

Тестовая документация. Багтрекеры

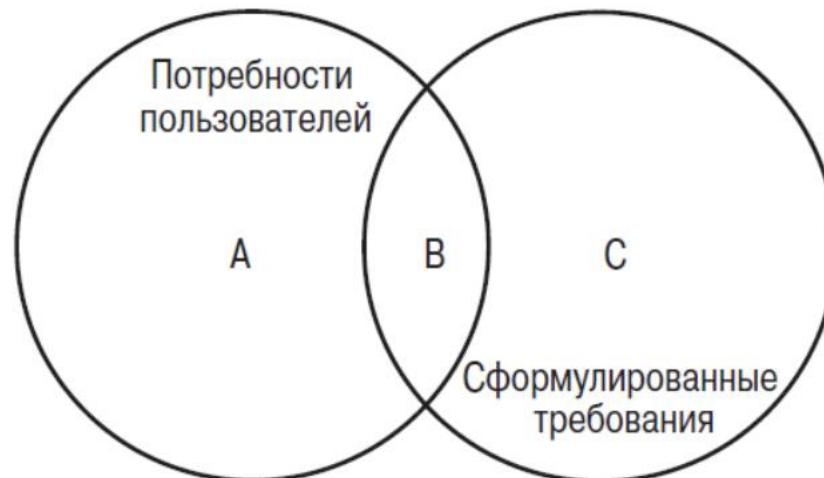


Содержание:

- I. Артефакты тестирования
- II. Дефекты в ПО, составление баг-репортов
- III. Приоритизация багов, системы отслеживания ошибок

Критерии составления требований:

- 1) корректность;
- 2) недвусмысленность;
- 3) полнота;
- 4) непротиворечивость;
- 5) упорядоченность по важности и стабильности;
- 6) проверяемость;
- 7) модифицируемость;
- 8) трассируемость;



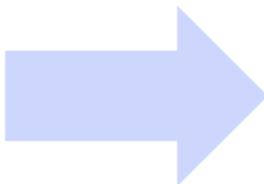
Если левый круг (область А) представляет множество потребностей пользователя, а правый (область С) — требования, то корректные требования будут находиться в области пересечения кругов, область В

Артефакты тестирования

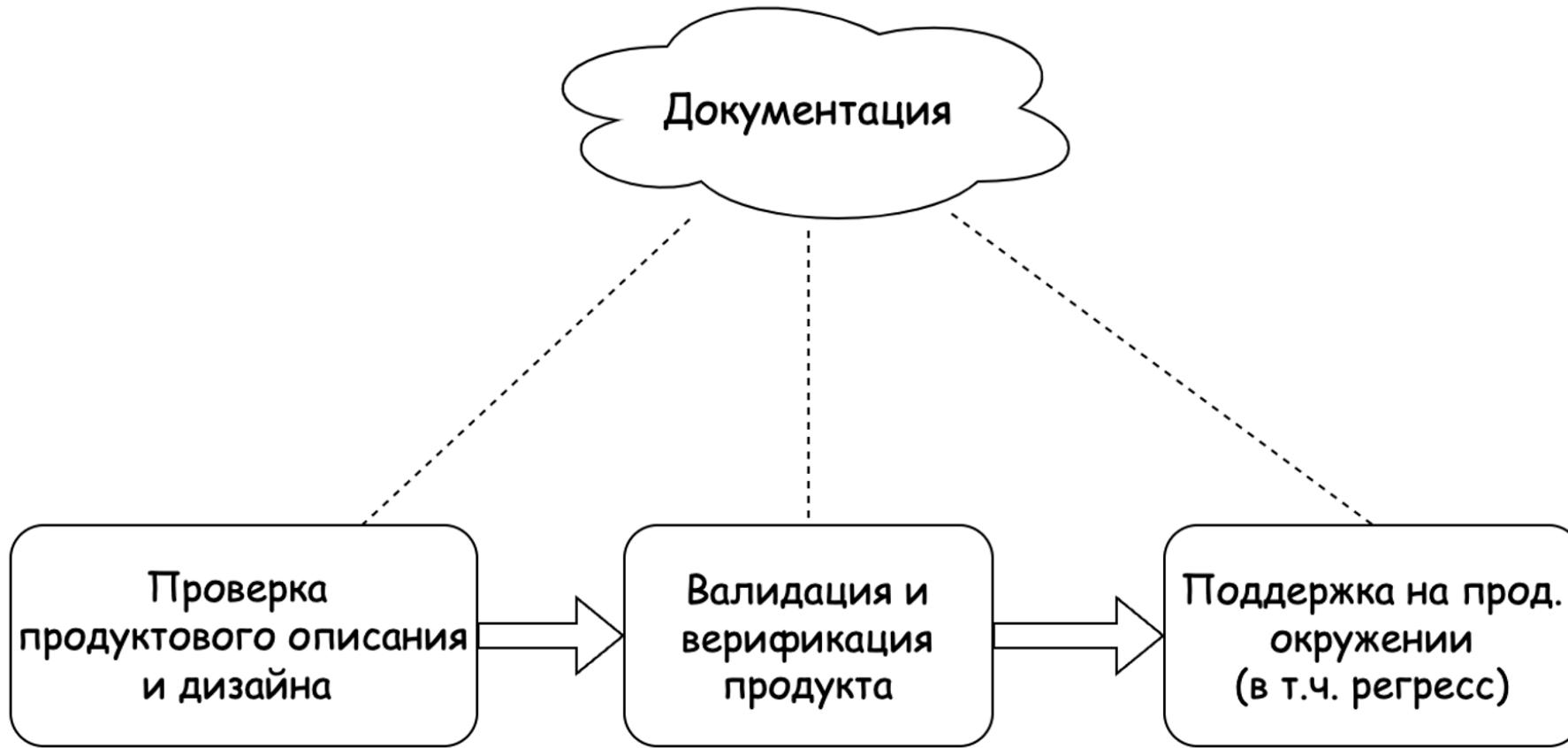


Тестирование - это

Проверка созданного программного продукта (фактического результата) на предмет соответствия эскизному проекту (ожидаемого результата).



Исследовательский процесс, целью которого является предоставление информации о качестве продукта заинтересованным лицам.



Наиболее распространенные тестовые артефакты:

1

2

3

4

Тест план

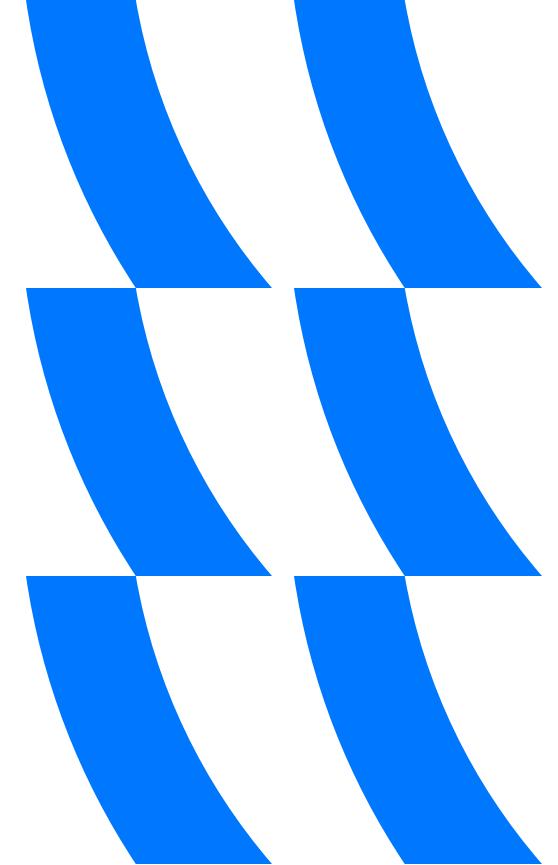
Тест-кейс

Чек-лист

Баг-репорт



Тест план



Что это?

Документ, описывающий **весь объем работ по тестированию** начиная с описания тестируемых объектов, стратегии, расписания, критериев начала и окончания тестирования, до необходимого в процессе работы оборудования, специальных знаний, а также оценки рисков с вариантами их разрешения



Для чего он нужен?

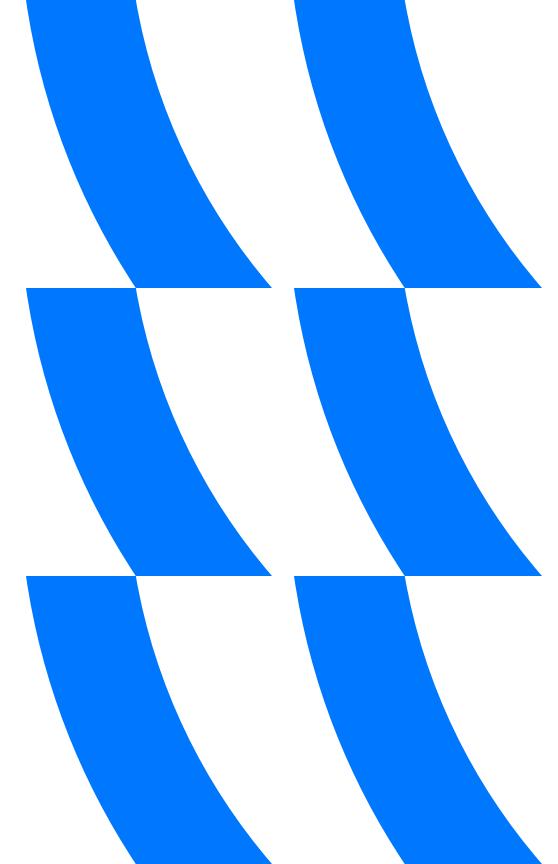
- 1) Согласование объёмов и стратегии тестирования различных составляющих тестируемого ПО с другими участниками проектной команды
- 2) Приоритизация задач по тестированию
- 3) Своевременное планирование ресурсозатрат на тестирование
- 4) Учёт требуемых ресурсов (ПО, оборудование), необходимых для тестирования
- 5) Заблаговременный учёт рисков, которые могут возникнуть в процессе
- 6) Реализации плана, и внедрение предупреждающей стратегии
- 7) Планирование использования ресурсов на тестирование.



Шаблон

- Назначение
- Объект тестирования
- Тестовая стратегия
- Применяемые виды тестирования
- Условия проведения тестирования
- Критерии начала и завершения тестирования
- План-график проведения тестирования
- Ресурсы, необходимые для выполнения тестирования
- Возможные риски

Тест-кейс



Что это?

Тест-кейс — это тестовый артефакт, суть которого заключается в **выполнении некоторого количества действий и/или условий**, необходимых для проверки определенной функциональности разрабатываемого приложения.

Написать тест кейс — значит создать текстовое описание процесса тестирования какой-то части или функции проекта.

Из чего состоит:

1

Порядковый номер
(ID)

4

Основные шаги

2

Заголовок
(Краткое описание)

5

Ожидаемый результат

3

Предусловия
Входные данные

6

Статус

Какой результат?

Положительный
(PASSED)



ВЫКАТЫВАЕМСЯ

Отрицательный
(FAILED)



ОТКАТЫВАЕМСЯ

Не завершен
(skipped, not run)

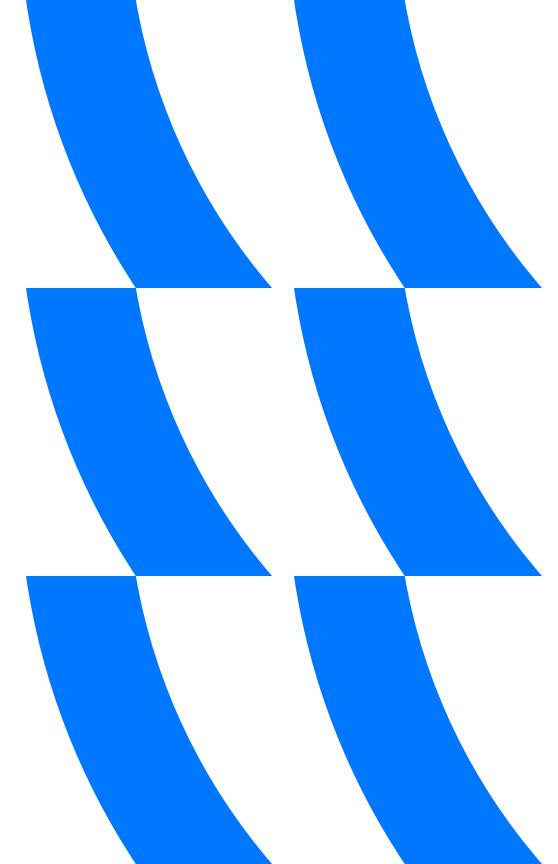
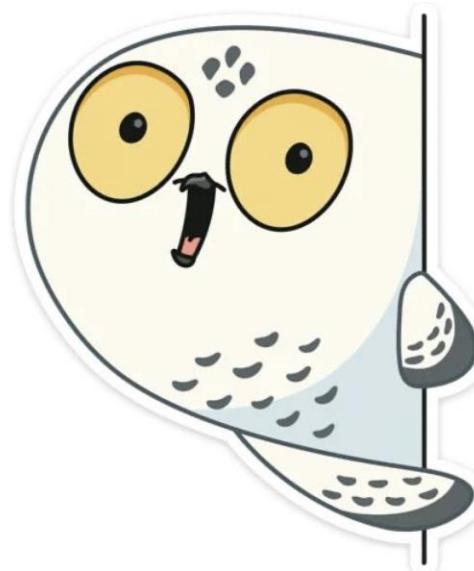


ЭТО ТОЧНО
НЕ БАГ?

