Звіт

З лабораторної роботи №2 «MongoDb & Mongoose.js»

Тема: Робота з документо-орієнтованою базою даних mongoDb

Мета: Отримання навиків підключення та роботи з базою даних MongoDb за допомогою NodeJS та Mongoose

Під час роботи над лабораторною роботою були використані напрацювання з минулої лабораторної роботи (№1).

Після встановлення MongoDb я встановив Mongoose за допомогою команди npm install mongoose. Mongoose – ORM для MongoDb. Далі в скрипті libs/mongoose.js я прописав основну роботу з бд :

* Створив підключення та відкрив його:

var mongoose = require('mongoose');

var log = require('./log')(module);

var config = require('./config');

mongoose.connect(config.get('mongoose:uri'));

var db = mongoose.connection;

db.on('error', function (err) {

    log.error('connection error:', err.message);

});

db.once('open', function callback () {

    log.info("Connected to DB!");

});

у файлі config.json прописав наступну інформацію для встановлення з’єднання (вказав порт та адресу бд на сервері mongoDb):

{

    "port" : 1337,

    "mongoose": {

        "uri": "mongodb://127.0.0.1:27017/test1"

    }

}

Після цього я створив опис статті за її сутністю за допомогою Схеми (Schema):

var Schema = mongoose.Schema; // Schemas

var Article = new Schema({

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

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

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

    modified: { type: Date, default: Date.now }

}); // validation

Та використав валідацію розміру заголовку статті:

Article.path('title').validate(function (v) {

    return v.length > 5 && v.length < 70;

});

Потім створив модель по схемі та поверув її значення:

var ArticleModel = mongoose.model('Article', Article);

module.exports.ArticleModel = ArticleModel;

У файлі серверу srv.js я прописав route, до якого відбувається звернення із запросом POST та даними, необхілними для заповнення полів моделі:

app.post('/api/articles', function(req, res) {

    var article = new ArticleModel({

        title: req.body.title,

        author: req.body.author,

        description: req.body.description,

        //images: req.body.images

    });

    article.save(function (err) {

        if (!err) {

            log.info("article created");

            return res.send({

                status: 'OK',

                article:article

            });

        } else {

            console.log(err);

            if(err.name == 'ValidationError') {

                res.statusCode = 400;

                res.send({ error: 'Validation error' });

            } else {

                res.statusCode = 500;

                res.send({ error: 'Server error' });

