

Безбажная пиццерия

Dodo Pizza, Юлия Ковалева
апрель 2018 года

Докладчик



 juliana_kov

Первая встреча с тестами 5 лет назад.

Автоматизирую и оптимизирую тесты последние 4 года.

Постоянно ищу новые способы проверить продукт лучше и быстрее, автоматизирую рутину.

В ближайшие 40 минут мы ...

- Познакомимся с Dodo Pizza
- Опишем текущую ситуацию с тестированием и тенденцию развития QA
- Узнаем, что такое Property Based Testing
- Рассмотрим примеры использования фреймворка FsCheck
- Разберемся, как же получать больше пользы от общения с QA-людьми
- Подведем итоги всему сказанному выше

Вы знакомы с нашим Распан?

The screenshot displays the Raspan restaurant management interface, divided into two main sections: a menu on the left and a table grid on the right.

Menu Section:

- Green Ghosts (55%):**
 - Овоши и г... 00:30
 - Пепперон... 00:30
 - Супермяс... 00:30
 - Мясная 35 00:30
 - Сырная 30 00:34
- Red Ghosts (45%):**
 - Сырн Цы... 00:34
 - Маргарит... 00:20
 - Мексика 30 00:20
 - Маргарит... 00:49
 - Мексика 30 00:49
- Orange Circle:**
 - Морская 30 00:49

Table Grid Section:

12:50	12:47	12:44	12:42	00:27
7 - 4 Лейла 2390 руб. Принимать	6 - 2 Фёдор 1010 руб. В процессе	5 - 2 Олеся 763 руб. Готов	4 - 3 Виктор 1123 руб. Готов	3 - 2 Гггггг 1240 руб. В процессе

Footer:

- Настройки
- Пользователи
- Заказов 7
- Ср. чек 1078 Р
- Ср. время приёма 00:19
- Навигация: < >
- Новый заказ

Dodo IS – облачная система



Сервисов много – тестировщиков мало

Ручной регресс

Для тестового окружения

Автотесты UI: 135

Автотесты API: 234





Кейсы на ручную проверку и автоматизацию проходят жесткий отбор

- Отбираем 1 пиццерию на ручной регресс
1 пиццерию на автоматический
- Вместо 10 стран тестируем одну
- Система не должна ломать работу сотрудников пиццерии и портить жизнь клиентам, все остальные кейсы – не в приоритете (проверяем раз в 4 релиза и не автоматизируем)

Нужны ли IT выделенные разработчики автотестов?

<https://events.withgoogle.com/gtac-2017/>



In the recent months, we've been taking a hard look at the discipline of Engineering Productivity as a logical next step in the evolution of test automation.

Нужны ли IT выделенные разработчики автотестов?

ГЕЙЗЕНБАГ

Круглый стол – Что должен знать
тестировщик в 2018 году?

#general

Юлия Атлыгина

ALM Works

Александр Хозя

Badoo

Николай Алименков

XP Injection

Никита Макаров

Одноклассники

Андрей Дмитриев

JUG.ru Group

Читать код разработчиков

Писать код не хуже разработчиков

Разбираться в инфраструктуре

Заменим тестировщика на сильного разработчика?

Хороший и стабильный код тестов

Позитивное мышление разработчика

Нет навыка поиска багов с позиции пользователя



Автотесты ходят

Следующий слайд, пожалуйста © Федор Овчинников

Заменим тестировщика на сильного разработчика?

Хороший и стабильный код тестов

Позитивное мышление разработчика

Нет навыка поиска багов с позиции пользователя



Автотесты ходят



Oops! Something went wrong.
(but we still love you)

**Ой! Кажется выручка считается
неправильно и учёт врет.**



Что же делать разработчикам?

1. Не идти в QA

Что же делать разработчикам?