Пример 1/3:

ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Статус
SC001	Ввод корректного города обучения без школы	<ol style="list-style-type: none">Перейти на главную дружбНажать на «поиск друзей по школе»В появившемся попапе в разделе «город» ввести городНажать «сохранить»	login: hIIII pass: qwertY1	Перехода не будет, поле добавления школы подсветится с ошибкой	PASSED
SC002	Ввод корректного города обучения со школой (добавление школы в профиль)	<ol style="list-style-type: none">Перейти на главную дружбНажать на «поиск друзей по школе»В появившемся попапе в разделе «город» ввести городДобавить школуДобавить года обученияОтметить в чекбоксе «добавить школу в профиль»Нажать «сохранить»	login: user pass: qwertY1	Все корректно	PASSED

Как можно улучшить?



Улучшения

- Не писать очевидные шаги (запустить приложение, авторизоваться и т.д. выносим в предварительные условия)
- Не писать лишние детали (название файла)

Пример 2/3:

ID	Описание	Шаги	Входные данные	Ожидаемые результаты	Статус
FR001	Добавить друга	<ol style="list-style-type: none">Перейти в профиль по url: https://ok.ru/profile/577961650829Добавить в друзья	login: friend pass: qwerty1	Кнопка «добавить» изменится на «запрос отправлен»	PASSED
SC002	Ввод корректного города обучения со школой (добавление школы в профиль)	<ol style="list-style-type: none">Перейти на главную дружбНажать на «поиск друзей по школе»В появившемся попапе в разделе «город» ввести городДобавить года обученияОтметить в чекбоксе «добавить школу в профиль»Нажать «сохранить»	login: school pass: qwerty1	Школа добавится в профиль. Перейдем на страницу школы с участниками-одноклассниками	PASSED

Пример 3/3:

Preconditions

Пользователь залогинен
Установлен статус
Открыт сайдбар группата, в котором пользователь участник

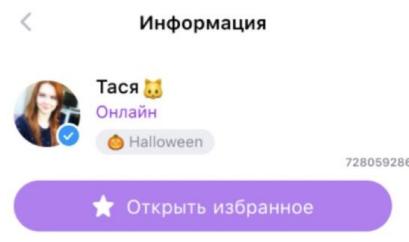
Steps

Step

- Нажать на себя в списке участников чата

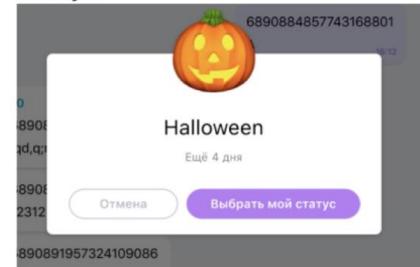
Expected Result

Открывается свой профиль
В профиле отображается статус

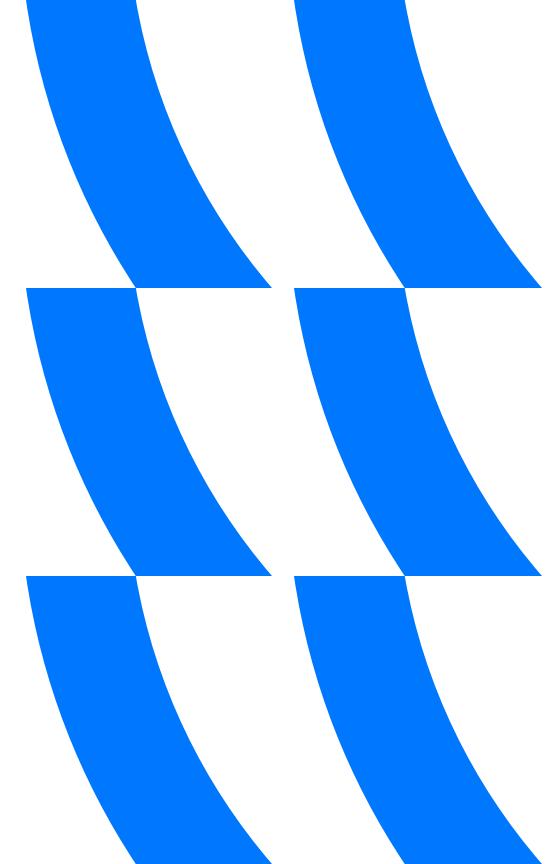


- Нажать на статус

Открывается попап просмотра статуса с кнопками "Отмена" и "Выбрать мой статус"



Чек-лист



Что это?

Чек-лист - это документ, описывающий, что должно быть протестировано

Детализация зависит от требований к отчетам, уровня знания продукта сотрудниками и сложности продукта.

ЕСЛИ НА КАЖДЫЙ ТАСК НУЖНО СОСТАВЛЯТЬ ЧЕК-ЛИСТ



ТО КАКОЙ ЧЕК-ЛИСТ НАДО СОСТАВЛЯТЬ НА ТАСК ПО СОСТАВЛЕНИЮ ЧЕК-ЛИСТА?

Из чего состоит:

1

Порядковый номер
(ID)

2

Заголовок
(Краткое описание)
может состоять из
заголовков тест-кейсов

В чек-листе нет(!) описания фактического результата.
Возможен статус – passed / failed

Плюсы и минусы

-  История тестов — можно выделить Failed и перепроверить только их
-  Слабый уровень детализации
-  Наглядность — легко оценить состояние продукта и стадию проверок
-  Отсутствие подробной документации — трудность для новых сотрудников

Пример 1/3:

Заглушки:

- Нет сети
- Нет друзей → есть кнопка "найти друзей"
- Никого на сайте
- Никого в поиске
- Заглушка в "заявки в друзья"
(пока нет новых запросов дружбы)

Пример 2/3:

Экран поиска города:

Ввод корректного города (есть выдача: похожий по названию/итоговый) - город указывается

Некорректный город (выдачи нет - добавить город нельзя)

Иконки городов по дизайну

Кнопка back корректная

Центрирование текста и иконок в выпадающем меню

Подсказка по выбору города работает от клавиатуры и от копирования\вставки текста.

Экран поиска школы:

Ввод корректной школы (есть выдача: похожие по названию + нужная) - школа указывается

Иконки + центрирование текста (вообще дизайн)

Выбор корректного года

Логика проставления некорректного года:

если год окончания раньше начала

или начало позже окончания - то подставляется "не выбрано" и дизайблится чекбокс

Ошибка, если школа не найдена

При отметке в чек-боксе школа добавляется в профиль

Без отметки - не добавляется

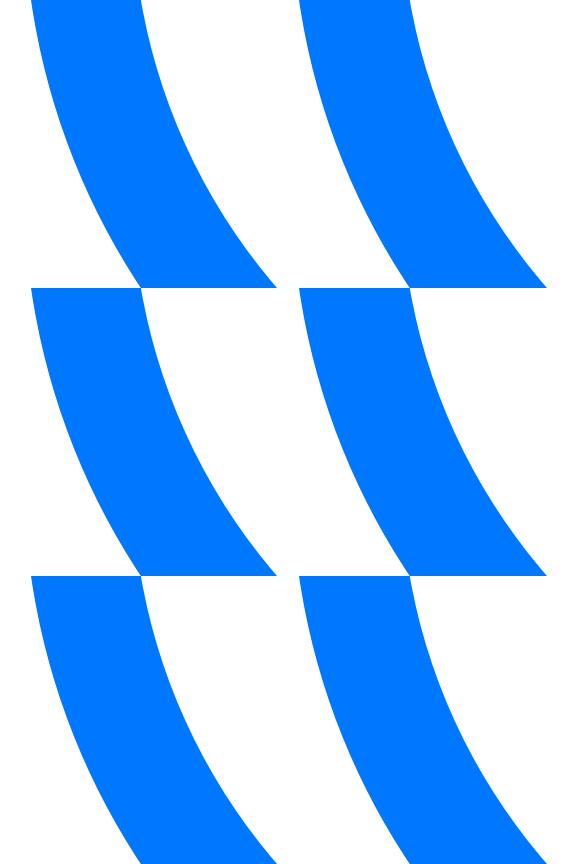
При disabled чек-боксе показывается текст "Выберите годы обучения в школе, если хотите добавить её в свой профиль"

Подсказка по выбору школы работает от клавиатуры и от копирования\вставки текста.

Пример 3/3:

	Адрес	Карта	Заказы	Несколько товаров (одинаковые/разные/разные фильтры одного товара)	Платная доставка
Корзина					
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	

Тест-кейс и чек-лист – в чем отличие?



Тест кейс и чек-лист – в чем отличие?

В чек-листе перечисляют аспекты ПО, которые нужно проверить. Когда составляют тест-кейс, описывают состояние программного обеспечения и то, как его изменяют. То есть чек-листом определяют, что тестировать.

А тест-кейсом — как тестировать. Чек-лист подойдет в качестве исходного документа, чтобы составить тест-кейсы.

Основы документации:

- I. Простота интерпретации
- II. Открытость документации
- III. Актуальность



Тестовое покрытие

Тестовое покрытие - это одна из метрик оценки качества тестирования, представляющая из себя плотность покрытия тестами требований, либо исполняемого кода.

Покрытие требований - оценка покрытия тестами функциональных и нефункциональных требований к продукту путем построения матриц трассировки.

Расчет тестового покрытия относительно требований проводится по формуле:

$$T_{cov} = (L_{cov} : L_{total}) * 100\%,$$

где:

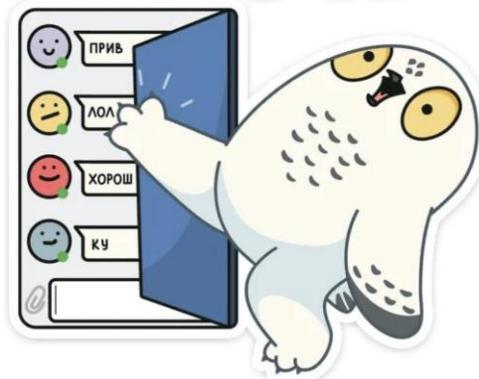
T_{cov} - тестовое покрытие

L_{cov} - количество требований, проверяемых тест кейсами L_{total} - общее количество требований

