
ozon{tech

# Allure-отчеты

# План занятия

# Общая теория

Отчеты о проведенных автотестах — обязательная часть работы на проекте. Информация о проведенном тестировании может быть полезна разным проектным ролям, и требования к ним в зависимости от роли могут меняться. Условно мы можем объединить эти роли в следующие группы:

- Тестировщики
- Разработчики
- Автоматизаторы тестирования
- Менеджеры
- Внешние пользователи

Группы выделены весьма условно и не привязаны к конкретным должностям. Иногда вы можете (и, скорее всего, будете) выполнять несколько функций. Например, QA-инженер может выступать в роли автоматизатора во время написания автотестов, тестировщика во время их прогона и менеджера во время планирования или принятия решения о релизе.

Рассмотрим подробнее каждую из них и выясним, что они хотят видеть в отчетах.

**Тестировщики**
Тестировщик в данной классификации — тот, кто занимается тест-дизайном и запускает тесты (автоматизированные и мануальные). В описаниях конкретных тест-кейсов тестировщиков прежде всего интересует наличие информации о предусловиях теста, какие шаги были выполнены и какие проверки проводились. В описаниях наборов тестов — покрытие, статусы тестов и их приоритет.

**Разработчики**

Разработчик — тот, кто разрабатывает непосредственно тестируемую систему. Как правило, подробно вникать в то, как тестировалась системы, ему не нужно, зато найденные ошибки исправлять приходится именно ему. Значит, разработчикам важна информация, помогающая в воспроизведении и локализации проблемы.

**Автоматизаторы тестирования**

Те, кто разрабатывает автотесты. В первую очередь этих специалистов интересуют метрики, характеристики самих автотестов — их стабильность, время выполнения, а также информация об инфраструктуре, на которой тесты запускаются.

**Менеджеры**

Люди, которым интересна ситуация с продуктом в целом. Тут важны данные, говорящие о том, необходимо ли дальнейшее тестирование, насколько продукт готов к релизу, и отчеты о результатах работы команды и загруженности ее членов.

**Внешние роли**

Те, кто не входит в команду, но тем не менее хочет получать информацию о результатах тестирования. Ими могут быть представители заказчика во время приемки, CEO, который решил узнать, как обстоят дела с тестированием, и так далее. Как правило, конкретных требований от внешних ролей нет, и тем не менее при проектировании системы репортинга их интересы иногда лучше учитывать.

# Проблемы отчетов автотестов

Что происходит, если репорт не удовлетворяет требованиям вышеперечисленных ролей? Если тесты проходят успешно, как правило, проблема не заметна. Тестировщики радуются “зелёному” репорту, продукт релизится, команда довольна.

При падении некоторых тестов может возникнуть ситуация, которую можно назвать «автотест Шредингера». Как мы знаем из теории, прогнанный тест может находиться в двух статусах — успешном и неуспешном. На практике тест может быть в третьем состоянии, когда непонятно, успешен он или нет, пока на него не посмотрит его автор. Вообще это комплексная проблема, решать которую стоит на уровне перестроения процессов в команде. При этом в ее разрешении могут сильно пригодиться правильно сделанные отчеты.

Какие проблемы в отчетах об упавших тестах могут приводить к такой ситуации?

- Недостаточно информации об ошибке — к примеру, тест помечается как FAILED, но ни информации о проверке, ни о причинах ее провала не отображается.
- Неинформативная информация о падении — данные об ошибке присутствуют, но либо неполны, либо непонятны читающему.
- Недостаточно информации о действиях, совершенных до непосредственной проверки.

В принципе все эти проблемы решаются подробными описаниями:

- Предусловий
- Шагов теста
- Проверки
- Ошибки, если она возникла

Возможна ситуация, когда мы имеем false-negative зеленый тест, лишь притворяющийся успешным из-за некорректных проверок. Причин тому может быть много: некорректный тест, ошибки в инфраструктуре и т. д. Но в ее предотвращении могут помочь хорошие репорты, отображающие детальную информацию о шагах теста и проверках, в том числе и для успешных тестов.

# Инструменты

Инструментов для генерации репортов много. У нас в основном используются два – Allure Report и Allure TestOps.

## Allure Report

Довольно зрелая система генерации репортов. Первая версия была сделана уже очень давно в Yandex, сейчас разрабатывается компанией Qameta Software. Позволяет указать, что мы хотим поместить в отчет прямо в коде автотестов, также может собирать различные агрегированные метрики о прогонах, предоставляет инструменты поиска в репортах, поддерживает множество языков программирования.

## Allure TestOps

Уже не только инструмент генерации отчетов, но полноценная Test Management System. Хранит результаты тестовых прогонов в облаке или на внутреннем сервере, интегрируется с CI и Jira, позволяет управлять не только запуском автотестов, но и мануальными кейсами. Также довольно неплохо интегрирован с популярными IDE.

Как работать с этими системами, лучше всего расскажет официальная документация:

- Allure Report
- Allure TestOps

Помимо очевидных советов, вроде необходимости проставления декораторов allure.title(), allure.suite() и т. д. и заворачивания шагов теста в контекстный менеджер allure.step() могу дать несколько рекомендаций:

- Удобно сделать линтер или скрипт, проверяющий наличие вышеперечисленных декораторов в тест-кейсах, так вы убедитесь, что не забыли ничего из этого в своих тестах.
- Также в линтер стоит встроить проверку на дублирующиеся allure.id(). Дублирующиеся ID ломают отчеты.
- Если вы параметризуете тесты каким-либо кастомным объектом, этому объекту стоит добавить метод __repr__(). Иначе параметр в отчете будет отображаться примерно как <__main__.MyObject object at 0x00000194CE522B60>. Причем при перепрогоне тестов последняя часть такого имени (адрес в памяти) для каждого теста будет другой и allure будет воспринимать это как новый параметр.
- Фичей в декораторе allure.feature() может быть как бизнесовая фича, так и ручка API.
- Из метрик полезен сбор информации о времени выполнения теста и процент успешных прогонов этого теста с сортировкой по убыванию в первом случае и возрастанию во втором. Так мы можем выявить первого кандидата на рефакторинг — самые нестабильные и долгие тесты.
- Allure позволяет линковать тесты с тикетами в Jira — удобно привязывать тесты к задачам на разработку или к дефектам.

## Grafana

Помимо Allure, в Ozon для репортинга также используется Grafana. Есть подробный доклад про это: https://sqadays.com/ru/talk/86550

# Заключение

На этом занятии мы рассмотрели:

- Общие принципы построения тестовых отчетов.
- Разобрали несколько типичных ошибок при их генерации.
- Познакомились с инструментом создания отчетов — Allure.