

Тестирование микросервисов

Golang Developer. Professional



Проверить, идет ли запись

Меня хорошо видно & слышно?







Хохлов Александр

Архитектор платформенных решений в ГК Иннотех

- **25+** лет в Информационных технологиях, последние 10 лет - архитектура и все что вокруг нее
- Занимаюсь вопросами проектирования, разработки (db/back/front/mobile), инфраструктуры, управления и развития технических команд.
- Преподаватель курсов:
 - Golang Developer. Basic
 - Golang Developer. Professional
 - System Design
 - Highload Architecture

 @khorost

 alexander.khokhlov@gmail.com

Правила вебинара



Активно
участвуем



Off-topic обсуждаем
в учебной группе **#канал**
группы



Задаем вопрос
в чат или голосом



Вопросы вижу в чате,
могу ответить не сразу

Карта курса



СТАРТ

Начало работы с Go

Concurrency в Go

Работа с сетью и БД

Стандартные библиотеки и практики

Микросервисы

Проектная работа

ФИНИШ



Маршрут вебинара

Юнит-тестирование vs интеграционное

Подходы к тестированию (TDD, BDD)

Примеры



Подходы к тестированию

Зачем?

Зачем нужны тесты?



Зачем нужны тесты

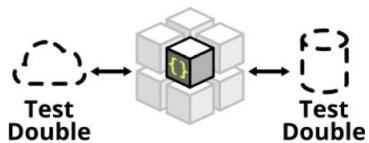
- Упрощают рефакторинг.
- Документируют код.
- Отделение интерфейса от реализации (mocks), менее связный код.
- Помогают найти неактуальный код.
- Помогают найти новые кейсы.
- Считают метрику для менеджмента (покрытие).
- Определяют контракт.
- Повышают качество кода.
- Придают уверенности при деплое в продакшен.



Думай, как тестировщик

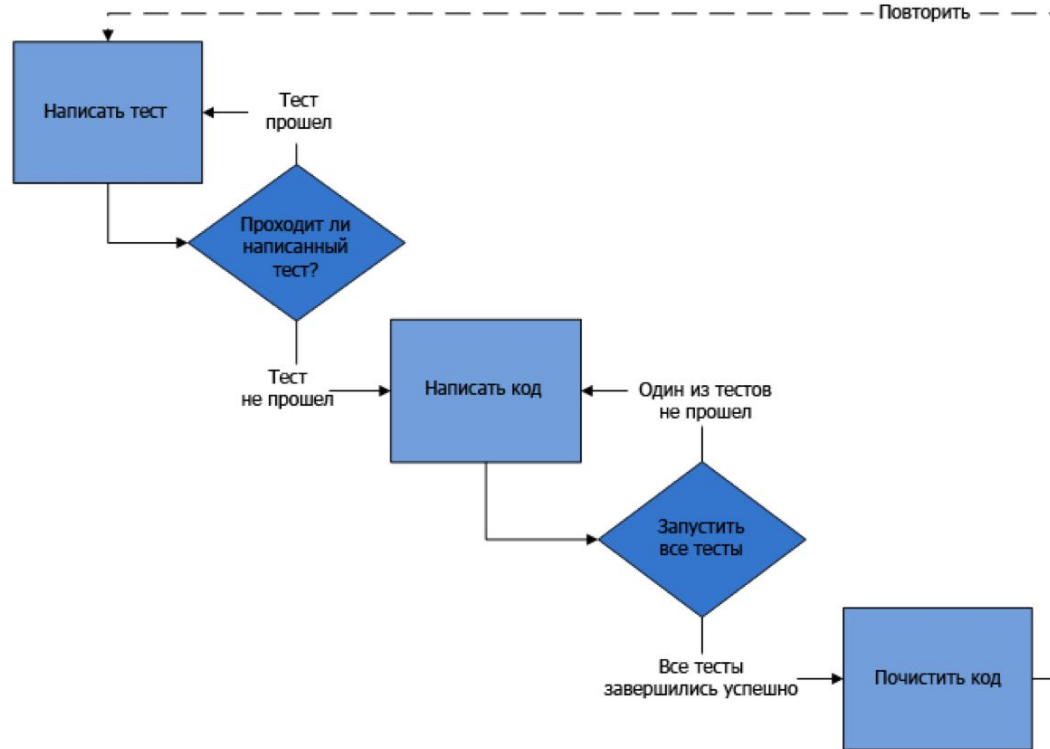
- Как хотелось бы, чтобы работало? (На что это похоже? Как бы я мог это использовать?) Не лазить в кишки.
- Как не должно работать? (Неправильные параметры, неправильный порядок вызовов) Негативные тест-кейсы.
- Что там на краю обрыва? (Самое маленькое/большое число, граница, на которой меняется состояние). Граничные условия.
- А что если? Странные сценарии использования.