1. ~~Не идти в QA~~

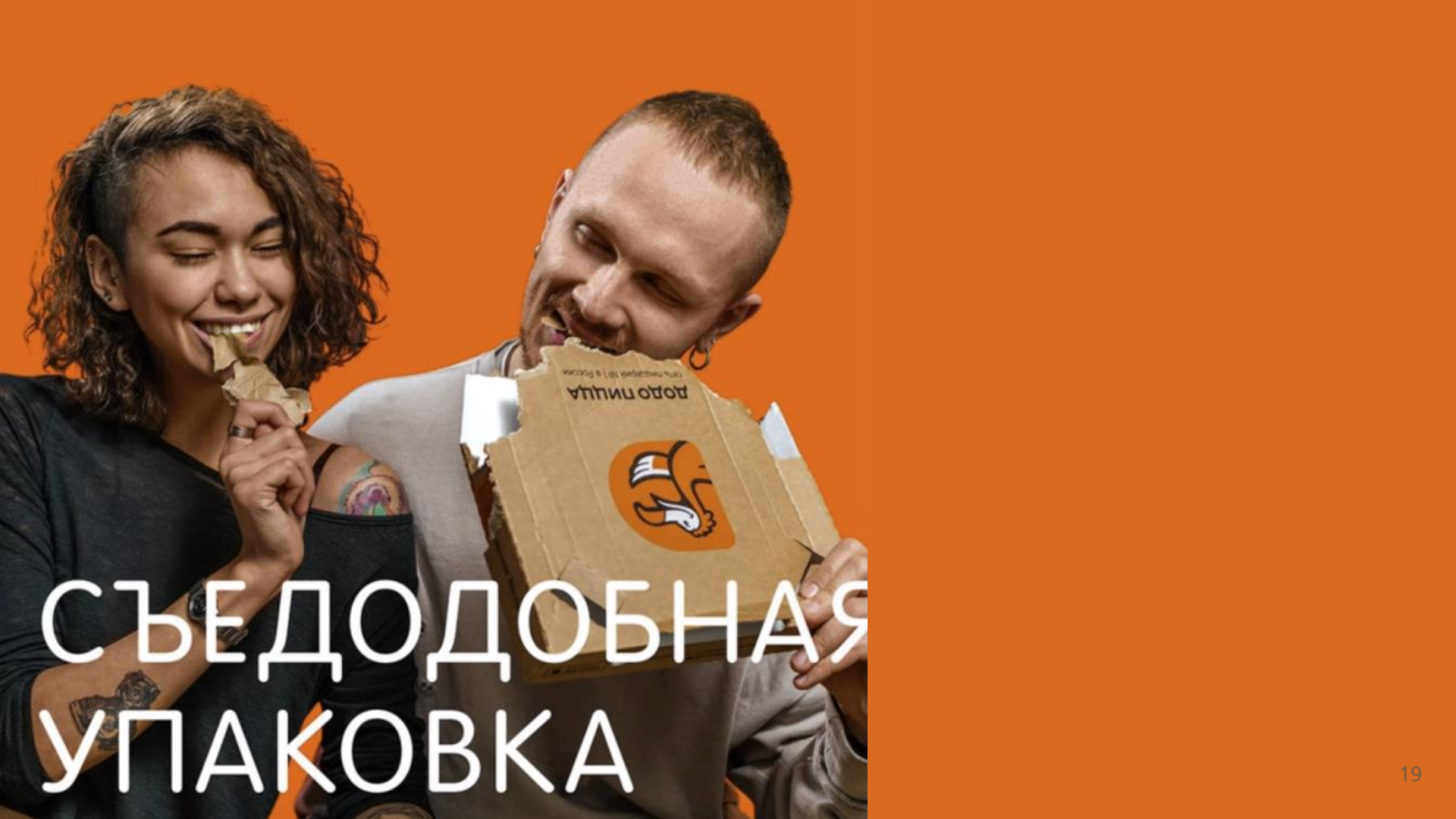
2. Перенять навыки тест-дизайна у своих тестировщиков

Что же делать разработчикам?

1. ~~Не идти в QA~~
2. ~~Перенять навыки тест-дизайна у своих тестировщиков~~
3. Писать собственные рандомизаторы для перебора входных данных

Что же делать разработчикам?

