

Manual QA: руководство для Junior QA (Web / E-commerce)

1. Теория качества: ISO/IEC 25010 (кратко и по делу)

ISO/IEC 25010 — международный стандарт модели качества ПО. Для Junior QA важно понимать не формулировки, а что именно проверять руками.

1.1 Основные характеристики качества

1. Функциональная пригодность (Functional suitability)

Работает ли функциональность так, как описано в требованиях.

2. Производительность (Performance efficiency)

Скорость загрузки страниц, время отклика.

3. Совместимость (Compatibility)

Работа в разных браузерах, отсутствие конфликтов.

4. Удобство использования (Usability)

Понятный интерфейс, читаемые тексты, логичные действия.

5. Надёжность (Reliability)

Система не падает, корректно обрабатывает ошибки.

6. Безопасность (Security)

Валидация данных, защита форм, доступы.

7. Сопровождаемость (Maintainability)

Косвенно: читаемые ошибки, предсказуемое поведение.

8. Переносимость (Portability)

Работа на разных устройствах и ОС.



Junior QA не обязан проверять всё глубоко, но должен думать в этих категориях.

2. Что такое ручное тестирование

Ручное тестирование — это проверка веб-приложения без автотестов, глазами и логикой пользователя.

Цель: - найти ошибки до пользователя - проверить сценарии использования - убедиться, что система ведёт себя ожидаемо

3. Сценарии тестирования

3.1 Позитивные сценарии

Проверяют корректную работу при **валидных данных**.

Пример: - добавить товар в корзину - оформить заказ с правильными данными

3.2 Негативные сценарии

Проверяют поведение системы при **ошибочных данных**.

Пример: - пустые обязательные поля - неверный email - недоступный товар

Ожидание: - понятная ошибка - отсутствие падений - данные не сохраняются

4. Пример тестирования e-commerce (практика)

4.1 Позитивный сценарий: оформление заказа

Название: Checkout — успешное оформление заказа

Предусловия: Товар есть в наличии

Шаги: 1. Открыть каталог 2. Добавить товар в корзину 3. Перейти в checkout 4. Заполнить все обязательные поля корректно 5. Подтвердить заказ

Ожидаемый результат: - заказ успешно создан - пользователь видит страницу подтверждения

4.2 Негативный сценарий: обязательное поле

Название: Checkout — пустое обязательное поле

Шаги: 1. Перейти в checkout 2. Оставить поле Email пустым 3. Нажать «Оформить заказ»

Ожидаемый результат: - поле подсвеченено - показано сообщение об ошибке - заказ не создан

5. Тест-план (Test Plan)

Тест-план — это краткое описание *что, как и в каком объёме мы тестируем*. Для Junior QA тест-план может быть простым, но он помогает структурировать работу и ожидания.

5.1 Назначение тест-плана

Тест-план отвечает на вопросы: - Что именно тестируем? - Что не входит в тестирование? - В каком окружении проводится тестирование? - Какие типы тестов используются?

5.2 Пример простого тест-плана (Junior уровень)

Проект: E-commerce store

Цель тестирования:

Проверить базовую работоспособность пользовательского оформления заказа.

Область тестирования:

- Каталог товаров
- Корзина
- Checkout

Не входит в тестирование:

- Платёжные системы (mock / sandbox)
- Административная панель

Типы тестирования:

- Ручное функциональное тестирование
- Позитивные и негативные сценарии
- UI / Usability проверки

Окружение:

- Browser: Chrome (latest)
- OS: Windows / macOS
- Environment: Staging

Критерии завершения:

- Все критические сценарии выполнены
- Блокирующих багов нет

6. Протокол тестирования (Test Report)

После тестирования Junior QA обычно оформляет **отчёт / протокол**.

5.1 Пример протокола

Проект: E-commerce store

Дата: 2026-01-XX

Тестировал: Junior QA

Проверенные модули:

- Каталог
- Корзина
- Checkout

Результат:

- Позитивные сценарии: OK
- Негативные сценарии: найдено 2 бага

Ссылки на баги:

- BUG-101
- BUG-102

6. Как правильно оформлять баг

Summary: Коротко, что не работает

Environment: Браузер, ОС

Steps to reproduce: Пошагово

Actual Result: Что происходит

Expected Result: Что должно быть

7. Бесплатные и условно бесплатные инструменты QA

7.1 Браузеры и DevTools

- Google Chrome
- Firefox
- DevTools: Console, Network, Elements, Mobile view

7.2 Документация и тест-кейсы

- Google Sheets
- Notion (Free)
- Markdown (Git / Confluence)

7.3 Баг-трекинг

- Jira (Free для маленьких команд)
 - GitHub / GitLab Issues
-

7.4 API и backend (базово)

- Postman (Free)
 - Insomnia
-

7.5 Responsive и визуальная проверка

- Chrome DevTools (Device Mode)
 - PerfectPixel (pixel-perfect)
-

7.6 Запись экрана / браузера (для багов)

- **OBS Studio** — бесплатно, видео экрана
- **Loom (Free)** — быстрая запись браузера
- **Встроенная запись экрана** (Windows / macOS)

Видео багов сильно упрощает жизнь разработчикам.

