use mydb – создать db

db.dropDatabase() – удалить базу данных

show dbs – показать созданные

show collections

db.mycollection.insert({“name”: “kot”})

[db.collection1.insertOne()](https://docs.mongodb.com/manual/reference/method/db.collection.insertOne/#db.collection.insertOne)

[db.collection1.insertMany()](https://docs.mongodb.com/manual/reference/method/db.collection.insertMany/#db.collection.insertMany)

[db.collection1.find()](https://docs.mongodb.com/manual/reference/method/db.collection.find/#db.collection.find)

db.collection1.find({name: “Egor”}).sort({size: -1}).limit(1).skip(1).pretty()

[db.collection1.updateOne()](https://docs.mongodb.com/manual/reference/method/db.collection.updateOne/#db.collection.updateOne)

[db.collection1.updateMany()](https://docs.mongodb.com/manual/reference/method/db.collection.updateMany/#db.collection.updateMany)

[db.collection1.replaceOne()](https://docs.mongodb.com/manual/reference/method/db.collection.replaceOne/#db.collection.replaceOne)

db.collection1.update({size: “M”}, {$set: {color: “Black”}}, {multi: true})

[db.collection1.deleteOne()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteOne/#db.collection.deleteOne)

[db.collection1.deleteMany()](https://docs.mongodb.com/manual/reference/method/db.collection.deleteMany/#db.collection.deleteMany)

db.colection1.remove({size: “M”})

В серверной части - Mongoose:

Npm install mongoose –save

В проекте создаем папку configs -> mongo.js

var mongoose = require('mongoose');  
  
//const dbURI = 'mongodb://username:password@locahost/mydatabase';  
const dbURI = 'mongodb://localhost:27017/products';  
var dbOptions = {  
 user: 'userName',  
 password: 'userPass'  
};  
//mongoose.connect(dbURI, dbOptions);  
mongoose.connect(dbURI);  
mongoose.connection.on('connected', function () {  
 console.log('Mongoose has connected to ' + dbURI)  
});  
mongoose.connection.on('error', function () {  
 console.log('Mongoose has not connected to ' + dbURI)  
});  
module.exports = mongoose;

В app.js:

Var mongoose = require(‘./configs/mongo’)

Создаем папку models и в ней создает product.model.js

В нем создаем схему и экспортируем модель Product

var mongoose = require('mongoose');  
var productSchema = mongoose.Schema({  
 \_id: mongoose.ObjectId,  
 name: String,  
 price: Number,  
 color: {  
 type: String,  
 enum: ["Black", "White", "Red"]  
 },  
 last\_updated: Date  
}, {collection: “myCol”});

//Если нам нужно до выполнения в методе пост функции save задать поля \_id и last\_update, которые не приходят из клиента, то используют hooks:

productSchema.pre('save', function (next) {  
 this.\_id = mongoose.mongo.ObjectId();  
 this.last\_updated = new Date();

next();  
});  
module.exports = mongoose.model("Product", productSchema);

Данные будут храниться в коллекциий "Product" с маленькой буквы во множественном числе, или можно сменить имя коллекции добавив collection: “myCol” в модель.

В products.js:

Var Product = require(‘./../models/product.model’);

router.get('/', function (req, res) {  
 Product.find({}, function (err, productsArray) {  
 res.status(200).send(productsArray);  
 })  
});  
  
router.get('/:id', function (req, res) {  
 Product.findOne({\_id: req.params.id}, function (err, product) {  
 res.status(200).send(product);  
 })  
});

router.post('/', function (req, res) {  
 var product = new Product(req.body);  
 product.save(function (err) {  
 if(err){  
 res.status(500).json(err)  
 } else {  
 res.status(201).send(product)  
 }  
 });  
});

router.delete('/:id', function (req, res) {  
 Product.deleteOne({\_id: req.params.id}, function (err) {  
 res.status(204).send();  
 })  
});  
  
module.exports = router;