

**Методы проектирования тестов**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Категория** | **Группа** | **Методы/Техники** | **Описание** | **Виды тестирования** |
| Методы, основанные на поведении | Чёрный ящик | * Эквивалентное разбиение, * Анализ граничных значений, * Тестирование с помощью таблиц решений, * Тестирование с помощью сценариев использования, * Тестирование с помощью схем переходов. | Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя требования, спецификации, сценарии использования и пользовательские истории.  Тестовые сценарии могут использоваться для определения несоответствий и отклонений между требованиями и реализацией.  Измерение покрытия основано на элементах базиса тестирования и методе проектирования, применяемом к базису тестирования. | Применимы как для функционального, так и для нефункционального тестирования. |
| Методы, основанные на структуре | Белый ящик | * Тестирование и покрытие операторов, * Тестирование и покрытие условий, * Тестирование и покрытие альтернатив, * Тестирование и покрытие множественных условий. | Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя код, архитектуру, детальную архитектуру или любой другой источник информации о структуре программного обеспечения.  Измерение покрытия основано на элементах структуры (коде, интерфейсах и так далее).  Спецификации используются как источник дополнительной информации для определения ожидаемых результатов тестовых сценариев. |  |
| Методы, основанные на опыте |  | * Исследовательское тестирование, * Предположение об ошибках. | Тестовые условия, тестовые сценарии и тестовые данные получаются из базиса тестирования, который может включать в себя знания и опыт тестировщиков, разработчиков, пользователей и других заинтересованных лиц. |  |

**Тестовые данные** — это не система и не элементы системы. Как правило, это артефакты, которые использует пользователь при выполнении шагов: логины, пароли, изображения, файлы и т. п.

**Верификация** (verification) — подтверждение экспертизой и предоставлением объективных доказательств того, что заданные требования выполнены.

**Валидация** (validation) — подтверждение экспертизой и предоставлением объективных доказательств того, что заданные требования для конкретного предполагаемого использования или применения были выполнены.

**Тест дизайн** (test design) — это процесс преобразования общих целей тестирования в осязаемые условия тестирования и тест-кейсы.

**Тест анализ** (test analysis) — это процесс изучения того, что может быть использовано для получения тестовой информации. Эта основа для тестов называется тестовый базис (тестовая база).

**Тестовый базис** (test basis) — тестовый базис включает всё, на чём основаны тесты. Это может быть системное требование, техническое задание, сам код (для структурного тестирование) или бизнес-процесс. Иногда тесты могут быть основаны на опытном знании пользователем системы, которое может быть не задокументировано.

**Объект тестирования** (test object или test item) — компонент или система, которую мы будем тестировать. Простыми словами — это то, что будем тестировать.

**Тест-кейс** — это набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определённой цели или тестового условия, таких как выполнения определённого пути программы или же для проверки соответствия определённому требованию.

Другими словами, тест-кейс — это документ, который описывает условия, шаги и ожидаемый результат, благодаря которым тестировщик проверяет, работает ли продукт в соответствии со спецификацией.

Существует разделение тест-кейсов на позитивные и негативные:

**Позитивный тест-кейс** проверяет, что при использовании корректных данных, приложение правильно выполнило функцию в соответствии со спецификацией.  
Пример: при вводе правильного формата электронной почты student@skillfactory.ru при регистрации поле «электронная почта» подсвечено зелёным цветом.

**Негативный тест-кейс** проверяет, что при использовании как корректных, так и некорректных данных, вызываемая функция не выполняется и срабатывает валидация.  
Пример: при вводе неправильного формата электронной почты student@skillfactory при регистрации на портале всплывает сообщение: «Пожалуйста, введите электронную почту в формате [student@skillfactory.ru»](mailto:student@skillfactory.ru).

**Тест-кейс** (Test Case) — это набор входных значений, предусловий выполнения, ожидаемых результатов и постусловий выполнения, разработанный для определённой цели или тестового условия, таких как выполнение определённого пути программы или же для проверки соответствия определенному требованию. Другие названия: тестовый случай, тестовая ситуация, вариант тестирования.