Отчет об ошибке *(баг-репорт)*



ВОШЁЛ В ЧАТ

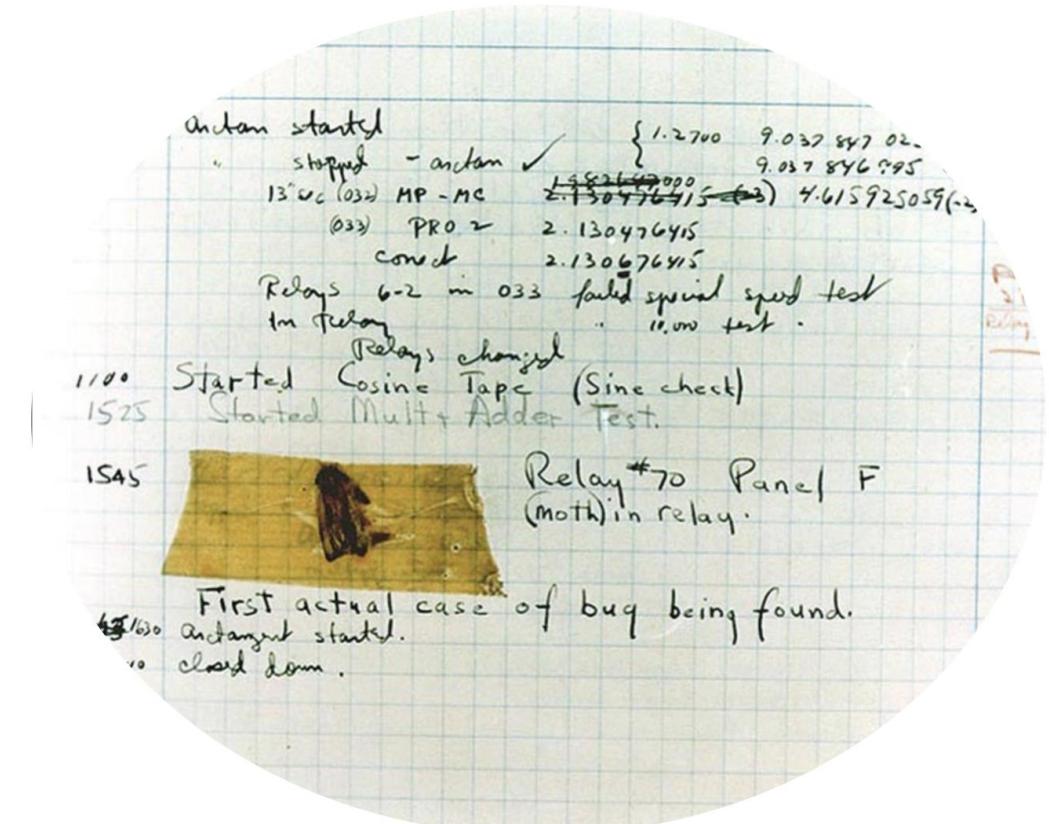


Дефекты в ПО,
составление баг-
репортов,
приоритизация багов

• • •

Предыстория:

9 сентября 1947 года операторы в Гарвардском университете, разобрали вычислительную машину Mark II и нашли мотылька, застрявшего между контактами электромеханического реле. Извлечённое насекомое было вклеено в технический дневник с сопроводительной надписью: «First actual case of bug being found».



Баг/Дефект

Дефект - это отклонение действительного результата от ожидаемого

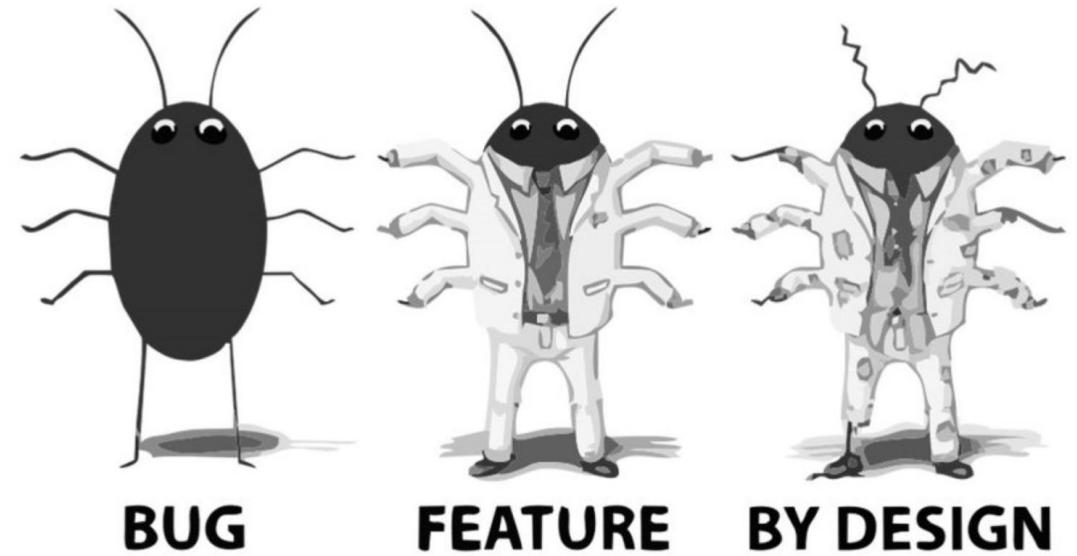
Тестирование **демонстрирует** наличие дефектов, но **не гарантирует** их отсутствие

Исчерпывающее тестирование невозможно

Зачастую дефекты живут группами

Это точно баг?

- Требования
- Требования
- Требования
- Пользовательский опыт



Отчет о найденном дефекте

Баг репорт - это документ, описывающий ситуацию или последовательность действий, приведшую к некорректной работе объекта тестирования с указанием причин и ожидаемого результата.

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)
- Статус (Status)
- Автор (Author)
- Назначен на (Assigned To)
- Описание (Description)
- Прикрепленный файл (Attachment)

Из чего состоит?

- Заголовок (Summary)

Короткое описание проблемы, явно указывающее на причину и тип ошибочной ситуации.

Принцип «Где? Что? Когда?»:

ГДЕ?

В каком месте интерфейса пользователя или архитектуры найдена проблема

ЧТО?

Что происходит или не происходит согласно спецификации или вашему представлению о нормальной работе программного продукта.

КОГДА?

В какой момент работы программного продукта, по наступлению какого события или при каких условиях проблема проявляется.

Хороший заголовок?

- Неправильная работа уведомлений
- Плохо работает поле ввода в настройках
- Падает приложение
- Не работает установка аватара на окне авторизации

Хороший заголовок?

- Неправильная работа уведомлений ✗
- Плохо работает поле ввода в настройках ✗
- Падает приложение ✗
- Не работает установка аватара на окне авторизации ✗

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)

Название тестируемого проекта/модуля

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)

Название части или функции тестируемого продукта

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)

Версия приложения/номер сборки, в
которой найден дефект
прим. 9.01(9410)

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность/серьезность (Severity)

S1 Блокирующий (Blocker)

S2 Критический (Critical)

S3 Значительный (Major)

S4 Незначительный (Minor)

S5 Тривиальный (Trivial)

Из чего состоит?

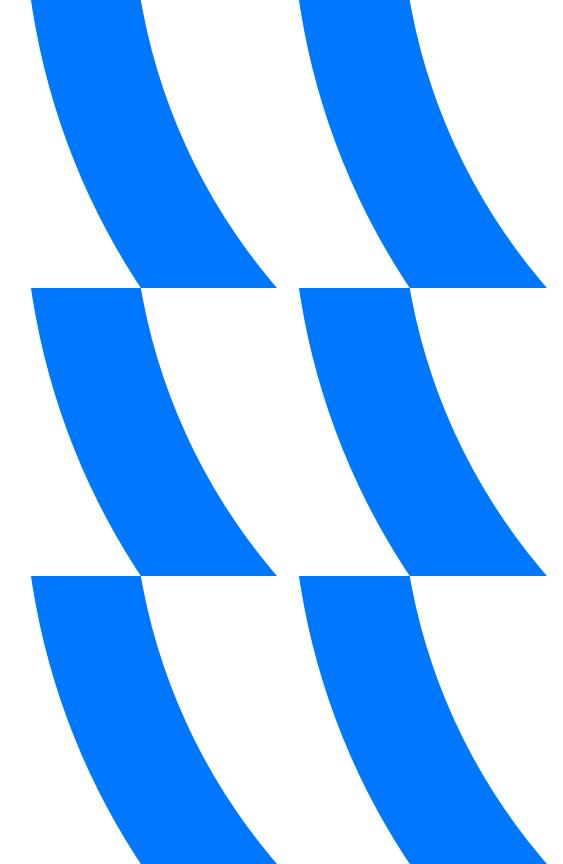
- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)

P1 Высокий (High)

P2 Средний (Medium)

P3 Низкий (Low)

В чем отличия?



В чем отличия?

Серьезность (Severity) - влияние дефекта на работоспособность приложения.

Приоритет (Priority) - очередность выполнения задачи или устранения дефекта (инструмент менеджера)



Как оценить серьезность?

- Базовый функционал
- Основной функционал
- Дополнительный функционал – как это узнать?

Как оценить серьезность?

- Базовый функционал
- Основной функционал
- Дополнительный функционал – как это узнать? → НИКАК

Подсказка:

Бизнес-требования и продуктовая воронка



Оформить и оплатить заказ

Как оценить серьезность?

Потери данных, изменения в данных

Crash

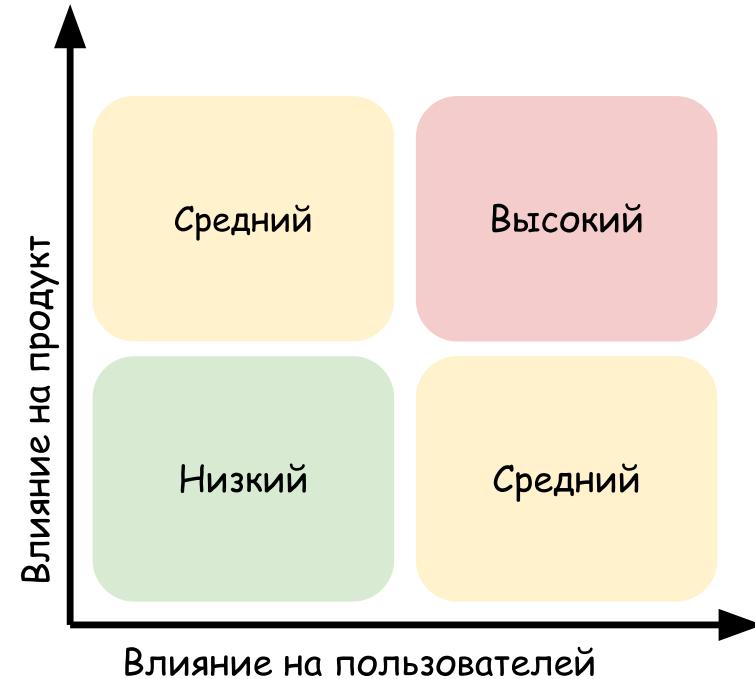
Не работает или неверно работает основной функционал

Не работает или неверно работает НЕ основной функционал

Неверно работает, но можно придумать костыль

Интерфейс не соответствует дизайну

Неправильный текст



Как много пользователей затрагивает найденная проблема?

Как оценить приоритет?

- Блокируется или затрудняется работа основной функции приложения
- “Заметность”
- Частота воспроизведения проблемы
- Повлиял ли апдейт?
- Репутационные потери
- Финансовые потери



Псевдо-блокеры

- Завысить приоритет, чтобы на баг обратили внимание
- Неправильно оценить затронутый функционал
- Состояние паники (AAAA, крэш! AAAA, у нас не работает)



Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)
- Статус (Status)

Новый (backlog)
Открыт (open)
В работе (in progress)
Отклонен (rejected)
Закрыт (done)

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)
- Статус (Status)
- Автор (Author)

Создатель

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)
- Статус (Status)
- Автор (Author)
- Назначен на (Assigned To)

Кто будет исправлять

Из чего состоит?

- Заголовок (Summary)
 - Проект (Project)
 - Компонент приложения (Component)
 - Номер версии (Version)
 - Критичность (Severity)
 - Приоритет (Priority)
 - Статус (Status)
 - Автор (Author)
 - Назначен на (Assigned To)
 - Описание (Description)
- Информация об окружении, на котором был найден баг
 - Шаги воспроизведения
 - Фактический результат
 - Ожидаемый результат

Из чего состоит?

- Заголовок (Summary)
- Проект (Project)
- Компонент приложения (Component)
- Номер версии (Version)
- Критичность (Severity)
- Приоритет (Priority)
- Статус (Status)
- Автор (Author)
- Назначен на (Assigned To)
- Описание (Description)
- Прикрепленный файл (Attachment)

Дополнительно:

1. Прикладываем видео/скриншоты, если можно их приложить
2. Ссылки на макеты/документацию, если ссылаемся на них
3. Дампы, ссылки на дампы, логи и другая дебажная
информация
4. Указываем версию приложения, версию системы, устройство,
если проблема Device Specific



Шаблон заведения баг репорта

Версия ОС

Девайс

Ссылка на билд

(номер билда)

Предварительно

Залогиниться ботом newBot/password

Шаги

- 1.
 - 2.
-

Фактический результат:

Ожидаемый результат:

Воспроизводится в проде

Да/нет

Окружение, на котором воспроизводится

Response

Ссылка на firebase/скриншоты/скринкаст/любая
доп.информация

Тип проблемы:

Укажите тип проблемы...



