ТЕСТОВАЯ ДОКУМЕНТАЦИЯ

Тестовая документация

Создание тестовой документации является вторым этапом жизненного цикла ПО.

Тестовая документация включает в себя:

- тест план;
- тестовая стратегия;
- чек-лист;
- тестовый сценарий;
- тестовый комплект;
- пользовательская история (User Story);
- отчет о дефекте.

Тест план

Хороший тест план должен описывать следующее:

- Что надо тестировать? Описание объекта тестирования: системы, приложения, оборудования.
- Что будете тестировать? Список функций и описание тестируемой системы и её компонент в отдельности.
- Как будете тестировать? Стратегия тестирования, а именно: виды тестирования и их применение по отношению к объекту тестирования.
- Когда будете тестировать? Последовательность проведения работ: подготовка, тестирование, анализ результатов в разрезе запланированных фаз разработки.

Тест план

Преимущества тест плана:

- Возможность приоритезации задач по тестированию.
- Построение стратегии тестирования, согласованной со всей командой.
- Возможность вести учет всех требуемых ресурсов (как технических, так и человеческих).
- Планирование использования ресурсов на тестирование.
- Просчет рисков, возможных при проведении тестирования.

Составляющей частью планирования тестирования (как отдельного документа или же процесса планирования в целом) является стратегия тестирования. Стратегия может быть:

- Частью общего тест-плана.
- Отдельным документом.

Тестовая стратегия

Тестовая стратегия определяет то, как мы тестируем продукт. Это набор мыслей и идей, которые направляют процесс тестирования.

В стратегии тестирования описывают:

- Тестовую среду.
- Анализ рисков проекта.
- Инструменты, которые будут использовать, чтобы провести автоматизированное тестирование и для других целей.
- План действий при непредвиденных обстоятельствах.

Чек-листы

Обязательно должен содержать в себе следующую информацию:

- идея проверок;
- набор входных данных;
- ожидаемые результаты;
- булевая отметка о прохождении/не прохождении тестового случая;
- булевая отметка о совпадении/несовпадении фактического и ожидаемого результата по каждой проверке.

Цель – обеспечить стабильность покрытия требований проверками необходимыми и достаточными для заключения о соответствии им продукта. Особенностью является то, что чек-листы компонуются теми тестовыми случаями, которые показательны для определенного требования.

Чек-листы

Чек-лист должен обладать рядом важных свойств:

- **Логичность.**Чек-лист пишется не «просто так», а на основе целей и для того, чтобы помочь в достижении этих целей.
- **Последовательность и структурированность.** Структурированность достигается за счёт оформления чек-листа в виде многоуровневого списка. Что касается последовательности, то, даже в том случае, когда пункты чек-листа не описывают цепочку действий, человеку всё равно удобнее воспринимать информацию в виде неких небольших групп идей, переход между которыми является понятным и очевидным.
- **Полнота и неизбыточность.** Чек-лист должен представлять собой аккуратную «сухую выжимку» идей, в которых нет дублирования и, в то же время ничто важное не упущено.

Чек-листы

Правила составления чек-листов:

- Одна операция.
- Пункты чек-листа - это минимальные полные операции. Например, заказать изготовление визиток и доставить визитки в офис - это 2 разных операции. Поэтому в чек-листе они отображаются отдельными пунктами: визитки заказаны и визитки доставлены в офис.
- Пункты пишутся в утвердительной форме. Цель чек-листа – проверка готовности задачи, поэтому лучше составлять пункты в утвердительной форме - «заказаны, доставлены». Сравните формулировку: «заказать визитки» и «визитки заказаны».
- Оптимальное количество пунктов - до 20. Чек-листы не должны быть длинными. Если все же это требуется, то лучше разбить задачу на несколько этапов и составить к каждому этапу отдельный чек-лист.

Чек-листы

Преимущества использования чек-листов:

- Структурирование информации у сотрудника. При записи необходимых действий у сотрудника чётко вырисовывается нужная последовательность задач.
- Повышение скорости обучения новых сотрудников. Не нужно повторять несколько раз последовательность операций. Достаточно провести короткий инструктаж и дать чек-лист для самостоятельной работы.
- Высокий результат, уменьшение количества ошибок. Как уже говорилось ранее, чек-листы помогают избежать проколов и ошибок по невнимательности.
- Взаимозаменяемость сотрудников.
- Экономия рабочего времени. Сотрудники будут значительно меньше времени тратить на переделывание задач.