**Позитивный тест-кейс** проверяет, что при использовании корректных данных приложение правильно выполнило функцию в соответствии со спецификацией.

**Негативный тест-кейс** проверяет, что при использовании как корректных, так и некорректных данных вызываемая функция не выполняется, и срабатывает валидация.

**Система управления тестовой документацией** (Test Management Software) — это специальная программа для создания и выполнения тест-кейсов, создания тест-планов и планирования тестовой деятельности, а также создания и хранения отчётов о тестировании.

**Тестовый набор/комплект** (Test Suite) — несколько (набор) тест-кейсов, объединённых между собой по типу тестирования, по приоритету, по тестируемой функциональности или модулю.

**Тест сценарий** — в ISTQB также можно найти термины тест сценарий (test scenario) и спецификация тестовой процедуры (test procedure specification), которые обозначают документ с последовательностью действий для выполнения тестов, также известные как тестовый сценарий (test script) или ручной тестовый сценарий (manual test script). Некоторые из этих терминов выглядят одинаково при переводе на русский язык, что может создать путаницу.

Чтобы правильно написать тест-кейс, нам важно определить:

* 1. предусловия выполнения,
  2. шаги проверки,
  3. ожидаемые результаты,
  4. постусловия выполнения.

Именно эти 4 пункта и формируют структурный состав тест-кейса.

**Тест-план** или **план тестирования** — это тестовый артефакт, рекомендуемый стандартом тестовой документации [ГОСТ Р 56922-2016/ISO/IEC/IEEE 29119-3:2013](http://docs.cntd.ru/document/1200134998).  
  
Согласно этому стандарту, **план тестирования** — это документ описывающий весь объём работ по тестированию, начиная с описания объекта, стратегии, расписания, критериев начала и окончания, конфигурации, знаний и навыков тестировщиков, а также оценки рисков и вариантов их разрешения.

## ****Что может быть задокументировано в плане тестирования?****

* + Определение объёма, целей и рисков тестирования,
  + Определение общего подхода к тестированию,
  + Интеграция и координация действий по тестированию в действия жизненного цикла программного обеспечения,
  + Принятие решений, что тестировать, о людях и других ресурсах, необходимых для выполнения различных тестовых действий, и как будут выполняться тестовые действия,
  + Планирование деятельности по анализу, проектированию, внедрению, выполнению и оценке тестов либо на определённые даты (например, в последовательной разработке), либо в контексте каждой итерации (например, в моделях разработки типа Agile/SCRUM),
  + Выбор метрик для мониторинга и контроля тестирования,
  + Бюджет тестовой деятельности,
  + Определение уровня детализации и структуры тестовой документации (например, путём предоставления шаблонов и примеров документов),
  + Критерии входа (Entry Criteria) и выхода (Exit Criteria) (желательно иметь критерии, определяющие, когда тестирование должно начинаться и когда оно завершается).

Типичные **критерии входа** включают в себя:

* + Наличие тестируемых требований, пользовательских историй (user stories) и/или моделей (например, при следовании стратегии тестирования на основе моделей),
  + Доступность элементов тестирования, соответствующих критериям выхода для любых предыдущих уровней тестирования,
  + Доступность тестовой среды,
  + Наличие необходимых инструментов тестирования,
  + Наличие тестовых данных и других необходимых ресурсов.

Типичные **критерии выхода** включают:

* + Запланированные тесты выполнены,
  + Достигнут определённый уровень покрытия (например, требований, пользовательских историй, критериев приемлемости, рисков, кода),
  + Количество неустранённых дефектов находится в пределах согласованного лимита,
  + Количество предполагаемых оставшихся дефектов достаточно мало,
  + Оценённые уровни надёжности, эффективности работы, удобства использования, безопасности и других соответствующих характеристик качества являются достаточными.

**Отчёт по тестированию**— документ, содержащий в себе информацию о выполненных тестах, результатах их выполнения, открытых дефектах и выводах о качестве разрабатываемого приложения. Отчёт по тестированию тесно связан с таким понятием, как фаза тестирования.