Заголовок:

Кратко опишите суть бага

Шаги воспроизведения:

1. Открыть раздел
2. Активировать поле ввода
- 3.

Фактический результат:

При совершении действия А, происходит Б

Ожидаемый результат:

При совершении действия А, должно происходить В



Скрыть документы из публичного доступа

Приоритет:

Средний



Устройства:



Apple iPhone SE SE

iOS 10

Добавить устройство

Платформы:

Укажите платформы, на которых воспроизводится баг



Пример 1/3

Заголовок

Crash приложения при открытии профиля
пользователя из чата

Шаги для воспроизведения

1. Открыть групповой чат
2. Получить сообщение от пользователя А
3. Нажать на аватарку пользователя А

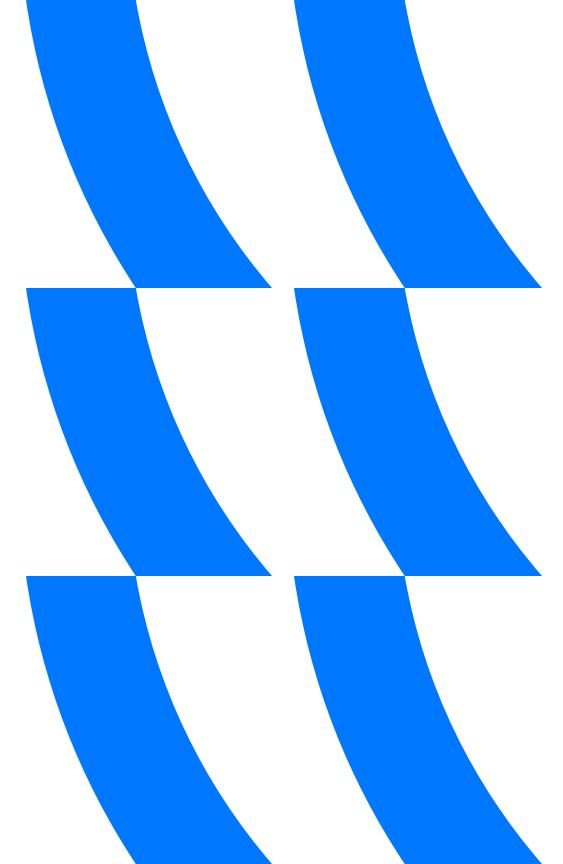
Фактический результат

Приложение падает [файл дампа, ссылка на дамп]

Ожидаемый результат

Открывается профиль пользователя А **Окружение**

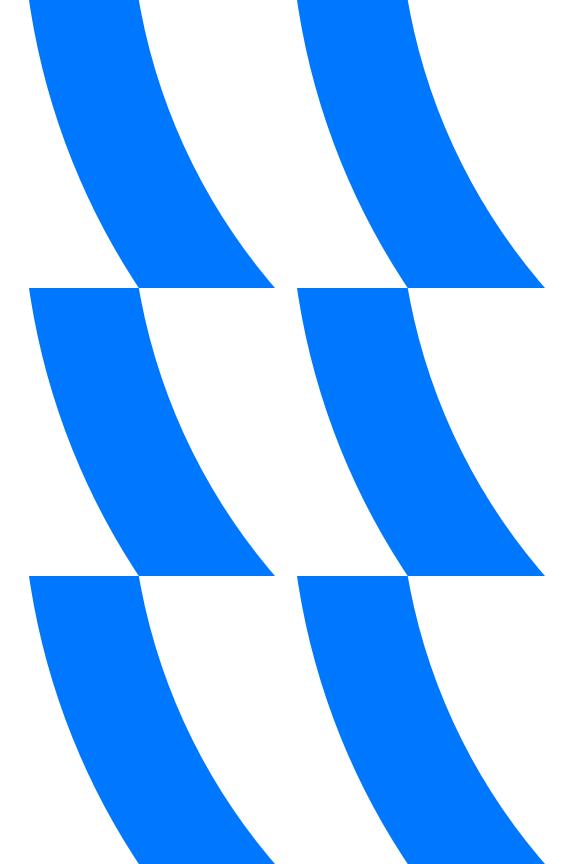
Версии 1.1.1, устройство/ОС



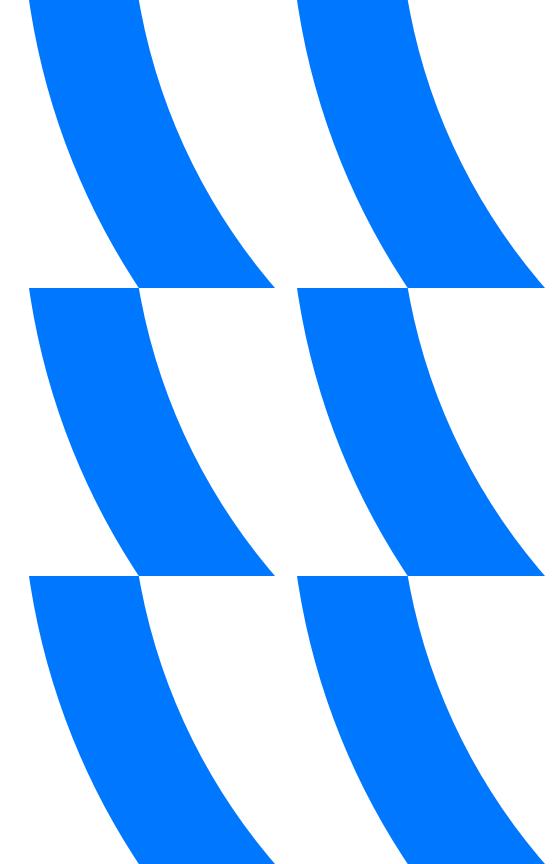
СОВА, КРУШИТЬ



- Что?
- Сайт для покупки товаров



Какой функционал
базовый, основной,
дополнительный?



Не работает кнопка «купить»

- S1 Блокирующий (Blocker)
- S2 Критический (Critical)
- S3 Значительный (Major)
- S4 Незначительный (Minor)
- S5 Тривиальный (Trivial)
- Уточнить у аналитика

Не работает кнопка «купить»

- **Блокирующий (Blocker)**
- Критический (Critical)
- Значительный (Major)
- Незначительный (Minor)
- Тривиальный (Trivial)
- Уточнить у заказчика

Не обновляется баббл при добавлении товара в корзину



- Р1 Высокий (High)
- Р2 Средний (Medium)
- Р3 Низкий (Low)
- Уточнить у разработчика

Не обновляется баббл при добавлении товара в корзину

- Высокий (High)
- Средний (Medium)**
- Низкий (Low)
- Уточнить у заказчика

Как должно отображаться окно для пользователя, который не авторизован?

- Заглушка и текст «чтобы просмотреть товары – нужно авторизоваться»
- Есть список товаров, их можно фильтровать, если кликнуть по карточке – ничего не произойдет
- Есть список товаров, их нельзя фильтровать, если кликнуть по карточке, то будет переход на окно авторизации.
- Уточнить у заказчика

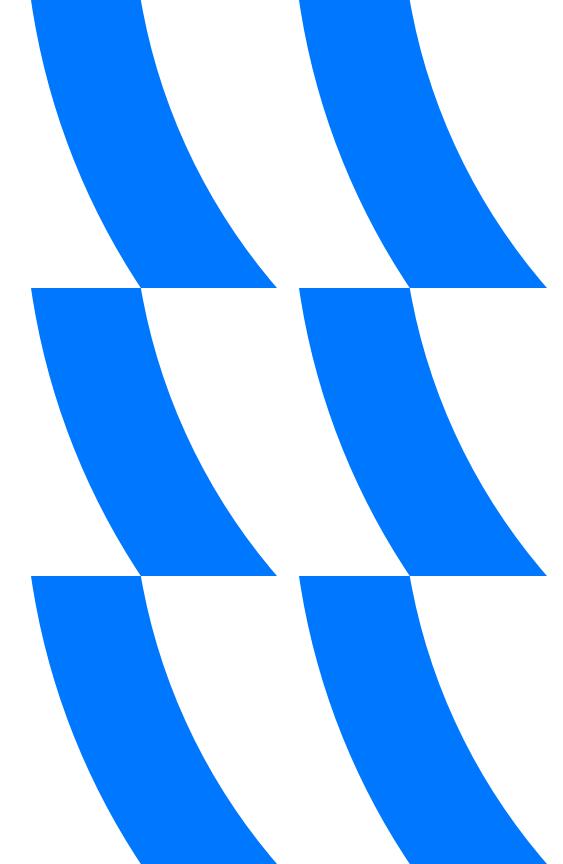
Как должно отображаться окно для пользователя, который не авторизован?

- Заглушка и текст «чтобы просмотреть товары – нужно авторизоваться»
- Есть список товаров, их можно фильтровать, если кликнуть по карточке – ничего не произойдет
- Есть список товаров, их нельзя фильтровать, если кликнуть по карточке, то будет переход на окно авторизации
- Уточнить у заказчика

Системы отслеживания ошибок

...

Как рассказать о найденном дефекте?



Способы передачи задачи

- Голосом
- Бумажным письмом
- Бумажной документацией
- Email
- В мессенджере
- В облачных редакторах

Какие недостатки?

- Нельзя быстро внести и согласовать изменения
- Не у всех участников процесса (или возможных участников) есть быстрый доступ к задаче
- Нельзя отследить изменения
- Сложно определить статус задачи
- Приватность данных может быть нарушена

Способы передачи задачи

- Голосом
- Бумажным письмом
- Бумажной документацией
- Email
- В мессенджере
- В облачных редакторах
- **В специальной системе**

Системы отслеживания ошибок

Bug tracking system — прикладная программа, разработанная с целью помочь разработчикам программного обеспечения (программистам, тестировщикам и др.) учитывать и контролировать дефекты, найденные в программах, а также следить за процессом устранения этих дефектов.

When developer
finds a bug



When tester
finds a bug



Зачем?

1. Каждый, кому следует знать о проблеме, должен узнавать о ней сразу же после составления отчета.
2. Ни одна из ошибок не должна остаться неисправленной просто потому, что о ней забыли или потому что так решил программист.
3. Минимум ошибок должны оставаться неисправленными из-за проблем взаимодействия сотрудников.