Пример чек-листа

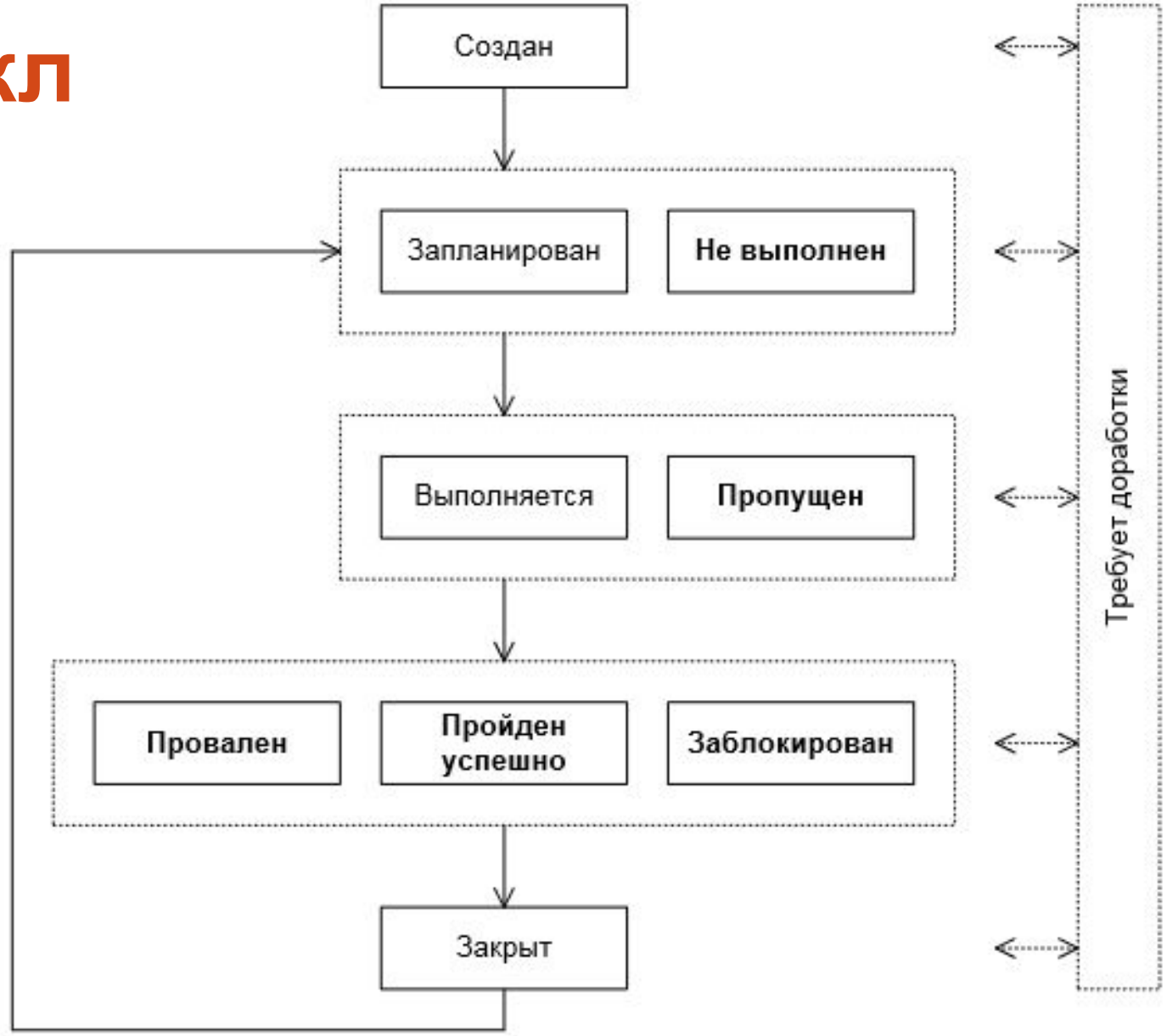
Проверка	Результат	Комментарии
Операции с файлами	Passed	
Создание файла	Passed	
Открытие файла	Passed	
Сохранение документа	Passed	
Печать	Passed	
Редактирование файлов	Failed	http://bt.qatestlab.com/view.php?id=1084
Отмена	Passed	
Копирование	Passed	
Вырезание	Passed	
Вставка	Passed	
Удаление	Passed	
Поиск	Failed	http://bt.qatestlab.com/view.php?id=1085
Поиск с заменой	Failed	http://bt.qatestlab.com/view.php?id=1086
Вставка даты	Passed	
Форматирование	Passed	
Перенос строки	Passed	
Изменение шрифта	Passed	
Справка	Passed	