- ~~1. Не идти в QA~~
- ~~2. Перенять навыки тест-дизайна у своих тестировщиков~~
- ~~3. Активнее пользоваться инструментами для параметризации тестов~~
4. Научиться описывать тесты как правила, чтобы фреймворк генерировал примеры за вас



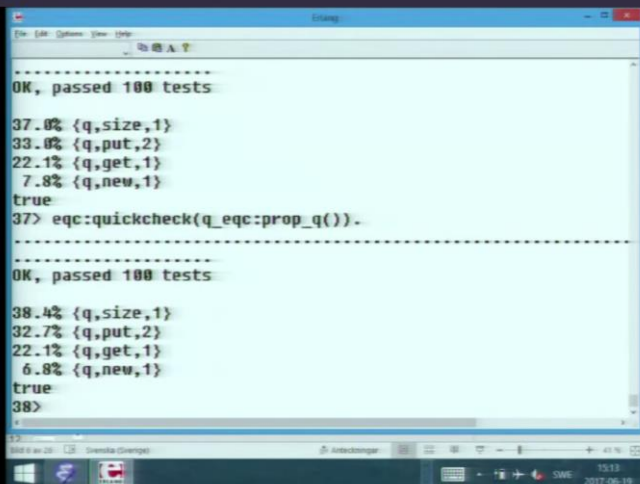
СЪЕДОДОБНАЯ
УПАКОВКА

John Hughes



Don't Write Tests!

John Hughes

A screenshot of a video player showing a terminal window. The terminal displays test results for a program. The first run shows 100 tests passed. The second run shows a quickcheck test failing, with a list of statistics for different test cases. The statistics are: 37.0% {q,size,1}, 33.0% {q,put,2}, 22.1% {q,get,1}, 7.8% {q,new,1}, and true. The quickcheck test is labeled as failed. The terminal window is titled 'Erlang' and has a menu bar with 'File', 'Edit', 'Options', 'View', and 'Help'. The video player interface at the bottom shows the video is at 15:13 and the date is 2017-06-19.

```
-----  
OK, passed 100 tests  
  
37.0% {q,size,1}  
33.0% {q,put,2}  
22.1% {q,get,1}  
7.8% {q,new,1}  
true  
37> eqc:quickcheck(q_eqc:prop_q()).  
  
-----  
OK, passed 100 tests  
  
38.4% {q,size,1}  
32.7% {q,put,2}  
22.1% {q,get,1}  
6.8% {q,new,1}  
true  
38>
```

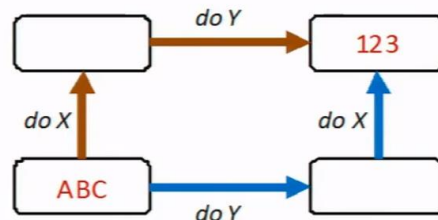
https://www.youtube.com/watch?v=hXnS_Xjwk2Y

John Hughes - Don't Write Tests

Scott Wlaschin



"Different paths, same destination"



<https://skillsmatter.com/skillscasts/6432>

The lazy programmer's guide to writing 1000's of tests: An introduction to property based testing

О чем же говорят эти спикеры?

- O^n сложность – трудоёмка для ручного придумывания кейсов

О чем же говорят эти спикеры?

- O^n сложность – трудоёмка для ручного придумывания кейсов
- Поэтому нужно научиться находить и формулировать инварианты

«для любых данных, таких, что ... выполняется условие ...»

О чем же говорят эти спикеры?

- O^n сложность – трудоёмка для ручного придумывания кейсов
- Поэтому нужно научиться находить и формулировать инварианты

«для любых данных, таких, что ... выполняется условие ...»

