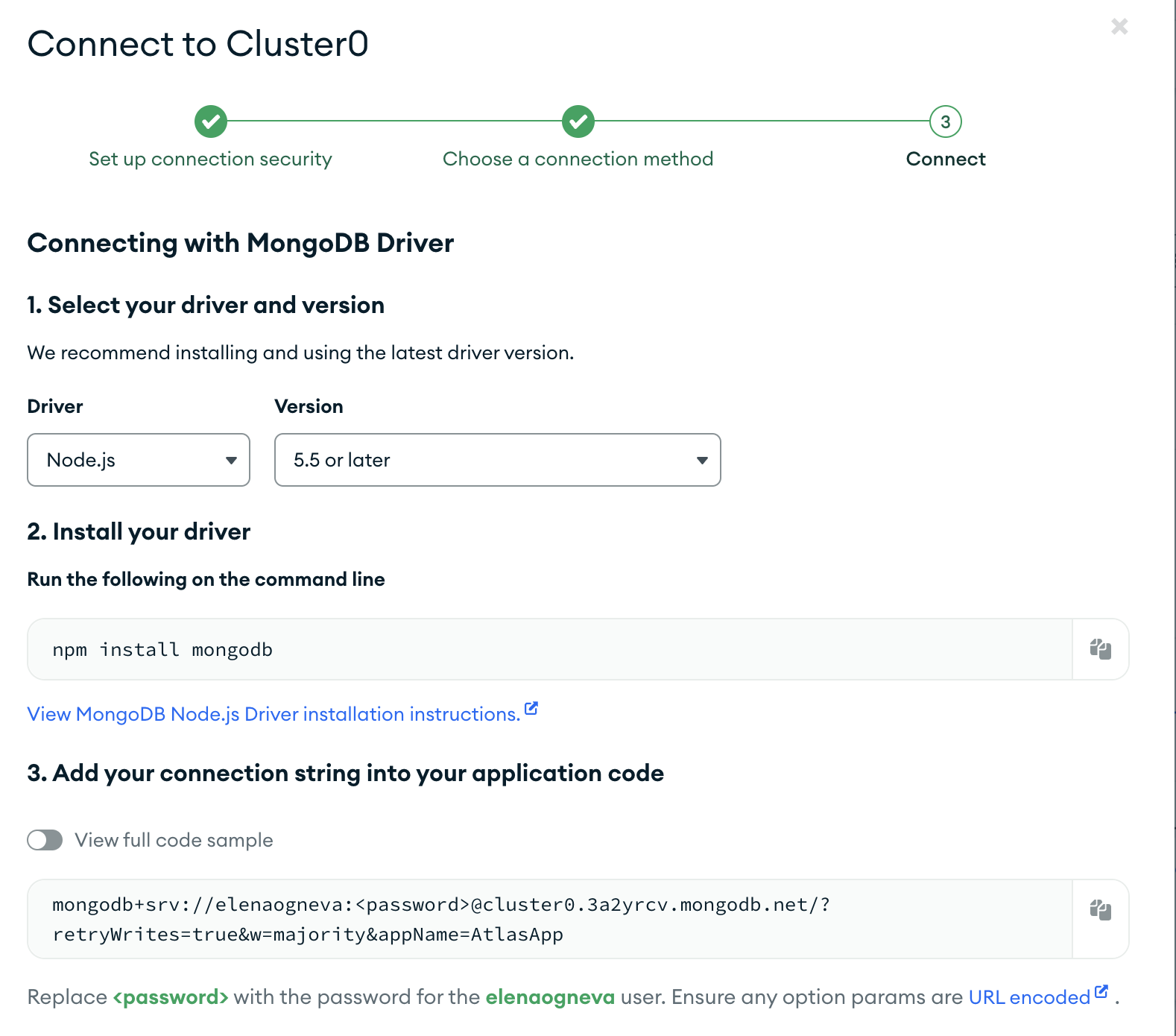
https://github.com/eogneva1/Assignment2\_CRUD

# Creating Database









# Creating mongo.env

MONGO\_URI="mongodb+srv://elenaogneva:G8fFJBluIFvyiHDV@cluster0.3a2yrcv.mongodb.net/Marketplace?retryWrites=true&w=majority&appName=AtlasApp"

# Changing App.js

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

var cors = require('cors');

const mongoose = require('mongoose');

// Load environment variables from the mongo.env file

require('dotenv').config({ path: './mongo.env' });

// Display the connection string for debugging

console.log("Attempting to connect using URI:", process.env.MONGO\_URI);

// Connect to MongoDB

mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => {

console.log('Connected to MongoDB');

})

.catch(err => {

console.error('Failed to connect to MongoDB', err);

});

var indexRouter = require('./routes/index');

var app = express();

// Middleware

app.use(cors());

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

// Routes

app.use('/', indexRouter);

// Catch 404 and forward to error handler

app.use(function(req, res, next) {

next(createError(404));

});

// Error handler

app.use(function(err, req, res, next) {

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

# Creating models/Products.js

const mongoose = require('mongoose');

const ProductSchema = new mongoose.Schema({

name: String,

description: String,

price: Number,

quantity: Number,

category: String

});

module.exports = mongoose.model('Product', ProductSchema);

# Creating controllers/productController.js

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

// Get all products

exports.getAllProducts = async (req, res) => {

try {

const products = await Product.find();

res.json(products);

} catch (error) {

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

}

};

// Create a new product

exports.createProduct = async (req, res) => {

const product = new Product({

name: req.body.name,

description: req.body.description,

price: req.body.price,

quantity: req.body.quantity,

category: req.body.category

});

try {

const newProduct = await product.save();

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

} catch (error) {

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

}

};

// Get a single product by ID

exports.getProductById = async (req, res) => {

try {

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

if (product == null) {

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

}

res.json(product);

} catch (error) {

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

}

};

// Update a product by ID

exports.updateProduct = async (req, res) => {

try {

const product = await Product.findByIdAndUpdate(req.params.id, req.body, { new: true });

if (product == null) {

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

}

res.json(product);

