*Тестирование веб-приложений обычно включает в себя несколько шагов, чтобы обеспечить его функциональность и корректную работу, прежде, чем его увидят пользователи.*

[Systems Sciences Institute](https://www.isixsigma.com/industries/software-it/defect-prevention-reducing-costs-and-enhancing-quality/) при IBM выявили, что стоимость исправления ошибки, выявленной после релиза, была в 4-5 раз выше, чем стоимость выявленной и исправленной ошибки во время разработки. Следовательно, стоит тщательно тестировать веб-приложение до его выпуска.

Ниже приведен обзор 6 шагов, которые помогут в этом.

**Шаг 1. Функциональное тестирование (Functional Testing)**

Первым шагом тестирования веб-приложения является проверка функциональности системы.

Функциональное тестирование (по Википедии)  — это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает.

Обычно функциональное тестирование включает в себя:

* выявление функциональности, которую должно выполнять приложение,
* ввод и вывод данных,
* выполнение тест-кейсов,
* анализ фактических результатов.

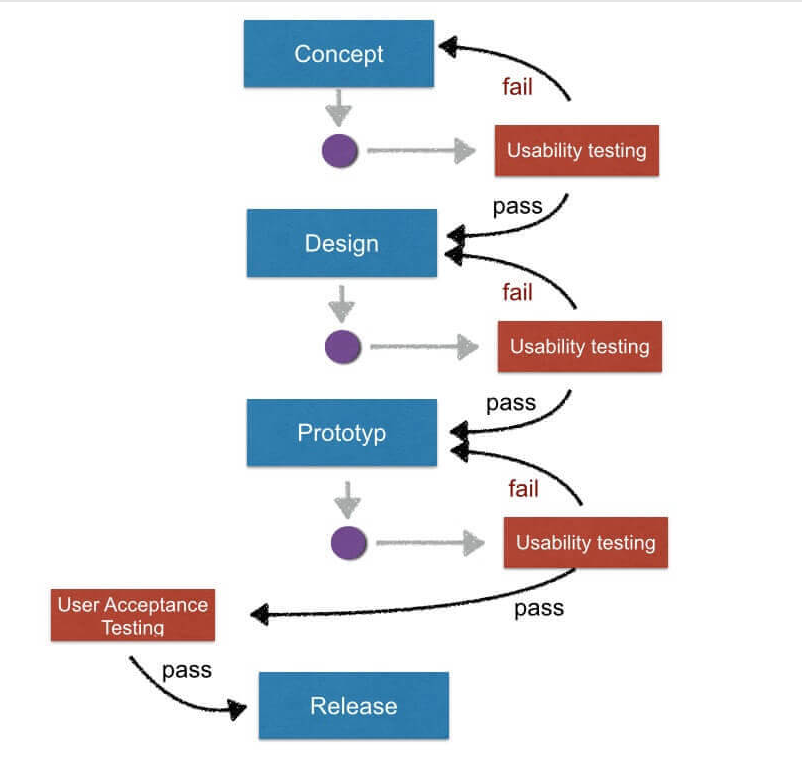
На протяжении функционального тестирования происходит имитация фактического использования системы. Цель этого типа тестирования — подойти как можно ближе к реальному использованию приложения и создать условия, которые близки к требованиям пользователя.

**Шаг 2. Юзабилити тестирование (Usability Testing)**

Юзабилити выходит за рамки функционального тестирования и сочетает в себе тестирование функциональности, а также общий пользовательский опыт. Юзабилити тестирование не следует путать с пользовательским приемочным тестированием (User Acceptance Testing). Несмотря на то, что они оба важны для успешной работы веб-приложения, каждое из них имеет разную цель и проводится на разных этапах жизненного цикла ПО.

Юзабилити тестирование включает в себя следующие шаги:

* разработку тестовой стратегии, которая регулирует проверку всех функций веб-приложения,
* подбор участников тестирования как внутри компании, так и извне,
* запуск теста под наблюдением экспертов,
* анализ результатов и улучшение веб-приложения.



**Шаг 3. Тестирование интерфейса (Interface Testing)**

Тестирование интерфейса позволяет убедиться, что все взаимодействия между веб-сервером и интерфейсом пользователя воспроизводятся в соответствии с требованиями. Также с помощью него проверяют корректность отображения сообщений об ошибке.

**Шаг 4. Тестирование совместимости (Compatibility Testing)**

Ключевой шаг в тестировании веб-приложений — это обеспечение совместимости приложения со всеми веб-браузерами и устройствами.

Совместимость с веб-браузерами.

Обеспечивает гарантию того, что приложение корректно функционирует на разных веб-браузерах. Оно позволяет убедиться, что JavaScript, AJAX, WebSockets, браузерные уведомления и запросы на аутентификацию работают так, как было указано в требованиях.