            }

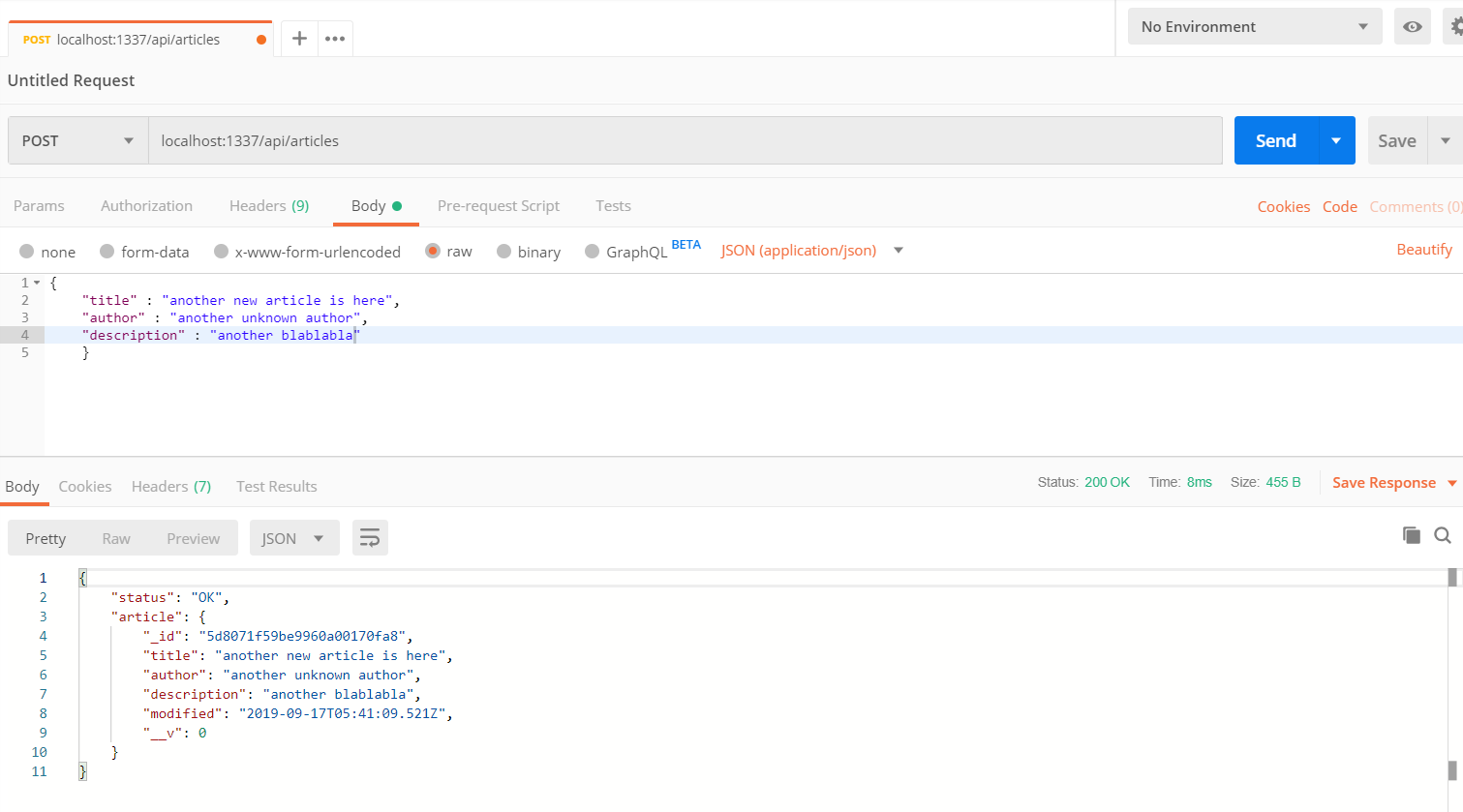
            log.error('Internal error(' + res.statusCode + '): ' + err.message);

        }

    });

});

Після цього зробив тест, відправивши дані через Postmen запросом POST:



Статус 200 – ОК повідомляє , що дані успішно збереглись в колекції Articles бази даних test1 і доступні за адресою “localhost:1337/api/articles/” + “id документу” (в нашому випадку це унікальний ключ "\_id": "5d8071f59be9960a00170fa8" )

Також за доступ до даної статті відповідає маршрут Express

app.get('/api/articles/:id', function(req, res) {

    return ArticleModel.findById(req.params.id, function (err, article) {

        if(!article) {

            res.statusCode = 404;

            return res.send({ error: 'Not found' });

        }

        if (!err) {

            return res.send({ status: 'OK', article:article });

        } else {

            res.statusCode = 500;

            log.error('Internal error(' + res.statusCode + '): ' + err.message);

