Шпаргалка: тест-дизайн

Введение в тест-дизайн

Тест-дизайн — проектирование тестовой документации.

Если тест-анализ — это ответ на вопрос «Что тестировать?», то тест-дизайн — «Как тестировать?».

Когда создаёшь тестовую документацию, ты проектируешь сценарии тестирования, или **тест-кейсы**. По ним ты будешь проверять приложение.

Правила тест-дизайна

В тест-дизайне есть пять важных правил.

1. **На одно требование может понадобиться несколько** тест-кейсов. Например, в требовании сказано: «Схема Яндекс.Метро масштабируется». Эту функциональность можно проверить по-разному: через кнопки «+» и «-» в интерфейсе, мышку, тачпад, жесты на мобильных устройствах.
2. **Один тест-кейс — одна проверка.** Если ты проверяешь, как масштабируется схема Яндекс.Метро, стоит создать два тест-кейса — на кнопку «+» и кнопку «-». Если объединить их в одном тест-кейсе, и в любой из кнопок окажется баг — будет непонятно, в какой именно.
3. **Проверки не должны дублировать друг друга.** Например, тебе удалось протестировать работу кнопки «+» в тест-кейсе «Масштабирование схемы Яндекс.Метро». Значит, тест-кейс «Отображение кнопки „+“ в интерфейсе» уже не нужен — понятно, что кнопка отображается.
4. **Проверки проектируют в рамках требований.** Если в требованиях не описано, как ведёт себя смартфон при температуре -120 градусов по Цельсию, — не пиши тест-кейс, где телефон погружают в жидкий азот.
5. **Проверки должны покрывать все требования**. Нужно убедиться, что вся функциональность есть в тест-кейсах.

Когда заканчивать тест-дизайн

Ориентируйся на эти правила, чтобы понять, когда заканчивать тест-дизайн. Если тест-кейсы покрывают все требования и не дублируются — значит, пора. Это критерии завершённости — definition of done.

Позитивные и негативные проверки

Во время тест-дизайна проектируют позитивные и негативные тест-кейсы.

Позитивные тест-кейсы проверяют, что приложение работает без ошибок в двух случаях:

- если использовать его по назначению и не пытаться сломать,
- если вводить корректные данные согласно требованиям.

Негативные тест-кейсы проверяют, как поведёт себя приложение, если:

- использовать приложение не так, как задумывали разработчики;
- вводить данные не из требований.

Когда выполняешь негативные проверки, важно убедиться, что приложение продолжает работать без ошибок и реагирует так, как описано в требованиях.

Порядок проверок

Сначала проводят позитивные проверки. Они важнее: так ты протестируешь, реализованы ли требования.

Эквивалентность

Одна из основных техник тест-дизайна — разбиение на классы эквивалентности.

Что такое эквивалентность

Эквивалентность — это равноценность объектов. В тестировании эквивалентными считаются значения, которые приложение обрабатывает одинаково.

Например, в требованиях сказано: «В поле „Фамилия“ можно ввести от 2 до 15 символов». Значит, приложение одинаково обработает значения от 2 до 15. Значения от 0 до 1 или больше 16 оно обработает по-другому — например, выведет сообщение об ошибке.

Такие значения объединяют в **классы эквивалентности** (сокр. — КЭ).

Пример: студенты университета учатся по десятибалльной системе и получают стипендию в зависимости от среднего балла за последнюю сессию. При среднем балле от 8 до 10 размер стипендии — максимальный, 100 долларов.

Здесь два класса эквивалентности:

- от 0 до 7 баллов — студент не получит 100 долларов;
- от 8 до 10 — студент получит 100 долларов.

Для чего используют классы эквивалентности

Чтобы сократить количество проверок. Можно не вводить в поле «Имя» все числа от 2 до 25, а выбрать только одно из этой комбинации. Приложение одинаково отреагирует на любое. Тестировщику остаётся убедиться, что если ввести корректные данные — например, число 10, — то ошибки не будет.

Как разбить требования на комбинации

В поле можно вводить разные символы: например, цифры и буквы. Ограничения полей отражают, какие комбинации символов корректны, а какие вызывают ошибку. Когда ты научишься разбивать такие ограничения на комбинации, сможешь выделять классы эквивалентности и тестировать быстрее.

Классы эквивалентности

Диапазон и набор значений

Класс эквивалентности может быть представлен в виде диапазона и набора значений.

- Диапазон — интервал чисел с границами: например, 2—25.
- Набор значений — множество значений, каждое прописано в требованиях. Например, в выпадающем списке на сайте можно выбрать один язык из трёх.

Один диапазон или один набор значений должен вызывать одинаковую реакцию приложения.

