**COMERCIO VERDE - SPRINT 3**

**Integrantes:**

Wilmer Yamid Orjuela Ortigoza

Edinson Fabian Barba Avila

Paula Katherine Feged Oviedo

Hernando Galeano González

Gina Paola Peña Gama

Manuel Andrey León Hernandez

**Objetivo general**

Configurar la estructura backend del proyecto Comercio Verde.

**Objetivos generales.**

* Elaborar diagrama relacional de la base de datos.
* Configurar la estructura básica del Backend.
* Implementar los Modelos y Controladores en Backend.
* Crear ruta CRUD.
* Realiza pruebas de las rutas usando Insomnia o Postman.

1. **Diagrama relacional de base de datos.**

El diagrama relacional de la base de datos se encuentra en los anexos a este documento.

1. **Configuración Backend**

**Configuración DataBase:**

const mongoose = require('mongoose');

const dbConnection = async () =>{

await mongoose.connect('mongodb://localhost:27017/ComercioVerde')

.then(() => console.log("Connect mongoDB ready"))

.catch(err => {

console.log(`Error connect mongoDB: ${ err }`);

throw new Error('Error al conectar la DB');

});

}

module.exports = {

dbConnection

}

**Configuración Helpers:**

const Categoria = require('../models/categoria');

const Artesania = require('../models/artesania');

const categoriaExist = async (categoria = '') =>{

let cate = await Categoria.findOne( { categoria } );

if(!cate){

throw new Error(`la categoria ${ categoria } no existe`);

}

}

const productoByIdExists = async (id = '') => {

let producto = await Artesania.findById(id);

if(!producto){

throw new Error(`El id ${ id } no es una artesania`);

}

}

module.exports={

categoriaExist,

productoByIdExists

}

**Configuración Middlewares:**

const { validationResult } = require('express-validator');

const validateDate = (req, res, next) => {

const err = validationResult(req);

if(!err.isEmpty()){

return res.status(400).json(err);

}

next();

}

module.exports = {

validateDate

}

**Configuración Server:**

const express = require('express');

const cors = require('cors');

const { dbConnection } = require('../database/config')

class Server {

constructor() {

this.app = express();

this.port = 3000;

this.artesaniaPath = '/api/artesanias';

this.alimentoPath = '/api/alimentos';

this.productPath = '/api/products';

this.initDB();

this.middlewares();

this.routes();

}

async initDB() {

await dbConnection();

}

middlewares() {

//Funciones intermedias entre la solicitud del request y el controlador

this.app.use(cors())

this.app.use(express.json());

this.app.use(express.static('public'));

}

routes() {

this.app.use(this.artesaniaPath, require('../routes/artesanias'));

this.app.use(this.alimentoPath, require('../routes/alimentos'));

this.app.use(this.productPath, require('../routes/products'));

}

listen() {

this.app.listen(this.port, () => {

console.log(`Example app listening at http://localhost:${ this.port }`)

});

}

}

module.exports = Server;

1. **Modelos y controladores Backend.**

**3.1 Modelos:**

**Modelo “Categoría”:**

const { Schema, model } = require('mongoose')

const categoriaSchema = Schema({

categoria: {

type: String

}

});

module.exports = model('Categoria', categoriaSchema);

**Modelo Alimentos:**

const { Schema, model } = require('mongoose')

const alimentoSchema = Schema({

categoria: {

type: String,

required: [true, '\*\*\*\* Db: La categoria es requerida']

},

nombre: {

type: String,

required: [true, '\*\*\*\* Db: El nombre es requerido'],

unique: true

},

precio: {

type: Number,

required: [true, '\*\*\*\* Db: El precio es requerido']

},

img: {

type: String

},

stock: {

type: Number,

required: true

},

dcto: {

type: Number,

default: 0

},

descripcion: {

type: String

}

});

module.exports = model('Alimento', alimentoSchema);

**Modelo Artesanías:**

const mongoose = require('mongoose');

const artesaniaSchema = mongoose.Schema({

categoria: {

type: String,

required: [true, '\*\*\*\* Db: La categoria es requerida']

},

nombre: {

type: String,

required: [true, '\*\*\*\* Db: El nombre es requerido'],

unique: true

},

precio: {

type: Number,

required: [true, '\*\*\*\* Db: El precio es requerido']

},

img: {

type: String,

default: ''

},

stock: {

type: Number,

required: true

},

dcto: {

type: Number,

default: 0

},

descripcion:{

type: String

},

comentarios:[{

type: mongoose.Schema.Types.ObjectId,

ref: 'comentario'

