Яндекс

Секреты строительства пирамид

Дмитрий Андриянов

Дмитрий Андриянов

- 7 лет в Яндексе
- 10 лет писал бэкенд на C#
- преподаю в Школе разработки интерфейсов
- консультирую по автотестам другие команды Яндекса


E2E
Integration testing
Unit testing

Яндекс

Автоматизированное тестирование single page веб-приложений

Дмитрий Андриянов

Яндекс

Автоматизированное тестирование single page веб-приложений

Дмитрий Андриянов

Яндекс

Автоматизированное тестирование single page веб-приложений

с примерами на react

Дмитрий Андриянов

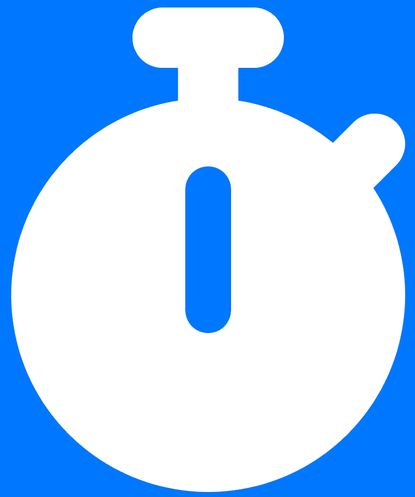
Яндекс

модульное тестирование

Автоматизированное тестирование single page веб-приложений

с примерами на react

Дмитрий Андриянов



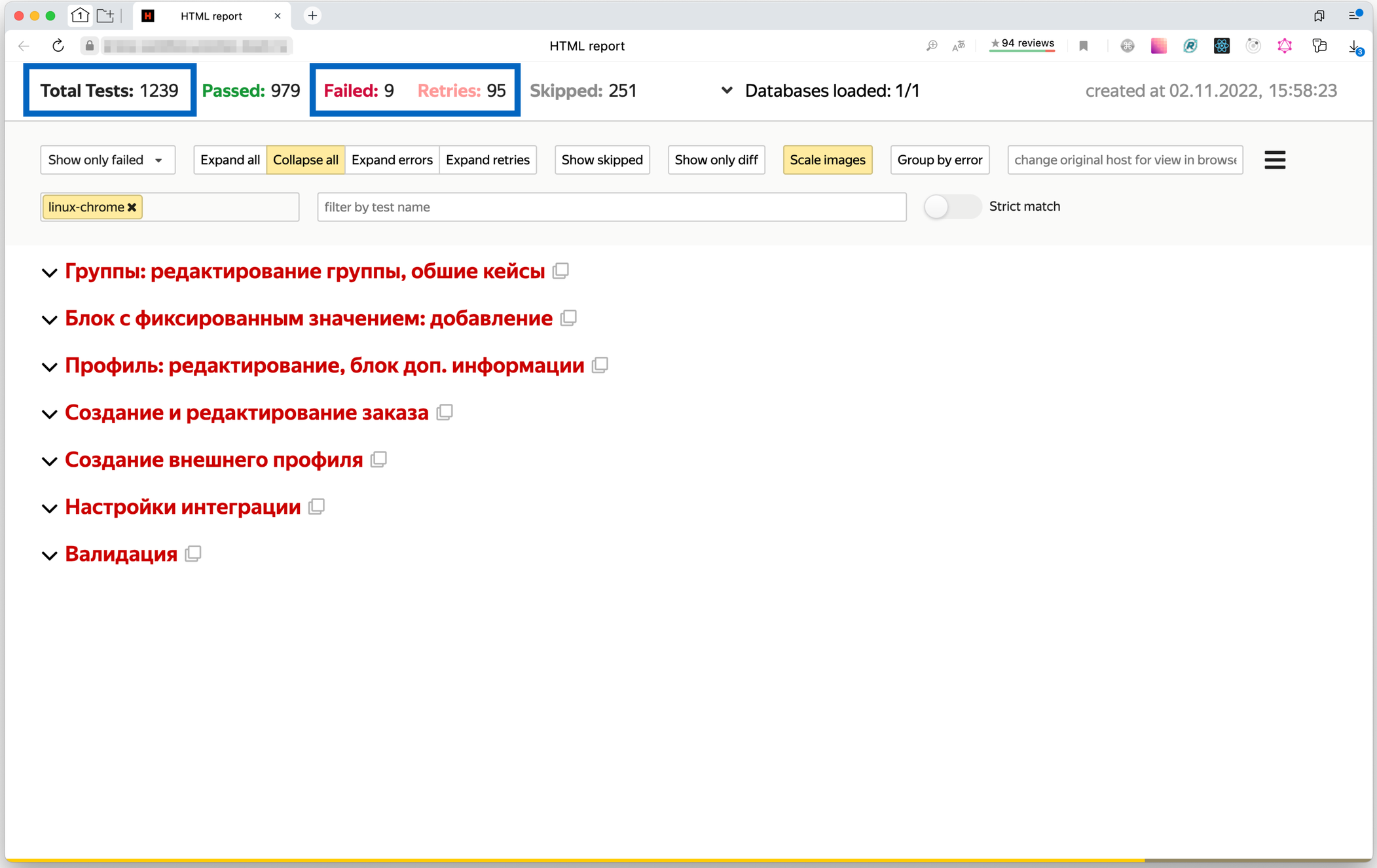
× 180

Автотесты

1. Проверяют продуктовые сценарии
2. Дёшево выполняют повторяющиеся проверки
3. Создают экземпляр тестируемой сущности и взаимодействуют с ним через GUI/API

Selenium

~2000 тестов

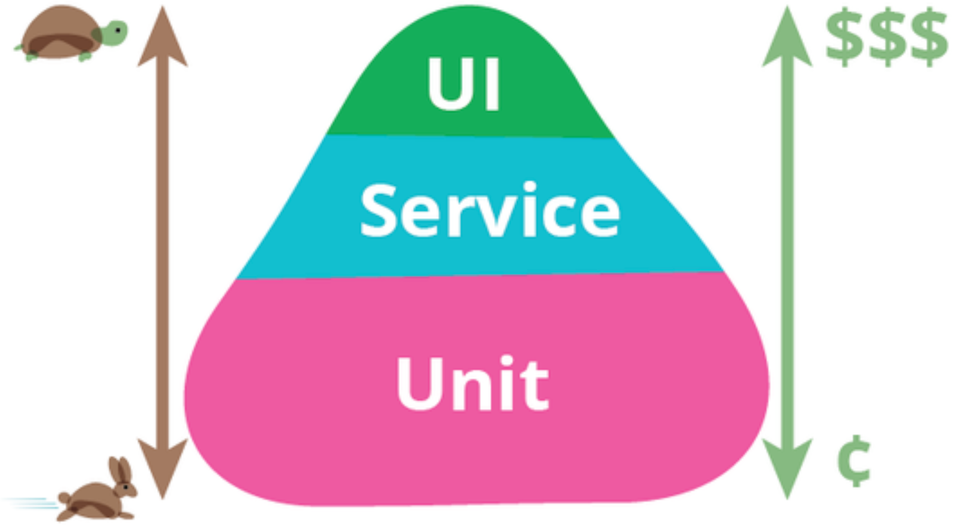

HTML report
HTML report
★94 reviews
Total Tests: 1239
Passed: 979
Failed: 9
Retries: 95
Skipped: 251
Databases loaded: 1/1
created at 02.11.2022, 15:58:23
Show only failed
Expand all
Collapse all
Expand errors
Expand retries
Show skipped
Show only diff
Scale images
Group by error
change original host for view in browse
linux-chrome
filter by test name
Strict match
Группы: редактирование группы, обшие кейсы
Блок с фиксированным значением: добавление
Профиль: редактирование, блок доп. информации
Создание и редактирование заказа
Создание внешнего профиля
Настройки интеграции
Валидация