Тест кейсы

Наличие тест-кейсов позволяет:

- Структурировать и систематизировать подход к тестированию.
- Вычислять метрики тестового покрытия и принимать меры по его увеличению (тест-кейсы здесь являются главным источником информации, без которого существование подобных метрик теряет смысл).
- Отслеживать соответствие текущей ситуации плану.
- Уточнить взаимопонимание между заказчиком, разработчиками и тестировщиками (тест-кейсы зачастую намного более наглядно показывают поведение приложения, чем это отражено в требованиях).
- Хранить информацию для длительного использования и обмена опытом между сотрудниками и командами.
- Проводить регрессионное тестирование и повторное тестирование.
- Повышать качество требований.
- Быстро вводить в курс дела нового сотрудника, недавно подключившегося к проекту.

Жизненный цикл тест кейса



Жизненный цикл тест кейса

Создан - типичное начальное состояние практически любого артефакта. Тест-кейс автоматически переходит в это состояние после создания.

Запланирован - в этом состоянии тест-кейс находится, когда он или явно включён в план ближайшей итерации тестирования, или, как минимум, готов для выполнения.

Не выполнен - нахождение тест-кейса в данном состоянии означает, что он готов к выполнению, но ещё не был выполнен.

Выполняется - если тест-кейс требует длительное время для выполнения, то он может быть переведён в это состояние для подчёркивания того факта, что работа идёт, и скоро можно ожидать её результатов.

Жизненный цикл тест кейса

Пропущен - бывают ситуации, когда выполнение тест-кейса отменяется по соображениям нехватки времени или изменения логики тестирования.

Провален - данное состояние означает, что в процессе выполнения тест-кейса был обнаружен дефект, заключающийся в том, что ожидаемый результат по как минимум одному шагу тест-кейса не совпадает с фактическим результатом.

Пройден успешно - данное состояние означает, что в процессе выполнения тест-кейса не было обнаружено дефектов, связанных с расхождением ожидаемых и фактических результатов его шагов.

