# Введение в тестирование

Виды тестирования. Тестирование API

Алексей Федин
Системный аналитик в ООО «Открытые решения»

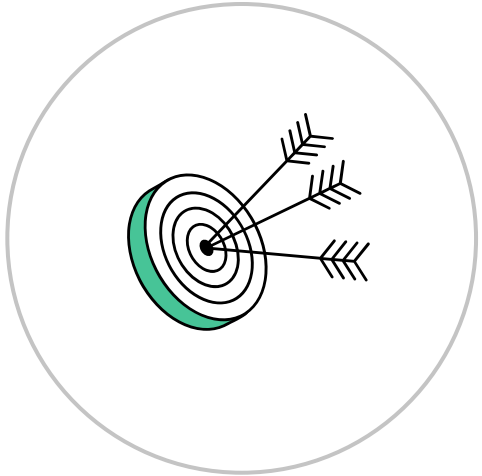
# Алексей Федин

О спикере:

- Системный аналитик
- Работает в IT с 2002 года
- Опыт администрирования и работы с сетями более 10 лет
- С 2008 года занимается проектированием и наладкой решений информационной безопасности в промышленности. Работал в компаниях: «РТСофт», Positive Technologies, iGrids, ElcomSoft

# Цели занятия

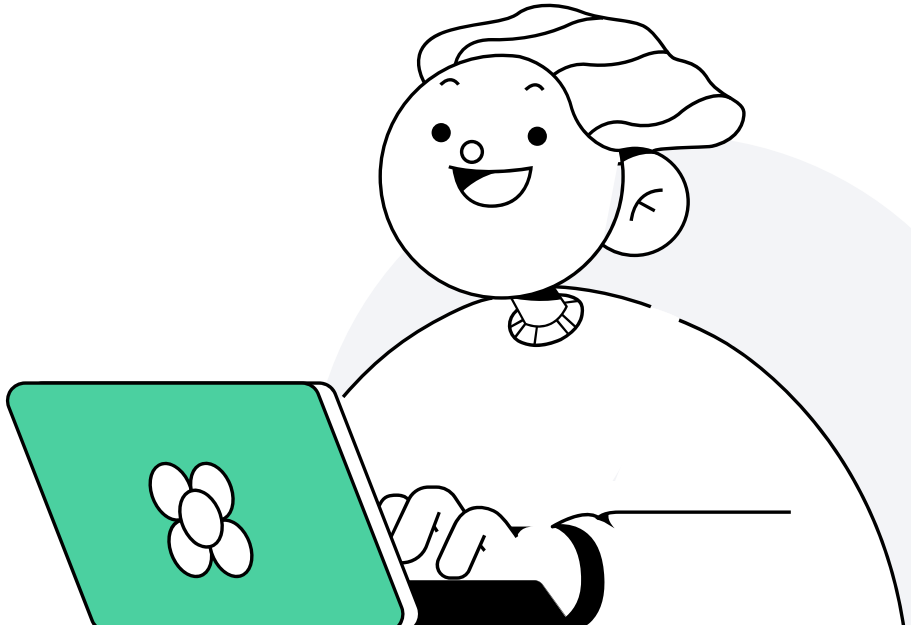
- Разобраться цели и задачи тестирования
- Узнать, что такое тест-дизайн
- Научиться работать с метриками тестирования



# План занятия

1. Введение в тестирование ПО
2. Тест-дизайн
3. Метрики тестирования
4. Тестирование API
5. Итоги
6. Домашнее задание

*Нажми на нужный раздел для перехода



# Введение в тестирование ПО

1

# Тестирование ПО

**Тестирование ПО** – это процесс проверки соответствия работы программного обеспечения требованиям, предъявляемым к разрабатываемому ПО

**Основная цель тестирования** – выявить ошибки и недочеты, которые могут возникнуть в процессе использования ПО

# Основные задачи тестирования

- Проверка соответствия функциональности ПО заявленным требованиям
- Проверка корректности работы ПО и отсутствия ошибок во время работы
- Обеспечение стабильности и надежности работы ПО
- Улучшение качества ПО и удовлетворенности пользователей
- Оценка производительности ПО
- Выявление ошибок интеграции со сторонними продуктами
- Выявление типовых проблем безопасности