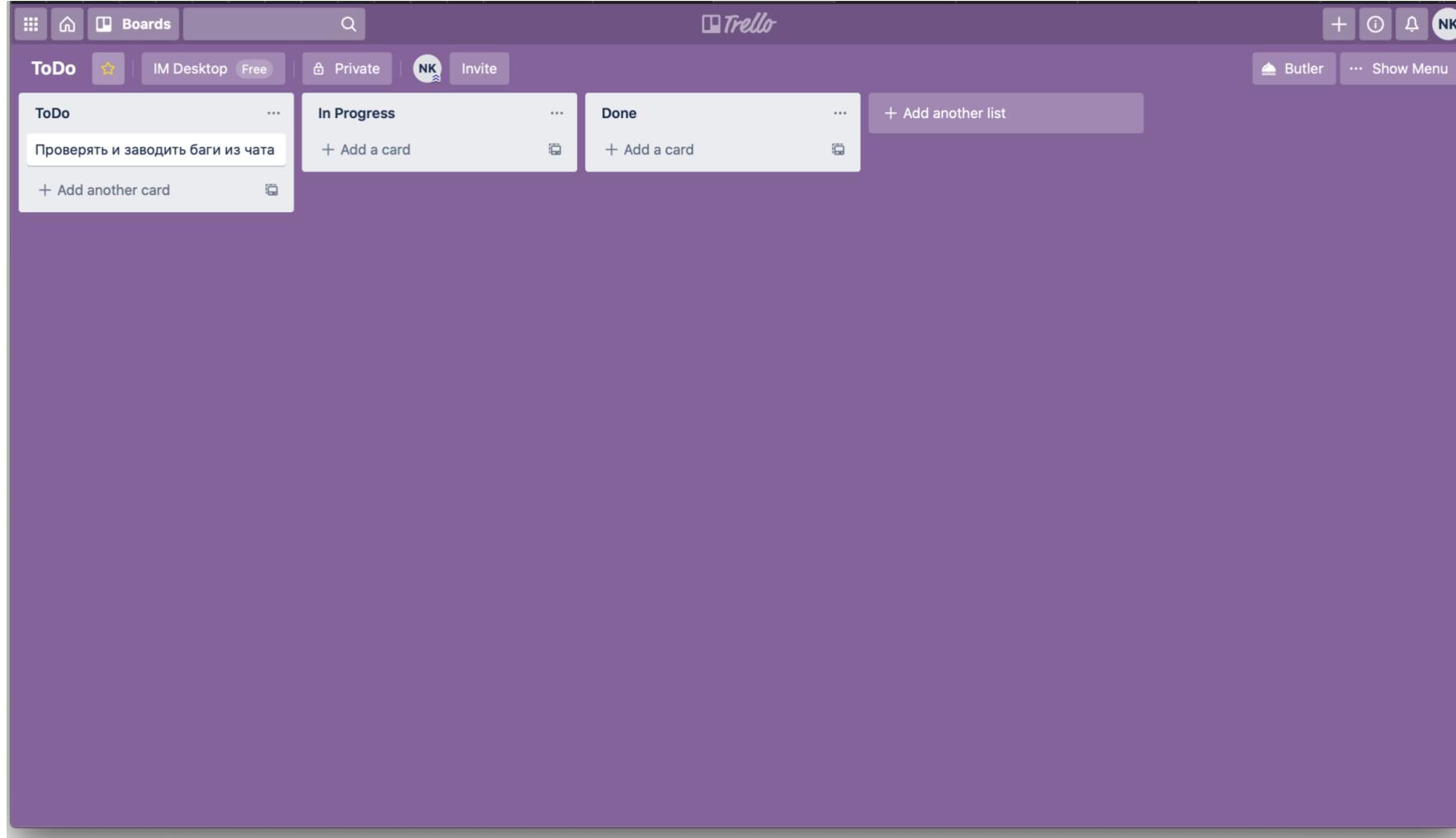
Какие бывают?



Какие бывают?



Trello



Trello

Проверять и заводить баги из чата
in list ToDo

Description

Add a more detailed description...

Activity

NK Write a comment...

ADD TO CARD

- Members
- Labels
- Checklist
- Due Date
- Attachment
- Cover

POWER-UPS

Get Power-Ups

Get unlimited Power-Ups, plus much more.

[Upgrade Team](#)

ACTIONS

- Move
- Copy
- Make Template
- Watch
- Archive
- Share

Какие бывают? §



Jira

Создание задачи

Настройте поля

Тип задачи* Задача ?

Связанный проект Не заполнено ▼

Тема*

Исполнитель Автоматически ▼

Назначить меня

Приоритет* Стандартный ?

Описание

Стиль ▼ | **B** *I* U A ^{A^o} _{A_o} |

Компоненты ▼

Создать еще одну Создать Отменить

Jira

Create Issue

Issue Type*	<input checked="" type="checkbox"/> Bug	<input type="button" value=""/>
Summary*	<input type="text"/>	
Priority	<input checked="" type="radio"/> None	<input type="button" value=""/>
Bug Environment*	<input type="text"/> None	
Platforms	<input type="text"/>	
Component/s	None	
Story Points	<input type="text"/>	
Measurement of complexity and/or size of a requirement.		
Business Value	<input type="text"/>	
Measurement of business value of a requirement.		
Description	<input type="text"/> B I U A A^o A^o^o P U I II @ + x	
<p>*Шаги воспроизведения: #</p> <p>*Фактический результат:</p> <p>*Ожидаемый результат:</p>		
Attachment	<input type="text"/> - ? Drop files to attach, or browse.	
Project affected*	<input type="text"/> None	
Assignee	<input type="text"/>	
Assign to me		
Responsible manager	<input type="text"/>	
Responsible designer	<input type="text"/>	
HTML Developer	<input type="text"/>	

Какие бывают?



<https://todoist.com/>



Поступочки в МОЕМ

Изменить

Название

Экзамен| в аспирантуру

Цвет

Зеленый

Добавить в избранное

Отображение

Список

Доска

Отображение общих проектов синхронизируется между участниками. [Узнать больше.](#)

Отмена Сохранить

Отобра...

детали в одном центральном, общем
проекте.

Экзамен в аспирантуру

Поделиться

Отображение

Комментарии

ooo

Математические основы ин... 1 ooo

Комп. технологии

Системный анализ 0

ooo

Добавить

Основные вопросы

0/4

Сохранить

Отмена

Добавить задачу

Добавить задачу

Название задачи

Описание



• Эк... / Мате...



Основные вопросы

≡ Описание

▼ Подзадачи 0/4

- Основы теории множеств и бинарных отношений. Множества конечные и бесконечные. Операции над множествами. Декартово произведение.
- Свойства бинарных отношений. Отношения эквивалентности. Частичноупорядоченные бинарные отношения. Экстремальные характеристики упорядоченных множеств
- Математическая логика. Основные законы математической логики.
- Булева алгебра. Логика высказываний. Булевы функции, канонические формы задания булевых функций. Понятие полной системы. Критерий п

+ Добавить подзадачу

k

Комментировать

Проект

• Экзамен ... / Математические...

Срок выполнения +

Приоритет

P4

Метки +

Напоминания PRO



Местоположение PRO



Какие бывают?



<https://clickup.com/>



Team Space > Projects > Project 1 1 of 3

TO DO ✓   Share ...

CREATED Oct 13, 1:08 am TIME TRACKED 0:00:00  

Задача 1

Write something or type '/' for commands

You created this task by copying #8678g9yjz (You Don't Have Access) 2 mins

You assigned to: You Just now

You added watcher: You Just now

To Do Add ▾ Mine

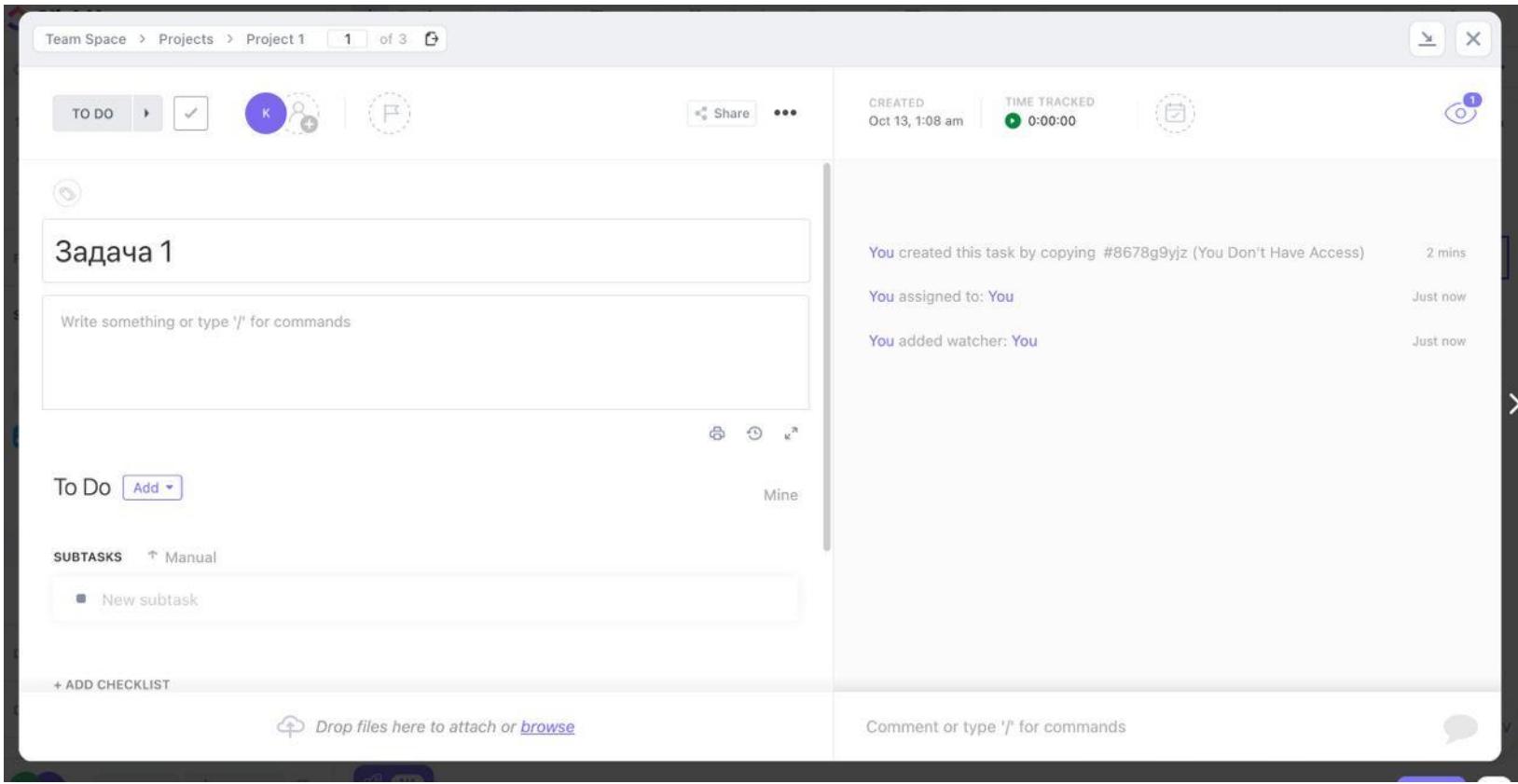
SUBTASKS Manual

New subtask

+ ADD CHECKLIST

Drop files here to attach or [browse](#)

Comment or type '/' for commands 





CREATED

Oct 13, 1:08 am

TIME TRACKED

0:00:00



Start date

Due date

Today

Fri

October 2023

Today



Later

3:12 am

SU

MO

TU

WE

TH

FR

SA

Tomorrow

Sat

1

2

3

4

5

6

7

This weekend

Sat

8

9

10

11

12

13

14

Next week

Mon

15

16

17

18

19

20

21

Next weekend

21 Oct

22

23

24

25

26

27

28

2 weeks

27 Oct

29

30

31

1

2

3

4

4 weeks

10 Nov

5

6

7

8

9

10

11

▼ Set Recurring

Общие процессы

- Назначить задачу на себя
- Переводить задачи в правильный статус
- Следить за оформлением задачи
(все вложения прикрепились, после тебя
задача на правильном исполнителе)

Jira Query Language

JQL — язык **Jira** для поисковых запросов. Удобно использовать для поиска задач, более гибкая настройка

- project = %1
- assignee = %1
- reporter = %1
- status = %1
- project in (%1, %2, ..., %n)

Jira Query Language

JQL — язык **Jira** для поисковых запросов. Удобно использовать для поиска задач, более гибкая настройка

- project = %1
- assignee = %1
- reporter = %1
- status = %1
- project in (%1, %2, ..., %n)

Примеры:

```
status = "Ready for Review" OR status = Committed  
resolved >= startOfDay(-1) and resolved <= endOfDay(-1)  
Resolved >= startOfDay()
```

Фильтры

- Быстрый поиск задач
- Отчеты на почту
- Графики

Настройка Dashboard

Визуальное представления задач из фильтра на одном экране

Можно вывести с помощью списка, диаграммы, графика



Техники-тест дизайна



Введение



Критерии составления требований:

Тест-дизайн – это этап процесса тестирования ПО, на котором проектируются и создаются тест кейсы, в соответствии с определёнными ранее критериями качества и целями тестирования.



Разработчик выпустил фикс
в пятницу вечером

Цель: Создать наборы тестовых случаев, обеспечивающих оптимальное тестовое покрытие.

- 1.Исключить непродуктивные тест-кейсы и сократить общее количество кейсов
- 2.Покрыть тестами как можно больше функциональности
- 3.Провести все тесты и не пропустить ничего важного



Роли в тест-дизайне

Тест-аналитик

"ЧТО тестировать?"



Тест-дизайнер

"КАК тестировать?"



Роли в тест-дизайне

Задача тест-аналитиков и дизайнеров: используя различные стратегии и техники тест-дизайна, создать набор тестовых случаев, обеспечивающий **оптимальное тестовое покрытие** тестируемого приложения. На большинстве проектов обе эти роли выполняет QA инженер.

"ЧТО и КАК тестировать?"