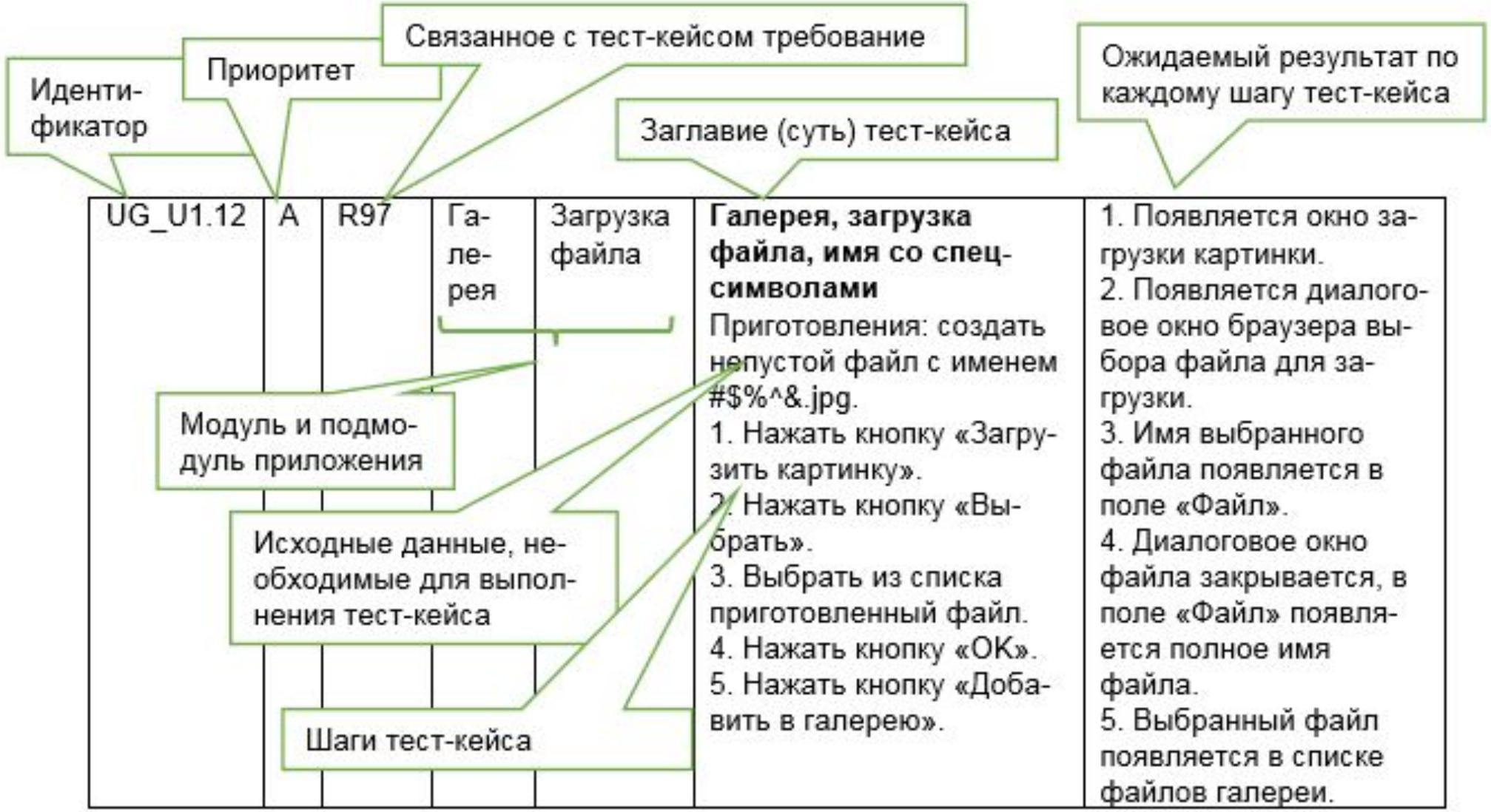
Жизненный цикл тест кейса

Заблокирован - данное состояние означает, что по какой-то причине выполнение тест-кейса невозможно.

Закрыт - в некоторых системах управления тест-кейс переводят в данное состояние, чтобы подчеркнуть тот факт, что на данной итерации тестирования все действия с ним завершены.

Требует доработки - в это состояние (или из него) тест-кейс может быть переведён в любой момент времени, если в нём будет обнаружена ошибка, если изменятся требования, по которым он был написан, или наступит иная ситуация, не позволяющая считать тест-кейс пригодным для выполнения и перевода в иные состояния.

Структура тест кейса



UG_U1.12	A	R97	Галерея	Загрузка файла	**Галерея, загрузка файла, имя со спецсимволами** Приготовления: создать непустой файл с именем #$%^&.jpg. 1. Нажать кнопку «Загрузить картинку». 2. Нажать кнопку «Выбрать». 3. Выбрать из списка приготовленный файл. 4. Нажать кнопку «OK». 5. Нажать кнопку «Добавить в галерею».	1. Появляется окно загрузки картинки. 2. Появляется диалоговое окно браузера выбора файла для загрузки. 3. Имя выбранного файла появляется в поле «Файл». 4. Диалоговое окно файла закрывается, в поле «Файл» появляется полное имя файла. 5. Выбранный файл появляется в списке файлов галереи.

Структура тест кейса

Идентификатор представляет собой уникальное значение, позволяющее однозначно отличить один тест-кейс от другого и используемое во всевозможных ссылках.

Приоритет показывает важность тест-кейса. Он может быть выражен буквами (A, B, C, D, E), цифрами (1, 2, 3, 4, 5), словами («крайне высокий», «высокий», «средний», «низкий», «крайне низкий») или иным удобным способом. Количество градаций также не фиксировано, но, чаще всего, лежит в диапазоне от трёх до пяти.

Связанное с тест-кейсом требование показывает то основное требование, проверке выполнения которого посвящён тест-кейс.

Модуль и подмодуль приложения указывают на части приложения, к которым относится тест-кейс, и позволяют лучше понять его цель.

Структура тест кейса

Заглавие (суть) тест-кейса призвано упростить и ускорить понимание основной идеи (цели) тест-кейса без обращения к его остальным атрибутам.

Исходные данные, необходимые для выполнения тест-кейса, позволяют описать всё то, что должно быть подготовлено до начала выполнения тест-кейса, например:

- состояние базы данных;
- состояние файловой системы и её объектов;
- состояние серверов и сетевой инфраструктуры.

Шаги тест-кейса описывают последовательность действий, которые необходимо реализовать в процессе выполнения тест-кейса.