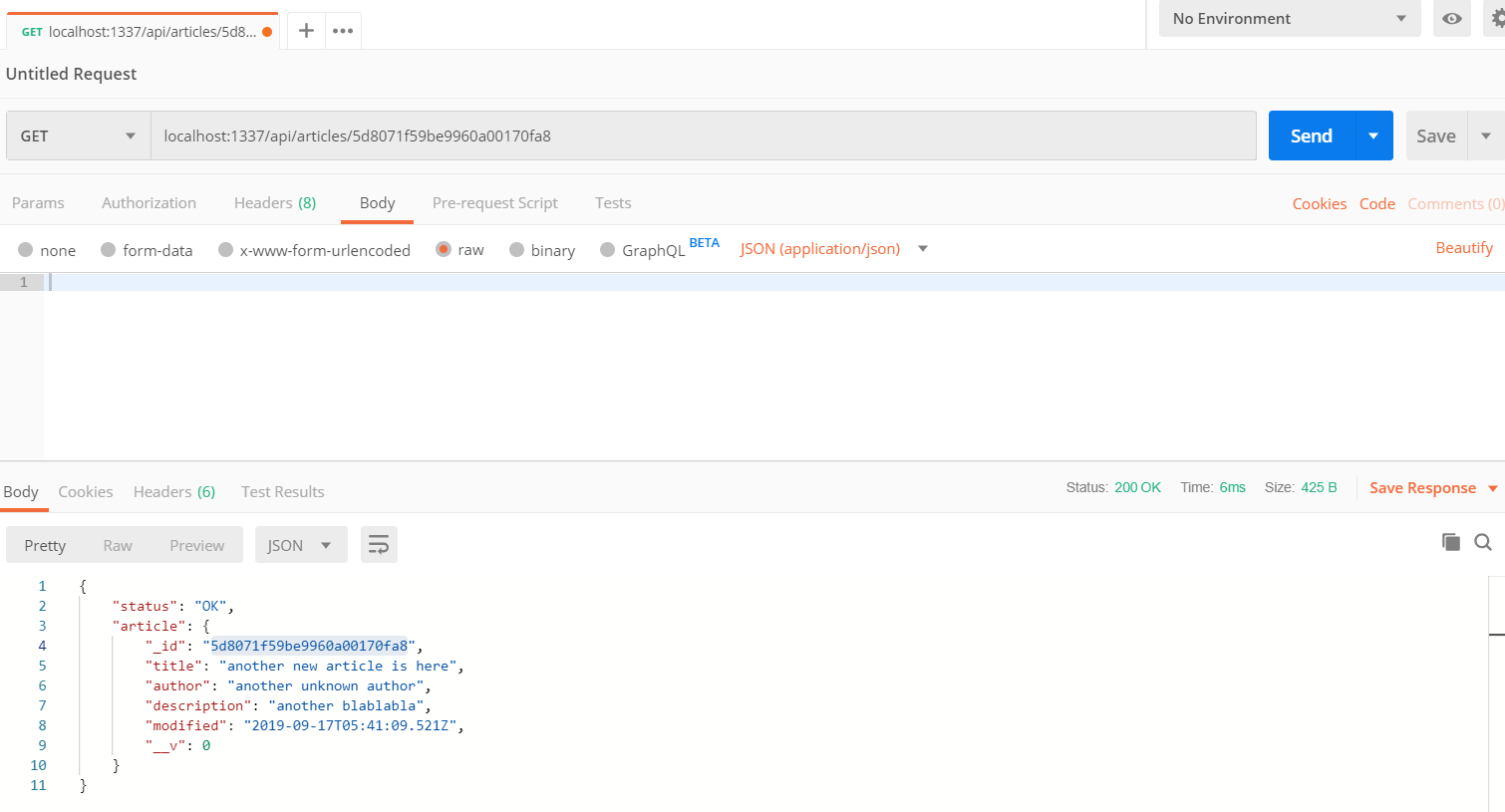
            return res.send({ error: 'Server error' });

        }

    });

});

При зверненні за даною адресою методом GET отримаєм наш документ:



Також в файлі srv.js я прописав код, який показує усі статті з колекції Articles:

app.get('/api/articles', function(req, res) {

    //console.log(req.body);

    return ArticleModel.find(function (err, articles) {

        if (!err) {

            return res.send(articles);

        } else {

            res.statusCode = 500;

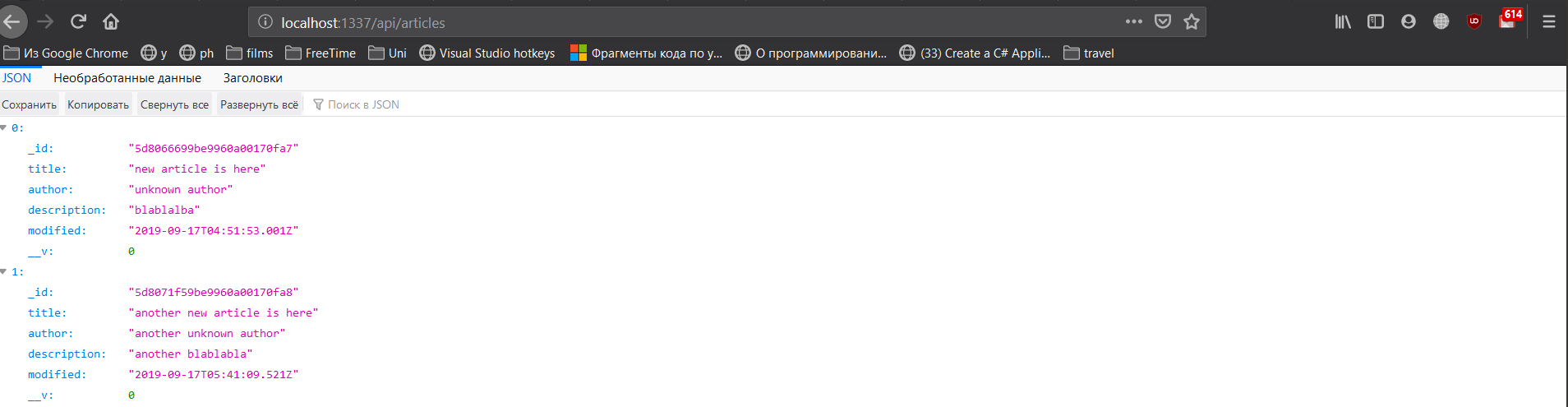
            log.error('Internal error(' + res.statusCode + '): ' + err.message);

