**Тестирование** — заключительный этап разработки программного обеспечения.

Он позволяет ответить на три важных вопроса:

* соответствует ли продукт изначальной задумке,
* все ли работает как надо
* и что нужно сделать, чтобы ответ на первые два вопроса был положительным.

Functional testing types include:

• Unit testing.

• Integration testing.

• System testing.

• Sanity testing.

• Smoke testing.

• Regression testing.

• Beta/Acceptance testing.

• Security testing.

Non-functional testing types include:

• Performance Testing (тестирование производительности).

• Load testing (нагрузочное тестирование).

• Stress testing (soak, spike).

• Volume testing (объемное тестирование).

• Recoverability testing (восстанавливаемость).

• Usability testing (удобство использования).

• Reliability (надежность).

• Portability (переносимость).

**Тест-кейс**

Атрибуты тест-кейса:

* Название — основная тема тест-кейса. Краткое описание его сути.
* Шаги — описание последовательности действий, которые должны привести нас к ожидаемому результату. Каждый шаг отвечает на вопрос «что сделать?» (например, «зайти на страницу „Новости"», «кликнуть на кнопку „Узнать больше"»).
* Ожидаемый результат — то, что должно произойти после выполнения всех шагов, если функционал работает правильно.
* Фактический результат — то, что происходит, если функционал работает некорректно (ошибка, баг).
* **Чек-лист**
* Документ, который описывает, что должно быть протестировано. Он может быть абсолютно разного уровня детализации — все зависит от требований к отчетности, уровня знания продукта сотрудниками и сложности разработки.
* **Баг-лист (баг-репорт)**
* Документ, описывающий ситуацию или последовательность действий, которые привели к некорректной работе объекта тестирования. В нем указываются причины и ожидаемый результат.
* **Фреймворк тестирования**
* Программная платформа или комплекс компонентов и моделей, которые упрощают реализацию продукта. С помощью фреймворков, позволяющих эмулировать поведение реальных пользователей из программной среды, строится автоматизация тестирования.

Фреймворки обычно состоят из двух частей:

* Программы или надстройки над браузером, которые позволяют управлять браузером.
* API, предоставляющего удобные функции для контролирования этой программы.

Возможности фреймворков, полезные для тестирования:

* Открытие новой вкладки или окна браузера, изменение размеров.
* Навигация между веб-страницами.
* Поиск на странице веб-элемента с указанными параметрами.
* Функции ожидания определенных событий, в частности, ожидания полной загрузки страницы или появления на ней определенного элемента.
* Эмуляция действий пользователя, например, нажатие кнопки мыши на определенный элемент или ввод последовательности символов с клавиатуры.
* Исполнение на странице команды JavaScript для более сложных действий.

Для автоматизации тестирования подходят разные фреймворки. Самые популярные — [Selenium](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://www.seleniumhq.org/projects/%26amp;sa%3DD%26amp;ust%3D1604587317117000%26amp;usg%3DAOvVaw3bHD5oDHmCN-afVMST9evA&sa=D&ust=1604587317155000&usg=AOvVaw0eRKVl3gPaRqufRqElygrL), [Watij](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttp://watij.com/%26amp;sa%3DD%26amp;ust%3D1604587317117000%26amp;usg%3DAOvVaw2jUc_0ZMAPy_OCIhmCUMJX&sa=D&ust=1604587317155000&usg=AOvVaw1DeK97mvChqJp7TrDAyi8e), [HtmlUnit](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttp://htmlunit.sourceforge.net/%26amp;sa%3DD%26amp;ust%3D1604587317117000%26amp;usg%3DAOvVaw1gDz1aOz3em1EoKOkvhh8_&sa=D&ust=1604587317155000&usg=AOvVaw3qzWfRagJ7loa5is-LgYQg), [Jamaleon](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttp://jameleon.sourceforge.net/%26amp;sa%3DD%26amp;ust%3D1604587317118000%26amp;usg%3DAOvVaw2PhgOyHpm6SEfY15YsdYUs&sa=D&ust=1604587317155000&usg=AOvVaw0OXR5cEOQcvemaUBvR7O9z), [Jest](https://www.google.com/url?q=https://www.google.com/url?q%3Dhttps://jestjs.io/%26amp;sa%3DD%26amp;ust%3D1604587317118000%26amp;usg%3DAOvVaw1Nv2UEVcfLa6rtedr7nT7K&sa=D&ust=1604587317155000&usg=AOvVaw2LNrOxfM9tdxw9VXSyXmjh).

**Уровни тестирования**

**Компонентное** (unit testing) — тестирование, при котором проверяются отдельные модули / компоненты (исходного кода программы) программного обеспечения. Обычно выполняется разработчиком. White box techniques. Хотя в практическом мире QA- специалисты также проводят модульное тестирование.

**Интеграционное** (integration testing) —

ISTQB — тестирование, выполняемое для обнаружения дефектов в интерфейсах и во взаимодействии между интегрированными компонентами или системами. тестирование, при котором программные модули интегрируются логически и проверяются как группа.

**Bottom-up testing** — последовательный подход к интеграционному тестированию, при котором компоненты нижнего уровня тестируются первыми и потом используется для облегчения тестирования компонентов более высокого уровня. It takes help of Drivers for testing.

**Top-down testing** — подход к интеграционному тестированию, при котором компоненты из верхнего уровня иерархии объектов тестируются в первую очередь, с использованием заглушек вместо компонентов более низкого уровня.

Протестированные компоненты используются для тестирования компонентов более низкого уровня и данный процесс повторяется до тех пор, пока не будет протестированы компоненты самого низшего уровня.

Sandwich — Стратегия сэндвич / гибрид представляет собой комбинацию подходов сверху вниз и снизу вверх. Здесь верхние модули тестируются с нижними модулями, а нижние модули интегрируются с верхними модулями и тестируются. Эта стратегия использует заглушки, а также драйверы.

**Big bang testing** — вид подхода к интеграционному тестированию, при котором элементы программного или аппаратного обеспечения, или и то и другое, собираются в компонент или в целую систему сразу, а не по этапам.

**Системное тестирование** (System) — Процесс тестирования системы в целом с целью проверки того, что она соответствует установленным требованиям.

Тестирование полностью интегрированных приложений, включая внешние периферийные устройства, чтобы проверить, как компоненты взаимодействуют друг с

knowledge of test frameworks: JMeter, Selenium, Mockito, TestNG, Junit, Protractor

knowledge of CI tools (Jenkins)

SQL

* **SELECT** - extracts data from a database
* **UPDATE** - updates data in a database
* **DELETE** - deletes data from a database
* **INSERT INTO** - inserts new data into a database
* **CREATE DATABASE** - creates a new database
* **ALTER DATABASE** - modifies a database
* **CREATE TABLE** - creates a new table
* **ALTER TABLE** - modifies a table
* **DROP TABLE** - deletes a table
* **CREATE INDEX** - creates an index (search key)
* **DROP INDEX** - deletes an index

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server. (Точка с запятой - это стандартный способ разделения каждого оператора SQL в системах баз данных, которые позволяют выполнять более одного оператора SQL в одном вызове сервера.)

Distinct

Count

As

In

Like

Between

Order by

Desc -  descending

Values

* % - The percent sign represents zero, one, or multiple characters
* \_ - The underscore represents a single character