Тестирование

Что такое тестирование?

Тестирование — это процесс проверки программного обеспечения на соответствие требованиям, ожиданиям и корректности работы.
Основная цель — найти ошибки до того, как продукт попадет к пользователю.

Зачем нужно тестирование?

- Выявление багов до релиза.
- Улучшение качества продукта.
- Повышение удовлетворенности пользователей.
- Экономия затрат на доработки в будущем.

Из чего состоит тестирование

1. **Анализ требований:**
   - Чтение технической документации, пользовательских историй, задач.
   - Определение, что именно и как нужно тестировать.

2. **Планирование тестирования:**
   - Определение стратегии: что покрываем, что не покрываем, как тестируем.
   - Назначение ресурсов и сроков.

3. **Проектирование тестов:**
   - Создание тест-кейсов (описание шагов, данных, ожидаемых результатов).
   - Настройка тестовых данных и окружения.

4. **Выполнение тестов:**
   - Ручное или автоматизированное выполнение тест-кейсов.
   - Фиксация результатов.
   -
5. **Оценка результатов и отчетность:**
   - Создание баг-репортов.
   - Анализ покрытия и качества.

6. **Завершение тестирования:**
   - Подведение итогов

Из чего состоит тестирование

6. **Завершение тестирования:**
   - Подведение итогов.
   - Архивация тестов.
   - Обратная связь команде.

Виды тестирования

По уровню:

1. **Модульное (Unit Testing)**
   - Проверка отдельных функций/методов.
   - Обычно пишется разработчиками.

2. **Интеграционное**
   - Проверка взаимодействия между модулями.

3. **Системное**
   - Проверка всей системы в целом.

4. **Приемочное (Acceptance Testing)**
   - Проверка готовности продукта заказчиком или QA-инженерами.

По способу выполнения:

1. **Ручное тестирование**
   - Выполняется вручную, шаг за шагом.

2. **Автоматизированное тестирование**
   - Использование скриптов и фреймворков (например, Selenium, JUnit, Cypress).

Типы тестирования

- **Функциональное** — проверка, как работает функционал по требованиям.
- **Нефункциональное** — проверка производительности, безопасности, удобства.
- **Регрессионное** — убеждение, что старый функционал не сломался после изменений.
- **Тестирование "черного ящика"** — без знания внутренней логики.
- **Тестирование "белого ящика"** — с учетом кода и архитектуры.
- **Тестирование API** — проверка взаимодействия через интерфейсы.
- **Тестирование UI** — как выглядит и работает пользовательский интерфейс.
- **Тестирование нагрузки** — как система справляется с большим объемом пользователей.

Функциональное тестирование

Цель: Проверка, что система делает то, что должна.
Как проводится: На основе требований или юзер-стори.
Пример: Проверка, что кнопка "Отправить" действительно отправляет форму.

Подтипы:

- Smoke testing (дымовое)
- Sanity testing (здравое)
- Regression testing (регрессионное)

Нефункциональное тестирование

Цель: Оценка *как* система работает, а не *что* она делает.
Пример: Сколько пользователей выдерживает сайт, сколько секунд открывается страница.

Сюда входят:

- Тестирование производительности (Performance testing)
- Нагрузочное (Load testing)
- Стресс-тестирование (Stress testing)
- Usability (удобство использования)
- Accessibility (доступность для людей с ОВЗ)

Регрессионное тестирование

Цель: Проверка, что после изменений (фиксы, новые фичи) **старый функционал всё ещё работает**.

Когда применяется:

- После каждого коммита в мастер
- После правок багов
- Перед релизом

Методы: часто автоматизируется (автотесты).

Безопасностное тестирование (Security Testing)

Цель: Проверка устойчивости системы к атакам, утечкам данных и несанкционированному доступу.

Пример:

- SQL-инъекции
- Попытки входа без авторизации
- Манипуляции с ID в URL

Используется в финтехе, госуслугах, здравоохранении.

Локализационное тестирование

Цель: Проверить правильность перевода, форматов, валют, локалей и т.д.

Пример:

- € вместо ₽
- Дата в американском формате вместо европейского
- Частично непереведённые элементы интерфейса

Кроссбраузерное и кроссплатформенное

Цель: Убедиться, что приложение одинаково работает в разных браузерах, ОС и устройствах.

Пример:

- В Safari не отображается выпадающее меню
- На Android кнопка уезжает за пределы экрана

Тестирование юзабилити (Usability Testing)

Цель: Понять, насколько удобно и понятно продуктом пользоваться.

Проводится с реальными людьми — им дают задания и наблюдают за действиями.

