Тестирование, как точка личного роста

Панфилов Александр

Обо мне

- 6 лет в IT разработке, последние 4 года на позиции фронтенд разработчика
- Любимые технологии: React, Next, Nest, CI/CD, MongoDB, MySQL

О чем поговорим

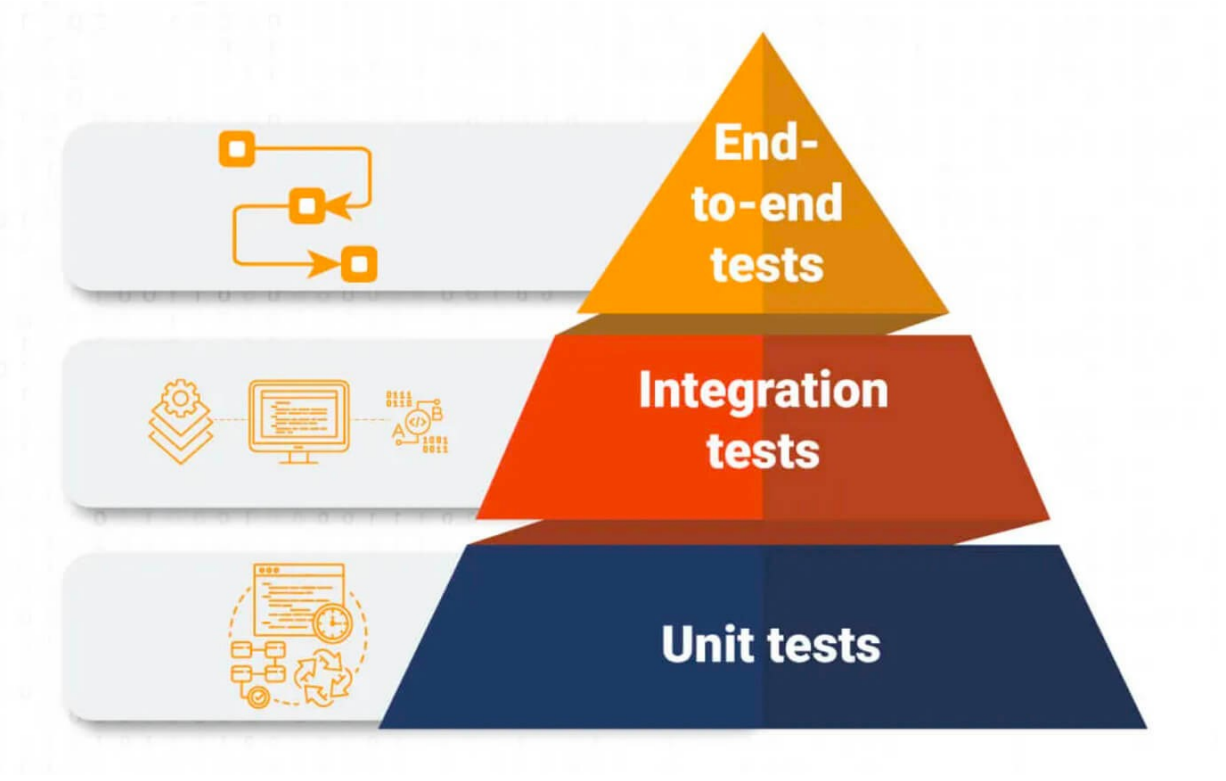
- Что такое тестирование и для чего оно нужно в проекте?
- Дерево тестирования.
- Технологии и среда разработки для тестов.
- Почему подходы разработки TDD и BDD не жизнеспособны?
- Что дают тесты в проекте, помимо надежности? Как они помогают развиваться и расти профессионально?
- Особенности тестирования клиентских и серверных приложений.
- Выводы.

Что такое тестирование и для чего оно нужно в проекте?

Тестирование (официальный термин) - это процесс исследования, испытания программного продукта

Тестирование - это проверка моего кода на корректность выполнения кода и помощь в том,что мои изменения не затронули другие места программы и ничего лишнего не сломали.