} catch (error) {

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

}

};

// Delete a product by ID

exports.deleteProduct = async (req, res) => {

try {

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

if (product == null) {

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

}

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

} catch (error) {

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

}

};

// Remove all products

exports.deleteAllProducts = async (req, res) => {

try {

await Product.deleteMany();

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

} catch (error) {

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

}

};

// Find products by name contains

exports.findProductsByName = async (req, res) => {

try {

const name = req.query.name; // Use req.query to get the 'name' parameter from the URL query

if (!name) {

return res.status(400).json({ message: 'Name parameter is missing' });

}

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

if (products.length === 0) {

return res.status(404).json({ message: 'No products found' });

}

res.json(products);

} catch (error) {

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

}

};

# Changing api.js

const express = require('express');

const router = express.Router();

const productController = require('../controllers/productController');

// CRUD Routes for Products

router.get('/products', productController.getAllProducts); // Get all products

router.get('/products/:id', productController.getProductById); // Get a single product by ID

router.post('/products', productController.createProduct); // Create a new product

router.put('/products/:id', productController.updateProduct); // Update a product by ID

router.delete('/products/:id', productController.deleteProduct); // Delete a product by ID

router.delete('/products', productController.deleteAllProducts); // Remove all products

router.get('/products', productController.findProductsByName); // Find products by name

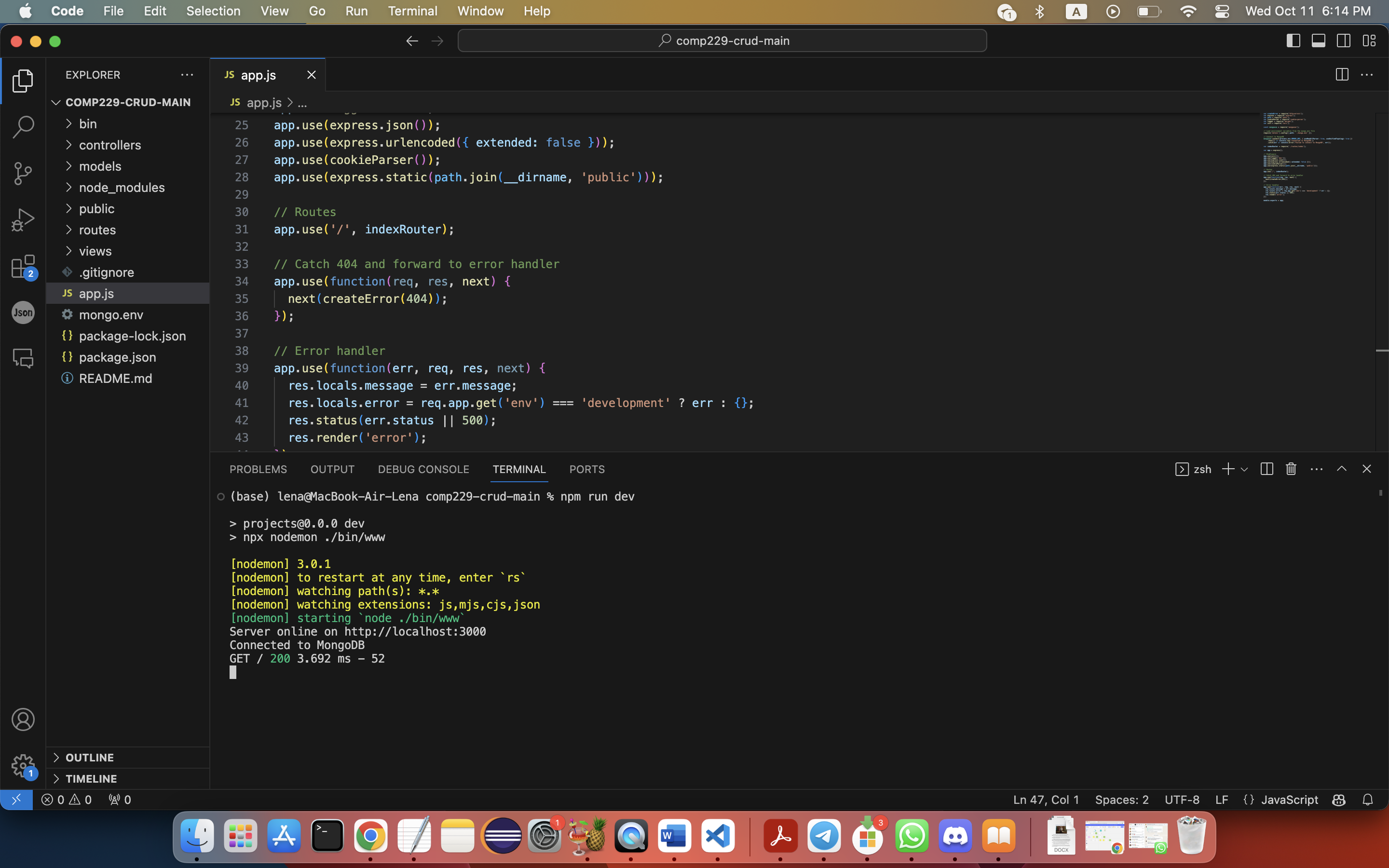
// Users routes

var usersRouter = require('./users');