# Виды тестирования: по автоматизации

- Ручное
- Автоматизированное

# Виды тестирования: по описанию объекта

- Тестирование чёрного ящика
- Тестирование белого ящика
- Тестирование серого ящика

# Виды тестирования: по отношению к системе

- Модульное тестирование
- Компонентное тестирование
- Интеграционное тестирование
- Системное тестирование

# Виды тестирования: по результату тестирования

- Позитивное тестирование
- Негативное тестирование

# Виды тестирования: по этапности

- Дымовое тестирование (smoke testing)
- Регрессионное тестирование
- Приёмочное тестирование
- Исследовательское тестирование

# Виды тестирования: по функциональности

- Функциональное тестирование
- Тестирование производительности
- Юзабилити-тестирование
- Тестирование безопасности
- Тестирование совместимости

# Документация: план тестирования

**План тестирования (Test Plan)** – документ, содержащий описание всего процесса тестирования, включая его цели, задачи, область применения, ресурсы и расписание.

В плане тестирования обязательно указываются:

- методы, используемые для проведения тестирования
- критерии, по которым будет оцениваться качество ПО
- любые требования, которые будут необходимы для данного проекта

# Документация: спецификация требований

**Спецификация требований на тестирование (Test Requirements Specification)** – документ, содержащий описание всех требований к тестированию продукта.

Спецификация состоит из списка функциональных и нефункциональных требований, которые должны быть протестированы, и описания того, как эти требования будут проверены

# Документация: чек-лист

**Чеклист (Check List)** – это формальный список проверок, которые нужно выполнить в процессе тестирования.

В чек-листе могут быть перечислены:

- тест-кейсы
- условия тестирования
- требования к продукту и многое другое

# Документация: тест-кейс

**Тест-кейс (Test Case)** – подробное описание проверки одного тестового случая.

Тест-кейс содержит:

- название
- предусловия
- шаги тестирования
- ожидаемый результат

# Документация: баг-репорт

**Баг-репорт, баг, ошибка (Bug Report)** – подробное описание ошибки ПО.

Баг-репорт содержит:

- описание ошибки
- важность ошибки
- критичность ошибки
- условия воспроизведения (версия разрабатываемого ПО, версия ОС, версия браузера, набор данных, и т.п.)
- шаги воспроизведения
- ожидаемый результат
- фактический резкъультат

# Тест-дизайн

2

# Тест-дизайн

**Тест-дизайн** – это процесс разработки тестовой документации (тест-кейсов) на основе требований и спецификаций.

Цель тест-дизайна – максимальное покрытие функциональности приложения тестами.

Основные техники тест-дизайна:

- классы эквивалентности
- граничные значения
- парное тестирование
- таблица принятия решений

# Тест-дизайн: классы эквивалентности

**Классы эквивалентности** – разбиение входных данных на классы, где каждый класс содержит эквивалентные значения.

Если для одного значения выполняются определенные условия, то они будут выполняться и для всех других значений этого класса (кнопка загрузки файла, несколько диапазонов цен, и т.д.).

# Тест-дизайн: граничные значения

**Граничные значения** – выбор граничных значений для каждого класса эквивалентности.

Позволяет выявить ошибки, связанные с переполнением или недостатком ресурсов.

Как правило, выбираются три значения:

- из диапазон допустимых значений
- ниже диапазона
- выше диапазона

# Тест-дизайн: парное тестирование

**Парное тестирование (pairwise testing)** –каждый проверяемый параметр должно быть протестирован на взаимодействие со всеми остальными параметрами.

Попарное тестирование является оптимальным по соотношению тестового покрытия и количества проверок.

Для парного тестирования лучше использовать специальные утилиты, например https://pairwise.teremokgames.com/

# Тест-дизайн: таблица принятия решений

**Таблица принятия решений** – способ представления проверки объекта со сложной логикой. Описывается как взаимосвязь между множеством условий и действий (чем-то похожа на сценарии использования).