Модульные VS Интеграционные



Тесты	Цель	Требует	Скорость	Сложность	Нужна настройка
Юнит-тесты	класс/метод	исходный код	очень быстро	низкая	нет
Интеграционные тесты	компонент/сервис	часть работающей системы	медленно	средняя	да

TDD Test-Driven Development



TDD: исправление багов

Ситуация:

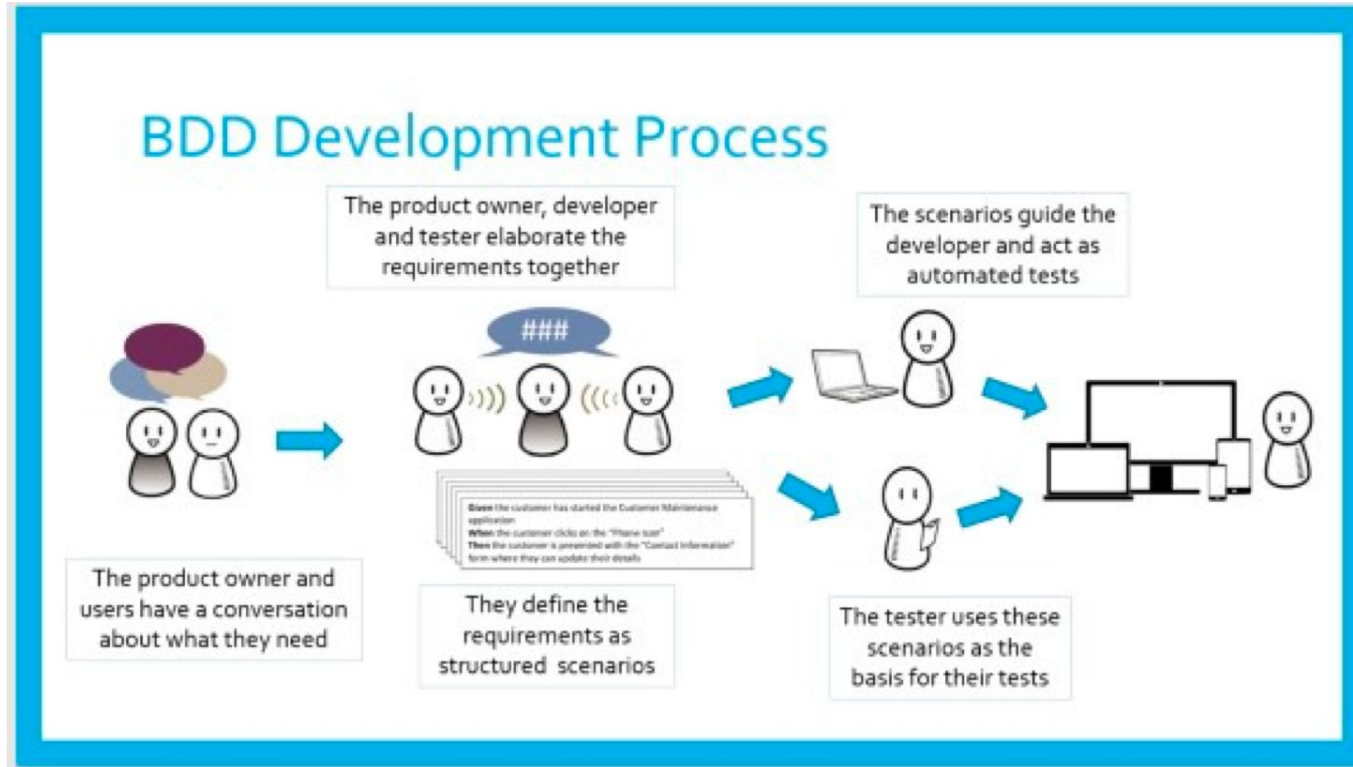
- Вам сообщили о проблеме и вы экстренно её решили.
- Написали тест на решение и он прошёл.
- Через какое-то время проблема повторилась.