Дерево тестирования



Пример подставленных данных для юнит теста

```
jest.mock("react-hook-form", () => {
  const mockModule = jest.requireActual("react-hook-form");

  return {
    ...mockModule,
    useForm: jest.fn().mockReturnValue({
      register: jest.fn(),
      handleSubmit: jest.fn(),
      formState: {
        errors: {
          usernameOrEmail: null,
        },
      },
      control: {},
    }),
    Controller: ({ name }: { name: string }) => <span>{name}</span>,
  };
});
```

Пример юнит теста на примере юай приложения

```
Run | Debug
describe("Signin", () => {
  Run | Debug
  it("check render component", async () => {
    const { findByText, findByRole } = render(<Signin />);

    expect(await findByText(/Авторизация в чате/i)).toBeInTheDocument();
    expect(await findByRole("textbox")).toBeInTheDocument();
    expect(await findByText(/Username\/email/i)).toBeInTheDocument();
    // второй InputWrapperError находится в пропсах, поэтому не отрисовывается в тестах
    expect(await findByText(/InputWrapperError/i)).toBeInTheDocument();
    expect(await findByText("password")).toBeInTheDocument();
    expect(await findByText("Отправить")).toBeInTheDocument();
    expect(await findByText("Регистрация")).toBeInTheDocument();
  });
});
```

Подставленные данные для юнит теста серверного приложения

```
const roomsRepository = jest.fn().mockReturnValue({
  getRoomsByInterlocutor: jest.fn().mockResolvedValue('getRoomsByInterlocutor'),
  getRoomById: jest.fn().mockResolvedValue('getRoomById'),
});
```

Инициализация данных на базе подставленных данных

```
beforeEach(async () => {
  const module = await Test.createTestingModule({
    providers: [
      RoomsService,
      {
        provide: RoomsRepository,
        useFactory: roomsRepository,
      },
    ],
  }).compile();

  roomsService = module.get<RoomsService>(RoomsService);
});

afterEach(() => jest.clearAllMocks());
```

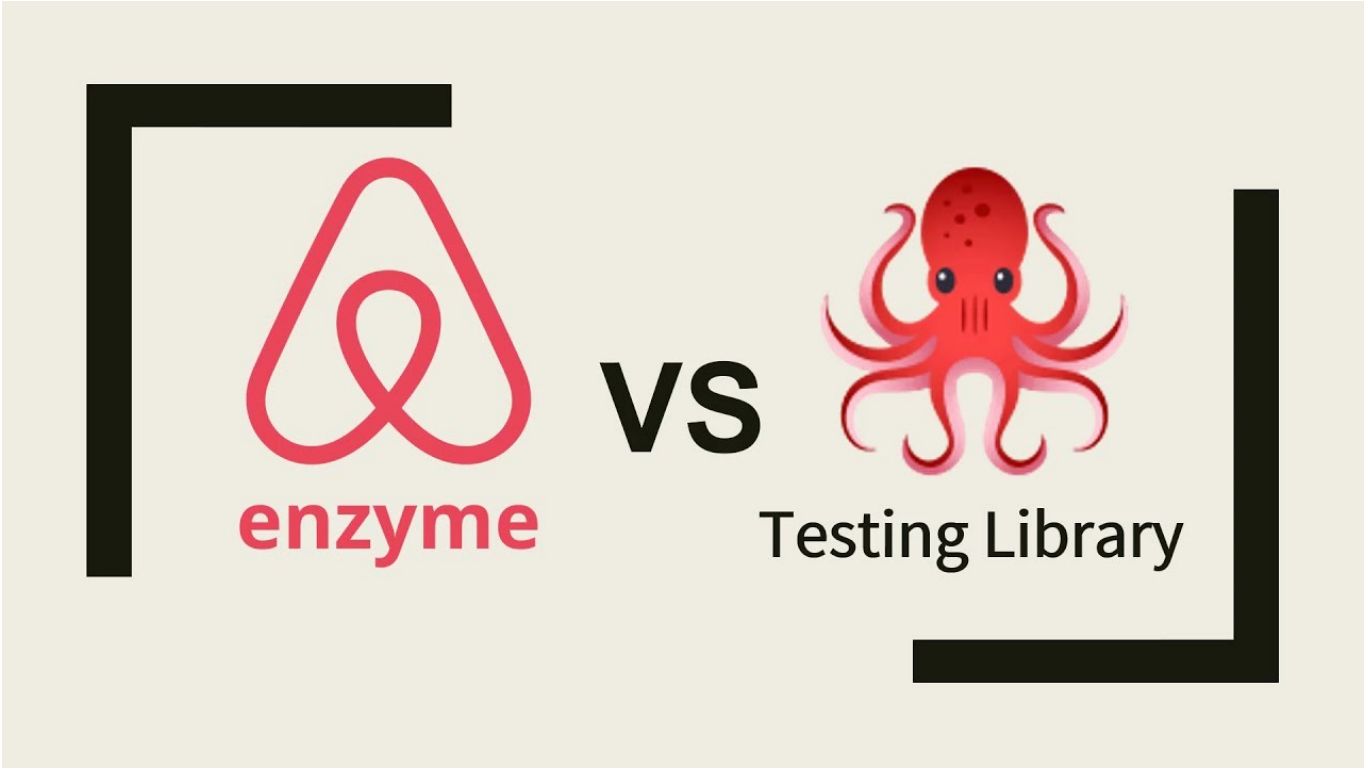
Юнит тесты серверного приложения

```
Run | Debug
it('getRooms', async () => {
  const result = await roomsService.getRooms({ userId: 'userId' });

  expect(roomsService.getRooms).toBeDefined();
  expect(result).toBe('getRoomsByInterlocutor');
});

Run | Debug
it('getRoomById', async () => {
  const result = await roomsService.getRoomById({ roomId: '111' });

  expect(roomsService.getRoomById).toBeDefined();
  expect(result).toBe('getRoomById');
});
```

