Тестирование Node.js-приложений — Jest, Supertest, и End-to-End

* Корректность логики
* Доступность API
* Поведение при ошибках
* Реакцию на невалидные данные

**🧰 7.1 Инструменты**

| **Инструмент** | **Назначение** |
| --- | --- |
| **Jest** | Фреймворк для Unit/Integration тестов |
| **Supertest** | HTTP-запросы к Express-приложениям |
| **MongoMemoryServer** | Временная MongoDB для изоляции |
| **Test Coverage** | Покрытие кода тестами |

**📦 7.2 Установка зависимостей**

bash

КопироватьРедактировать

npm install --save-dev jest supertest mongodb-memory-server

Добавь в package.json:

json

КопироватьРедактировать

"scripts": {

"test": "jest"

}

**📁 7.3 Структура проекта с тестами**

pgsql

КопироватьРедактировать

rest-api-project/

├── \_\_tests\_\_/

│ └── users.test.js

├── app.js ← отделён от index.js

├── index.js

**⚙️ 7.4 Рефакторинг: выносим Express-приложение**

📄 app.js

js

КопироватьРедактировать

require('dotenv').config();

const express = require('express');

const mongoose = require('mongoose');

const app = express();

app.use(express.json());

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

module.exports = { app, mongoose };

📄 index.js

js

КопироватьРедактировать

const { app, mongoose } = require('./app');

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

.then(() => {

app.listen(process.env.PORT || 3000, () => {

console.log(`🚀 Сервер на порту ${process.env.PORT}`);

});

})

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

**🧪 7.5 Тест: users.test.js**

📄 \_\_tests\_\_/users.test.js

js

КопироватьРедактировать

const request = require('supertest');

const { app } = require('../app');

const mongoose = require('mongoose');

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

const { MongoMemoryServer } = require('mongodb-memory-server');

let mongoServer;

beforeAll(async () => {

mongoServer = await MongoMemoryServer.create();

const uri = mongoServer.getUri();

await mongoose.connect(uri);

});

afterAll(async () => {

await mongoose.disconnect();

await mongoServer.stop();

});

beforeEach(async () => {

await User.deleteMany();

});

test('Создание нового пользователя', async () => {

const res = await request(app)

.post('/users')

.send({ name: 'Ivan', email: 'ivan@example.com' });

expect(res.statusCode).toBe(201);

expect(res.body.name).toBe('Ivan');

});

test('Получение пользователей', async () => {

await User.create({ name: 'Anna', email: 'anna@example.com' });

const res = await request(app).get('/users');

expect(res.body.length).toBe(1);

expect(res.body[0].email).toBe('anna@example.com');

});

**📊 7.6 Проверка покрытия (coverage)**

Добавь в package.json:

json

КопироватьРедактировать

"scripts": {

"test": "jest --coverage"

}

Запуск:

bash

КопироватьРедактировать

npm test

Ты получишь:

matlab

КопироватьРедактировать

File | % Stmts | % Branch | % Funcs | % Lines |

--------------|---------|----------|---------|---------|

routes/users.js | 100% | 100% | 100% | 100% |

**🛑 7.7 Тесты на ошибки и валидацию**

Допиши в users.test.js:

js

КопироватьРедактировать

test('Не должен создавать без email', async () => {

const res = await request(app).post('/users').send({ name: 'NoEmail' });

expect(res.statusCode).toBe(500);

});

**🧱 7.8 Unit-тест функции**

Если у тебя есть утилита:

📄 utils/capitalize.js

js

КопироватьРедактировать

module.exports = function capitalize(str) {

return str.charAt(0).toUpperCase() + str.slice(1);

}

📄 \_\_tests\_\_/capitalize.test.js

js

КопироватьРедактировать

const capitalize = require('../utils/capitalize');

test('capitalize работает', () => {

expect(capitalize('ivan')).toBe('Ivan');

});

**🧩 7.9 Best Practices**

| **Рекомендация** | **Почему** |
| --- | --- |
| Подключай MongoMemoryServer | Изоляция, без "реальной" Mongo |
| Удаляй данные beforeEach | Независимость тестов |
| Не запускай app.listen в тестах | Используй app, не сервер |
| Используй describe блоки | Группировка |
| Используй supertest | Быстро, легко, надёжно |

**✅ 7.10 Вывод главы**

Теперь ты можешь:

* Создавать Unit и Integration тесты
* Проверять API с Supertest
* Использовать временную MongoDB
* Проверять ошибки и валидацию
* Повышать надёжность проекта