Почему?



Behavior-Driven Development

BDD Behavior-Driven Development



BDD: язык Gherkin

Feature: Guess the word

Scenario: Maker starts a game

When the Maker starts a game

Then the Maker waits for a Breaker to join

Scenario: Breaker joins a game

Given the Maker has started a game with the word "silky"

When the Breaker joins the Maker's game

Then the Breaker must guess a word with 5 characters



BDD Behavior-Driven Development

- Похож на TDD.
- Описание идёт через спецификацию поведения.
- Стандарт для спецификации de facto – язык Gherkin.
- Наиболее известная компания, продвигающая фреймворки для BDD - Cucumber.
- BDD придуман, чтобы бизнесу был ближе к программистам. (как на самом деле?)



BDD: возможный тест на Gherkin

История: Отсылка email-уведомления

Как клиент API сервиса регистрации

Чтобы понимать, что пользователю приходит подтверждение регистрации

Я хочу получать события из соответствующей очереди

Сценарий: Получаем событие от сервиса уведомлений

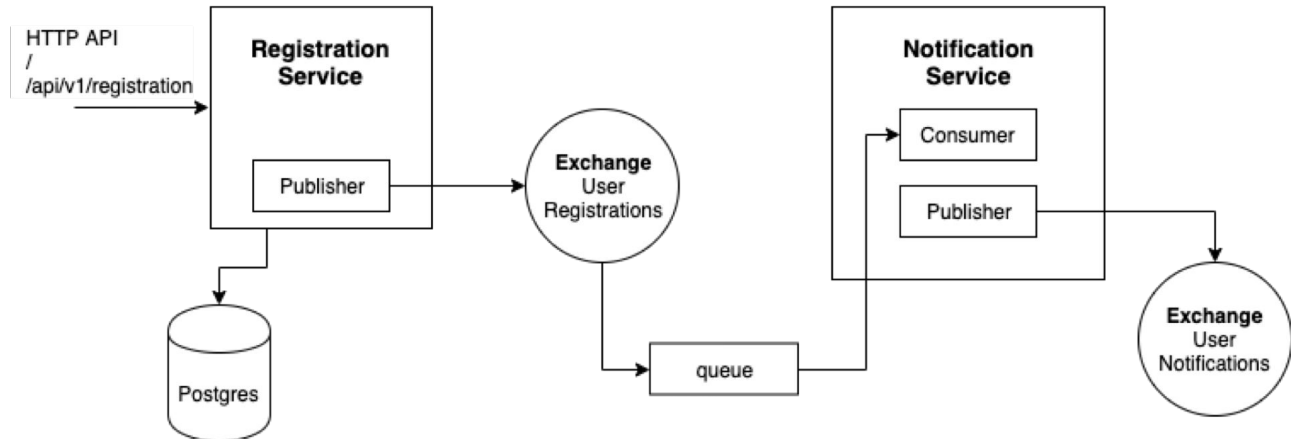
Когда я отсылаю POST-запрос с пользовательским JSON в сервис регистрации

Тогда ответ от сервиса должен быть 200 OK

И я должен получить событие из очереди, содержащее email-пользователя

BDD: пример

1. Клиент API посылает запрос на регистрацию пользователя в **RegistrationService**
2. **RegistrationService** сохраняет пользователя в базу и публикует событие, что произошла новая регистрация
3. **NotificationService** уведомляет пользователя о регистрации (например смс, email и пр.) и публикует событие, что такой-то пользователь был проинформирован.