}]

});

module.exports = mongoose.model('Artesania', artesaniaSchema);

**Modelo “Productos”:**

const { Schema, model } = require('mongoose')

const productSchema = Schema({

categoria: {

type: String,

required: [true, '\*\*\*\* Db: La categoria es requerida']

},

nombre: {

type: String,

required: [true, '\*\*\*\* Db: El nombre es requerido'],

unique: true

},

precio: {

type: Number,

required: [true, '\*\*\*\* Db: El precio es requerido']

},

img: {

type: String

},

stock: {

type: Number,

required: true

},

dcto: {

type: Number,

default: 0

},

descripcion: {

type: String

}

});

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

**3.2 Controladores:**

**Controladores “Alimentos”:**

const Alimento = require('../models/alimento')

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

const alimentos = await Alimento.find({ categoria: 'alimento' })

res.json(alimentos);

}

const alimentoPost = (req, res) => {

const { name, author } = req.body;

res.json({ msg: "Post alimento controller", name, author });

}

// const alimentosGetBySKU = (req, res) => {

// res.json({ msg: "get by SKU alimentos controller" });

// }

const alimentoPut = (req, res) => {

res.json({ msg: "put alimento controller" });

}

const alimentoDelete = (req, res) => {

res.json({ msg: "delete alimento controller" });

}

module.exports = {

alimentoGet,

alimentoPost,

alimentoPut,

alimentoDelete

}

**Controladores “Artesanías”**

const Artesania = require('../models/artesania')

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

const artesanias = await Artesania.find({categoria: 'artesania'})

res.json(artesanias);

}

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

const { categoria, nombre, precio, img, stock ,descripcion} = req.body;

const nuevaArtesania = new Artesania({categoria,nombre,precio,img,stock,descripcion});

await nuevaArtesania.save()

res.json({ nuevaArtesania });

}

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

const { id } = req.params;

const { \_id, dcto ,...data} = req.body

const ArteActu = await Artesania.findByIdAndUpdate(id,data,{new:true})

res.json( {ArteActu} );

}

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

const { id } = req.params;

const productoEliminado = await Artesania.findByIdAndDelete(id)

res.json({ "mensaje":"producto eliminado",productoEliminado });

}

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

const { id } = req.params;

const artesaniaF = await Artesania.findById(id)

res.json( artesaniaF );

}

module.exports = {

artesaniaGet,

artesaniaPost,

artesaniaPut,

artesaniaDelete,

artesaniaGetByid

}

**Controladores “Productos”**

const productsGet = (req, res) => {

res.json({ msg: "get Products controller" });

}

const productsPost = (req, res) => {

const { name, author } = req.body;

res.json({ msg: "Post Products controller", name, author });

}

const productsGetBySKU = (req, res) => {

res.json({ msg: "get by SKU Products controller" });

}

const productsPut = (req, res) => {

res.json({ msg: "put Products controller" });

}

const productsDelete = (req, res) => {

res.json({ msg: "delete Products controller" });

}

module.exports = {

productsGet, productsPost, productsGetBySKU, productsPut, productsDelete

}

La evidencia de instalación del ambiente de trabajo se encuentra en anexos.

1. **Ruta CRUD**

**Ruta CRUD Alimentos:**

const { Router } = require('express');

const { check } = require('express-validator');

const {

categoriaExist

} = require('../helpers/validaciones')

const { validateDate } = require('../middlewares/validacionerrores')

const router = Router();

const {

alimentoGet,

alimentoPost,

alimentoPut,

alimentoDelete

} = require('../controllers/alimentos')

router.get('/', alimentoGet);

router.post('/', [

check('categoria').custom(categoriaExist),

validateDate

], alimentoPost);

router.put('/', alimentoPut);

router.delete('/', alimentoDelete);

module.exports = router;

**Ruta CRUD Artesanías:**

const { Router } = require('express');

const { check } = require('express-validator');

const { categoriaExist,

productoByIdExists

} = require('../helpers/validaciones')

const { validateDate } =require('../middlewares/validacionerrores')

const router = Router();

const { artesaniaGet,

artesaniaPost,

artesaniaPut,

artesaniaDelete,

artesaniaGetByid

} = require('../controllers/artesanias')

router.get('/', artesaniaGet);

router.get('/:id',[

check('id', 'No es un Id valido').isMongoId(),

check('id','hola').custom(productoByIdExists),

validateDate

],artesaniaGetByid)

router.post('/',[

check('categoria').custom(categoriaExist),

check('stock', 'El stock es requerido o es invalido').notEmpty().isInt({ min:1 }),

check('nombre', 'El nombre es requerido').notEmpty(),

check('precio', 'El precio es requerido').notEmpty(),

validateDate

], artesaniaPost);

router.put('/:id',[

check('id', 'No es un Id valido').isMongoId(),

check('id').custom(productoByIdExists),

validateDate

], artesaniaPut);

router.delete('/:id',[

check('id', 'No es un Id valido').isMongoId(),

check('id').custom(productoByIdExists),

validateDate

], artesaniaDelete);

module.exports = router;

router.put('/', artesaniaPut);

router.delete('/', artesaniaDelete);

module.exports = router;