Каждый тест-кейс должен проверять конкретный функциональный аспект приложения.

| | Тариф 1 | Тариф 2 | Тариф 3 |
|---|---|---|---|
| Сумма | 500 - 1000 | 1000 - 5000 | 5000 - ... |
| Период | до обеда | днём | вечером |
| Скидка | 5% | 10% | 15% |

# Метрики тестирования

3

# Метрики тестирования: тестовое покрытие

**Тестовое покрытие** (test coverage) показывает, какая часть функциональности приложения покрыта тестами.

Чем выше тестовое покрытие, тем больше вероятность, что найденные ошибки будут связаны с функциональными проблемами, а не с проблемами синтаксиса. Поэтому многие команды разработчиков стремятся к 100% тестовому покрытию, чтобы убедиться в высоком качестве своего программного обеспечения

# Метрики тестирования: количество дефектов

**Количество дефектов** отображает количестве ошибок, обнаруженных в процессе тестирования.

Дает представление о качестве кода и о том, сколько времени будет затрачено на исправление ошибок

# Метрики тестирования: скорость выполнения

**Скорость выполнения** показывает, насколько быстро работает приложение в определенных условиях.

Может быть использована для того, чтобы сравнивать производительность разных версий приложения

# Метрики тестирования: соотношение успешных и неуспешных тестов

**Соотношение успешных и неуспешных тестов** используется для оценки отношения удачно пройденных тестов к завершившимся с ошибками.

Метрика помогает оценить качество тестов и готовность продукта

# Метрики тестирования: соотношение открытых и закрытых ошибок

**Соотношение открытых и закрытых ошибок** показывает оценку скорости устранения ошибок и может позволить выявить причины, по которым ошибки остаются незакрытыми.

Метрика помогает оценить скорость работы\загруженность команды разработки.

# Метрики тестирования: количество невыполненных тестов

**Количество невыполненных тестов** показывает количество тестов, которые не могут быть запущены либо из-за неготовности кода, либо из-за блокировки другими ошибками.

Метрика помогает оценить готовность кода и приложения. Влияет на покрытие

# Метрики тестирования: распределение ошибок по параметрам

**Распределение ошибок по параметрам** показывает распределение ошибок по основным параметрам, например:

- важность
- критичность
- функциональные
- нефункциональные
- и прочие

Метрика помогает оценить общее состояние тестирования проекта

# Метрики тестирования: количество перекрытых ошибок

**Количество перекрытых ошибок** показывает количество заново появившихся ошибок, которые были исправлены ранее.

Метрика помогает оценить общее квалификацию разработчиков и выявить сложные места проекта

# Метрики тестирования: среднее время прохождения теста

**Среднее время прохождения теста** показывает среднее время работы одного теста.

Метрика помогает оценить длительность этапа тестирования. Особенно важно при оценки регрессионого тестирования

# Метрики тестирования: соотношение запланированного и потраченного времени

**Соотношение запланированного и потраченного времени** учитывает время, затраченное на проверки.

Метрика помогает оценить состояние и квалификацию команды тестирования

# Тестирование API

6

# Демонстрация работы

Пример тестирования простой веб-службы

# Итоги

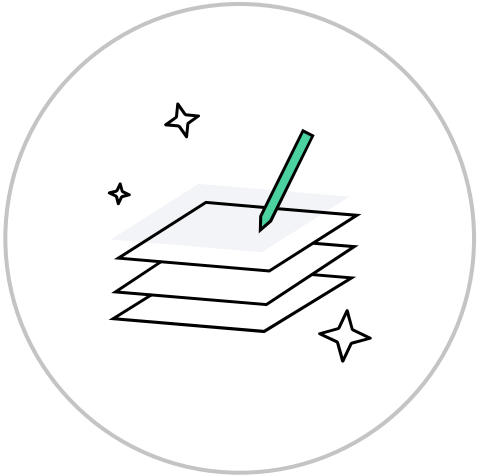
Сегодня мы:

1. узнали, что такое тестирование программного обеспечения
2. поговорили о  техниках тест-дизайна и метриках тестирования
3. рассмотрели пример тестирования web-приложения

# Домашнее задание

Давайте посмотрим ваше домашнее задание.

1. Вопросы по домашней работе задавайте в чате группы
2. Задачи можно сдавать по частям
3. Зачёт по домашней работе ставят после того, как приняты все задачи



# Задавайте вопросы и пишите отзыв о лекции

Алексей Федин
Системный аналитик в ООО «Открытые решения»