Среда разработки

- Jest — разработана под react. Быстрая, легко настраиваемая, не требует сторонних программ для запуска тестов.
- Mocha был разработан для тестирования приложений, работающих на Node.js. Он обеспечивает гибкую и расширяемую среду тестирования. Он хорошо подходит для сложных задач тестирования.
- Jasmine — среда тестирования для небольших проектов, используется в тестировании angular, требует сторонние программы для запуска тестов, например Kafka.

Библиотеки для написания тестов



Разница между react test library (RTL) и enzyme

- Enzyme предоставляет низкоуровневый подход к инструментам и доступ к внутренностям компонента;
- RTL предназначена для тестирования компонентов с использованием как можно меньшего количества их реализации

Пример теста, написанного на enzyme

```
import React from "react";
import Post from "./post";

const setUp = (props) => shallow(<Post {...props} />);
```

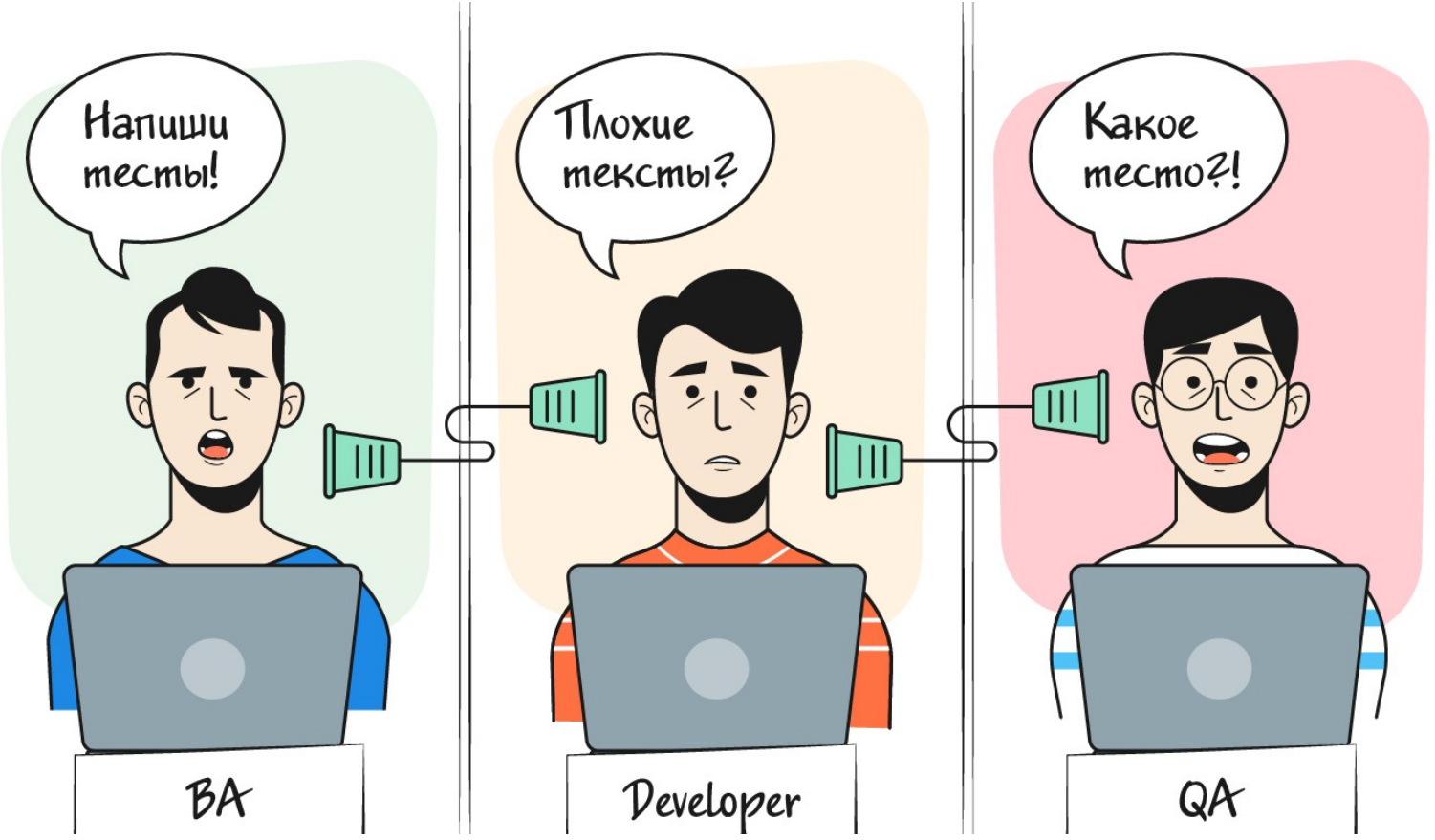
```
describe("should render Post component", () => {
  let component;
  beforeEach(() => {
    component = setUp();
  });

  it("should contain .post wrapper", () => {
    const wrapper = component.find(".post");
    expect(wrapper.length).toBe(1);
  });

  it("should contain link", () => {
    const wrapper = component.find("a");
    expect(wrapper.length).toBe(1);
  });

  it("should render created date", () => {
    const created_at = "01-03-2020";
    component = setUp({ created_at });
    const date = component.find(".date");
    expect(date.text()).toBe(new Date(created_at).toLocaleDateString());
  });
});
```