1 May 2012

Martin Fowler

TESTING

The test pyramid is a way of thinking about how different kinds of automated tests should be used to create a balanced portfolio. Its essential point is that you should have many more low-level UnitTests than high level BroadStackTests running through a GUI.



Пример бесполезного модульного теста

```
it("новые элементы должны добавляться в список", () => {

    const prevState: Todo[] = [
        { text: "Купить молоко", done: true }
    ]

    const newState = todoReducer(prevState, todoAdded("Приготовить ужин"))

    expect(newState).toEqual([
        { text: "Купить молоко", done: true },
        { text: "Приготовить ужин", done: false },
    ])
});
```

jsdom - npm

www.npmjs.com

jsdom - npm

Norvell's Public Machinations

Products Pricing Documentation

npm

Search packages

Search

jsdom DT

20.0.2 • Public • Published 4 days ago

Readme | Explore BETA | 26 Dependencies | 5,679 Dependents | 246 Versions

jsdom

jsdom is a pure-JavaScript implementation of many web standards, notably the WHATWG DOM and HTML Standards, for use with Node.js. In general, the goal of the project is to emulate enough of a subset of a web browser to be useful for testing and scraping real-world web applications.

The latest versions of jsdom require Node.js v14 or newer. (Versions of jsdom below v20 still work with previous Node.js versions, but are unsupported.)

Basic usage

```
const jsdom = require("jsdom");
```

Install

```
npm i jsdom
```

Repository

github.com/jsdom/jsdom

Homepage

github.com/jsdom/jsdom#readme

Weekly Downloads

20,140,289

Version

20.0.2

License

MIT

Unpacked Size

Total Files

Пример бесполезного теста для GUI

```
import { render } from "@testing-library/react"
import pretty from "pretty"

it("проверяем разметку TodoItem", () => {

  const item = <TodoItem text="Сделать зарядку" checked={true} />

  const { container } = render(item)

  const html = pretty(container.outerHTML)

  expect(html).toMatchInlineSnapshot()
});
```

Пример бесполезного теста для GUI

```
import { render, screen } from "@testing-library/react"
import events from "@testing-library/user-event"

it("Клик по галочке вызывает обработчик клика", () => {
   const handleClick = jest.fn()

   render(<TodoItem text="Сделать зарядку" onClick={handleClick} />)

   events.click(screen.getByTestId("item-checkbox"))

   expect(handleClick).toBeCalled()
})
```

```
expect(umbrellaOpens).toBe(true)

tests: 1 passed, 1 total

**all tests passed**
```

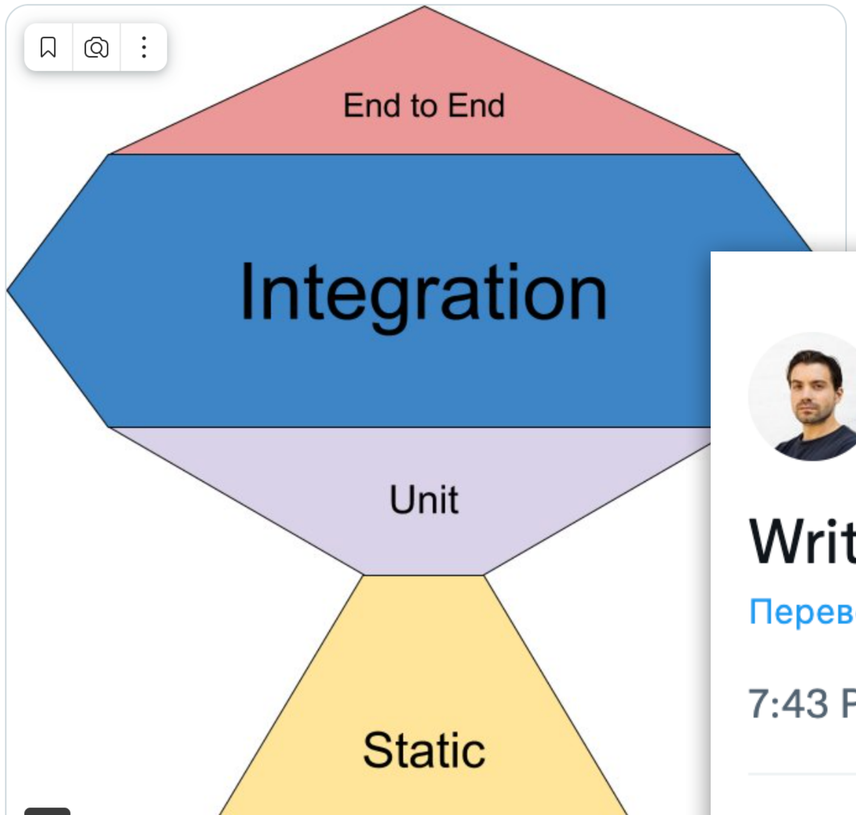
Kent C. Dodds
@kentcdodds

"The Testing Trophy" 🏆

A general guide for the **return on investment** 💰 of the different forms of testing with regards to testing JavaScript applications.

- End to end w/ @Cypress_io ⚫
- Integration & Unit w/ @fbjest 🃏
- Static w/ @flowtype *F* and @geteslint ⬢

Перевести твит



Lean Testing or Why U ×

medium.com Lean Testing or Why Unit Tests are Worse than You Think | by Eugen Kiss | Medium ★45 reviews

Get started

Search

Eugen Kiss

Jun 30, 2018 · 8 min read · Listen

Lean Testing or Why Unit Tests are Worse than You Think

Testing is a controversial topic. People have strong convictions about testing approaches. Test Driven Development is the most prominent example. Clear empirical evidence is missing which invites strong claims. I advocate for an **economic perspective** towards testing. Secondarily, I claim that focussing too much on unit tests is not the most economic approach. I coin this testing philosophy **Lean Testing**.