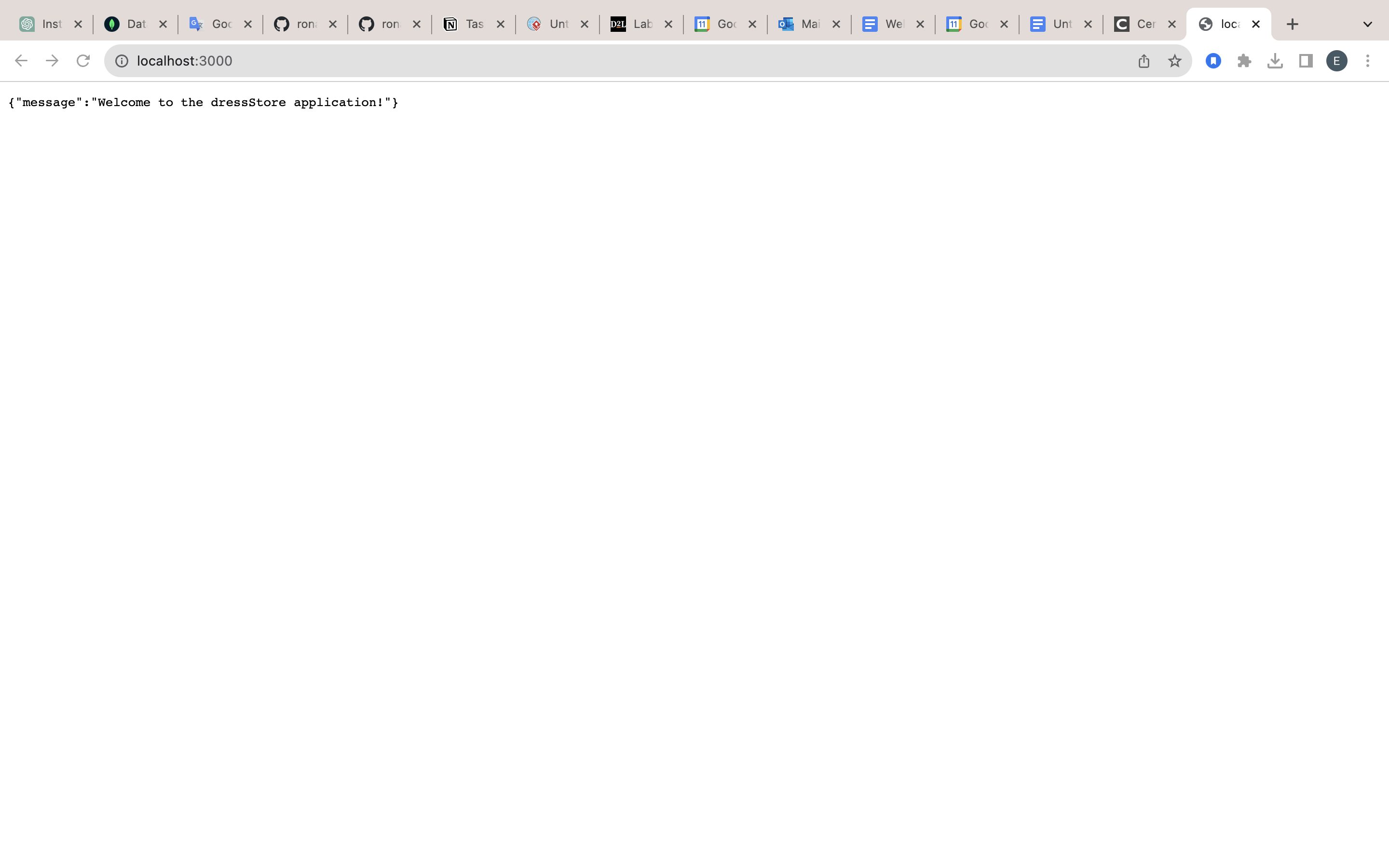
router.use('/users', usersRouter);

module.exports = router;

