Цель: настроить полноценный **REST API на Express + MongoDB + Redis**, контейнеризованный в Docker Compose, с логированием, кешем, JWT, Nginx.

**🔧 ЧТО ТЫ СОЗДАШЬ:**

| **Компонент** | **Назначение** |
| --- | --- |
| Node.js API | Express-сервер с /auth, /users |
| MongoDB | Хранилище пользователей |
| Redis | Кеширование /users, JWT хранилище |
| Nginx | Reverse proxy + HTTPS |
| JWT | Авторизация |
| Docker Compose | Связывает всё вместе |

**✅ ЗАДАНИЯ:**

**🔹 1. Создай Express-приложение с REST API**

📄 routes/users.js

js

КопироватьРедактировать

const express = require('express');

const router = express.Router();

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

const redis = require('../lib/redis');

// GET /users (с кешем)

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

const cached = await redis.get('users');

if (cached) return res.json(JSON.parse(cached));

const users = await User.find();

await redis.set('users', JSON.stringify(users), 'EX', 60); // 1 минута

res.json(users);

});

module.exports = router;

**🔹 2. Настрой MongoDB и mongoose**

📄 models/User.js

js

КопироватьРедактировать

const mongoose = require('mongoose');

module.exports = mongoose.model('User', new mongoose.Schema({

name: String,

email: String

}));

**🔹 3. Подключи Redis**

📄 lib/redis.js

js

КопироватьРедактировать

const redis = require('redis');

const client = redis.createClient({ url: 'redis://redis:6379' });

client.connect();

client.on('error', err => console.error('Redis error', err));

module.exports = client;

**🔹 4. Добавь JWT-аутентификацию**

📄 routes/auth.js

js

КопироватьРедактировать

const jwt = require('jsonwebtoken');

const router = require('express').Router();

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

const token = jwt.sign({ id: 1 }, process.env.JWT\_SECRET, { expiresIn: '1h' });

res.json({ token });

});

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

const auth = req.headers.authorization;

if (!auth) return res.sendStatus(401);

try {

const decoded = jwt.verify(auth.split(' ')[1], process.env.JWT\_SECRET);

res.json(decoded);

} catch {

res.sendStatus(403);

}

});

module.exports = router;

**🔹 5. Настрой Dockerfile**

Dockerfile

КопироватьРедактировать

FROM node:18-alpine

WORKDIR /app

COPY package\*.json ./

RUN npm install

COPY . .

EXPOSE 3000

CMD ["node", "index.js"]

**🔹 6. Напиши docker-compose.yml**

yaml

КопироватьРедактировать

version: '3.8'

services:

api:

build: .

ports:

- "3000:3000"

depends\_on:

- mongo

- redis

environment:

- PORT=3000

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

- JWT\_SECRET=supersecret

volumes:

- ./logs:/app/logs

healthcheck:

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

interval: 10s

retries: 3

mongo:

image: mongo

volumes:

- mongo-data:/data/db

redis:

image: redis

volumes:

mongo-data:

**🔹 7. Добавь логирование в файл**

📄 logger.js

js

КопироватьРедактировать

const fs = require('fs');

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

module.exports = (...args) => {

const msg = `[${new Date().toISOString()}] ${args.join(' ')}\n`;

stream.write(msg);

console.log(...args);

};

**🔹 8. Настрой Nginx + HTTPS (опционально)**

📄 nginx/nginx.conf

nginx

КопироватьРедактировать

server {

listen 443 ssl;

ssl\_certificate /etc/nginx/ssl/cert.pem;

ssl\_certificate\_key /etc/nginx/ssl/key.pem;

location / {

proxy\_pass http://api:3000;

proxy\_http\_version 1.1;

}

}

Добавь в Compose nginx и сертификаты.

**🔹 9. Проверь авторизацию**

1. POST /login → получи token
2. GET /me → добавь Authorization: Bearer <token>

**🔹 10. Протестируй кеш Redis**

* Сделай два запроса к /users
* Первый — получит из Mongo
* Второй — из Redis (быстрее)

**✅ РЕЗУЛЬТАТ УРОВНЯ 2:**

| **Функция** | **Статус** |
| --- | --- |
| REST API | ✅ |
| Mongo + Redis | ✅ |
| Кеширование | ✅ |
| JWT авторизация | ✅ |
| Compose с 3 сервисами | ✅ |
| Healthcheck + Logging | ✅ |
| Reverse Proxy | 🔄 опционально |