**Фазы тестирования** или **жизненный цикл тестирования приложения** (*STLC*) — это определённый набор задач, объединённых в контролируемую фазу проекта. В тестировании можно выделить следующие фазы, которые коррелируются с жизненным циклом продукта:

* + фаза планирования,
  + фаза анализа,
  + фаза проектирования тестов,
  + фаза реализации и выполнения тестов,
  + фаза анализа результатов тестирования (оценка критериев выхода) и отчётов,
  + фаза завершения тестирования.

Отчёт по тестированию содержит следующую информацию:

* + спецификацию документа,
  + сводка выполненного тестирования,
  + отклонения от Плана Тестирования,
  + оценка завершения тестирования,
  + препятствующие факторы,
  + показатели тестирования,
  + остаточные риски,
  + практические результаты тестирования,
  + активы тестирования, допускающие повторное использование,
  + накопленный опыт.

## ****Элементы, которые могут быть внутри веб-страницы****

|  |  |  |  |
| --- | --- | --- | --- |
| **Название UI-элемента** | **Вариант внешнего вида** | **Как работает** | **Дополнительно** |
| **Checkbox**  **Чек-бокс** | img | Позволяет выбрать один или несколько пунктов. | Если чек-бокс недоступен, он должен быть серого или иного неяркого цвета. |
| **Radio buttons**  **Переключатель** | img | Позволяет выбрать только один из вариантов. | Если переключатель недоступен, он должен быть серого или иного неяркого цвета. |
| **Dropdown lists**  **Выпадающий список** | img | При нажатии раскрывает список значений, из которых можно выбрать одно или несколько. | Если можно выбрать сразу несколько значений — это лучше указать. |
| **List**  **Список** | img | Список значений, из которых можно выбрать одно или несколько. |  |
| **Buttons**  **Кнопки** | imgimg  img | Вызывают какое-то действие, чаще всего — отправку данных на сервер. | По иконке, надписи на кнопке должно быть ясно, для чего она.  Если кнопка неактивна, это должно быть явно видно пользователю (серый цвет, невозможно кликнуть). |
| **Toggle / Switches**  **Тумблер/ выключатель** | img | Переключатель может выглядеть иначе, чем Radio buttons, и иметь только два состояния. |  |
| **Text field**  **Поле для ввода текста** | imgimg | Позволяет вводить текст пользователю.  Поля ввода логина, пароля, строка поиска, «оставьте ваше сообщение» и прочие. | Если определенный ввод недопустим, под полем должна появляться подсказка для пользователя или суть ошибки. |
| **Date and time pickers**  **Поле для ввода даты (времени)** | img | Помогает вводить пользователю дату и время. | Если заблокирован ввод даты с клавиатуры, надо проверить, насколько удобен ввод дат в далёком прошлом (дней рождения). |
| **Breadcrumb**  **«**[**Хлебные крошки**](https://ru.wikipedia.org/wiki/Навигационная_цепочка)**»** | img  img | Элемент навигации (навигационная цепочка), помогает пользователю понять, где в структуре сайта он находится, и вернуться на вышестоящую в иерархии страницу. | Обычно клик по элементу переводит на соответствующую ему страницу. |
| **Pagination**  **Пагинация, нумерация страниц** | img | Разбивает содержимое длинной страницы или множество результатов поиска на «странички», которые можно просматривать по очереди. | Все «странички» должны загружаться по щелчку на номере. |
| **Slider**  **Слайдер, ползунок, бегунок** | img | Позволяет менять значение элемента в определённых диапазонах и с заданным шагом. |  |
| **Icon**  **Иконка** | img | Графическое обозначение действия или элемента. |  |
| **Image Carousel**  **Карусель изображений** | img | Часто встречающийся вид для галереи, позволяющий просмотреть по очереди изображения, похоже на бесконечную (или конечную) ленту слайдов. |  |
| **Notifications**  **Уведомления (счётчик уведомлений)** | img | Отображает число уведомлений (напоминаний) для какого-то элемента или приложения. | Число уведомлений соответствует количеству непрочитанных уведомлений. |
| **Progress Bar**  **Панель прогресса** | img | Позволяет визуально отобразить состояние растянутых во времени, длительных процессов (скачивание большого файла и тому подобное). |  |
| **Tool Tip**  **Тултип, подсказка, всплывающая подсказка** | img | Отображает в появляющемся окошке какую-то полезную для пользователя информацию. |  |

Бывает, что веб-приложение ещё разрабатывается и тестировщик его в глаза не видел. Даже тогда тестировщик занимается тест-дизайном — планированием и проектированием тестов для проверки реализованной функциональности в соответствии со спецификацией.