1. Running the project

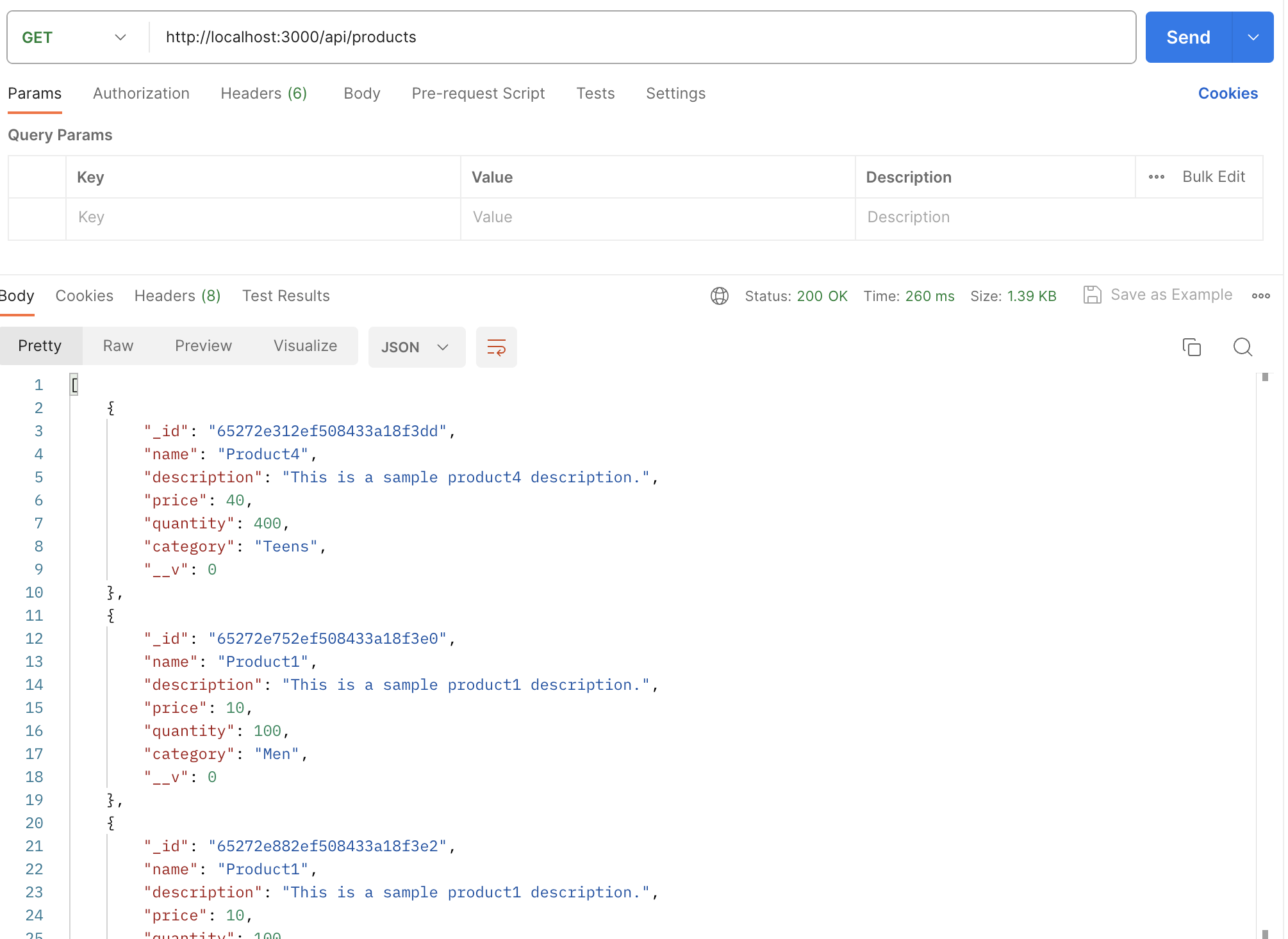


1. Result



# Testing the REST APIs using Postman

## **Get all products**



Response JSON:  
[

{

"\_id": "65272e312ef508433a18f3dd",

"name": "Product4",

"description": "This is a sample product4 description.",

"price": 40,

"quantity": 400,

"category": "Teens",

"\_\_v": 0

},

{

"\_id": "65272e752ef508433a18f3e0",

"name": "Product1",

"description": "This is a sample product1 description.",

"price": 10,

"quantity": 100,

"category": "Men",

"\_\_v": 0

},

{

"\_id": "65272e882ef508433a18f3e2",

"name": "Product1",

"description": "This is a sample product1 description.",

"price": 10,

"quantity": 100,

"category": "Men",

"\_\_v": 0

},

{

"\_id": "65272fc32ef508433a18f3e6",

"name": "Product kwa-kwa",