Низкоуровневая реализация enzyme

```
const DEFAULT_PAGE = 10;
let component;
let instance;

beforeEach(() => {
  component = setUp();
  instance = component.instance();
});
```

```
it("should handle search input value", () => {
  expect(component.state().searchQuery).toBe("");
  instance.handleInputChange({ target: { value: "test" } });
  expect(component.state().searchQuery).toBe("test");
});

it("should handle change of hits per page", () => {
  expect(component.state().hitsPerPage).toBe(20);
  instance.handleHitsChange({ target: { value: String(DEFAULT_PAGE) } });
  expect(component.state().hitsPerPage).toBe(DEFAULT_PAGE);
});

it("should handle change page if 'Enter' clicked", () => {
  instance.setState({ page: DEFAULT_PAGE });
  instance.getSearch({ key: "Enter" });
  expect(component.state().page).toBe(0);
});

it("should not change page if 'a' button clicked", () => {
  instance.setState({ page: DEFAULT_PAGE });
  instance.getSearch({ key: "a" });
  expect(component.state().page).toBe(DEFAULT_PAGE);
});
```

Почему TDD и BDD не работает



Почему TDD и BDD не работает

- Аналитика (тех. радание) редко сразу полностью описывают весь функционал;
- Разработчик может не правильно понять функционал;
- Изменение функционала по ходу разработки;
- Уровень команды.

Как тесты помогают расти и профессионально развиваться?

- Лучше понимаешь работу компонента и его узкие места;
- Прокачивается навык декомпозиции компонентов и понимания архитектуры;
- Повышается насмотренность на работу компонентов и увеличиваешь в последствии скорость разработки.
- Прогнозируешь четкие эстимейты по задачам.
- Более глубже понимаешь работу языка программирования.

Разница в тестировании между ui и бекенд приложеним

Клиентское приложение	Серверное приложение
Компонент	Класс
Мокаются все внешние источники данных (хуки, селекторы и т.п.)	Мокаются все внешние слои
Проверяется отрисовка и простой функционал	Проверяется наличие метода и его функционал

Выводы

- К тестированию нужно относится не только как к проверке самого себя, а и как к поиску своих слабых точек;
- Есть три среды разработки: jest, mocha, jasmine. Jest используется, в основном в react, mocha — node.js, jasmine — angular.
- TDD и BDD — не применимы в реальной жизни.
- Написание тестов помимо проверки кода, позволяют развить насмотренность и найти точки роста у себя.
- Принципы написания тестов одинаковы для, как для клиентского приложения, так и для бекенд приложения.

Спасибо за внимание

Блог

Канал на дзене

Канал в телеграме