- А генерацию примеров оставить фреймворку

Меньше ручного труда — больше мыслей

- Один генератор вместо тысячи строк примеров

Меньше ручного труда — больше мыслей

- Один генератор вместо тысячи строк примеров
- Специальные случаи автоматически будут учтены условиями вашего генератора

Меньше ручного труда — больше мыслей

- Один генератор вместо тысячи строк примеров
- Специальные случаи автоматически будут учтены условиями вашего генератора
- Минимальный невалидный кейс будет найден автоматически

Меньше ручного труда — больше мыслей

- Один генератор вместо тысячи строк примеров
- Специальные случаи автоматически будут учтены условиями вашего генератора
- Минимальный невалидный кейс будет найден автоматически
- Проще оценить покрытие тестами, ведь разбиение тестов на классы осуществляется фреймворком

Property Based подход

Для мира .Net - FsCheck



- Поддерживает языки C#, F# и VB
- BSD лицензия
- Интегрирован с NUnit, XUnit, SpecFlow и другие

**Рассмотрим
пример**



Открываем сайт dodopizza.ru для города Киров



Пиццы Комбо Закуски Десерты Напитки Акции Контакты Франшиза

0 ₽
0 товар



i

Двойная пепперони

Томатный соус (x),
моцарелла (x) и
пикантная пепперони (x)

35 см

30 см

25 см

Традиционное

Тонкое

635 ₽

В корзину



i

Сырная

Томатный соус (x) и
моцарелла (x)

35 см

30 см

25 см

Традиционное

Тонкое

535 ₽

В корзину

i

Грибы и ветчина

Томатный соус (x),
моцарелла (x), ветчина (x) и
шампиньоны (x)

35 см

30 см

25 см

Традиционное

Тонкое

495 ₽

В корзину

i

Гавайск

Томатный
моцарелл

35 см

Традици

535 ₽

31

Составляем инвариант

Для любых браузеров,
любой пиццерии в России и
любого продукта доступного к продаже

Таких, что

Продукт из СПИСКА ПРОДУКТОВ для ПИЦЦЕРИИ

Браузер из СПИСКА БРАУЗЕРОВ

Выполняется условие

В корзину можно добавить продукт и он в корзине будет только один

Структура обычного Unit - теста

```
[Test]
public void ProductsInCart ()
{
    // Выбран продукт "Додстер" с доставкой в Самару-1
    Arrange () ;

    // Кладем "Додстер" в корзину в Самаре-1
    Act () ;

    // Проверяем, что в корзине 1 "Додстер"
    Assert () ;
}
```

Получаем тест на FsCheck

```
Prop.ForAll (SetProducts (), unitProduct =>
{
    GoToUnitSite (unitProduct);
    AddToCart (GetProductCard (unitProduct));
    SelectCarryoutStore (unitProduct);
    GoToCart ();

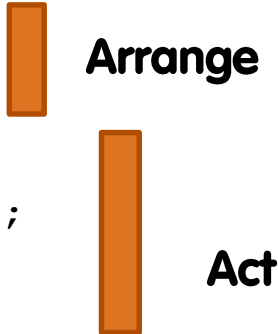
    ...
    return
        quantity.Equals ("1")
            .Label ($"Продуктов в корзине: {quantity}")
            .And (productsInCart.Contains (unitProduct.Name)
                .Label ($"Наименование с сайта: {productsInCart}"))
    }) .Check (_configuration);
```



Arrange

Получаем тест на FsCheck

```
Prop.ForAll (SetProducts(), unitProduct =>  
{  
    GoToUnitSite (unitProduct);  
    AddToCart (GetProductCard (unitProduct));  
    SelectCarryoutStore (unitProduct);  
    GoToCart ();  
    ...  
    return  
    quantity.Equals ("1")  
    .Label ($"Продуктов в корзине: {quantity}")  
    .And (productsInCart.Contains (unitProduct.Name)  
    .Label ($"Наименование с сайта: {productsInCart}")  
}) .Check (_configuration);
```