Eugen Kiss
421 Followers
eugenkiss.com

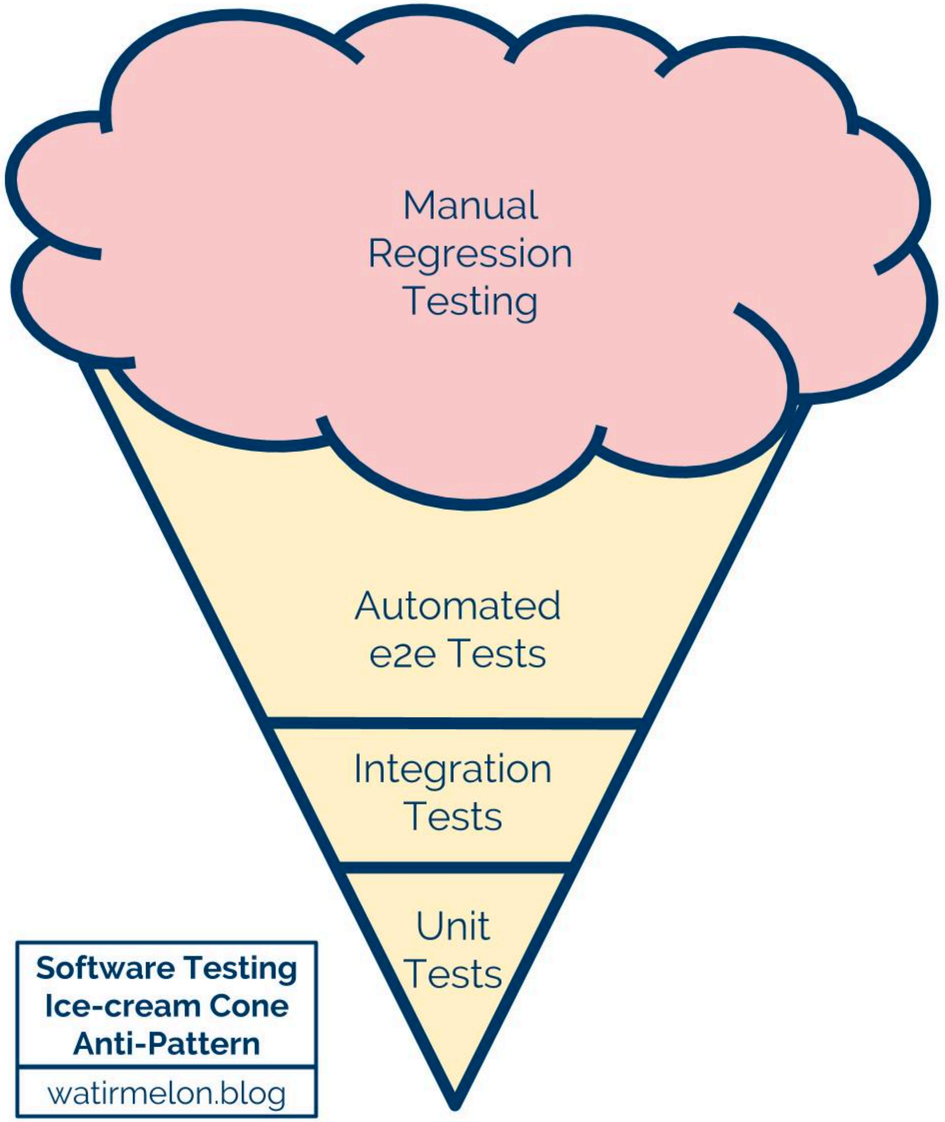
Follow

Guillermo Rauch
@rauchg

Write tests. Not too many. Mostly integration.

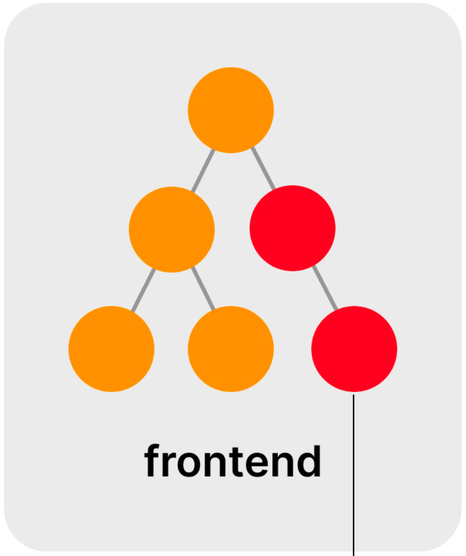
Перевести твит

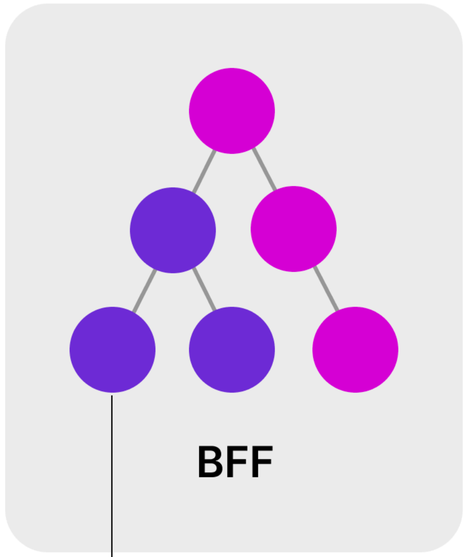
7:43 PM · 10 дек. 2016 г. из San Francisco, CA · Twitter Web Client