Для работы с кодом (white-box)

- Покрытие операторов
- Покрытие условий
- Покрытие путей
- Покрытие функций
- Покрытие вход/выход
- Покрытие значений параметров

В работе с требованиями (black-box)

- Классы эквивалентности
- Граничные значения
- Попарное тестирование
- Таблица принятия решений
- Диаграмма состояний и переходов
- Тестирование вариантов использования
- Доменное тестирование.

Классы эквивалентности



Классы эквивалентности

“The ART of Software testing” Гленфорда Майерса: выделяя класс эквивалентности, мы утверждаем, что если ошибка не найдена во время теста одного из элементов класса, весьма маловероятно, что мы найдем его в другом элементе класса.

“Тестирование программного обеспечения” Сэм Канер: если мы от выполнения двух тестов ожидаем одинаковый результат, можно считать их эквивалентными.

Группа тестов образует класс эквивалентности, если вы утверждаете, что:

- Они все проверяют одно и то же;
- Если какой-то элемент класса ловит баг, то другой элемент тоже поймает его.
- Если какой-то элемент класса не ловит баг, то другой элемент тоже не поймает его.

Классы эквивалентности

- направлены на поиск одной и той же ошибки;
- если один из тестов обнаруживает ошибку, другие скорее всего, тоже её обнаружат;
- если один из тестов не обнаруживает ошибку, другие, скорее всего, тоже её не обнаружат;
- тесты используют схожие наборы входных данных;
- для выполнения тестов мы совершаем одни и те же операции;
- тесты генерируют одинаковые выходные данные или приводят приложение в одно и то же состояние;
- все тесты приводят к срабатыванию одного и того же блока обработки ошибок;
- ни один из тестов не приводит к срабатыванию блока обработки ошибок.

Классы эквивалентности (пример)

Допустимые значения (от 1 до 100);

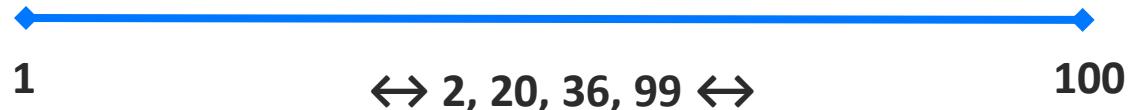


Классы эквивалентности (пример)

Допустимые значения (от 1 до 100);

Недопустимые значения:

- от $-\infty$ до 0;
- от 101 до $+\infty$;
- специальные символы (# @ + - / _ : ; “ ‘ и т.д.);
- буквы.



Классы эквивалентности (пример)

При оплате инвойса:

- В течение **24** часов: получает скидку 15% на пакет рекламы, премиум-поддержку и месячную поддержку бухгалтерии.
- В течение **48** часов: получает скидку 10% на пакет рекламы и месячную поддержку бухгалтерии.
- В течение **72** часов: получает скидку 10% на пакет рекламы.
- В течение **96** часов: получает скидку 5% на пакет рекламы.
- **Более 96** часов: инвойс надо запросить повторно.

Классы эквивалентности:

1. 24-48 часов
2. 49-72 часа
3. 73 часа-96 часов
4. 97 часов

Классы эквивалентности (пример)

При оплате инвойса:

- В течение **24** часов: получает скидку 15% на пакет рекламы, премиум-поддержку и месячную поддержку бухгалтерии.
- В течение **48** часов: получает скидку 10% на пакет рекламы и месячную поддержку бухгалтерии.
- В течение **72** часов: получает скидку 10% на пакет рекламы.
- В течение **96** часов: получает скидку 5% на пакет рекламы.
- **Более 96** часов: инвойс надо запросить повторно.

~~Классы эквивалентности:~~

1. ~~24-48 часов~~
 2. ~~49-72 часа~~
 3. ~~73 часа-96 часов~~
 4. ~~97 часов~~
-
1. **0-23 часа 59 минут 59 секунд**
 2. **24 часа - 47 часов 59 минут 59 секунд**
 3. **48 часов - 71 час 59 минут 59 секунд**
 4. **72 часа - 95 часов 59 минут 59 секунд**
 5. **96 часов-??**

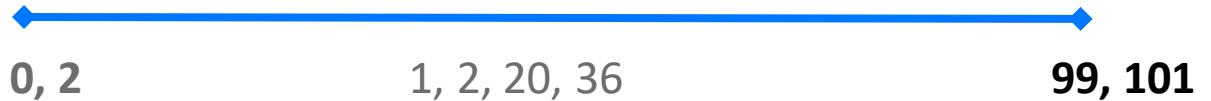
Границы значение



Анализ граничных значений

Граничные значения — это те места, в которых один класс эквивалентности переходит в другой

1. Выделить классы эквивалентности;
2. Определить граничные значения этих классов;
3. Понять, к какому классу будет относиться каждая граница;
4. Провести тесты по проверке значения до границы, на границе и сразу после границы



Анализ граничных значений

Важно! Есть разные типы границ.

- **Логическая:** ограничение, накладываемое логикой работы приложения, а не самой программой.

Пример: 0 символов для поля ввода является логической границей, так как отрицательной длины строки не бывает.

- **Технологическая:** ограничение, накладываемое используемой технологией.

Пример: количество открытых соединений в БД, ограничение для длины поля в Oracle БД в 5000 символов.

- **По требованиям:** ограничение, накладываемое разработчиком/аналитиком/заказчиком.

Пример: Допуск к порталу лицам от 18 лет.

Анализ граничных значений (пример)

При оплате инвойса мы выделили классы эквивалентности:

1. 0-23 часа 59 минут 59 секунд
2. 24 часа - 47 часов 59 минут 59 секунд
3. 48 часов - 71 час 59 минут 59 секунд
4. 72 часа - 95 часов 59 минут 59 секунд
5. 96 часов-??

Граничные значения:

0

1 секунда

23 часа 59 минут 59 секунд

24 часа 0 секунд

24 часа 1 секунда

47 часов 59 минут 59 секунд

48 часов 0 секунд

48 часов 1 секунда

71 час 59 минут 59 секунд

72 часа 0 секунд

72 часа 1 секунда

95 часов 59 минут 59 секунд

96 часов 0 секунд

96 часов 1 секунда

Доменное тестирование



Доменное тестирование

Методика разработки тестов, относящаяся к методу черного ящика, использующаяся для определения действенных и эффективных тестовых сценариев в случаях, когда множественные параметры могут или должны быть протестированы одновременно. Методика базируется и обобщает методы эквивалентного разбиения и анализа граничных значений.

Анализ различных значений, поиск их взаимосвязи и составление эффективных тестов.

Суть: разделить набор условий тестирования на те значения, которые можно считать одинаковыми, и за счет этого протестировать эффективней.

Доменное тестирование (пример)

Возьмем для примера такие требования:

- Размер файла: до 200 МБ
- Имя файла: от 5 до 24 символов, только латиница
- Форматы файлов: только изображения

В этом случае нужны такие проверки:

- 1.Загрузить файл менее 200 МБ
- 2.Загрузить файл с именем hexlet
- 3.Загрузить файл с расширением .jpg

Можно заменить одной проверкой:

- 1.Загрузить файл менее 200 МБ, с именем hexlet.jpg

Таблица принятия решений



Таблица принятия решений

Decision Table (таблица принятия решений) — техника, помогающая наглядно изобразить комбинаторику условий из ТЗ. Способ компактного представления модели со сложной логикой; инструмент для упорядочения сложных бизнес требований, которые должны быть реализованы в продукте.

Это взаимосвязь между множеством условий и действий.

Таблица принятия решений содержит следующие элементы:

- **Условия** — список возможных условий
- **Варианты** — комбинация из выполнения и/или невыполнения условий этого списка
- **Действия** — список возможных действий (вариантов исхода)

Таблица принятия решений

Как это работает?

1. Наглядность — таблица нагляднее текста.
2. Нарисовал таблицу = записал тест-кейсы. Поменял в заголовках слово «правило» на «тест-кейс» = готовые кейсы.
3. Наглядность поможет найти баги в документации.
4. Таблица помогает взглянуть на ТЗ свежим взглядом, по-новому. Пока мы пытаемся перекомбинировать условия, составляя таблицу, мы можем заметить пропущенный ранее баг.

Таблица принятия решений (пример)

Как это работает?

Пример из практики: определенный вид страховки на машину выдаётся людям, удовлетворяющим трём **условиям**:

- Возраст: 18-60 лет
- Гражданство: Россия
- Судимости: отсутствуют

Варианты: да/нет

Действия: одобляем страховку/нет

Представляем данную информацию в виде следующей таблицы:

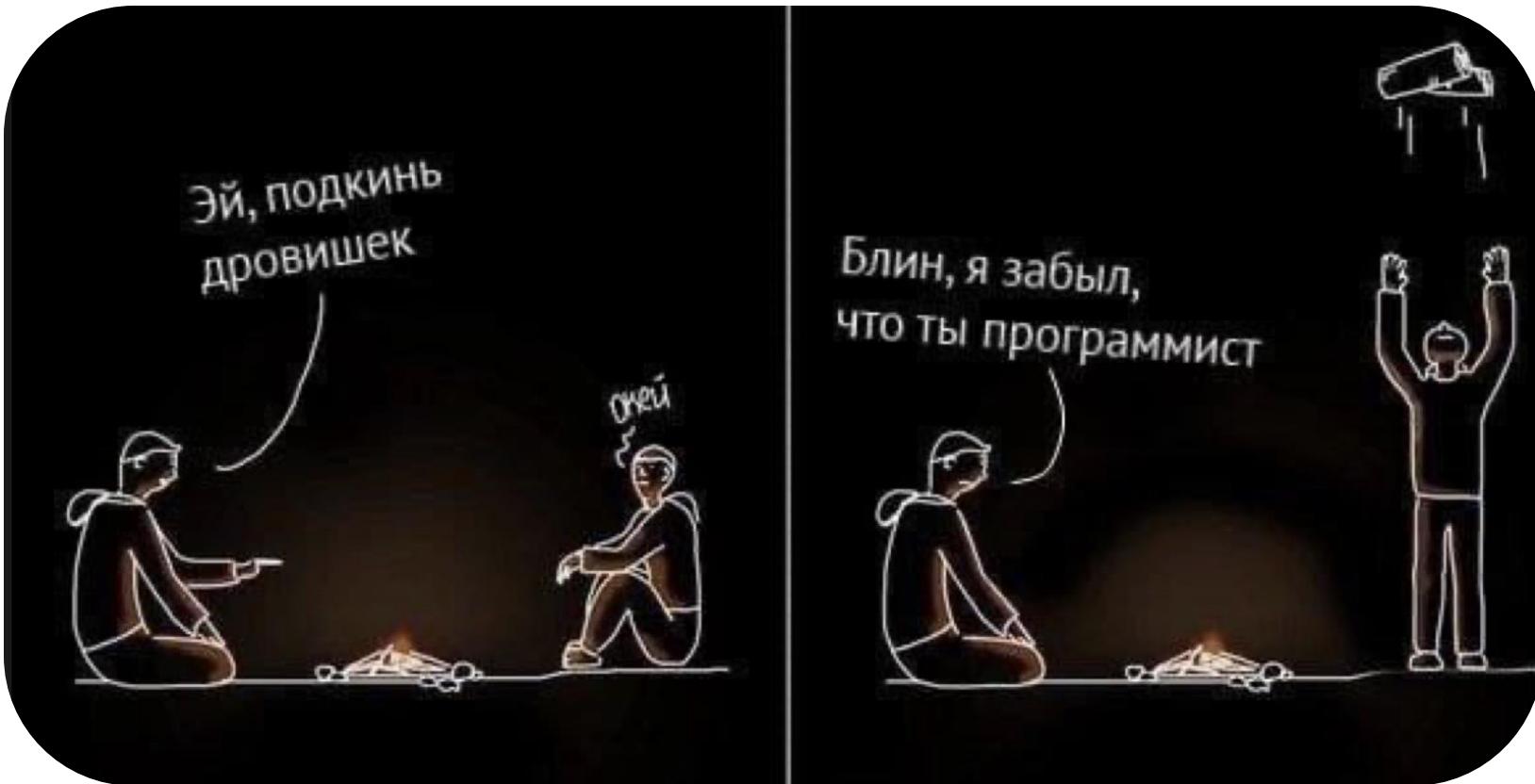
Таблица принятия решений (пример)

Условие	Значение 1	Значение 2	Значение 3	Значение 4
Возраст 18-60	да	да	нет	да
Гражданство РФ	да	да	да	нет
Судимости	нет	да	нет	нет
Действия				
Одобрение	да	нет	нет	нет

Предугадывание ошибок



Предугадывание ошибок



Предугадывание ошибок

Предугадывание ошибки (Error Guessing - EG). Это когда тест аналитик использует свои знания системы и способность к интерпретации спецификации на предмет того, чтобы "предугадать" при каких входных условиях система может выдать ошибку.