Этапы выделения классов эквивалентности

Чтобы выделить класс эквивалентности, ответь на четыре вопроса:

1. Что перед тобой: диапазон или набор значений?
2. Какие числа или значения считать допустимыми?
3. Какие числа или значения считать недопустимыми?
4. Что будет, если ввести недопустимые числа или значения?

Подбор тестовых значений

В тестировании есть гипотеза: «Если приложение обработало верно одно значение класса, оно обработает верно и остальные».

Чтобы подобрать тестовое значение, возьми одно значение внутри класса.

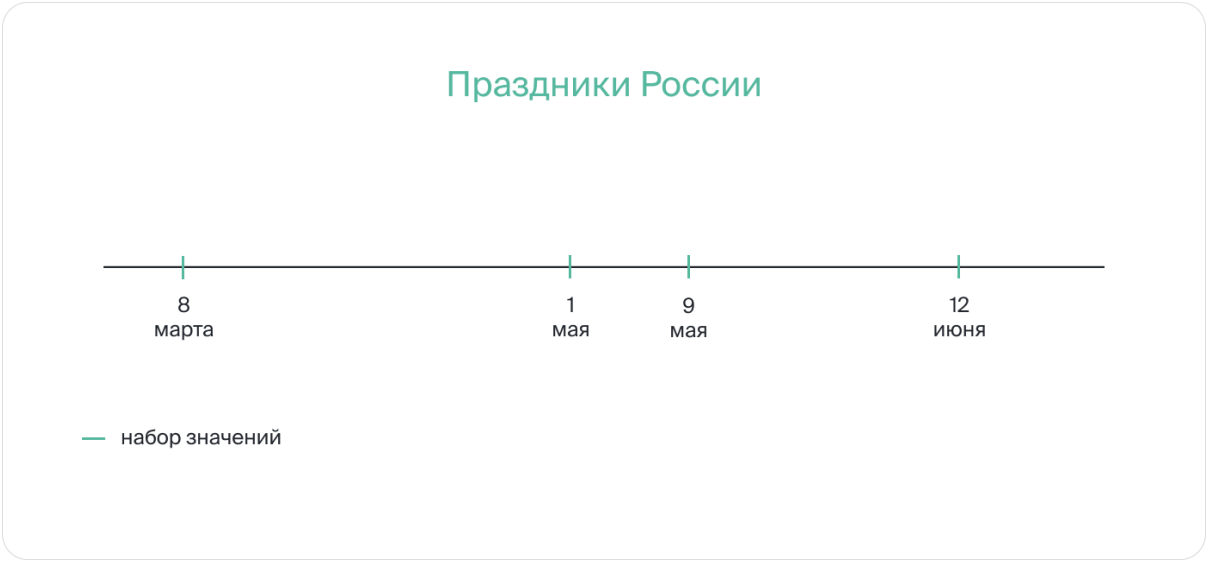
Граничные значения

Граничные значения (сокр. — ГЗ) — первая и последняя точки класса эквивалентности. Их важно проверять, потому что на границах часто возникают ошибки.

Обрати внимание: граничные значения есть только у диапазона. У наборов их не бывает.

Например, в требованиях написано: пароль может содержать от 8 до 20 символов. Значит, 8 и 20 — границы класса.

Если класс — это набор, прописывают все значения, которые в него входят. Например, список праздников.



Как определять границы

Определить границы можно только у классов, которые представляют собой диапазон, а не набор значений.

В диапазонах допустимо несколько типов данных, например:

- целые числа (1, 2, 3...);
- дробные числа (0.1, 0.2, 0.3...);
- временные интервалы (часы, дни, месяцы).

Границы в требованиях могут прописывать по-разному.

Явно на интервале чисел — есть точные значения начала и конца числового интервала. Например, пароль может содержать от 8 до 20 символов.

Явно на множестве чисел — в требованиях написано, что операцию можно выполнить при любом целом положительном числе. Например, пополнить счёт мобильного телефона можно не меньше чем на 1 рубль. У диапазона от 1 до бесконечности есть только минимальная граница — 1: у бесконечности нет границы.

Неявно на интервале чисел — в требованиях написаны примерные характеристики интервала, которые нужно уточнить дополнительно. Например, кредит выдают совершеннолетним и не дают пенсионерам.

Неявно на множестве чисел — в требованиях написаны примерные характеристики интервала, у которого нет границ или одной из границ. Например, по закону можно получить загранпаспорт в любом возрасте. Минимальная граница — 0 лет, а максимальной нет — нельзя принять за границу определённое число лет.

Проектирование тестов

Например, кредит выдают только совершеннолетним и не выдают пенсионерам. Законодательные ограничения: совершеннолетие — с 18 лет, пенсионный возраст — с 65 лет.
Чтобы узнать о кредитных возможностях, нужно ввести свой возраст в поле на странице банка.