"description": "This is a sample product kwa-kwa description.",

"price": 220,

"quantity": 2200,

"category": "Women",

"\_\_v": 0

},

{

"\_id": "65272fd22ef508433a18f3e8",

"name": "Product 55kwkw",

"description": "This is a sample product 55kwkwa description.",

"price": 230,

"quantity": 3500,

"category": "Men",

"\_\_v": 0

},

{

"\_id": "65272ffe2ef508433a18f3ea",

"name": "Product2",

"description": "This is a sample product1 description.",

"price": 140,

"quantity": 1400,

"category": "Men",

"\_\_v": 0

},

{

"\_id": "6527304c2ef508433a18f3ec",

"name": "Product3",

"description": "This is a sample product1 description.",

"price": 60,

"quantity": 600,

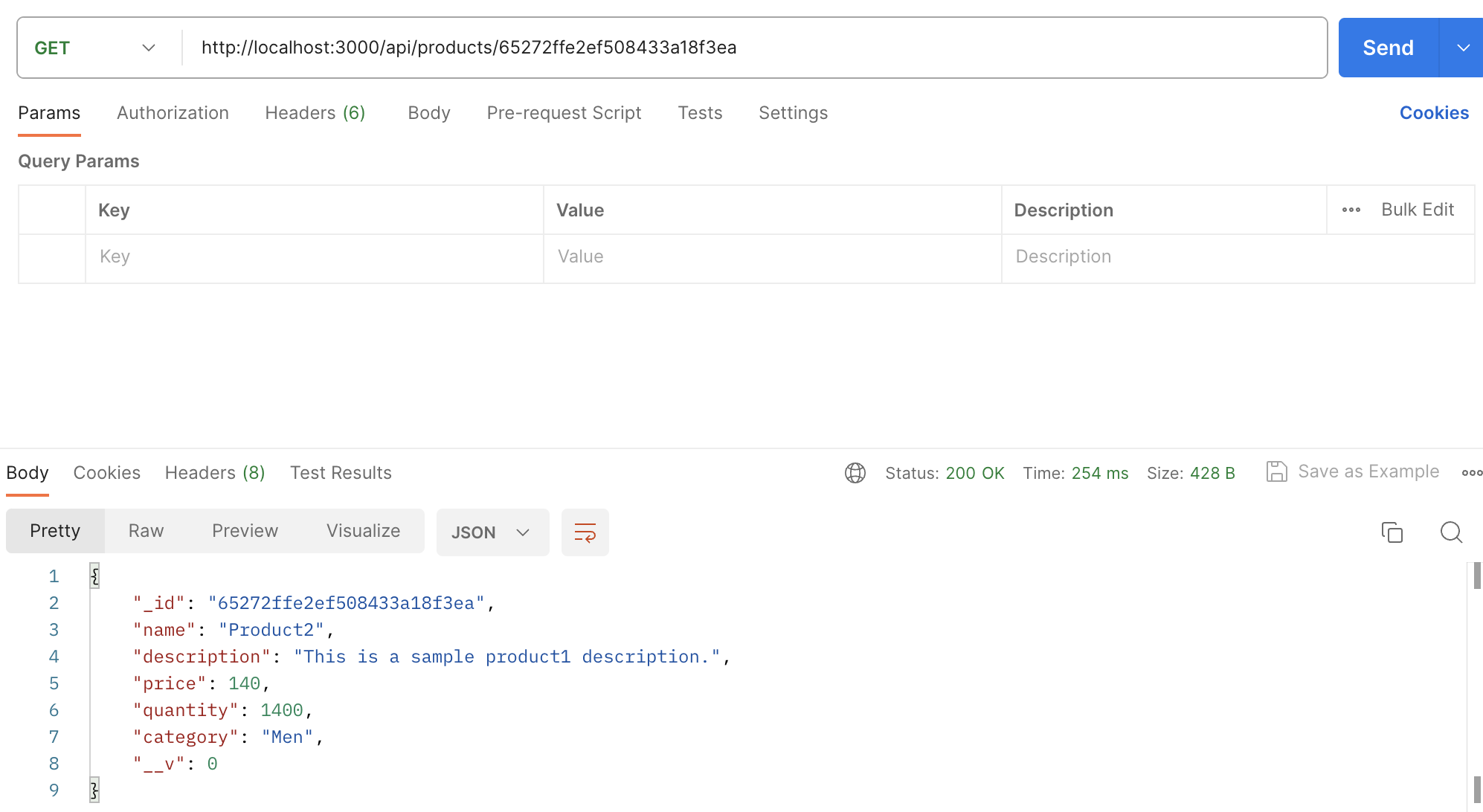
"category": "Teens",

"\_\_v": 0

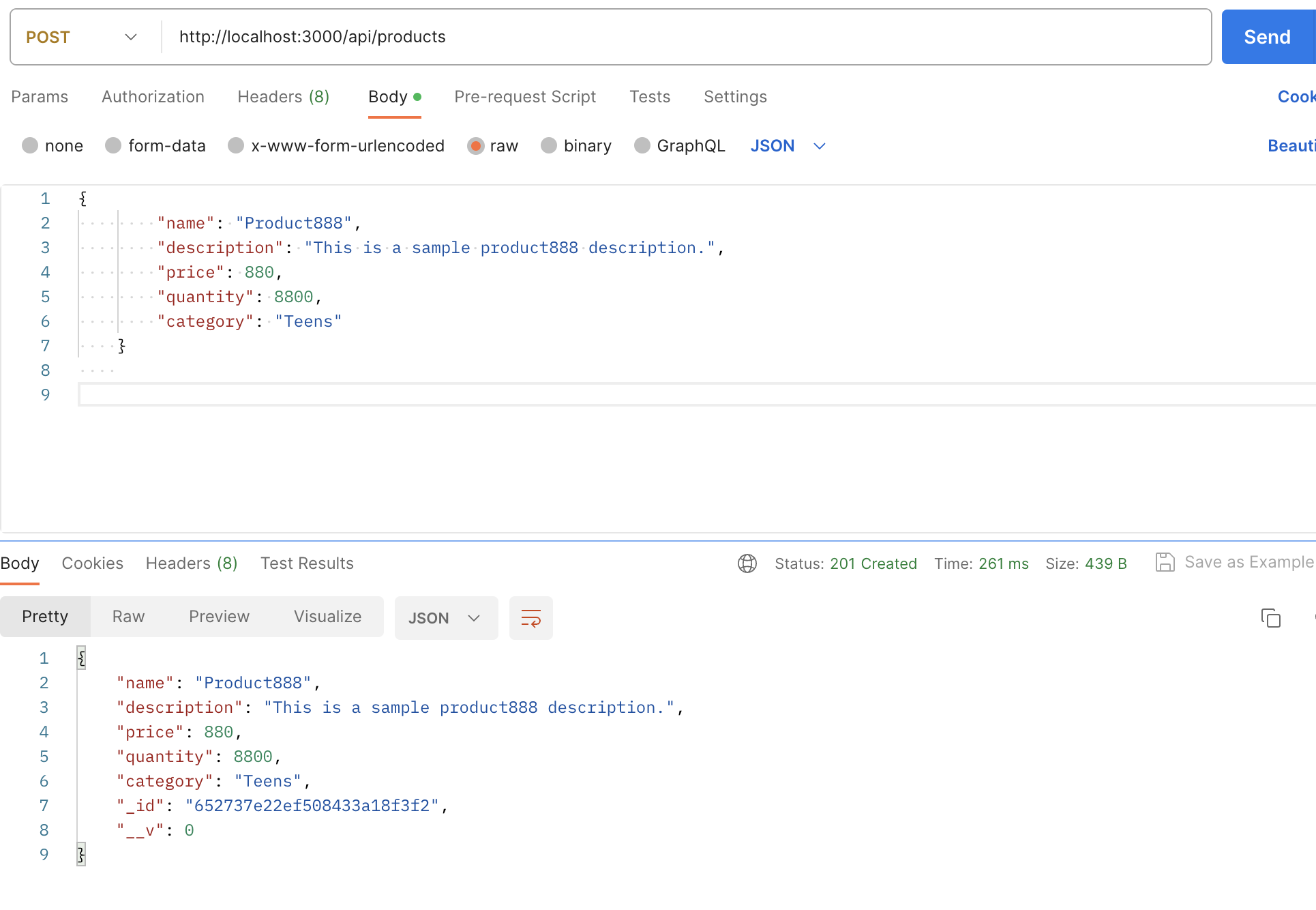
}

]