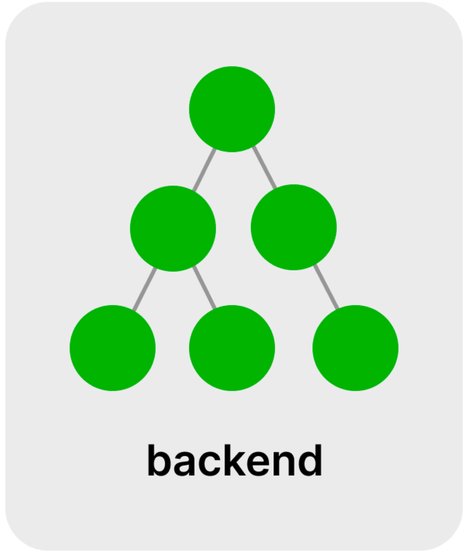


https://alisterbscott.com/kb/testing-pyramids

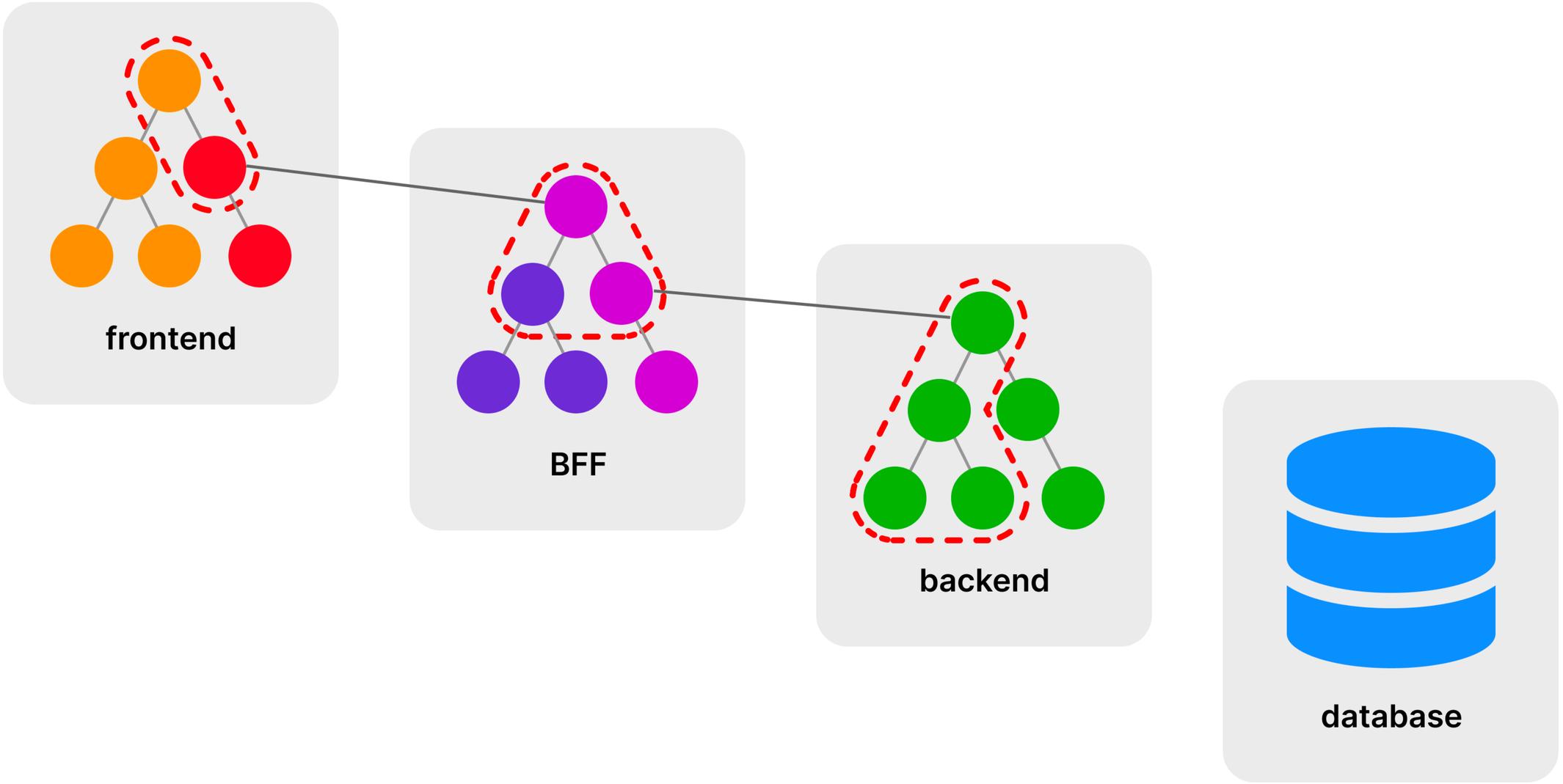

UI
Service
Unit
End to End
Integration
Unit
Static



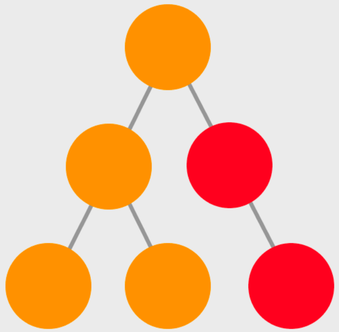




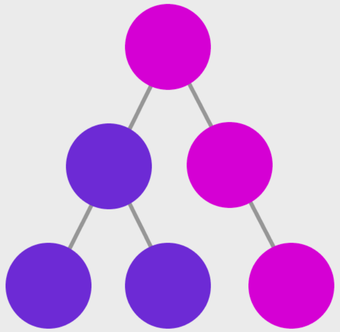
код сторонних библиотек


frontend