Предугадывание ошибок

Предположение об ошибках – это способ предотвращения ошибок, дефектов и отказов, основанный на знаниях тестировщика, включающих:

- Историю работы приложения в прошлом.
- Наиболее вероятные типы дефектов, допускаемых при разработке.
- Типы дефектов, которые были обнаружены в схожих приложениях.

Предугадывание ошибок

В предугадывании ошибок **нет четкой и логической схемы**, которая позволила бы нам составить тест-кейсы.

Эта техника основывается на опыте тестировщика и на его умении думать креативно и деструктивно.

Суть данного метода заключается в том, чтобы найти как можно больше ошибок, попытаться сломать работающий функционал

Обычно используется вкупе с другими методиками, так как является не структурированным способом обнаружения багов.

(freestyle)

Предугадывание ошибок

- Помните о ранее проблемных областях. Как правило, в приложениях определенного типа возникают некоторые распространенные ошибки.
- Используйте информацию об окружении (сервер, операционная система, база данных), в которой размещено приложение.
- Ищите не только ошибки в коде, но и ищите ошибки и неясности в требованиях, дизайне, сборке и т.д.
- Оценивайте исторические данные и результаты испытаний
- Следите за типичными ошибками реализации

Причина - следствие



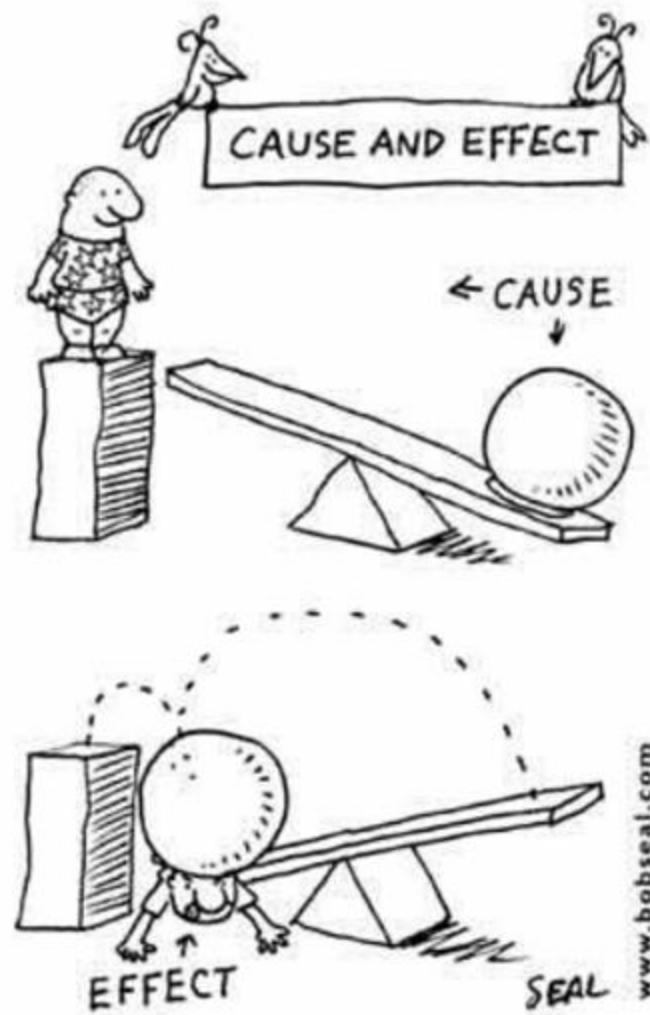
Причина-следствие

Ввод комбинаций условий (причин), для получения ответа от системы (следствие).



ждёт реакцию

Причина-следствие



Причина-следствие

Обычно входит в общую канву разработку тест-кейсов.

План разработки тест кейсов предлагается следующий:

- Анализ требований.
- Определение набора тестовых данных на основании Классов эквивалентности, Границных значений и Предугадывания ошибок.
- Разработка шаблона теста на основании Причины-следствия.

Причина-следствие

Причина	Следствие
Вводим имя “Валерий” в поле “Имя” и нажимаем кнопку “Добавить”	В БД добавляется новый пользователь со значением “Валерий”
Редактируем значение “Имя” в профиле пользователя > “Иннокентий”	Значение поля “БД” изменяется с “Валерий” на “Иннокентий”
Удаляем строку “Иннокентий” из БД	В БД удаляется строка “Иннокентий”

Матрица соответствия требований



Матрица соответствия требований



Оно же – “Traceability Matrix”, оно же – “матрица трассировки требований”

По определению матрица трассируемости — двумерная таблица, содержащая соответствие функциональных требований продукта (functional requirements) и подготовленных тестовых сценариев (test cases).

На пересечении соответствующих строки и столбца ставится отметка, обозначающая, что данное требование покрывается данным тест-кейсом.

Матрица соответствия требований

Req Root Folder: Contract processing NULL Folder Mode: True False

Test Root Folder: Modeling NULL Show Weight: True False

[View Report](#)

14 4 1 of 1 > >|| 100% Find | Next

<YOUR COMPANY NAME> HP Quality Center Report

Requirements Traceability Matrix

Requirements Traceability Matrix		Requirement	#	Sales order															
				Total	Req	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Root Folder: Modeling		Covered	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
		Test	#	Test	Relate	4	4	4	4	4	4	2	2	2	1	0	0	0	0
Contact processing - path 2	1	1	X	10	X	X	X							X	X				
Contact processing - path 1	2	1	X	8				X	X	X	X								
Agree on	3	1	X	2	X														
Check	4	1	X	2		X													
Create contact	5	1	X	2			X												
Determine	6	1	X	2				X											
See customer off	7	1	X	2					X										
Send contact	8	1	X	2						X									
Send original	9	1	X	2								X							
Sign contact	10	1	X	2							X								
Contact processing	11	1	X	1									X						

Матрица соответствия требований

	Тест-кейс 1	Тест-кейс 2	Тест-кейс 3	Тест-кейс 4	Тест-кейс 5
Требование 1					
Требование 2					
Требование 3					
Требование 4					
Требование 5					

Матрица соответствия требованиям

Таким образом, таблица дает визуальное отображение двух параметров:

- наличие в системе требований, которые еще не покрыты (если у требования нет ни одного пересечения с тест-кейсами (достаточное условие);
есть ли в системе избыточное тестирование — если требования имеет несколько пересечений (необходимое условие).

При работе с требованиями это помогает нам:

- визуализировать актуальное состояние реализации;
- разбивать требования на более атомарные и структурировать их;
- отслеживать, есть ли требования, на которые еще не запланирована разработка (пропуск реализации);
- отслеживать, реализовано ли требование в данный момент;
- отслеживать, покрыто ли требование тест-кейсом (пропуск тестирования);
- наглядно отображать приоритезацию требований.

Матрица соответствия требованиям

Привязка требования и тест-кейса может быть:

- 1 к 1 (атомарное требование, которое покрывается одним тест-кейсом, данный тест-кейс покрывает только это требование);
- 1 к n (требование, которое покрывается несколькими тест-кейсами, данные тест-кейсы покрывают только это требование);
- n к n (требование, которое покрываетсяическими тест-кейсами, данные тест-кейсы покрывают это и другие требования).

Если для оценки покрытия мы используем метрику “отношение количества требований к количеству тестовых артефактов”, то связи в матрице должны быть “1 к 1”, а требования максимально декомпозированы.

Матрица соответствия требований



Как составить матрицу требований?

- Начинаем с требований: они декомпозируются и подлежат приоритезации командой QA и\или product-owner. Результатом этапа становится структурированный и приоритезированный список всех требований по данной функциональности.
- Общаемся с командой разработки и проставляем задачи из таск трекера на разработку в матрицу к соответствующим требованиям. В результате мы можем отследить трассируемость требований и задач на разработку.
- Разрабатываем тест-кейсы и\или чек-листы.
- Заполняем матрицу тест-кейсами. По результатам всего процесса мы получаем задачи на разработку, тест-кейсы на тестирование и матрицу трассируемости, объединяющую их и требования.
- Поддерживаем матрицу в актуальном состоянии. Изменения должны вноситься при любых модификациях требований. Также следует учитывать интеграционные связи между двумя матрицами, которые описывают разные фичи или модули, и при изменении в одной обязательно проверять, нет ли необходимости правки второй.

Попарное тестирование



Попарное тестирование



Оно же – “**pairwise**”

Это техника формирования наборов тестовых данных.

Сформулировать суть можно, например, таким образом: формирование таких наборов данных, в которых каждое тестируемое значение каждого из проверяемых параметров хотя бы единожды сочетается с каждым тестируемым значением всех остальных проверяемых параметров.

Достаточно проверить комбинации пар входных параметров, потому что ошибки чаще всего находятся именно на перекрестке двух параметров. Исключения бывают, но они достаточно редкие.

Обеспечивает полное тестовое покрытие.

Попарное тестирование

ISTQB определяет попарное тестирование как технику тест-дизайна методом черного ящика, при которой тест-кейсы создаются таким образом, чтобы выполнить все возможные отдельные комбинации каждой пары входных параметров.

Попарное тестирование

Пример:

- email (Поле 1)
- Пол (Поле 2)
- Чек-бокс (Поле 3)

Поле 1	Поле 2	Поле 3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Поле 1	Поле 2	Поле 3
0	0	1
0	1	0
1	0	0
1	1	1

	Адрес	Карта	Заказы	Несколько товаров (одинаковые/разные/разные фильтры одного товара)	Платная доставка
Корзина					
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	
				одинаковые	
				разные	
				разные	
				одинаковые	

Попарное тестирование

Обычно техника Pairwise используется с применением инструментов, так как пар может быть много. Они помогают автоматизировать проверки.

- PICT – Попарное независимое комбинаторное тестирование от Microsoft Corp.
- IBM FoCuS – Единое решение для функционального покрытия от IBM.
- ACTS – Расширенная комбинаторная система тестирования от NIST, агентства правительства США.
- Hexawise
- Jenny
- Pairwise от Inductive AS
- VPTag – бесплатный инструмент попарного тестирования

Диаграмма состояний и переходов



Диаграмма состояний и переходов



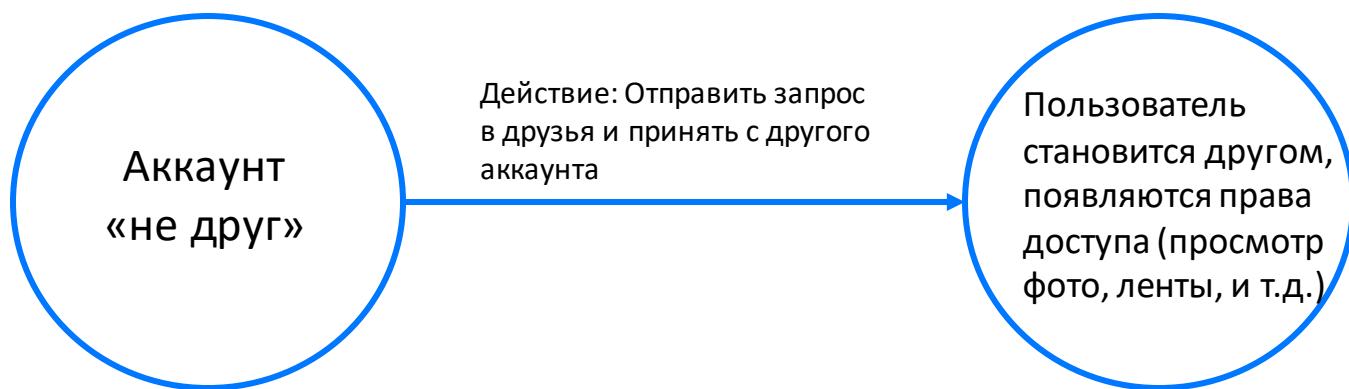
State & Transition Diagram, S&T — это схема переходов и состояния, специальная техника для перехода ТЗ из одного статуса в другой. С ее помощью пользователь в наглядной форме может просматривать переход продукта из одной стадии в другую.