Чтобы проверить, как работает приложение на границах:

1. Выдели диапазон значений: от 18 до 64 лет включительно. Это диапазон, когда человек совершеннолетний и не достиг пенсионного возраста.
2. Проверь, что приложение работает согласно требованиям на границах диапазона, — значения 18 и 64.
3. Проверь, что вне границ диапазона приложение работает по другой логике, — значения 17 и 65.
4. Проверь, что приложение работает согласно требованиям внутри границ диапазона, — значения 19 и 63.

Значит, в тестирование поля нужно включить шесть проверок: 18 и 64; 17 и 65; 19 и 63.

Общий алгоритм проектирования тестов по граничным значениям

Например, нужно проверить, что минутная стрелка в часах работает согласно требованиям в диапазоне от 15 до 45 минут. Тип данных — временной интервал. {{Шаг}} — 1 минута.

1. Определи тип данных и шаг: временной интервал и 1 минута.
2. Проверь, что приложение работает на границах диапазона: 15 и 45.

3. Сделай шаг за границы диапазона и проверь эти значения: 14 и 46.
4. Сделай шаг внутрь границ диапазона и проверь эти значения: 16 и 44.

Порядок применения техник КЭ и ГЗ

Как проверить пополнение счёта, оптимально применяя техники КЭ и ГЗ?

1. **Проверь значения внутри классов:** 50.00, 1000.00 и 5000.00. Если проверки прошли успешно, переходи к следующему шагу. Если нет — проверять дальше нет смысла: приложение не работает.
2. **Проверь значения на границах:** 0.00 и 99.99; 100.00 и 2999.99; 3000.00. Если проверки прошли успешно, для этих значений приложение работает корректно. В последнем классе нет максимальной границы, поэтому её не проверяют.
3. **Проверь значения на один шаг вне границ:** 100.00; 99.99 и 3000.00; 2999.99. Если проверки прошли успешно, тебе удалось подтвердить, что границы верны. В первом классе нельзя задать одно значение вне границ: отрицательную сумму нельзя положить на счёт.
4. **Проверь значения на один шаг внутрь границ:** 0.01 и 99.98; 100.01 и 2999.98; 3000.01. Если проверки прошли успешно, приложение работает согласно требованиям и внутри всего диапазона: на значения из одного класса система реагирует одинаково.

Постарайся запомнить порядок применения техник КЭ и ГЗ:

1. Сначала выделяют классы эквивалентности.
2. Потом у классов с типом диапазон отмечают граничные значения.
3. Затем формулируют тестовые значения для всех классов — и диапазонов, и наборов значений.

Оптимизация проверок

Основной принцип оптимизации — сократить дубликаты перед проверкой.

Сложные случаи

Бывают случаи, когда в документации одно требование пересекается с другим.

Банк разрешает переводить деньги с карты на карту с двумя условиями:

- За один раз можно перевести не меньше 50 и не больше 15 000 рублей.
- Сумма перевода не может превышать сумму, которая есть на счёте.

Два требования пересекаются. Проверить работу сервиса с учётом двух ограничений можно, если разделить интервал на классы в несколько этапов.

По первому условию нужно разделить интервал на три класса с диапазоном значений:

N КЛАССА	ДИАПАЗОН, РУБ.	ПОВЕДЕНИЕ СИСТЕМЫ
Класс А	0 - 49	Перевод невозможен
Класс Б	50 - 15000	Перевод возможен
Класс В	15001 - ∞	Перевод невозможен

Классы спроектированы с допущением, что сумма на счёте не ограничена.

Чтобы соблюсти и второе условие, нужно взять число из диапазона [50; 15 000]. Например, на карте осталось 7000 рублей. Класс Б разделится ещё на два с диапазонами значений Б1 и Б2.

N- КЛАССА	ДИАПАЗОН, РУБ.	ПОВЕДЕНИЕ СИСТЕМЫ
Класс Б1	50 - 7000	Перевод возможен
Класс Б2	7001 - 15000	Перевод невозможен

Чтобы проверить, как работает система с учётом сразу двух условий, и оптимизировать количество проверок, нужно соединить обе группы классов.

Обрати внимание на столбец «Сумма на счёте» — он содержит условия, которые тоже нужно учитывать при проверке диапазонов.

ДИАПАЗОН, РУБ.	СУММА НА СЧЁТЕ	ПОВЕДЕНИЕ СИСТЕМЫ
0 - 49	не ограничена	Перевод невозможен
50 - 7000	7000	Перевод возможен
7001 - 15000	7000	Перевод невозможен
15001 - ∞	не ограничена	Перевод невозможен

Тебе удалось выполнить одно из главных условий тест-дизайна: покрыть требования минимальным количеством проверок.