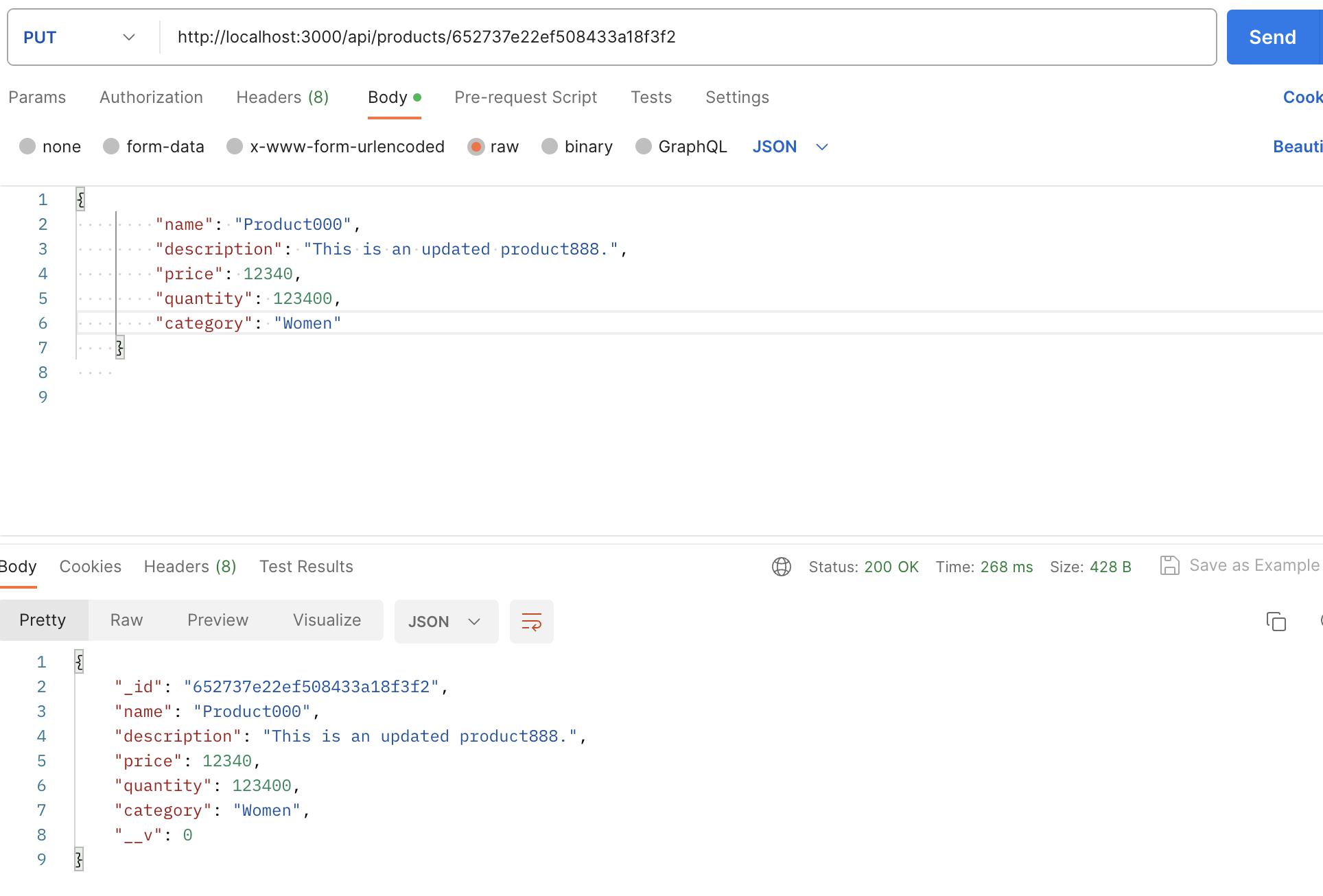
## **GET product by id**



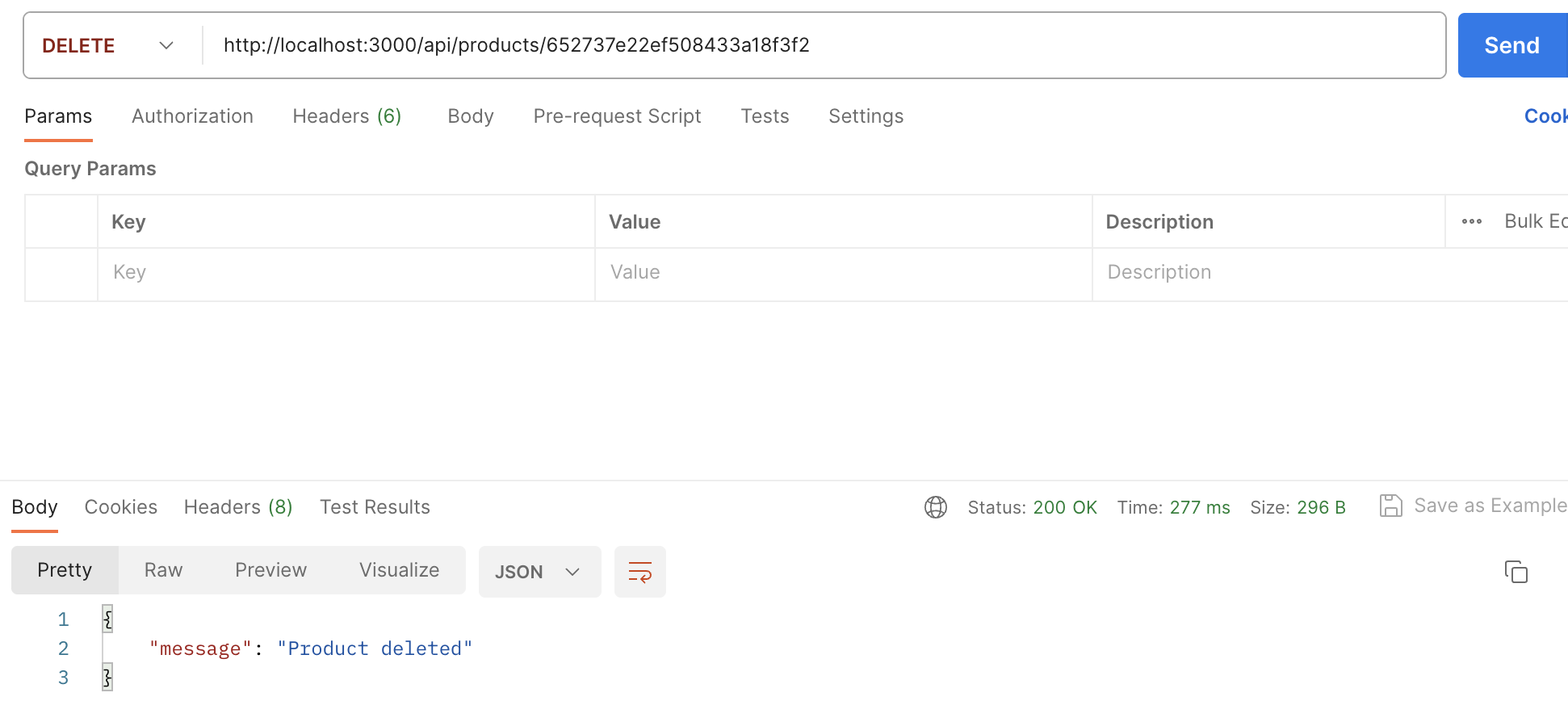
## **Add new Product**



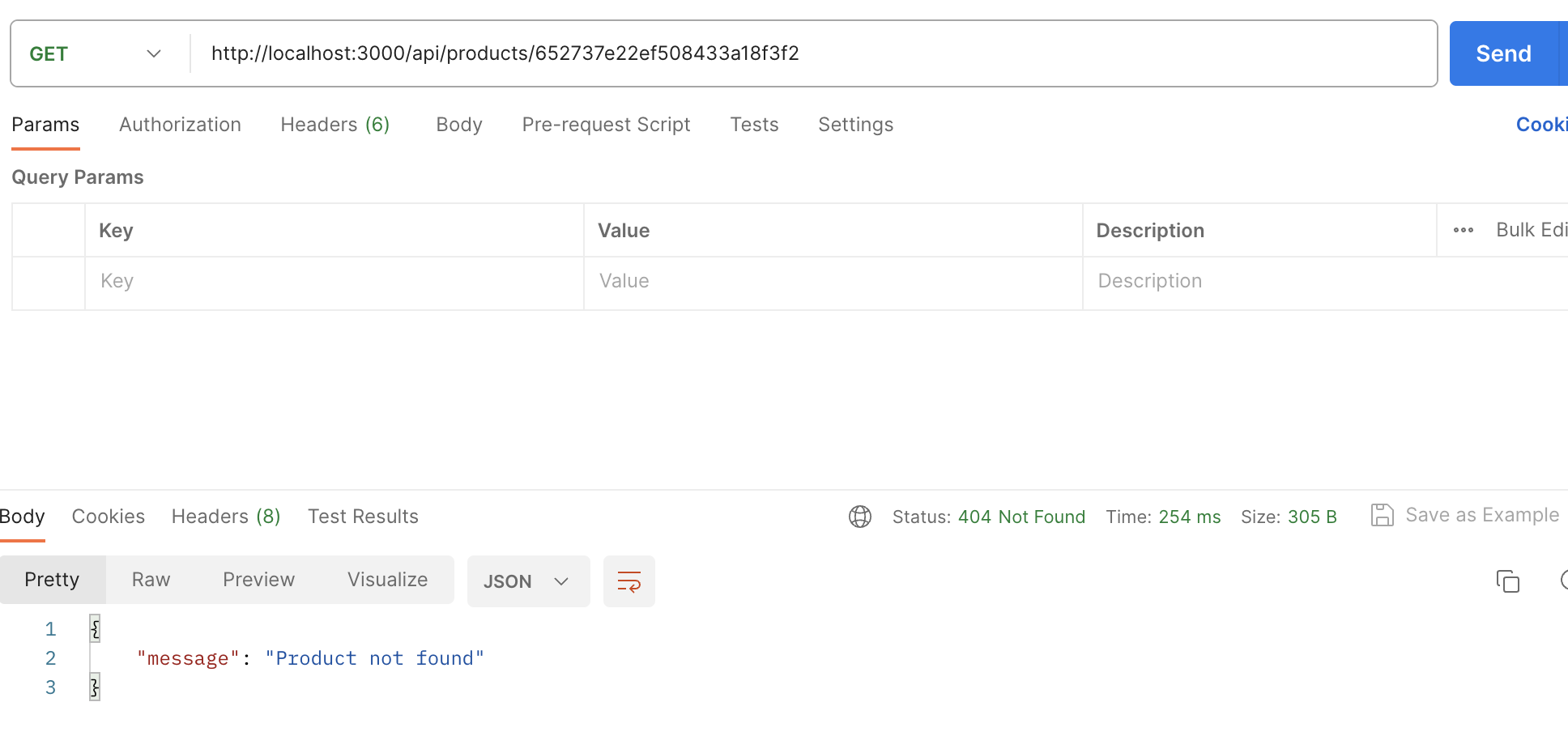