В зависимости от того, что и как предстоит тестировать, можно выделить три больших группы проверок на:

* 1. **Функциональность** — как элементы интерфейса выполняют свои функции.
  2. **Соответствие стандартам** — общий стиль, принятые обозначения.
  3. **Usability** или **практичность** — удобство использования.

Как правило, стараются покрыть тестами все три группы.

К примеру, нам предстоит протестировать веб-приложение распространённого типа — интернет-магазин.

Вначале, применяя технику декомпозиции, мы выделяем логические элементы, поведение которых объединено общими признаками. В таблице представлено условное визуальное макетирование условного сайта. На сайте действующего интернет-магазина разделы могут размещаться иначе.

|  |  |  |
| --- | --- | --- |
| **Меню разделов** — открывает разделы с разной продукцией, сгруппированной по определённому признаку. | | |
| **Контакты**  Номера телефонов, иные способы связи для заказа/решения вопросов от покупателя. | **Поиск по сайту** — помогает найти нужный товар. | |
| **Список фильтров**  Помогает отобрать продукт по определённым критериям, отбросив то, что сейчас неинтересно пользователю. | **Реклама**  Баннеры с промоакциями и распродажами привлекают внимание к товарам, которые сейчас продвигает магазин. | **Корзина**  Содержит счётчик добавленных товаров, опционально — сумму покупок. При клике ожидается подробная информация о добавленных товарах, их стоимости, способах оплаты и доставки. |
| **Служебная и правовая информация**  Информация о магазине, доставке, гарантиях, условиях возврата и прочем. | Фильтры для сортировки товаров по цене, названию, рейтингу.  **Карточки товаров** — должны содержать как минимум фото товара, название и цену. | |

**Функциональность**: все пункты меню должны быть кликабельными. Пункты меню не должны вести на несуществующие страницы.

**Стандарты**: пункты меню должны быть одинаково выровнены. Каждый пункт должен умещаться в меню, не обрываясь на каком-то фрагменте. Иконки для пунктов меню должны быть схожими с теми, что обычно используют на родственных сайтах.

**Юзабилити**: пункты меню легко читаемы, понятны для пользователя. Если есть выпадающее меню, они не «убегают», их легко нажать. Каждый пункт меню ведёт к товарам. Если есть пустые разделы, незаполненные товаром, их лучше скрыть, чтобы не раздражать пользователя тем, чего нет.

POSTMAN

скрипты

* **“Как использовать переменные из *csv* файла в скриптах во вкладке *Tests?*”:**

function ShowIndentNames () {

let jsonData = pm.response.json();

let all\_e = jsonData.pets.length;

**for** (let i = **0**; i < all\_e; i++) {

**if** (jsonData.pets[i].name == pm.variables.get('name')) {

console.log(jsonData.pets[i].id)

}

}

};

ShowIndentNames ();

*Вам необходимо использовать конструкцию pm.variables.get('name') Она смотрит шире — вне коллекции и в первую очередь в прикреплённый файл.*

* **Как обратиться к последнему элементу массива:**

pm.test("Последний эл-т массива", *function* () {

*let* jsonData = pm.response.json();

              pm.collectionVariables.set("petid", jsonData.pets[jsonData.pets.length-1].id)

});

Посчитали весь массив, команда lenth, и обратились к последнему -1

* ***Выводит в консоль id всех карточек с именем «Бобер»***

*function* ShowAll (){

*let* jsonData = pm.response.json();

*let* all = jsonData.pets.length;

    for (*let* i = 0; i<all; i++){

        if (jsonData.pets[i].name == "Бобер"){

*console*.log(jsonData.pets[i].id);

        }

    }

}

ShowAll ();