BFF
backend
database

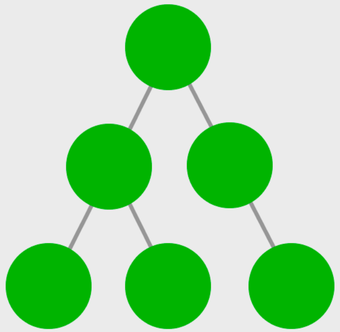
интеграционные тесты



frontend



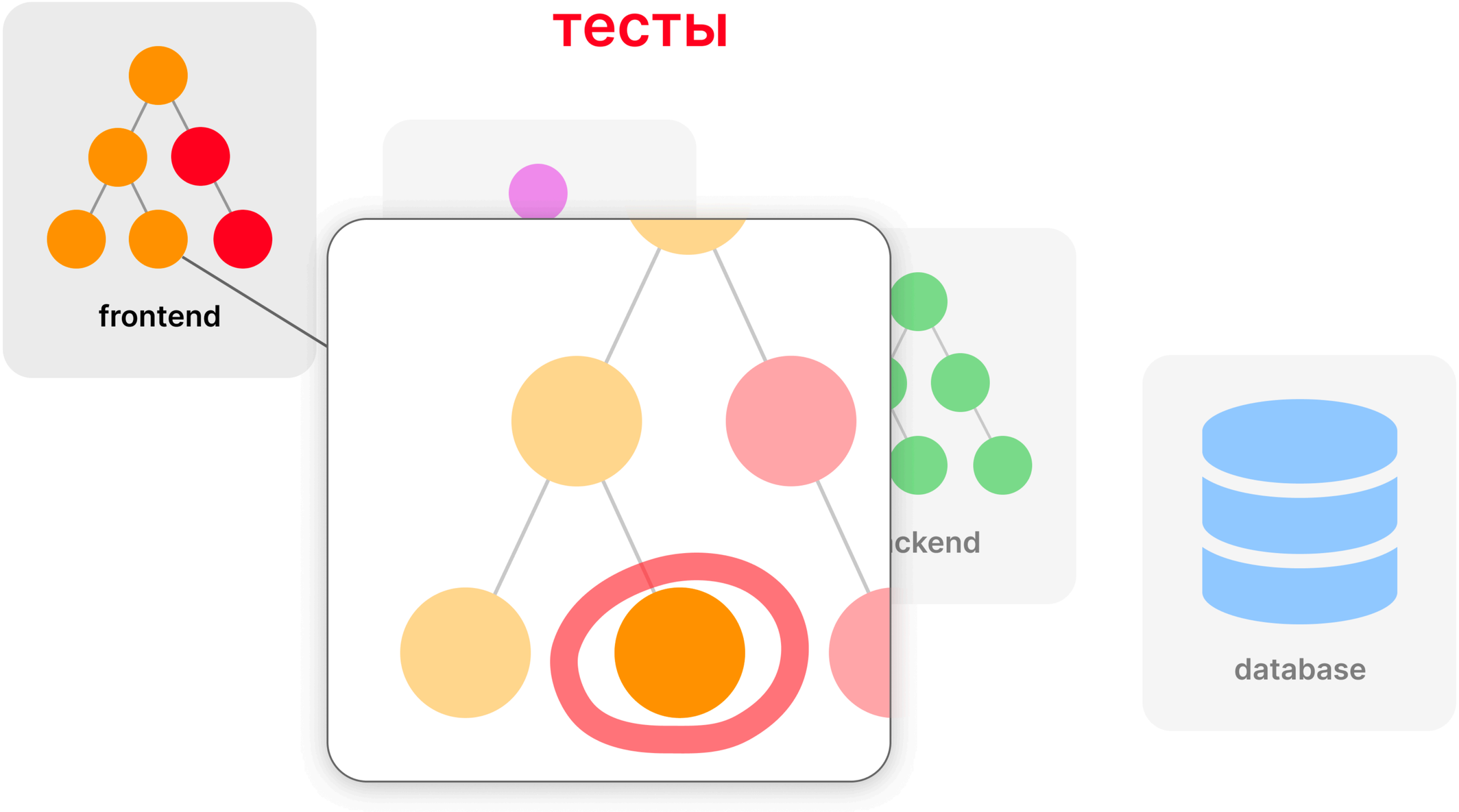
BFF



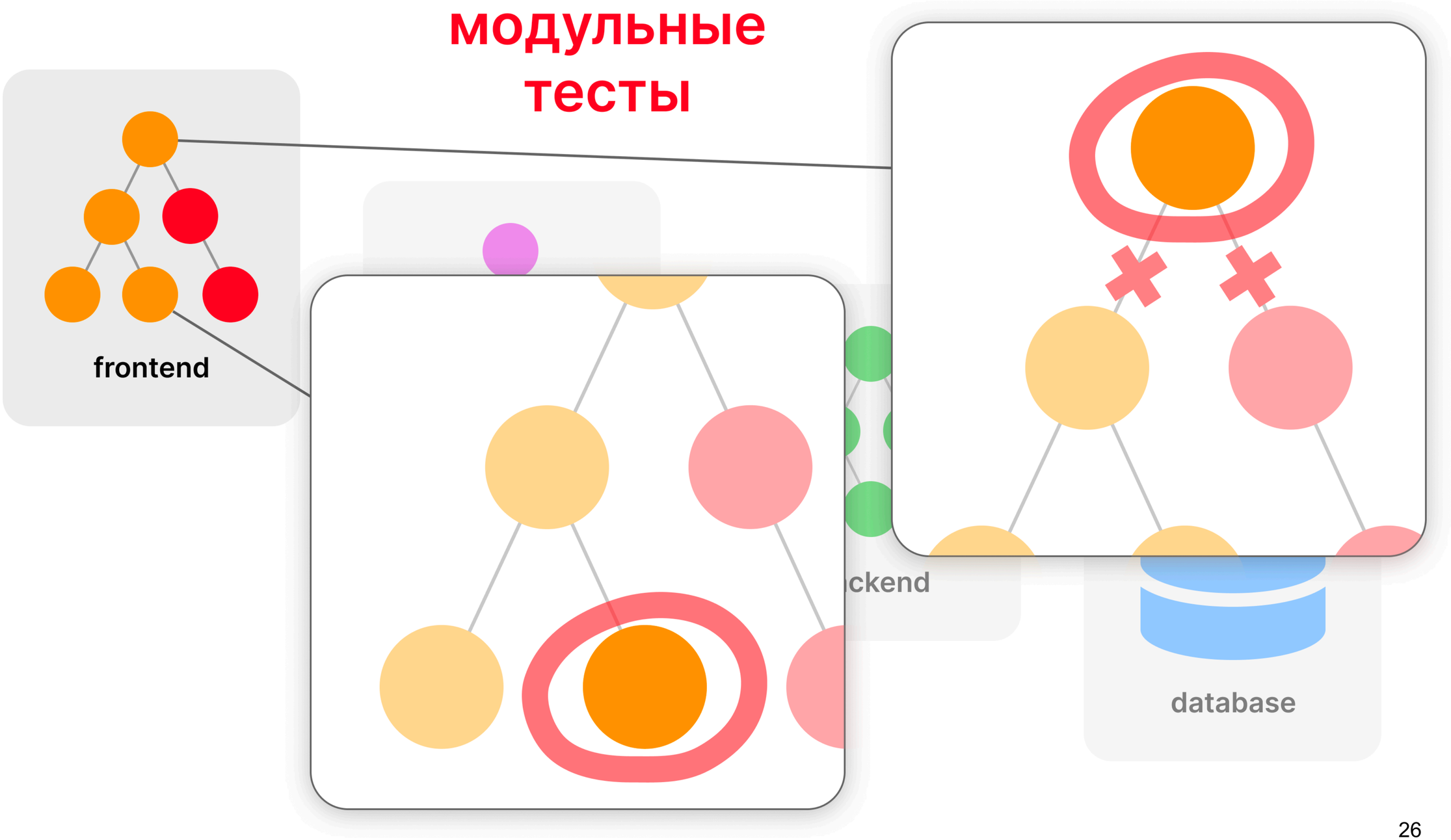
backend

database

модульные тесты

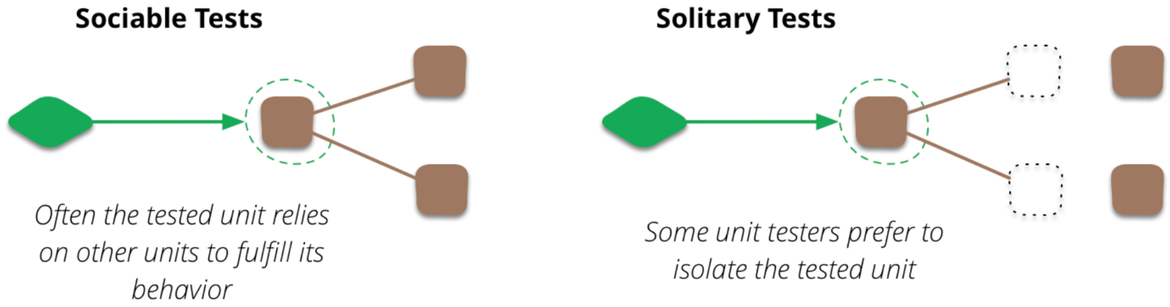


модульные тесты



Solitary or Sociable?