Кроме тестирования приложения на различных браузерах (и даже на Internet Explorer! :)), следует убедиться, что и другие версии одного и того же браузера обеспечивают корректную работу приложения.

Совместимость с мобильными устройствами.

Существенной частью тестирования веб-приложений также является проверка работы приложения и на разных мобильных устройствах, которые работают на Android, iOS и других мобильных операционных системах.

**Шаг 5. Тестирование производительности.**

После проверки работы веб-приложения на различных браузерах и устройствах, следует проверить его на производительность во время высокой нагрузки. С помощью данного вида тестирования, можно проследить, как работает приложение при различной скорости интернета, а также как приложение себя поведёт при нормальных и пиковых нагрузках (load testing).

Чтобы определить предел работы приложения, оно подвергается высокому (стрессовому) уровню нагрузки до тех пор, пока не нарушится его функциональность (stress testing). Также это помогает проверить, как приложение восстанавливается после сбоев.

**Шаг 6. Тестирование безопасности.**

Завершающий шаг в тестировании веб-приложений. Он помогает убедиться в том, что приложение защищено от несанкционированного доступа и вредоносных действий с помощью вирусов или другого вредоносного ПО.

Тестирование безопасности включает в себя следующие шаги:

* проверка возможности доступа защищенных страниц без авторизации,
* проверка того, что все сессии заканчивают свою работу после прекращения пользовательской активности (закрытия приложения),
* проверка SSL-протокола приложения,
* проверка того, что файлы с ограниченным доступом невозможно загрузить без предварительной авторизации.

Большую пользу приносит чек-лист тестирования безопасности, так как он поможет организовать и структурировать тестовую работу. Чек-лист должен включать в себя задачи из таких областей:

* Безопасная передача данных;
* Аутентификация;
* Управление сеансом;
* Авторизация;
* Криптография;
* Проверка данных;
* Отказ в обслуживании (Denial of Service);
* Специфические функциональные тесты;
* Обработка ошибок.

#### Подходы к функциональному тестированию Веб-приложений

Функциональное тестирование (functional testing) – процесс верификации соответствия функционирования продукта его начальным спецификациям. Характерным примером может быть проверка того, что программа подсчета выплат по банковской ссуде выдает корректные выкладки на любые введенные сумму ссуды и срок ее возврата. Обычно подобные проверки проводятся вручную, иногда к этому подключаются конечные пользователи в качестве бета-тестеров. Однако программные системы становятся все сложнее, а комбинации различных входных параметров и поддерживаемых операционных систем нередко исчисляются десятками и сотнями.

Перечислим некоторые из методов функционального тестирования веб-приложений :

1. *Record & Play* – основан на возможности средств автоматизации тестирования автоматически генерировать код.
2. *Functional Decomposition* – в основе лежит разбиение всех компонент фреймворка по функциональному признаку на бизнес-функции (реализуют/проверяют бизнес-функциональность приложения), user-defined функции (вспомогательные функции, которые еще имеют привязку к тестируемому приложению или к конкретному проекту), утилиты (функции общего назначения, не привязанные к конкретному приложению, технологии, проекту).
3. *Data-driven* – основан на том, что к некоторому тесту или группе тестов привязывается источник данных, и этот тест или набор тестов циклически выполняется для каждой записи из этого источника данных. Вполне может применяться в комбинации с другими подходами.
4. *Keyword-driven* – представляет собой фактически движок для обработки посылаемых ему команд, а сами инструкции выносятся во внешний источник данных.
5. *Object-driven* – основан на том, что основные ходовые части фреймворка реализованы в виде объектов, что позволяет собирать тесты по кирпичикам.
6. *Model-based* – основан на том, что тестируемое приложение (или его части) описывается в виде некоторой поведенческой модели.

Функциональное тестирование направлено на то, чтобы каждая функция веб-сайта работала в соответствии с требованиями спецификации. Тестирование функциональности веб-сайта показывает **«Что делает система».**

Попробуем создать чек-лист для тестирования функциональности веб-сайта.

**Тестирование ссылок**

Необходимо проверить:

* Исходящие ссылки
* Корректность внутренних ссылок
* Отсутствие ссылок, ведущих к одной странице
* Ссылки, которые используются для отправки электронной почты админам сайта
* Есть ли страницы, на которые не указаны ссылки
* Отсутствие неработающих ссылок

**Тестирование форм для всех страниц**

Вы используете формы для интерактивного общения с вашими клиентами. Итак, необходимо проверить следующие моменты:

* Действительность входных данных
* Допустимые значения для поля данных
* Недопустимые входные значения для поля данных
* Параметры форм, в которых возможно удаление или любая другая модификация данных.

**Тестирование cookies**

Cookies представляют собой небольшие файлы, которые хранятся на компьютере пользователя после посещения веб-страницы.