Arrange

Act

Получаем тест на FsCheck

```
Prop.ForAll (SetProducts(), unitProduct =>
{
    GoToUnitSite (unitProduct);
    AddToCart (GetProductCard (unitProduct));
    SelectCarryoutStore (unitProduct);
    GoToCart ();
    ...
    return
    quantity.Equals ("1")
    .Label ($"Продуктов в корзине: {quantity}")
    .And (productsInCart.Contains (unitProduct.Name)
    .Label ($"Наименование с сайта: {productsInCart}")
})) .Check (_configuration);
```



Arrange



Act



Assert

Получаем тест на FsCheck

```
Prop.ForAll (SetProducts(), unitProduct =>
{
    GoToUnitSite (unitProduct);
    AddToCart (GetProductCard (unitProduct));
    SelectCarryoutStore (unitProduct);
    GoToCart ();
    ...
    return
    quantity.Equals ("1")
    .Label ($"Продуктов в корзине: {quantity}")
    .And (productsInCart.Contains (unitProduct.Name))
    .Label ($"Наименование с сайта: {productsInCart}")
}) .Check (_configuration);
```

Arrange

Act

Assert

Loop

РА
ДОДО
СТЬ



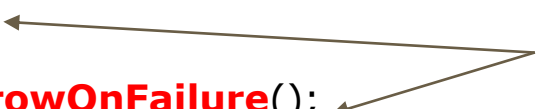
Способы запуска

property.**Check**(configuration); - ЗАВИСИТ ОТ СВОЙСТВ КОНФИГА

property.**QuickCheck**();

property.**QuickCheckThrowOnFailure**();

сгенерированные данные не
печатаются



property.**VerboseCheck**();

property.**VerboseCheckThrowOnFailure**();

сгенерированные
данные печатаются



Как воспроизвести упавший тест?

Falsifiable, after 3 tests (0 shrinks)
(738508685, 296411070):

```
global::FsCheck.Random.StdGen sidId =  
global::FsCheck.Random.StdGen.NewStdGen(738508685, 296411070);
```

```
var configuration = Configuration.VerboseThrowOnFailure;  
configuration.Replay = sidId;  
// -- создание свойства
```

```
property.Check(configuration);
```

Если же стартовать запуск из консоли, то можно использовать **--params**

ИСПОЛЬЗУЕМ
НАТУРАЛЬНЫЙ
СОУС
100% ТОМАТОВ
И СПЕЦИЙ

НЕ ЗАМОРАЖИВАЕМ
ТЕСТО: ОНО ЛЕГКОЕ
И ВКУСНОЕ

ДЕЛАЕМ ПИЦЦУ
ЧИСТОМИ РУКАМИ

ЗАКАЖИТЕ
НА ВСЮ

Много зависимостей в рамках одного правила

Пример: Проверка создания заказа (orders add метод).

- Смена (int) должна быть открыта
- Отдел (int) должен уметь принимать заказы
- Могут быть разные типы заказа (int), от чего зависит должен ли передаваться Адрес доставки (int/null)
- И т.д.

Много зависимостей в рамках одного правила

FsCheck: Gen.ArrayOf, Gen.Four, Gen.Three и подобное.

FsCheck: Filter & MapFilter

Фильтры замедляют процесс генерации нужного примера,
особенно когда учитывается много условий

Попробуйте генерировать только 1 пример и в таком виде
отправить его в CI/CD.

Недостатки

Придется разобраться в подходе и хитростях работы с инструментом

Для Arbitrary не поддерживается Shrinking

Неудачно написанный генератор будет долго искать вам подходящий пример

Однако

- ✓ Если примеров даже при ручном создании получается не меньше 10
- ✓ Если вам необходимо придумывать объект с > 10 полями и > 2 зависимостей
- ✓ Если вы боитесь упустить специальные случаи (пустые строки, max/min)
- ✓ Если вы хотите быстро получать информацию о покрытии тестами на основе их запуска

Однако

- ✓ Если примеров даже при ручном создании получается не меньше 10
- ✓ Если вам необходимо придумывать объект с > 10 полями и > 2 зависимостей
- ✓ Если вы боитесь упустить специальные случаи (пустые строки, max/min)
- ✓ Если вы хотите быстро получать информацию о покрытии тестами на основе их запуска