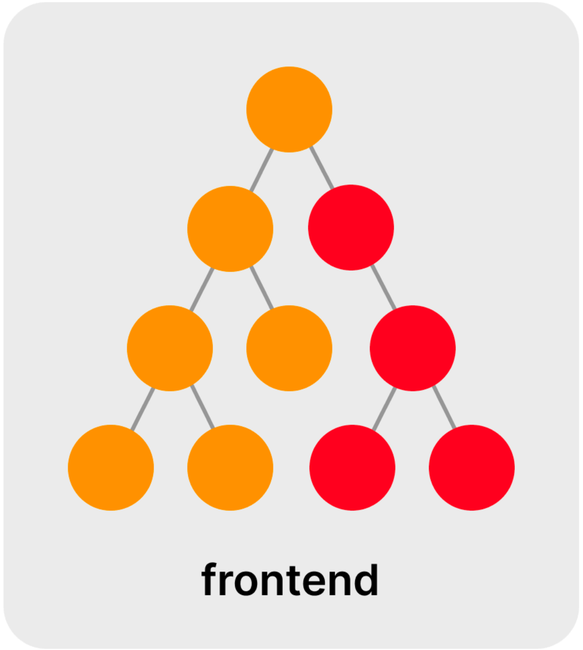
A more important distinction is whether the unit you're testing should be sociable or solitary [3]. Imagine you're testing an order class's price method. The price method needs to invoke some functions on the product and customer classes. If you like your unit tests to be solitary, you don't want to use the real product or customer classes here, because a fault in the customer class would cause the order class's tests to fail. Instead you use TestDoubles for the collaborators.

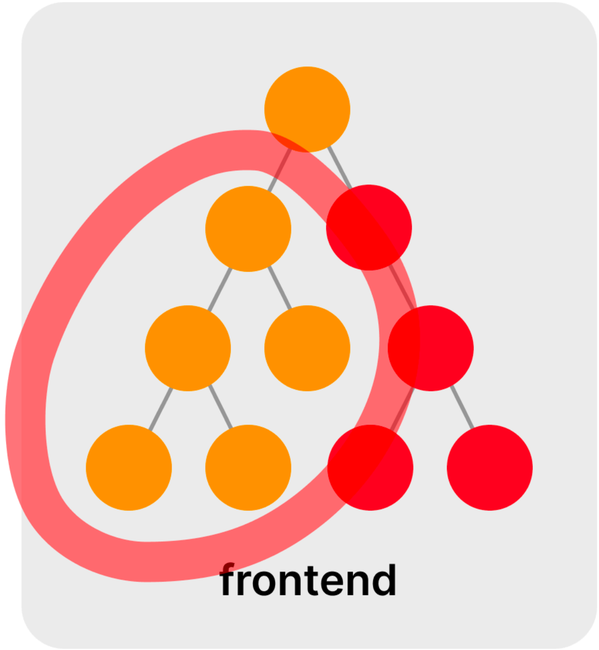


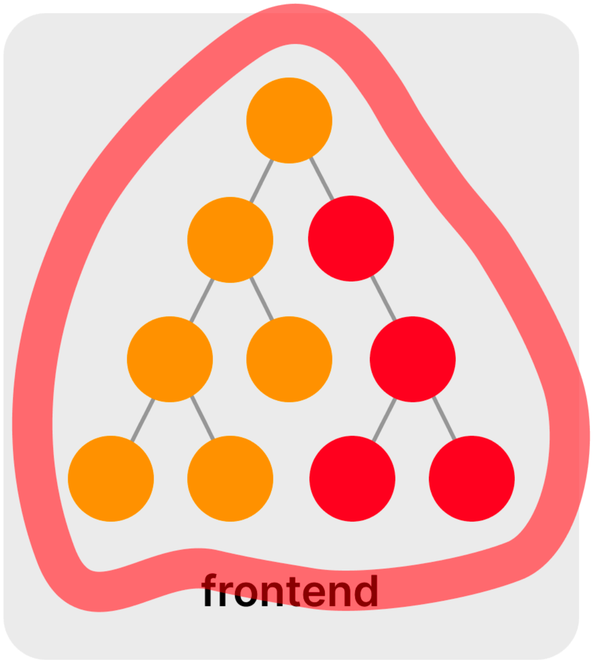
But not all unit testers use solitary unit tests. Indeed when xunit testing began in the 90's we made no attempt to go solitary unless communicating with the collaborators was awkward (such as a remote credit card verification system). We didn't find it difficult to track down the actual fault, even if it caused neighboring tests to fail. So we felt allowing our tests to be sociable didn't lead to problems in practice.

Indeed using sociable unit tests was one of the reasons we were criticized for our use of the term "unit testing". I think that the term "unit testing" is appropriate because

А что если отрендерить в модульном тесте целиком страницу приложения?


frontend


frontend


frontend

Модульный тест на всё приложение

```
import { render } from "@testing-library/react"
import { Application } from "./Application"

it("<название теста>", () => {
   // 1. рендерим всё приложение
   render(<Application />)

   // 2. взаимодействуем с приложением

   // 3. проверяем результат

   // 4. PROFIT
})
```

DEMO: проверяем unit-тестом продуктовый сценарий

https://github.com/dima117/unit-demo-cra

Внешние зависимости

это не только запросы по сети, но и...