            return res.send({ error: 'Server error' });

        }

    });

});

Тому при зверненні за адресою localhost:1337/api/articles ми можемо глянути всі документи колекції Articles: 

Також я прописав маршрут для зміни даних PUT :

app.put('/api/articles/:id', function (req, res){

    return ArticleModel.findById(req.params.id, function (err, article) {

        if(!article) {

            res.statusCode = 404;

            return res.send({ error: 'Not found' });

        }

        article.title = req.body.title;

        article.description = req.body.description;

        article.author = req.body.author;

        // article.images = req.body.images;

        return article.save(function (err) {

            if (!err) {

                log.info("article updated");

                return res.send({ status: 'OK', article:article });

            } else {

                if(err.name == 'ValidationError') {

                    res.statusCode = 400;

                    res.send({ error: 'Validation error' });

                } else {

                    res.statusCode = 500;

                    res.send({ error: 'Server error' });

                }

                log.error('Internal error(' + res.statusCode + '): ' + err.message);

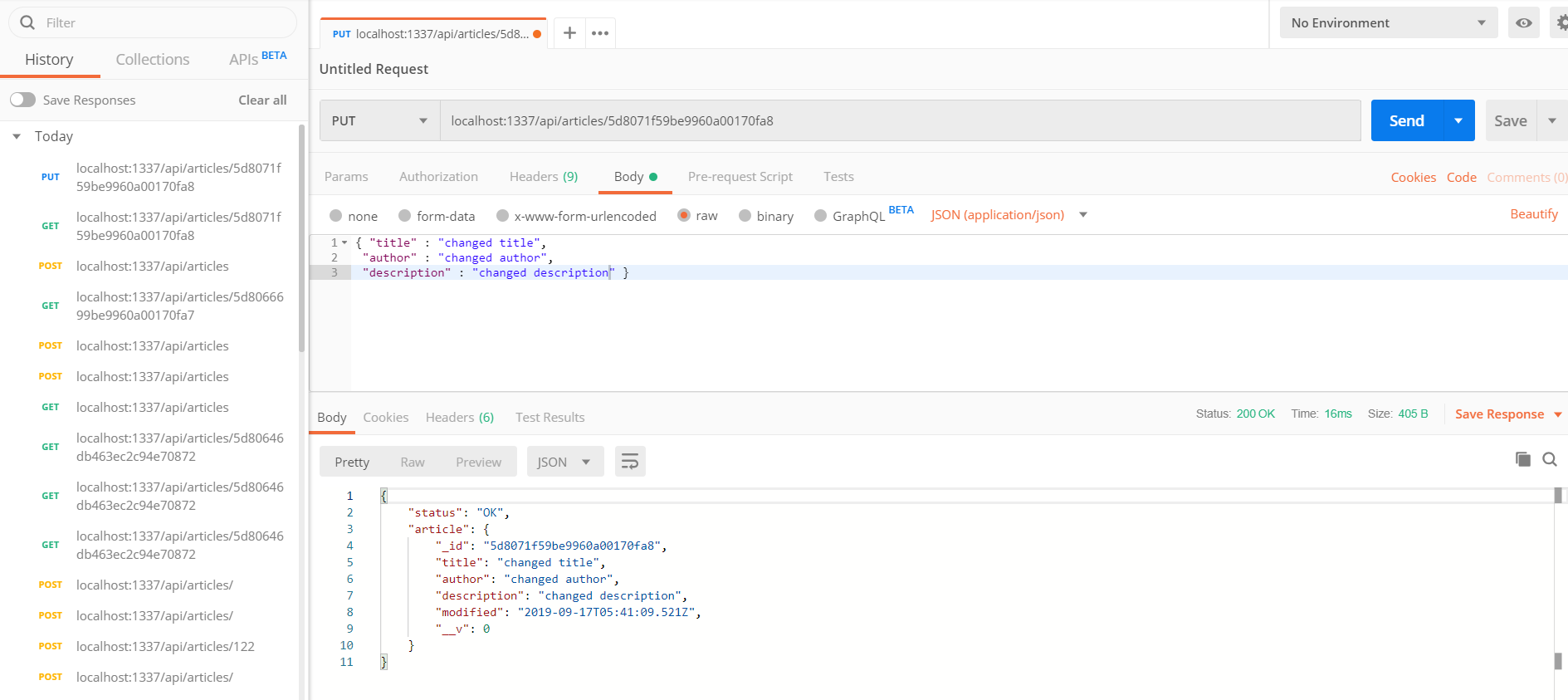
            }

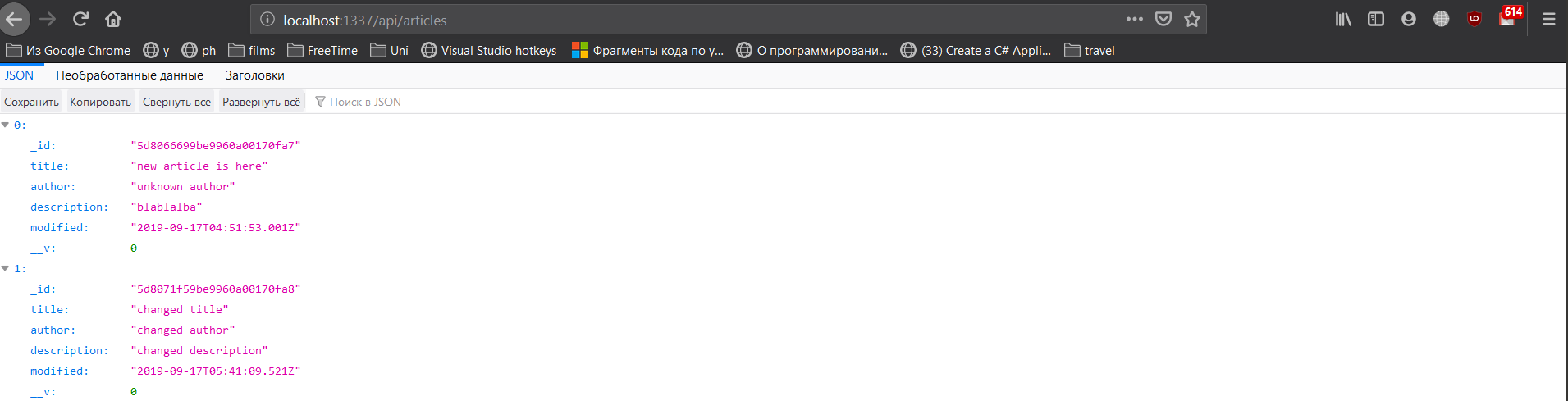
        });

    });

});

Якого ми “годуємо” зміненими даними через Postmen :





А відповідно за маршрутом

app.delete('/api/articles/:id', function (req, res){

    return ArticleModel.findById(req.params.id, function (err, article) {

        if(!article) {

            res.statusCode = 404;

            return res.send({ error: 'Not found' });

        }

        return article.remove(function (err) {

            if (!err) {

                log.info("article removed");

                return res.send({ status: 'OK' });

            } else {

                res.statusCode = 500;

                log.error('Internal error(' + res.statusCode + '): ' + err.message);

                return res.send({ error: 'Server error' });