**Ruta CRUD Productos:**

**const { Router } = require('express');**

**const router = Router();**

**const {**

**productsGet,**

**productsPost,**

**productsGetBySKU,**

**productsPut,**

**productsDelete**

**} = require('../controllers/products')**

**router.get('/', productsGet);**

**router.post('/', productsPost);**

**router.get('/:sku', productsGetBySKU);**

**router.put('/:sku', productsPut);**

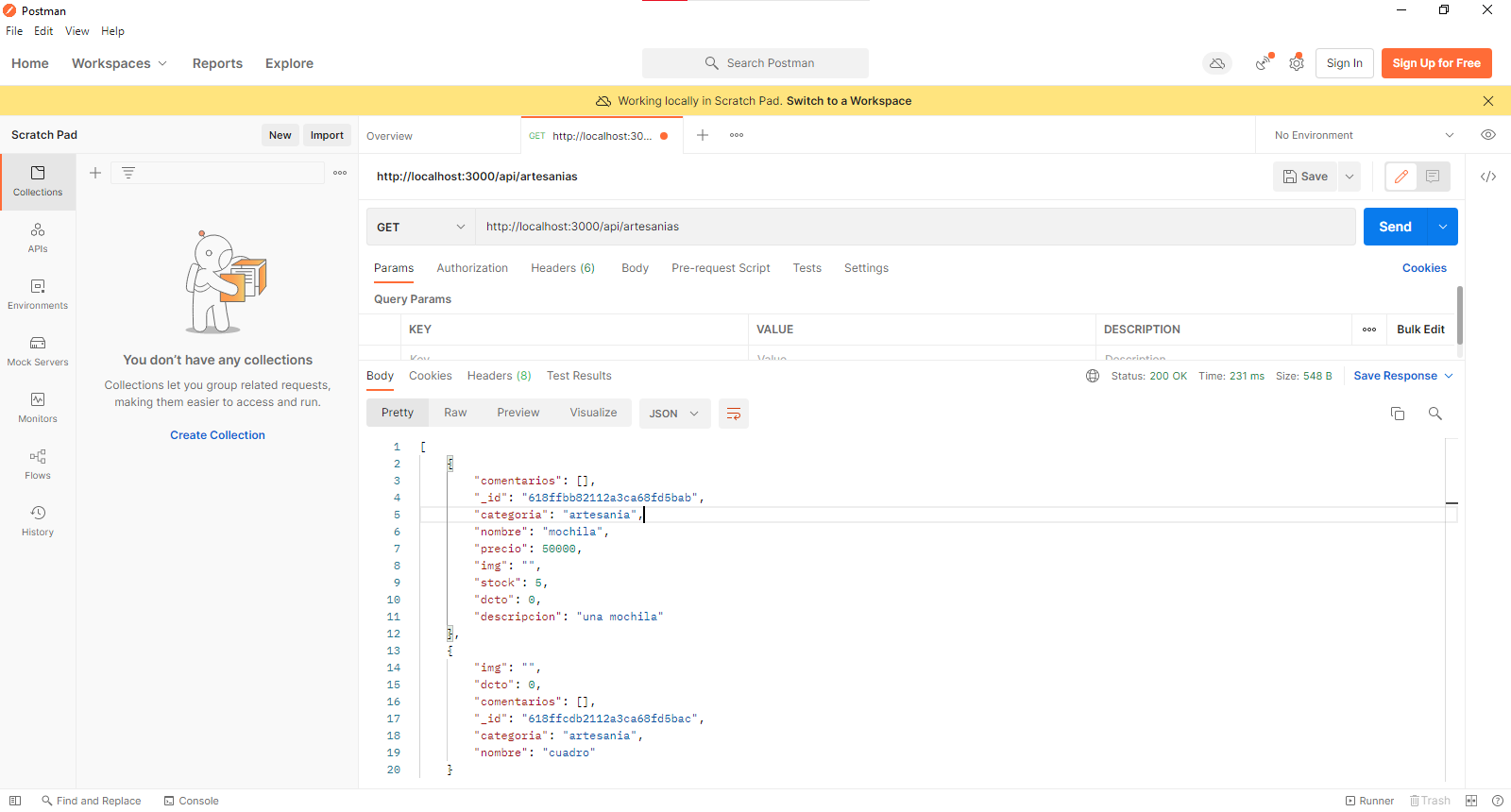
**router.delete(':sku', productsDelete);**