## **Update product by id**



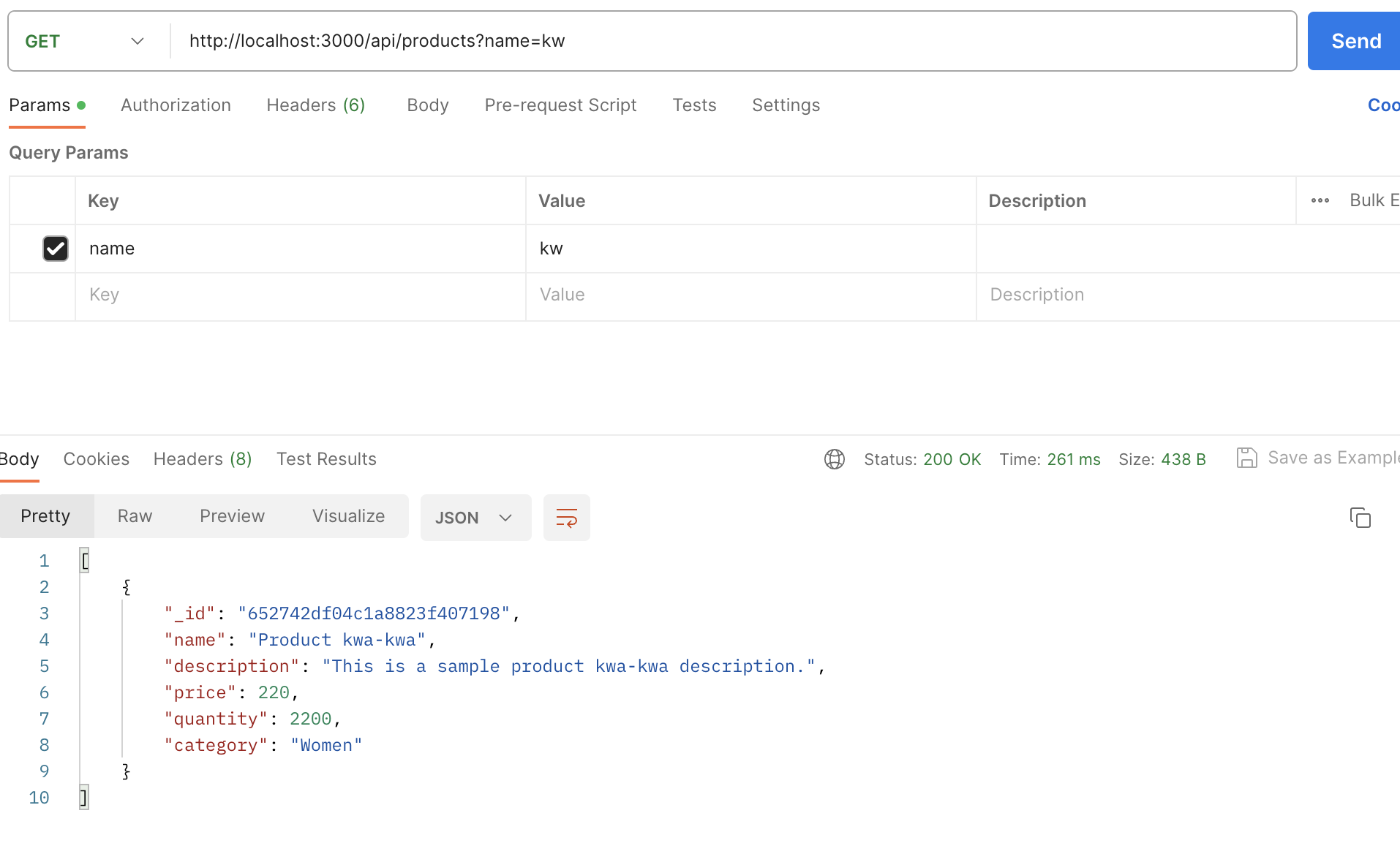
## **Remove product by id**



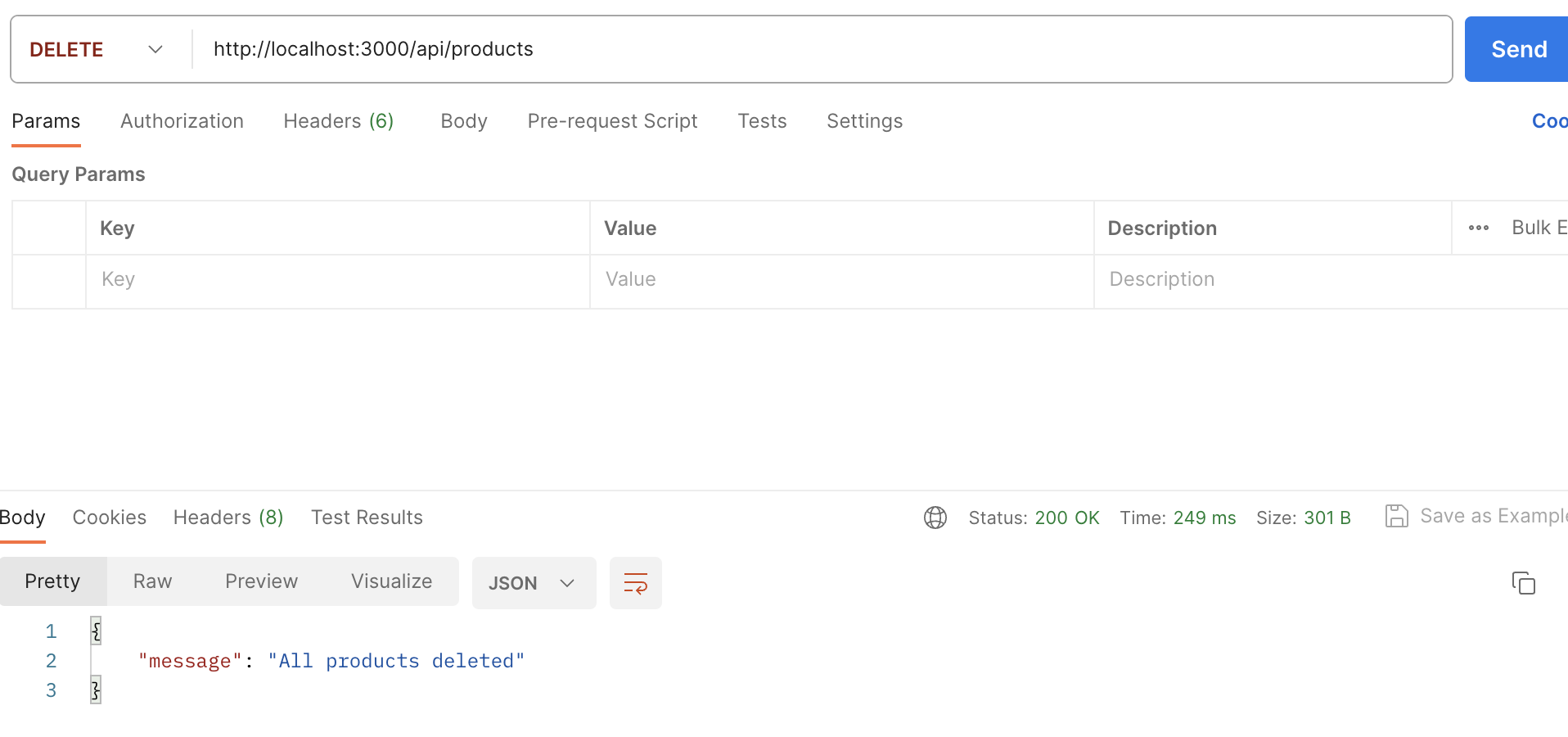
### **Check if it was deleted:**



## **Find all products which name contains 'kw'**



## **Remove all products**



### **Check if all products were deleted:**