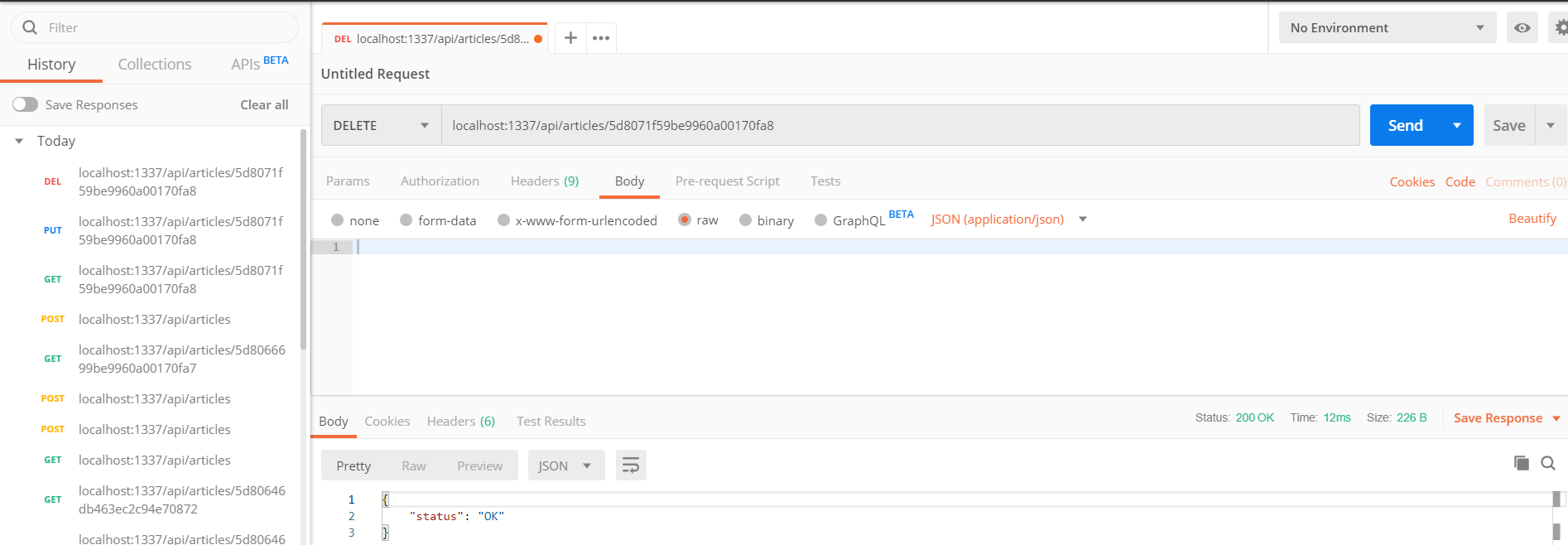
            }

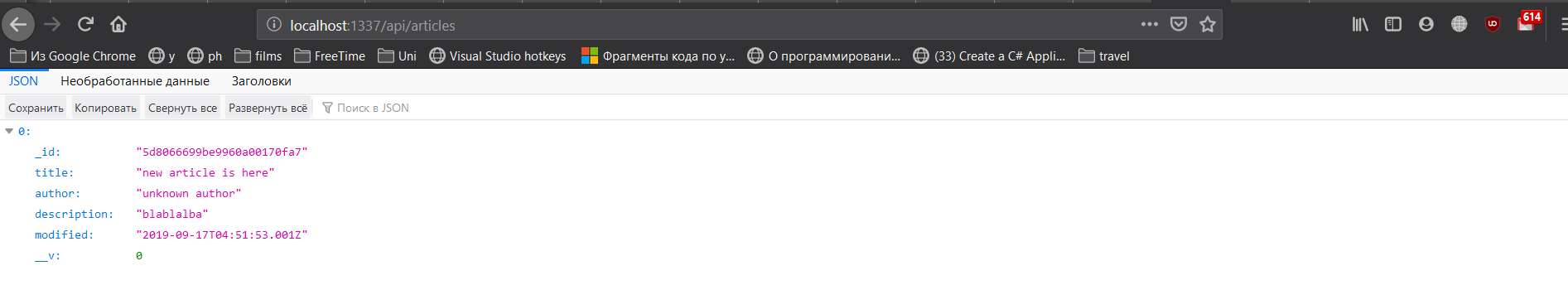
        });

    });

});

Ми маємо змогу видалити нашу статтю:





ВИСНОВОК