**module.exports = router;**

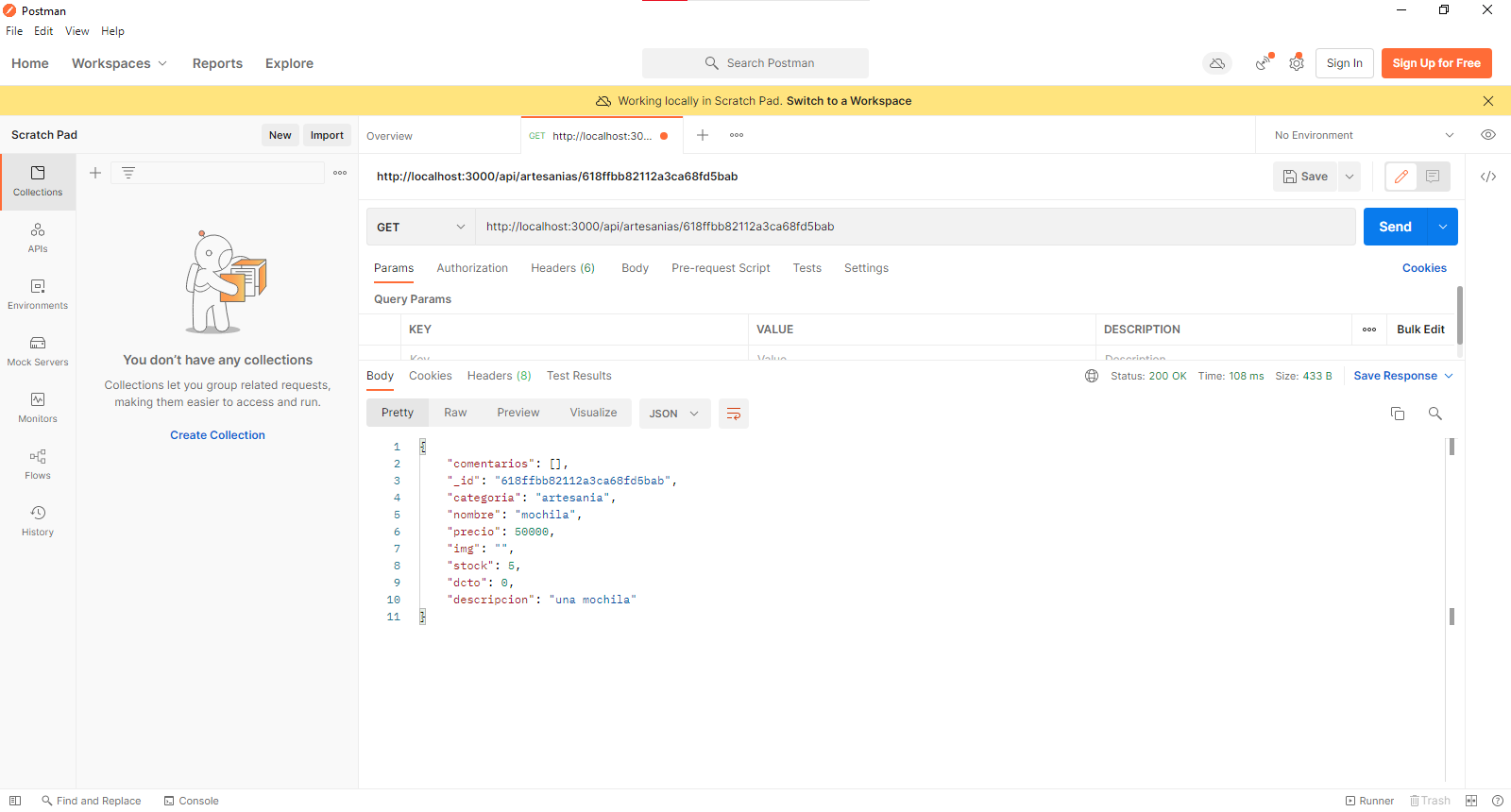
1. **Pruebas de las rutas:**

Artesanias:

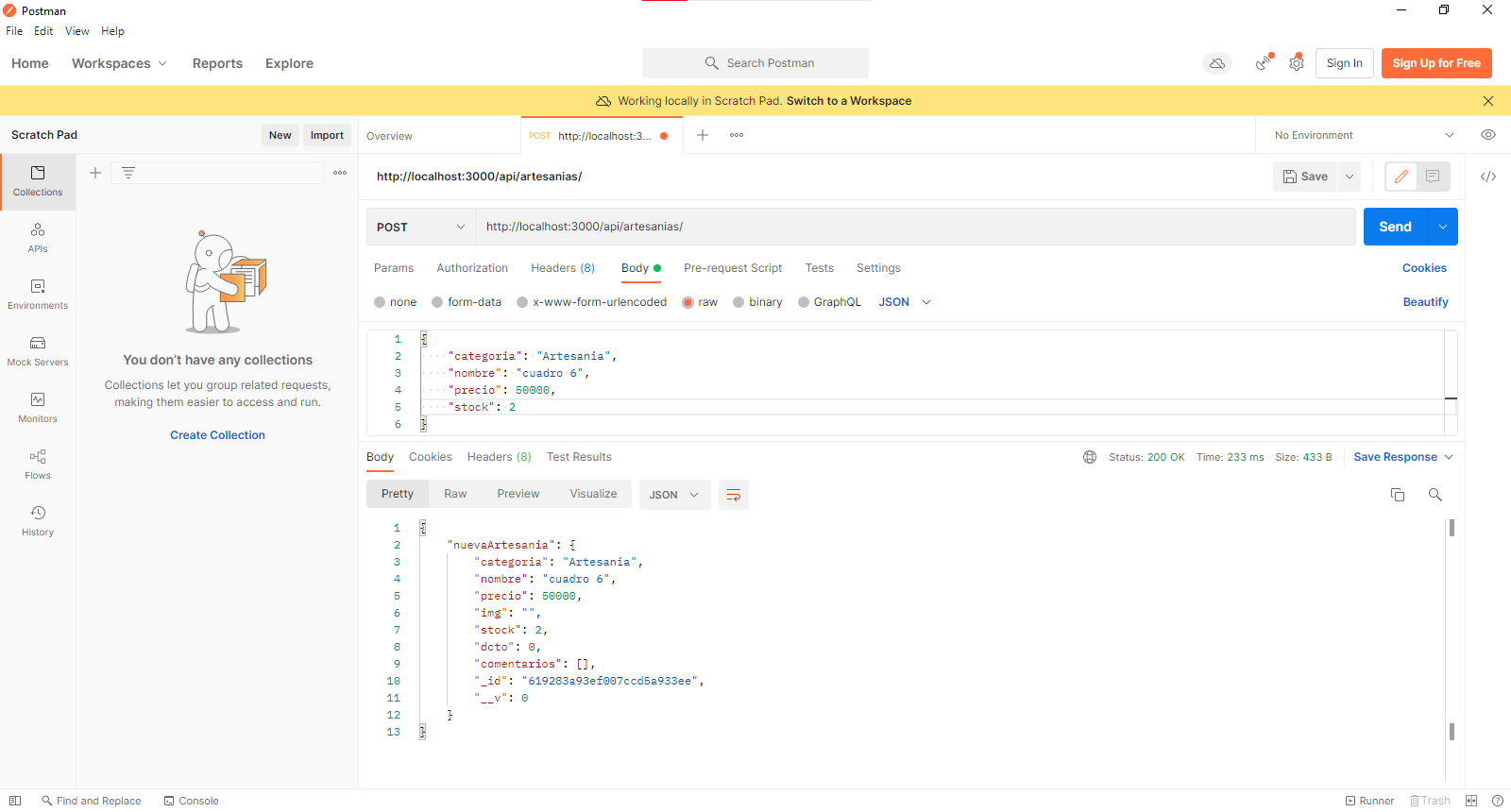
Metodo GET para traer todos los productos en la base de datos