Оценивается:

- Ясность интерфейса
- Логика навигации
- Минимум кликов до цели

Исследовательское тестирование (Exploratory Testing)

Цель: Найти нестандартные ошибки без заранее написанных тест-кейсов.
Как: Тестировщик изучает систему "на ходу", как пользователь.

Полезно на старте проекта, при ограниченном времени или если документации мало.

Вспомогательные типы

Smoke Testing — первичная проверка "жизнеспособности" билда

Sanity Testing — поверхностная проверка только что исправленного

A/B Testing — сравнение двух версий для анализа поведения пользователей

Артефакты тестирования

Артефакты тестирования — это документы, файлы или отчеты, которые создаются на разных этапах тестирования. Они фиксируют знания, процессы, результаты и служат коммуникационными инструментами внутри команды.

Основные артефакты:
Тест-план (Test Plan)
- Описание стратегии тестирования, объемов, подходов, рисков и сроков.
- Может быть в виде отдельного документа или в виде задачи в трекере.

Тест-кейсы (Test Cases)
- Подробное описание шагов, которые нужно выполнить, чтобы проверить функционал.
- Включает: шаги, входные данные, ожидаемый результат, статус выполнения.

Чек-листы (Test Checklists)
- Упрощённая форма тест-кейсов — список, по которому проверяется логика работы без детального описания каждого шага.
- Удобны для быстрого регресса.

Артефакты тестирования

Баг-репорты (Bug Reports / Defect Reports)
- Описание ошибок, найденных в системе.
- Включают: шаги воспроизведения, окружение, скриншоты/видео, приоритет/серьезность.

Отчеты о тестировании (Test Reports)
- Сводка результатов тестирования: что протестировано, что прошло/провалилось, сколько багов найдено.
- Часто предоставляются после окончания спринта или релиза.

Матрица трассировки (Traceability Matrix)
- Таблица, связывающая требования и тест-кейсы, чтобы убедиться, что все требования покрыты тестами.

Тестовые данные (Test Data)
- Наборы данных, которые используются для проверки (например, учетные записи, заказы, документы).

Артефакты тестирования

Автотесты и скрипты

- Код, выполняющий автоматическую проверку функционала.
- Являются частью CI/CD пайплайнов или запускаются вручную.

Логи тестирования

- Логи из систем, консоли, браузера или тестовых фреймворков, особенно важны при автоматизации.

Виды багов и их влияние

Баг (ошибка, дефект) — это любое несоответствие между фактическим поведением системы и ожидаемым.
Даже мелкий баг может негативно сказаться на опыте пользователя или привести к сбоям, особенно если он дойдёт до продакшена.

Виды багов

Функциональные баги
- Поведение системы не соответствует требованиям (кнопка «Сохранить» не сохраняет данные)

UI/UX баги
- Проблемы в интерфейсе или логике взаимодействия (текст вылезает за границы кнопки, слишком мелкий шрифт на мобильных)

Логические баги
- Нарушение бизнес-логики или неверная реализация алгоритмов (скидка применяется не к тому товару)

Кроссбраузерные и адаптивные баги
- Отображение и поведение компонентов зависит от браузера или устройства (сайт работает в Chrome, но ломается в Safari)

Баги производительности
- Система медленно откликается или «виснет» под нагрузкой (страница грузится 10+ секунд при 1000 товарах)

Виды багов

Безопасностные баги

- Уязвимости, которые могут привести к утечке данных, взлому, доступу без авторизации (можно войти в чужой аккаунт, просто изменив ID в ссылке)

Баги локализации

- Ошибки перевода, отображения валют, дат, форматов и др. (дата отображается в американском формате, хотя стоит русская локаль)

Регрессионные баги

- Ошибки, появившиеся после внесения новых изменений (работавшая ранее кнопка «Оплатить» перестала работать после обновления)

Почему критично находить баги на этапе тестирования?

- **Исправить баг до релиза — дешевле, чем после.**
- **Раннее выявление = меньше стресса для команды.**
- **Баги, попавшие в прод, подрывают доверие к продукту.**
- **QA — это последняя линия обороны.**

Роли в процессе тестирования

- QA Engineer (инженер по качеству) — тестирует, пишет тест-кейсы, ищет баги.
- Automation QA — пишет автотесты.
- Manual QA — ручное тестирование.
- Test Lead — организует процесс тестирования в команде.
- DevOps / Developer — могут участвовать в CI/CD, писать юнит-тесты.

Инструменты, которые часто используются

- **Для ручного тестирования:** TestRail, Jira, Xray.
- **Для автоматизации:** Selenium, Cypress, Playwright, JUnit, Postman.
- **CI/CD:** Jenkins, GitLab CI, GitHub Actions.