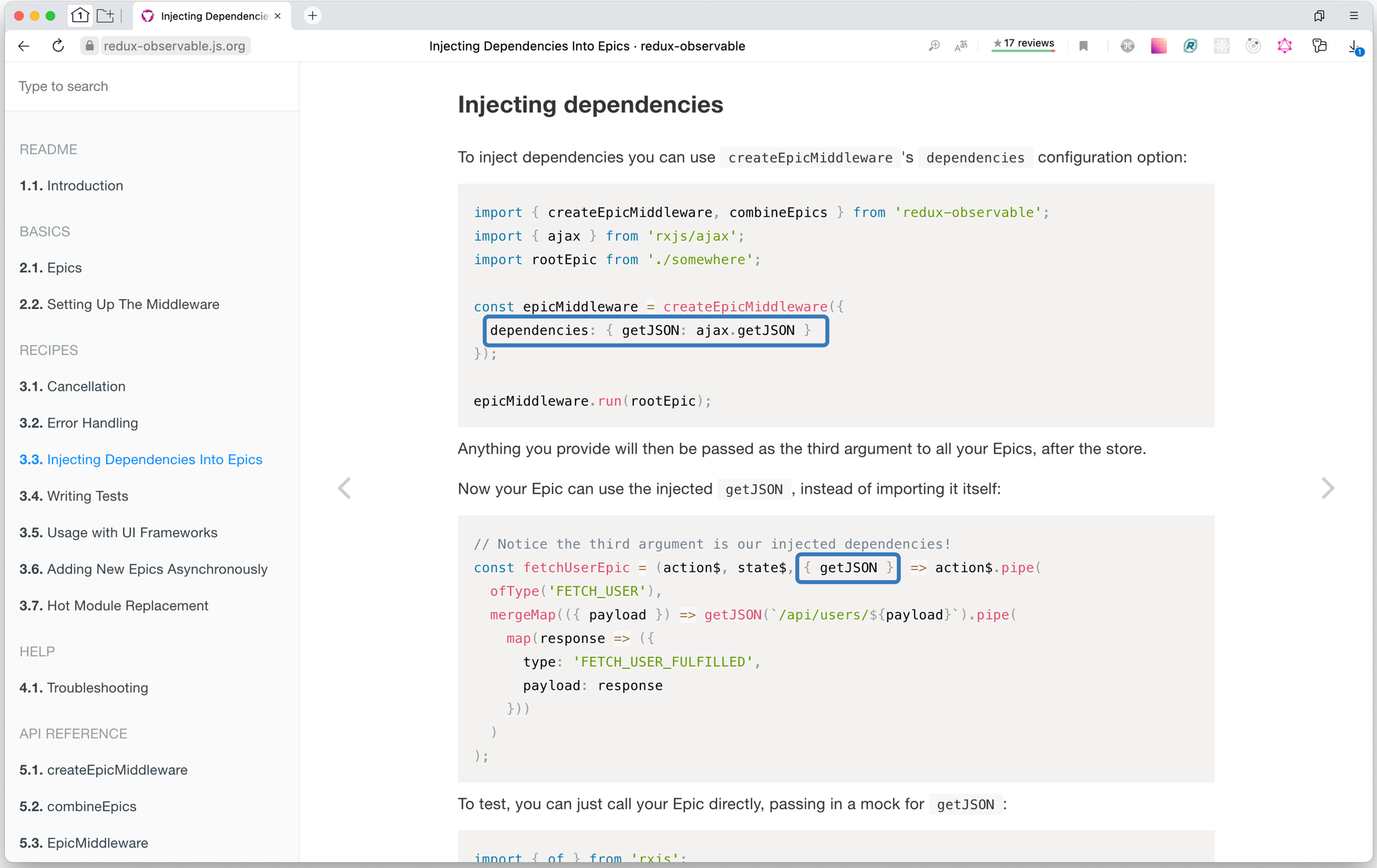
- сайд-эффекты (роутер, cookies, localStorage)
- Math.random
- системное время
- ... а ещё **глобальные переменные**

Избавляемся от глобальных переменных

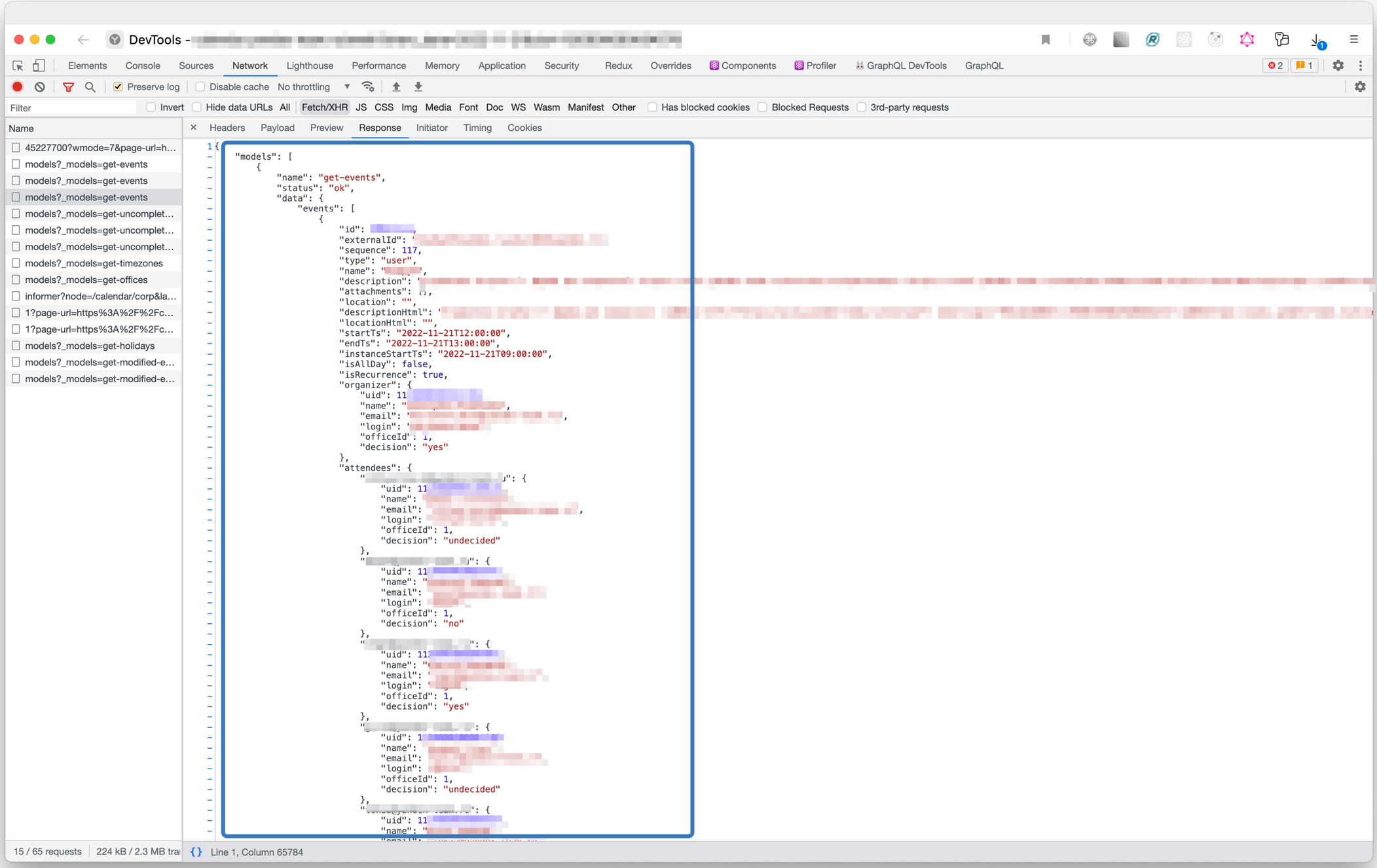
```
// было

import api from './api'

export const MyComponent = () => {
   const [data, setData] = useState()

   useEffect(() => {
     api.loadData().then(setData)
   }, [])

   // ...
}
```

Избавляемся от глобальных переменных

```
// стало

export const MyComponent = () => {
    const [data, setData] = useState()

    const api = useContext(MyContext)

    useEffect(() => {
        api.loadData().then(setData)
    }, [])

    // ...
}
```