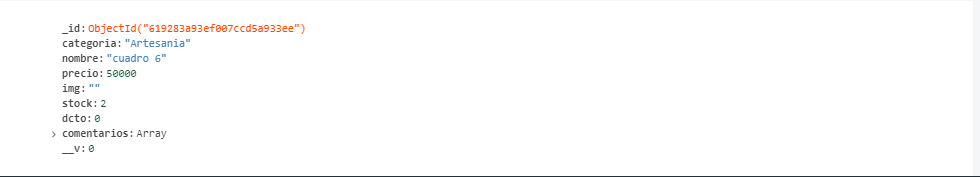
Metodo GET busqueda por ID

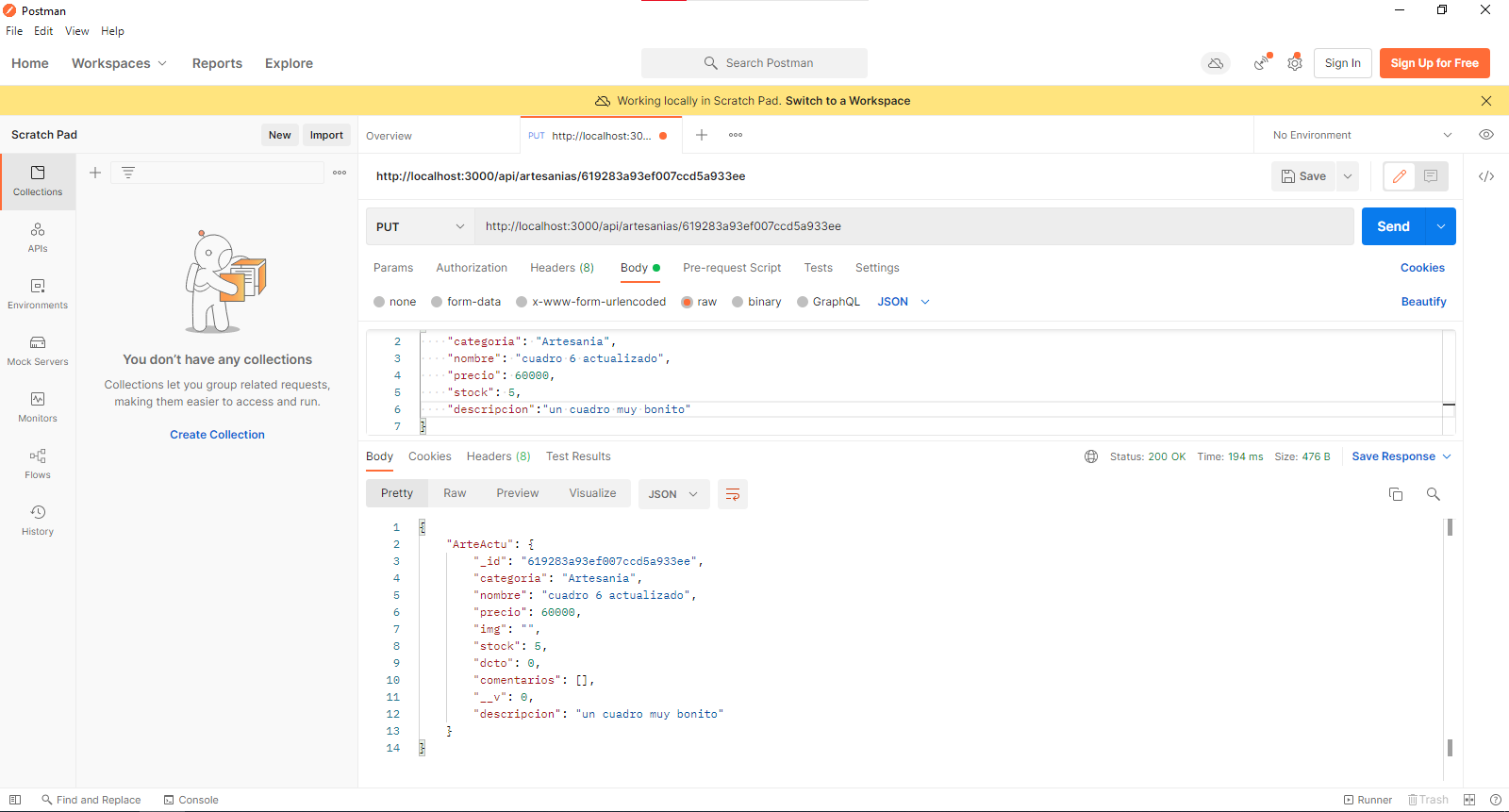


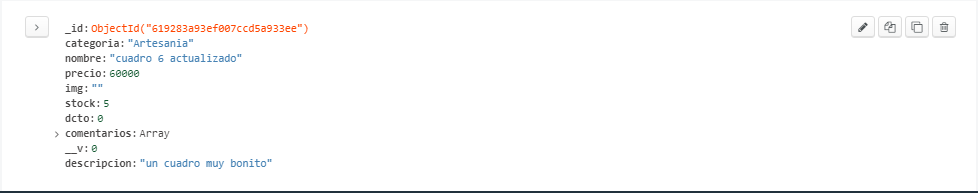
Metodo POST crear una nueva artesania



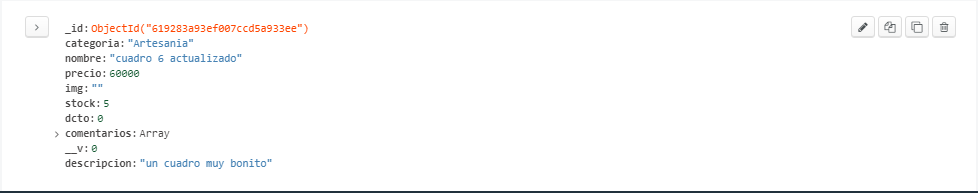
Metodo PUT actualización de una artesania