8. Советы Junior QA

- Тестируй как пользователь
 - Проверяй крайние случаи
 - Не бойся задавать вопросы
 - Лучше простой баг-репорт, чем умный, но непонятный
-

9. Роли в обеспечении качества: QA, QC и тестировщик

Этот раздел помогает Junior QA понять **кто за что отвечает** и чем отличаются роли, которые часто путают.

9.1 QA (Quality Assurance)

QA — обеспечение качества процесса разработки.

QA отвечает не только за поиск ошибок, а за то, чтобы ошибки появлялись как можно реже.

Назначение QA: - выстроить процессы качества - определить стандарты тестирования - предотвратить дефекты на ранних этапах

Основные задачи QA: - участие в анализе требований - разработка тест-стратегии и тест-планов - определение критериев качества - улучшение процессов разработки и тестирования

Ответственность QA: - качество процесса - соответствие стандартам (например, ISO) - прозрачность и управляемость тестирования

QA думает как сделать продукт качественным ещё до тестирования.

9.2 QC (Quality Control)

QC — контроль качества готового продукта.

QC фокусируется на проверке результата, а не процесса.

Назначение QC: - обнаружить дефекты в продукте - подтвердить соответствие требованиям

Основные задачи QC: - выполнение тест-кейсов - проверка функциональности - фиксация дефектов

Ответственность QC: - найденные дефекты - корректность проверок - отчётность по результатам тестирования

QC отвечает на вопрос: «Соответствует ли продукт требованиям?»

9.3 Тестировщик (Manual Tester)

Тестировщик — это практическая роль, чаще всего относящаяся к QC.

Назначение тестировщика: - проверить работу приложения глазами пользователя

Основные задачи тестировщика: - тестирование по сценариям (позитивным и негативным) - поиск и описание багов - повторная проверка после исправлений (re-test) - базовая проверка UI и логики

Ответственность тестировщика: - корректные баг-репорты - воспроизводимость ошибок - соблюдение тест-плана

В реальных проектах Junior QA чаще всего начинает именно с роли тестировщика.

9.4 Краткое сравнение ролей

| Роль | Фокус | Отвечает за |
|-------------|----------|---------------------|
| QA | Процессы | Качество разработки |
| QC | Продукт | Найденные дефекты |
| Тестировщик | Проверка | Баги и отчёты |

9.5 Важно для Junior QA

- В вакансиях **QA** и **тестировщик** часто используют как синонимы
- На практике Junior выполняет задачи **QC / тестировщика**
- Понимание различий — плюс на собеседовании

10. Основные принципы тестирования (ISTQB)

Ниже приведены **7 классических принципов тестирования**, которые важно знать каждому Junior QA. Эти принципы часто спрашивают на собеседованиях и используют в реальной работе.

10.1 Тестирование показывает наличие дефектов

Тестирование может показать, что **дефекты существуют**, но не может доказать их отсутствие.

Заблуждение: «Если багов не нашли — значит их нет»

10.2 Исчерпывающее тестирование невозможно

Невозможно протестировать **все комбинации** данных, состояний и сценариев.

Поэтому QA выбирает наиболее важные и рискованные сценарии

10.3 Раннее тестирование экономит ресурсы

Чем раньше найден дефект, тем **дешевле и быстрее** его исправить.

Ошибка в требованиях стоит дешевле, чем баг на продакшене

10.4 Скопление дефектов (Defect Clustering)

Большая часть дефектов обычно сосредоточена в **ограниченном числе модулей**.

Часто 80% багов находятся в 20% системы

10.5 Парадокс пестицида

Если одни и те же тесты выполнять постоянно, они перестают находить новые дефекты.

Тест-кейсы нужно регулярно пересматривать и обновлять

10.6 Тестирование зависит от контекста

Подходы и методы тестирования зависят от типа продукта, рисков и целей проекта.

Нет универсального способа тестировать всё одинаково

10.7 Заблуждение об отсутствии ошибок

Даже если дефекты не найдены, продукт может быть **неудобным или бесполезным** для пользователя.

«Соответствует ТЗ» не означает «качественный продукт»

11. Итог

Этот документ можно использовать как: - учебный материал для Junior QA - внутренний QA-гайд - основу для PDF / DOC

Хороший QA — это не про инструменты, а про внимание к деталям.