Финальный проект — REST API + MongoDB + Docker + Monitoring

* Express-сервер с REST API
* Подключение MongoDB
* Контейнеризацию с Dockerfile и Compose
* Healthcheck, Restart-политику, логи
* Расширяемую архитектуру

**🧱 6.1 Структура проекта**

pgsql

КопироватьРедактировать

rest-api-project/

├── Dockerfile

├── docker-compose.yml

├── .env

├── package.json

├── index.js

├── models/

│ └── User.js

├── routes/

│ └── users.js

├── logs/

│ └── app.log

**📦 6.2 package.json**

json

КопироватьРедактировать

{

"name": "rest-api",

"version": "1.0.0",

"main": "index.js",

"scripts": {

"start": "node index.js"

},

"dependencies": {

"dotenv": "^16.0.3",

"express": "^4.18.2",

"mongoose": "^7.6.1"

}

}

**📄 6.3 .env**

ini

КопироватьРедактировать

PORT=3000

MONGO\_URI=mongodb://mongo:27017/usersdb

**🧩 6.4 Модель пользователя (models/User.js)**

js

КопироватьРедактировать

const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({

name: String,

email: String

});

module.exports = mongoose.model('User', userSchema);

**🌐 6.5 Роуты (routes/users.js)**

js

КопироватьРедактировать

const express = require('express');

const router = express.Router();

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

// GET all users

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

const users = await User.find();

res.json(users);

});

// POST new user

router.post('/', async (req, res) => {

const newUser = new User(req.body);

await newUser.save();

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

});

module.exports = router;

**🧠 6.6 Главный файл (index.js)**

js

КопироватьРедактировать

require('dotenv').config();

const express = require('express');

const mongoose = require('mongoose');

const fs = require('fs');

const app = express();

const PORT = process.env.PORT || 3000;

// логирование в файл

const logStream = fs.createWriteStream('./logs/app.log', { flags: 'a' });

console.log = (...args) => {

logStream.write(`[INFO] ${args.join(' ')}\n`);

process.stdout.write(`[INFO] ${args.join(' ')}\n`);

};

app.use(express.json());

app.use('/users', require('./routes/users'));

mongoose.connect(process.env.MONGO\_URI)

.then(() => console.log('✅ Mongo подключена'))

.catch(err => console.error('❌ Mongo ошибка', err));

app.get('/', (req, res) => res.send('REST API работает'));

app.listen(PORT, () => console.log(`🚀 Сервер на порту ${PORT}`));

**🐳 6.7 Dockerfile**

Dockerfile

КопироватьРедактировать

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["npm", "start"]

**⚙️ 6.8 docker-compose.yml**

yaml

КопироватьРедактировать

version: '3.8'

services:

app:

build: .

ports:

- "3000:3000"

restart: always

volumes:

- ./logs:/app/logs

environment:

- PORT=3000

- MONGO\_URI=mongodb://mongo:27017/usersdb

healthcheck:

test: ["CMD", "curl", "-f", "http://localhost:3000"]

interval: 10s

timeout: 5s

retries: 3

mongo:

image: mongo

ports:

- "27017:27017"

volumes:

- mongo-data:/data/db

volumes:

mongo-data:

**🔨 6.9 Запуск проекта**

bash

docker-compose up -d --build

⏳ Подожди несколько секунд — сначала Mongo и Express инициализируются

Проверь:

* 📄 <http://localhost:3000> — должно быть REST API работает
* 📄 <http://localhost:3000/users> — пустой массив

**🧪 6.10 Примеры запросов**

**POST /users**

bash

КопироватьРедактировать

curl -X POST http://localhost:3000/users \

-H "Content-Type: application/json" \

-d '{"name":"Ivan","email":"ivan@example.com"}'

**GET /users**

bash

КопироватьРедактировать

curl http://localhost:3000/users

**✅ 6.11 Результат**

Ты получил:

* Настроенный REST API
* MongoDB с volume-хранилищем
* Healthcheck + restart
* Логирование в файл
* Один YAML для развёртывания