* Проверьте сайт с отключенными cookies
* Проверьте сайт с включенными cookies
* Убедитесь, что файлы cookies зашифрованы перед записью на компьютер пользователя
* Проверьте аспекты безопасности при удалении файлов cookies.
* Если cookies имеют продолжительность действия, то следует проверить, активны ли они в указанный период времени.

**HTML / CSS валидация**

* Синтаксические ошибки HTML
* Убедитесь, что сайт доступен для поисковых машин.
* Убедитесь, что ваша веб-страница имеет точную карту сайта в формате XML и HTML

#### Общая постановка задачи для функционального тестирования веб приложения

#### Проверка всех ссылок

* Проверьте ссылки, исходящие от всех страниц к конкретному домену.
* Внутренние ссылки.
* Ссылки на другие элементы, расположенные внутри страниц.
* Ссылки для отправления электронной почты администратору или другим пользователям веб-страниц.
* Проверьте, нет ли ссылок на изолированные страницы.

#### Проверка форм

Формы используются для получения информации от пользователей и взаимодействия с ними.

Что нужно проверить в формах:

* Правильность работы валидации в каждом поле формы.
* Значения полей, используемые по умолчанию.
* Опции для создания форм, удаления, просмотра и редактирования форм (если такие имеются ).

Рассмотрим пример проекта поисковой системы. В проекте есть этапы регистрации рекламодателей и партнеров. Каждый шаг регистрации отличается от других, но зависит от остальных этапов. Поэтому весь процесс регистрации должен проходить правильно.

Есть различные виды валидации, например, проверка электронной почты, финансовой информации пользователя и т.д. Все поля с валидацией нужно протестировать в ручном или автоматическом режиме.

#### Тестирование файлов cookie

Cookie - это небольшие файлы, хранящиеся на компьютере пользователя. Чаще всего они используются для поддержки сеансов с авторизацией. Проверьте приложение, выключая и включая cookies в опциях браузера.

Проверьте, шифруются ли Cookie перед записью на компьютере. Протестируйте сеансы регистрации и статистику пользователя, когда сеанс посещения сайта закончится. Проверьте, влияет ли на безопасность приложения удаление файлов cookie .

#### Проверьте HTML/CSS

Если вы оптимизируете сайт для поисковых систем, то валидация HTML/CSS особенно важна. Первым делом проверьте сайт на наличие синтаксических ошибок в HTML-коде. Проверьте, доступен ли сайт для различных поисковых систем.

#### Тестирование базы данных

Взаимодействие веб-приложения с базой данных является очень важным моментом. Проверьте целостность данных и проведите тестирование сайта на наличие ошибок при редактировании, удалении, изменении форм или других действиях, имеющих отношение к базе данных. Проверьте, все ли запросы к базе данных выполняются правильно, данные

Полезные инструменты для проведения функционального тестирования: [Selenium](http://www.seleniumhq.org/), [Linux Test Project](https://linux-test-project.github.io/), [JUnit,](http://junit.org/junit4/) [Sprinter by Hewlett Packard Entreprise](https://saas.hpe.com/ru-ru/software/sprinter-manual-software-testing) (ручное тестирование), [Browserstack](http://www.browserstack.com/) (ручное и автоматизированное тестирование), [Usersnap](https://usersnap.com/) (ручное тестирование).

Самым распространенным является подход, называемый Capture & Playback (другие названия – Record & Playback, Capture & Replay) . Суть этого подхода заключается в том, что сценарии тестирования создаются на основе работы пользователя с тестируемым приложением. Инструмент перехватывает и записывает действия пользователя, результат каждого действия также запоминается и служит эталоном для последующих проверок. При этом в большинстве инструментов, реализующих этот подход, воздействия (например, нажатие кнопки мыши) связываются не с координатами текущего положения мыши, а с объектами HTML-интерфейса (кнопки, поля ввода и т.д.), на которые происходит воздействие, и их атрибутами. При тестировании инструмент автоматически воспроизводит ранее записанные действия и сравнивает их результаты с эталонными, точность сравнения может настраиваться. Можно также добавлять дополнительные проверки – задавать условия на свойства объектов (цвет, расположение, размер и т.д.) или на функциональность приложения (содержимое сообщения и т.д.).

Основное достоинство этого подхода – простота освоения. Создавать тесты с помощью инструментов, реализующих данный подход, могут даже пользователи, не имеющие навыков программирования.

Вместе с тем, у подхода имеется ряд существенных недостатков. Для разработки тестов не предоставляется никакой автоматизации; фактически, инструмент записывает процесс ручного тестирования. Если в процессе записи теста обнаружена ошибка, то в большинстве случаев создать тест для последующего использования невозможно, пока ошибка не будет исправлена (инструмент должен запомнить правильный результат для проверки). При изменении тестируемого приложения набор тестов трудно поддерживать в актуальном состоянии, так как тесты для изменившихся частей приложения приходится записывать заново.