Рекомендации по написанию тест-кейса

1. Начинайте с понятного и очевидного места, не пишите лишних начальных шагов (запуск приложения, очевидные операции с интерфейсом и т.п.).
2. Даже если в тест-кейсе всего один шаг, нумеруйте его.
3. Если вы пишете на русском языке, то используйте безличную форму (например, «открыть», «ввести», «добавить»).
4. Соотносите степень детализации шагов и их параметров с целью тест-кейса, его сложностью, уровнем и т.д. В зависимости от этих и многих других факторов степень детализации может варьироваться от общих идей до предельно чётко прописанных значений и указаний.
5. Ссылайтесь на предыдущие шаги и их диапазоны для сокращения объёма текста (например, «повторить шаги 3–5 со значением…»).
6. Пишите шаги последовательно, без условных конструкций вида «если… то…».

Структура тест кейса

Ожидаемые результаты по каждому шагу тест-кейса описывают реакцию приложения на действия, описанные в поле «шаги тест-кейса». Номер шага соответствует номеру результата.

По написанию ожидаемых результатов можно порекомендовать следующее:

- Описывайте поведение системы так, чтобы исключить субъективное толкование (например, «приложение работает верно» — плохо, «появляется окно с надписью…» — хорошо).
- Пишите ожидаемый результат по всем шагам без исключения, если у вас есть хоть малейшие сомнения в том, что результат некоего шага будет совершенно тривиальным и очевидным.
- Пишите кратко, но не в ущерб информативности.
- Избегайте условных конструкций вида «если… то…».

Наборы тест-кейсов

Набор тест-кейсов - совокупность тест-кейсов, выбранных с некоторой общей целью или по некоторому общему признаку.

Наборы тест-кейсов можно разделить на **свободные** (порядок выполнения тест-кейсов не важен) и **последовательные** (порядок выполнения тест-кейсов важен).

Преимущества свободных наборов:

- Тест-кейсы можно выполнять в любом удобном порядке, а также создавать «наборы внутри наборов».
- Если какой-то тест-кейс завершился ошибкой, это не повлияет на возможность выполнения других тест-кейсов.

Преимущества последовательных наборов:

- Каждый следующий в наборе тест-кейс, в качестве входного состояния приложения, получает результат работы предыдущего тест-кейса, что позволяет сильно сократить количество шагов в отдельных тест-кейсах.
- Длинные последовательности действий куда лучше имитируют работу реальных пользователей, чем отдельные «точечные» воздействия на приложение.

Пользовательские истории

Пользовательские истории (User Story) — способ описания требований к разрабатываемой системе, сформулированных как одно или более предложений на повседневном или деловом языке пользователя. Пользовательские истории используются гибкими методологиями разработки программного обеспечения для спецификации требований.

User Story — это короткая формулировка намерения, описывающая что-то, что система должна делать для пользователя.

Структура юзер стори

Текст самой юзер стори должен объяснять роль/действия юзера в системе, его потребность и профит, который юзер получит после того как история случится.

К примеру: Как, <роль/персона юзера>, я <что-то хочу получить>, <с такой-то целью>

Пользовательские истории

Правила на написание User Story

- Есть один actor.
- Есть одно действие.
- Есть одна ценность / value / impact.

Actor - Вы выделили персоны, или у вас есть роли, и вы легко их вписываете в начало истории. Убери часть истории про актера. Если история ничего при этом не потеряла - значит эта часть бесполезна (например, пользователь амортизировался в системе).

Действие - это суть истории, "что нужно сделать". Что можно улучшить. Действие должно быть одно - основное. Нет смысла описывать" авторизуется и выполняется поиск" или "указывает параметры поиска и выполняет поиск".

Ценность - главная проблема с User Story. Вы всегда знаете первую часть истории, но всегда сложно указать для чего это делается. Согласно шаблону User Story потому вы пишите "чтобы ...". Уберите эту часть из истории. Если ничего не потеряли - значит формализация ценности в истории была бесполезна.

Практические советы по написанию пользовательских историй

- Лучше написать много историй поменьше, чем несколько громоздких.
- Каждая история в идеале должна быть написана избегая технического жаргона—чтобы клиент мог приоритезировать истории и включать их в итерации.
- Истории должны быть написаны таким образом, чтобы их можно было протестировать
- Тесты должны быть написаны до кода.
- Как можно дольше стоит избегать UI. История должна выполняться без привязки к конкретным элементам.
- Каждая история должна содержать оценку.
- История должна иметь концовку—т.е. приводить к конкретному результату.
- История должна вмещаться в итерацию.