BDD: реализация

Реализация примера:

https://github.com/OtusGolang/webinars_practical_part/tree/master/31-integration-testing

Для BDD используем godog (читайте внимательно README):

<https://github.com/DATA-DOG/godog>



BDD Behavior-Driven Development

- Позволяет взглянуть со стороны (мышление тестировщика)
- Тулинг для тестов отличается от тулинга для продукта

- Сложно писать нетривиальные кейсы
- Неудобный рефакторинг

Вопросы

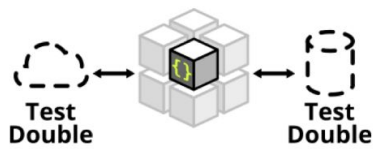


если есть вопросы

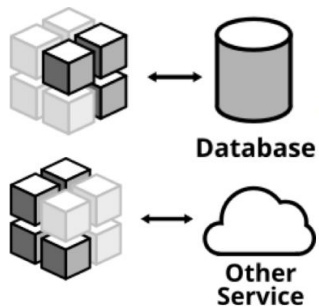


если вопросов нет

Интеграционные тесты



Тесты	Цель	Требует	Скорость	Сложность	Нужна настройка
Юнит-тесты	класс/метод	исходный код	очень быстро	низкая	нет
Интеграционные тесты	компонент/сервис	часть работающей системы	медленно	средняя	да



Интеграционные тесты: окружение

Варианты:

- Поднимаем сервисы, базу, кеши и пр. локально
- У нас есть виртуалка или тестовое окружение, куда мы можем раскатиться
- Docker (docker-compose или kubernetes)

Что делать с сервисами, которые ходят во внешнюю сеть (стороннее API и пр.)?

docker-compose: полезные команды

```
docker-compose [-f file] up [-d] [-build] [--exit-code-from service]
```

```
docker-compose [-f file] down
```

```
docker-compose logs [-f service]
```

```
docker-compose ps [-a]
```

```
docker-compose [-f file] run service [command]
```

```
docker-compose [-f file] exec service [command]
```



Интеграционные тесты: примеры

live-кодинг сессия по написанию интеграционных тестов

<https://www.youtube.com/watch?v=AV6xAeHQVI4>

https://github.com/kulti/otus_ol_int_tests

Пример с общим сьютом для юнит и интеграционных тестов

<https://github.com/kulti/task-list>



LIVE

Интеграционные тесты: резюме

- Используйте окружение максимально похожее на прод
- Не используйте тестируемый код
- Используйте спецификации и кодогенеренных клиентов
- Мокайте ненужные сервисы
- Пишите интеграционные тесты на том языке, на каком удобнее

Вопросы



если есть вопросы



если вопросов нет

E2E

Пирамида тестирования



E2E

- Покрывают все функции приложения с точки зрения клиента
- Могут эмулировать действия клиента (нажатие на кнопки, открытие страниц и тп) - например selenium
- Довольно сложные в написании



Е2Е для монолита

- Фактически представляет интеграционные тесты
 - API
 - СУБД
- Может усложниться если используем selenium для автоматизации тестирования фронта



Е2Е для микросервисов



Е2Е для микросервисов



- расписание
- стенды

Ограничения от DevOps



- Сложность поддержки
- Много ресурсов

Решение



- service-mesh, istio, consul
- маршрутизация трафика по меткам в http заголовков

Вопросы



если есть вопросы



если вопросов нет

Итоги занятия

Рефлексия

Список материалов для изучения

1. [Пирамида тестов на практике / Хабр](#)
2. [Антипаттерны тестирования ПО / Хабр](#)
3. https://ru.wikipedia.org/wiki/Разработка_через_тестирование
4. https://en.wikipedia.org/wiki/Behavior-driven_development
5. <https://cucumber.io/docs/gherkin/reference/>



Рефлексия



С какими впечатлениями уходите с вебинара?



Как будете применять на практике то, что узнали на вебинаре?

Приглашаем на следующий вебинар

System Design



Ссылка на вебинар будет
в ЛК за 15 минут



Материалы
к занятию в ЛК —
можно изучать



Обязательный материал
обозначен красной
лентой

Заполните, пожалуйста, опрос о занятии

Мы читаем все ваши сообщения
и берем их в работу 🖋️❤️

**You are our
OTUS heroes**