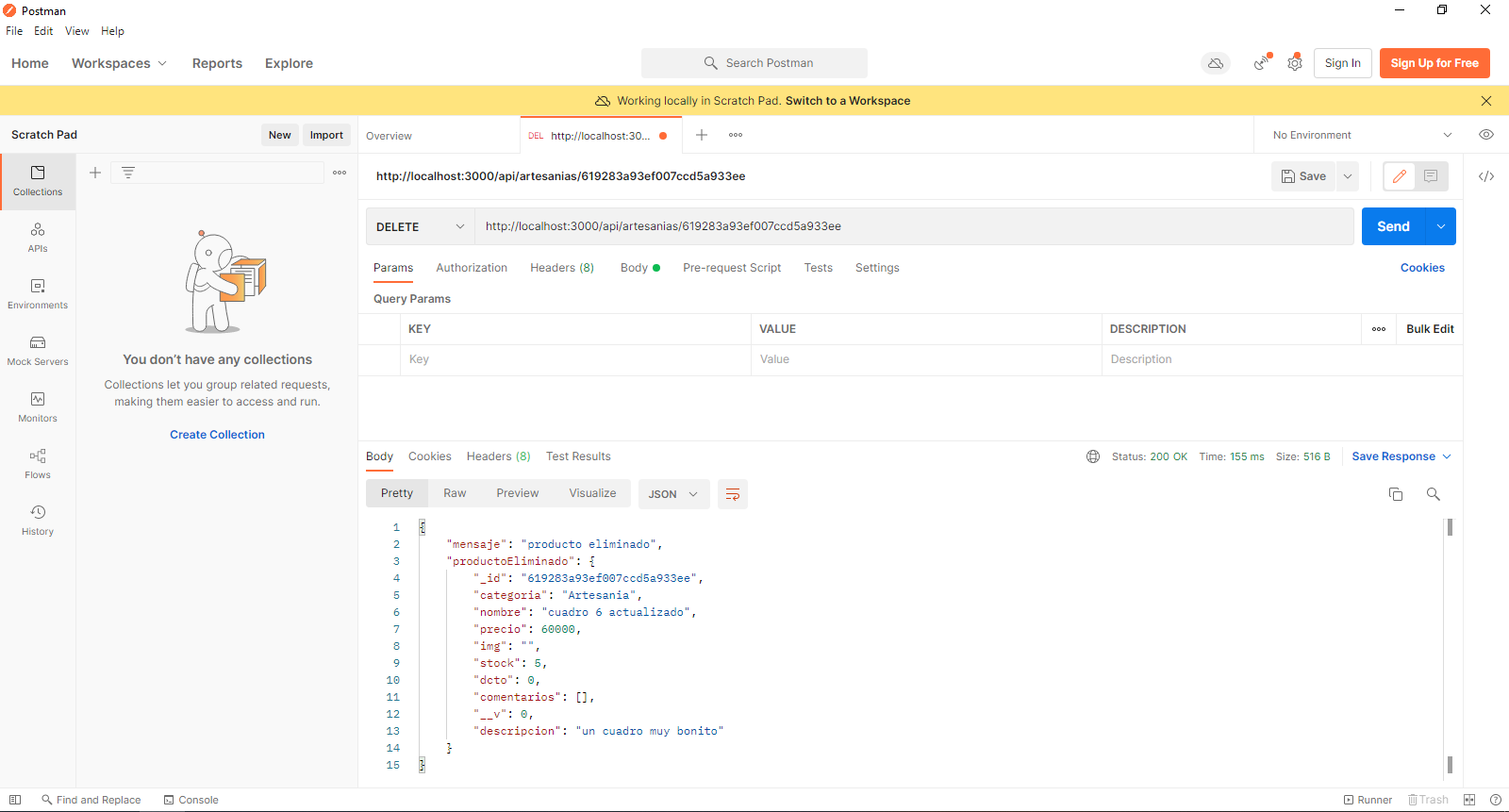




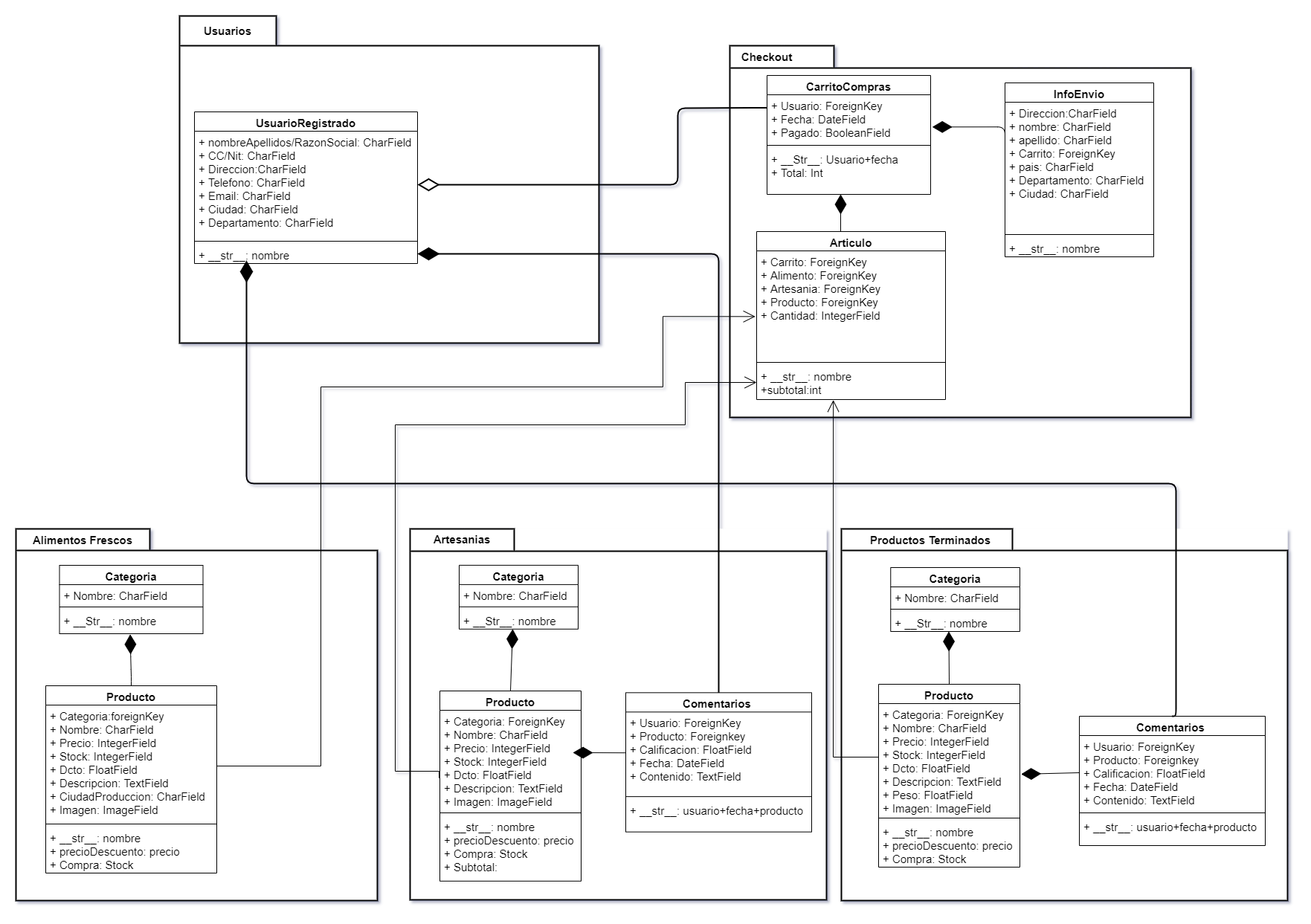


Metodo DELETE





**ANEXOS:**

**Imagen 1 Diagrama de clases UML ComercioVerde**

Repositorios:

[**https://github.com/psicohernando/comercio-verde-B29**](https://github.com/psicohernando/comercio-verde-B29)

[**https://github.com/Manuel-Leon-H/comercio-verde-B29**](https://github.com/Manuel-Leon-H/comercio-verde-B29)