Например, вода. Какие состояния есть у воды?

1. Лед
2. Вода
3. Пар

Итого, 3 состояния. Первый шаг сделан, мы выделили **состояния**. Дальше необходимо определить, какие действия изменяют ее состояние, позволяя одному состоянию переходить в другое. Например, что необходимо сделать, чтобы вода превратилась в пар? **Нагреть** до определенной температуры. А чтобы в лед? **Охладить**. Таким образом, необходимо найти все действия, которые влияют на состояния.

Диаграмма состояний и переходов



Схематически подобную методику отображают в форме кругов и стрелочек, где:

Кружочки — это текущее состояние объекта;

Стрелочки — ситуация, событие или процесс, благодаря которым объект может двигаться из стадии А в стадию В. Это своего рода действие, которое может выполняться как пользователем, так и системой.

Диаграмма состояний и переходов

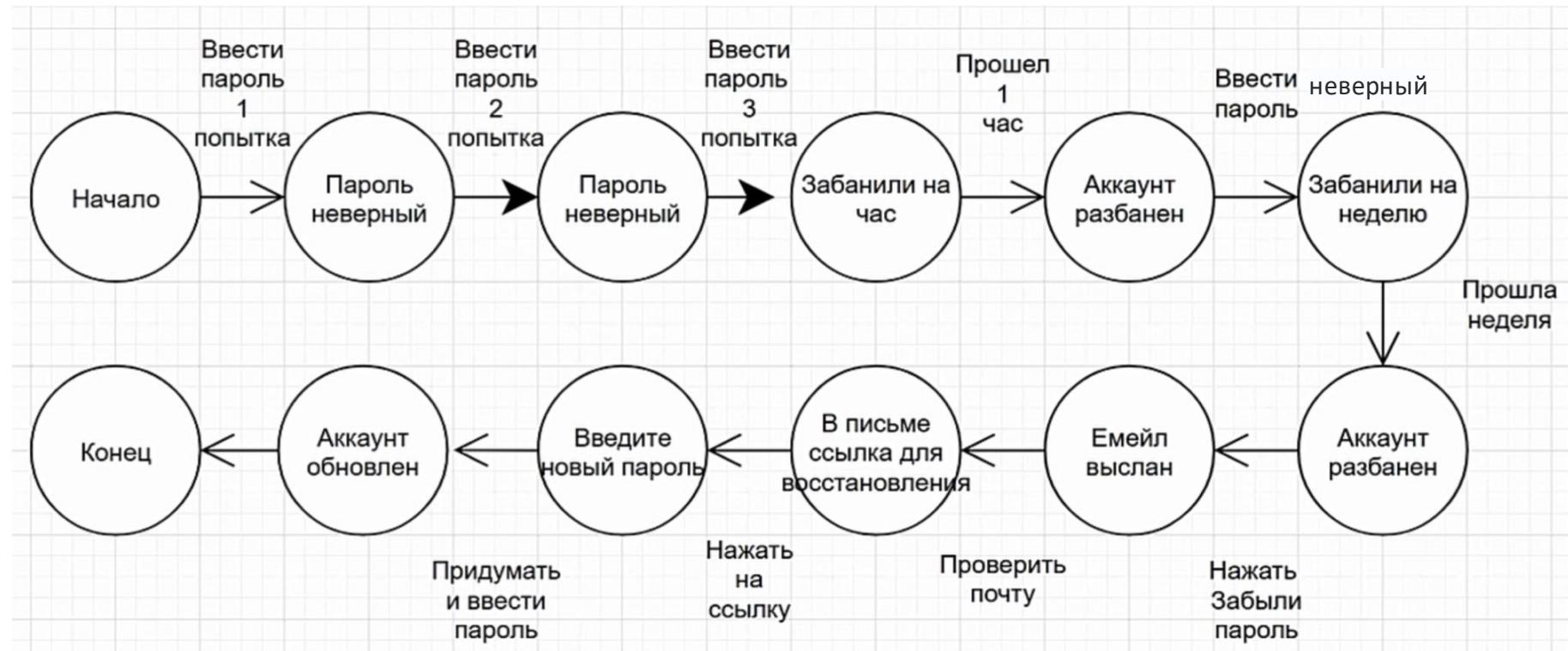


Диаграмма состояний и переходов

Важно: S&T рисуется определенный объект. Объект — это практически всегда строка в базе данных.

Шаг 1. Вы выбираете объект в своём проекте.

Шаг 2. Думаете, какие у него состояния. Они не должны пересекаться, то есть: объект не может быть разом в двух состояниях, и при этом он всегда хоть в каком-то одном есть

Шаг 3. Рисуете эти состояния кружочками.

Шаг 4. Соединяете их стрелочками. Стрелочки - это действия, их надо подписать.

Шаг 5. Смотрите, что получилось и анализируете, есть ли у объекта другие состояния? А другие действия между текущими состояниями? Переход на шаг 2.

Диаграмма вариантов использования



Диаграмма вариантов использования

Оно же - UseCase diagram.

Основные элементы диаграммы - участник (actor) и прецедент (вариант).

Эта методика тест-дизайна направлена на понимание способов использования ваших систем именно участником системы.

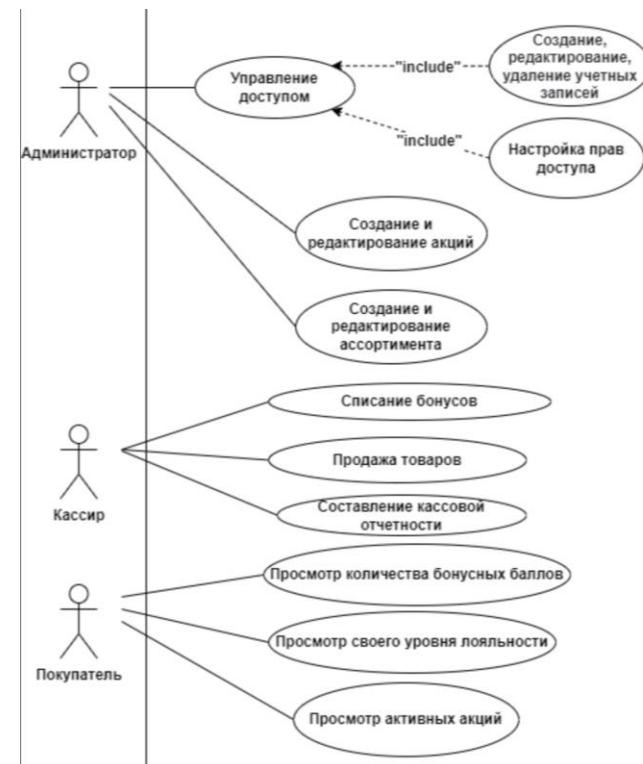


Диаграмма вариантов использования



Участник - это множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек или другая система, подсистема или класс, которые представляют нечто вне сущности. Графически участник изображается "человечком".

Прецедент (use case) - описание множества последовательных событий (включая варианты), выполняемых системой, которые приводят к наблюдаемому участником результату. Прецедент представляет поведение сущности, описывая взаимодействие между участниками и системой. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется. Прецеденты обозначаются очень простым образом - в виде эллипса, внутри которого указано его название.

Диаграмма вариантов использования

Это техника, в ходе которой строится диаграмма, описывающая, какая функциональность программы доступна каждой группе пользователей. Представьте перед собой закрытое сообщество ВКонтакте. В нём мы можем встретить администратора, редактора, подписчика и желающего подписаться пользователя. Каждый из них имеет уникальный набор прав, которого нет у представителей других групп.

Изобразим наших героев на схеме, а рядом с ними добавим доступные им действия. От действий проводим стрелочки к тем полям, к которым для ролей имеется доступ. И не будем забывать про негативные кейсы — проверки, которые выполняются с теми тестовыми данными, которые программа не ожидает. Конечный результат должен выглядеть так:

Диаграмма вариантов использования

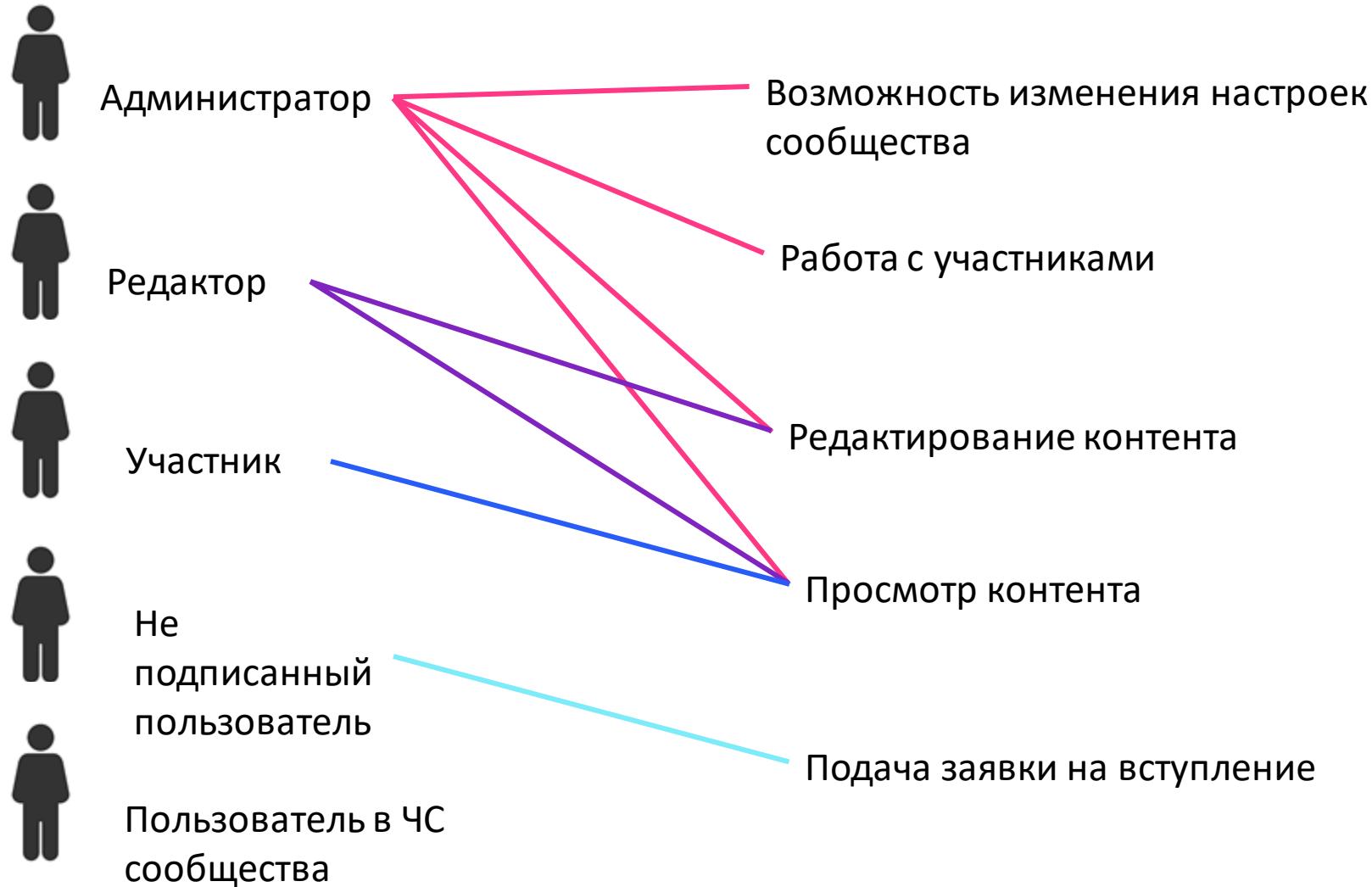


Диаграмма вариантов использования

Диаграммы Use Case применяются не только как методика тест-дизайна, но и при бизнес-анализе для моделирования видов работ, выполняемых организацией, и для моделирования функциональных требований к ПС при ее проектировании и разработке.

Построение модели требований при необходимости дополняется их текстовым описанием. При этом иерархическая организация требований представляется с помощью пакетов use cases.

Спасибо за внимание!

Напоминание отметить
на портале