спешит на помощь



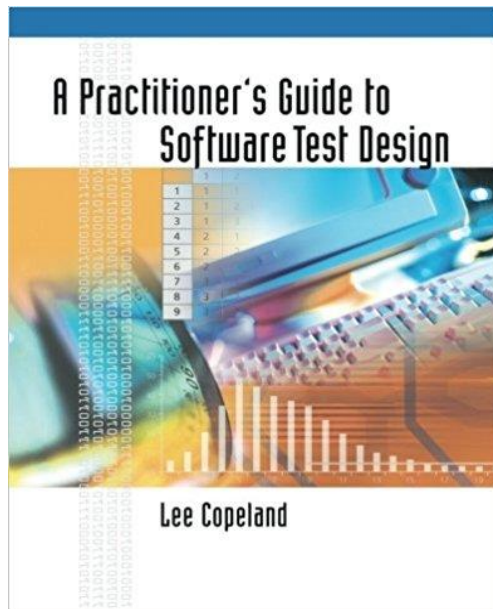
**Немного рекомендаций,
если все же не помогло ...**

Работаем в паре с тестировщиком

- Составляем вместе кейсы
- Помогаем с автотестами
- Анализируем логи
- Обучаем тестировщика правильным практикам
- Меняем верстку



Пишем тесты на основе статистики с прода и техник тест-дизайна



https://books.google.ru/books/about/A_Practitioner_s_Guide_to_Software_Test.html?id=dMX_C8z9PfMC&redir_esc=y

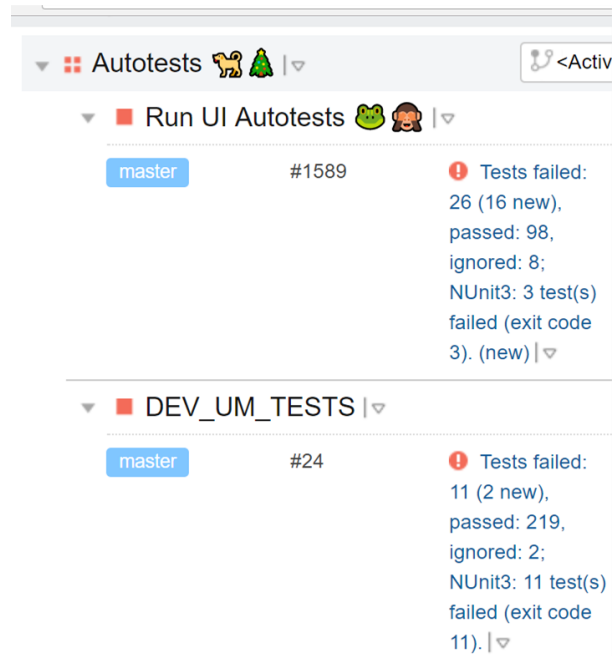
Пингуем все сервисы до регресса

- До этапа регрессионного тестирования доходит рабочий код и рабочее окружение;
- Меньше "псевдо" flasky-тестов.



Автотесты UI и API запускаем на Dev-окружениях

- Разработчик сможет запустить полный регресс (глазами тестировщика) на своем окружении до того как это попадет в тестинг;
- Хотфиксы станут не такими болезненными.



Тестовый стенд под нагрузкой

Часть ошибок уходят в Production, так как стенд на котором проводилось тестирование совсем не похож на боевой.



Как же добиться безбашности?

- Пишите тесты, используя инструменты и подходы, решающие именно ваши проблемы
- Если примеров получается много, то подключайте FsCheck
- Тестировщика поместите в команду
- Следите за тенденциями развития тестирования ПО, они очень быстро
меняется

Следующий слайд © Федор Овчинников

Вопросы?

<https://fscheck.github.io/FsCheck>

<https://github.com/akustic/dotnext-fscheck>



j.kovaleva@dodopizza.ru
juliana_kov (Skype)