Injecting Dependencie
redux-observable.js.org
Injecting Dependencies Into Epics · redux-observable
17 reviews
Type to search
README
1.1. Introduction
BASICS
2.1. Epics
2.2. Setting Up The Middleware
RECIPES
3.1. Cancellation
3.2. Error Handling
3.3. Injecting Dependencies Into Epics
3.4. Writing Tests
3.5. Usage with UI Frameworks
3.6. Adding New Epics Asynchronously
3.7. Hot Module Replacement
HELP
4.1. Troubleshooting
API REFERENCE
5.1. createEpicMiddleware
5.2. combineEpics
5.3. EpicMiddleware
Injecting dependencies
To inject dependencies you can use createEpicMiddleware 's dependencies configuration option:
import { createEpicMiddleware, combineEpics } from 'redux-observable';
import { ajax } from 'rxjs/ajax';
import rootEpic from './somewhere';
const epicMiddleware = createEpicMiddleware({
dependencies: { getJSON: ajax.getJSON }
});
epicMiddleware.run(rootEpic);
Anything you provide will then be passed as the third argument to all your Epics, after the store.
Now your Epic can use the injected getJSON , instead of importing it itself:
// Notice the third argument is our injected dependencies!
const fetchUserEpic = (action$, state$, { getJSON }) => action$.pipe(
ofType('FETCH_USER'),
mergeMap(({ payload }) => getJSON(`/api/users/${payload}`).pipe(
map(response => ({
type: 'FETCH_USER_FULFILLED',
payload: response
}))
)
);
To test, you can just call your Epic directly, passing in a mock for getJSON :
import { of } from 'rxjs';

Думайте о тестировании, когда проектируете архитектуру и выбираете библиотеки


DevTools -
Elements
Console
Sources
Network
Lighthouse
Performance
Memory
Application
Security
Redux
Overrides
Components
Profiler
GraphQL DevTools
GraphQL
Preserve log
Disable cache
No throttling
Filter
Invert
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
3rd-party requests
Name
45227700?wmode=7&page-url=h...
models?_models=get-events
models?_models=get-events
models?_models=get-events
models?_models=get-uncomplet...
models?_models=get-uncomplet...
models?_models=get-uncomplet...
models?_models=get-timezones
models?_models=get-offices
informer?node=/calendar/corp&la...
1?page-url=https%3A%2F%2Fc...
1?page-url=https%3A%2F%2Fc...
models?_models=get-holidays
models?_models=get-modified-e...
models?_models=get-modified-e...
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
{
"models": [
{
"name": "get-events",
"status": "ok",
"data": {
"events": [
{
"id":
"externalId":
"sequence": 117,
"type": "user",
"name":
"description":
"attachments":
"location": "",
"descriptionHtml":
"locationHtml": "",
"startTs": "2022-11-21T12:00:00",
"endTs": "2022-11-21T13:00:00",
"instanceStartTs": "2022-11-21T09:00:00",
"isAllDay": false,
"isRecurrence": true,
"organizer": {
"uid": 11
"name":
"email":
"login":
"officeId":
"decision": "yes"
},
"attendees": {
"uid": 11
"name":
"email":
"login":
"officeId": 1,
"decision": "undecided"
},
"uid": 11
"name":
"email":
"login":
"officeId": 1,
"decision": "no"
},
"uid": 11
"name":
"email":
"login":
"officeId": 1,
"decision": "yes"
},
"uid": 1
"name":
"email":
"login":
"officeId": 1,
"decision": "undecided"
},
"uid": 11
"name":
15 / 65 requests
224 kB / 2.3 MB tra
Line 1, Column 65784

Заглушки для данных

```ts
interface User {
    id: number;
    login: string;
    first_name_ru: string;
    last_name_ru: string;
    country: Country;

    // ...
    // много других полей
}
```

```ts
interface Country {
    code: string;
    group_id: number;
    name_ru: string;

    // ...
    // много других полей
}
```

Заглушки для данных

```
const getUser = (): User => {
    return {
        id: 123,
        login: "test",
        first_name_ru: "Иван",

        // много других полей
    }
}
```

Заглушки для данных

```
const getUser = (): User => {
    return {
        id: 123,
        login: "test",
        first_name_ru: "Иван",
        country: getCountry(),

        // много других полей
    }
}
```

```
const getCountry = (): Country => {
    return {
        code: "ru",
        name_ru: "Россия",

        // много других полей
    }
}
```

Заглушки для данных

```ts
const getUser = (
    data?: Partial<User>
): User => {
    return {
        id: 123,
        login: "test",
        first_name_ru: "Иван",
        country: getCountry(),

        // много других полей

        ...data,
    }
}
```

```ts
const getCountry = (
    data?: Partial<Country>
): Country => {
    return {
        code: "ru",
        name_ru: "Россия",

        // много других полей

        ...data,
    }
}
```

Заглушки для данных

```
it("название теста", () => {

   const user = getUser({
      login: "cow17",
   });

   // ...
});
```

Заглушки для данных

```
it("название теста", () => {

   const user = getUser({
      login: "cow17",
      country: getCountry({ code: "kg" }),
   });

   // ...
});
```

Заглушки для данных

```
it("название теста", () => {

   const user = getUser({
      login: "cow17",
      country: getCountry({ code: "kg" }),
   });

   // ...

   expect(loginLink).toHaveTextContent("@cow17");
   expect(selectCountry).toHaveAttribute("value", "kg");
});
```

Резюме

Выводы

1. Получили новый инструмент для тестирования продуктовых сценариев в памяти, без реального браузера — тесты надёжнее и работают в сотни раз быстрее

2. Этим способом нельзя проверить вёрстку и сценарии, затрагивающие несколько слоёв системы — такие сценарии проверяем тестами в браузере, как раньше

3. Получилось распределение сценариев по видам тестирования, как в Пирамиде


E2E
Integration testing
Unit testing

Что дальше?

1. Посмотрите код примера и попробуйте запустить
   https://github.com/dima117/unit-demo-cra

2. Попросите разработчиков написать один такой тест в вашем проекте

3. Приходите в дискуссионную зону
   (а если возникнут вопросы позже, то напишите мне)

Полезные ссылки

- Лекция по автотестам в Школе разработки интерфейсов Яндекса
  https://www.youtube.com/watch?v=DFLXBdfnAeE
- Статьи Мартина Фаулера
  https://martinfowler.com/articles/2021-test-shapes.html
  https://martinfowler.com/bliki/UnitTest.html
- Блог про автотесты, React, TypeScript
  https://kentcdodds.com/blog
- Testing Library
  https://testing-library.com/docs

Яндекс

Спасибо

Дмитрий Андриянов

Разработчик интерфейсов

 dima117a@gmail.com

 @dima117a