|  |  |  |
| --- | --- | --- |
| Специализация | Чем занимается | Ключевые навыки и инструменты |
| Ручной тестировщик (Manual QA) | Основа основ. Проверяет функционал "вручную", без написания скриптов. Создает тест-кейсы, баг-репорты, проводиет исследовательское тестирование. | • Чек-листы, тест-кейсы, техники тест-дизайна  • DevTools браузера, Charles/Fiddler (для перехвата трафика)  • Postman (для простых запросов к API)  • Jira, TestRail |
| Автоматизатор (QA Automation Engineer) | Пишет код для автоматического запуска тестов. Экономит время на регрессионном тестировании. Работает с UI, API, базой данных. | Языки: Java, Python, C#, JavaScript  Фреймворки: Selenium, Cypress, Playwright (UI), pytest, TestNG/JUnit, Appium (мобильное)  Инструменты: REST Assured, Postman, Git, Docker, системы CI/CD (Jenkins) |

**1. Классификация по Уровню Погружения и Автоматизации.**

**2. Классификация по Типу Продукта / Домена**

Здесь начинается узкая специализация. Ваш пример "тестировщик игр на C# Unity" находится именно в этой категории.

Тестировщик Игр (QA Game Tester)

Это огромная индустрия со своими особенностями. Специализация внутри нее может быть очень глубокой.

|  |  |  |
| --- | --- | --- |
| Специализация | Чем занимается | Ключевые навыки и инструменты |
| Функциональный тестировщик игр | Проверяет базовую функциональность: не падает ли игра, работают ли квесты, открываются ли двери, сохраняется ли прогресс. Аналог ручного тестировщика в IT. | • Чек-листы и сценарии (часто в Confluence или подобных)  • Умение описывать баги максимально четко  • Jira или специализированные системы (DevTrack, Hansoft) |
| Геймплейный тестировщик / Балансировщик | Самое интересное! Проверяет "ощущения" от игры. Сбалансировано ли оружие? Интересны ли противники? Не слишком ли сложная головоломка? Требует глубокого понимания игровой механики. | • Аналитическое мышление  • Понимание основ геймдизайна  • Часто требуются глубокие знания конкретного жанра (RPG, шутеры, стратегии) |
| Тестировщик на совместимость | Проверяет, как игра работает на разных видеокартах, процессорах, мониторах, контроллерах. Особенно актуально для ПК-игр. | • Знание железа  • Умение работать с драйверами, настройками графики |
| Тестировщик локализации | Проверяет не только перевод текста, но и культурные особенности. Подходят ли шутки? Корректно ли отображаются символы? Не выступает ли текст за границы кнопок? | • Знание языка и культуры целевой страны<br• • Внимательность к деталям |
| Технический тестировщик игр / Перформанс-инженер | Ищет узкие места в производительности: падение FPS, утечки памяти ("memory leaks"), долгие загрузки. | • Знание архитектуры ПК/консолей  • Инструменты: Profiler в Unity/Unreal Engine, RenderDoc, GPUView |

**3. Классификация по Техническим Нишам (Глубокие Специализации)**

Эти роли требуют продвинутых технических знаний.

|  |  |
| --- | --- |
| Специализация | Чем занимается |
| Тестировщик Производительности (Performance QA) | Проверяет, как система ведет себя под нагрузкой. Определяет "узкие места". |
| Тестировщик Безопасности (Security QA) | Ищет уязвимости в приложениях: SQL-инъекции, XSS, CSRF-атаки и т.д. |
| Тестировщик API (API QA) | Специализируется на тестировании бэкенда через API-интерфейсы, без участия графического интерфейса. |
| Тестировщик Базы Данных (Database QA) | Проверяет целостность данных, корректность работы запросов, миграций. |
| QA DevOps / Инженер по Качеству (Quality Engineer) | Строит и поддерживает инфраструктуру для тестирования: настраивает CI/CD пайплайны, контейнеризацию тестов, управление тестовыми средами. |

Итог: Как выбрать специализацию?

1) Начальный уровень: Начните с ручного тестирования, чтобы понять фундаментальные процессы.

2) Любите игры? Идите в геймдев. Будьте готовы к рутине, но зато сможете работать над любимыми проектами. Специализация "QA Automation в Unity" — одна из самых востребованных и высокооплачиваемых в игровой индустрии.

3) Нравится код? Осваивайте автоматизацию (Java/Python + Selenium) или производительность (JMeter).

4) Интересуетесь хакерством? Вам дорога в Security QA.

5) Любите работать с "железом"? Смотрите в сторону мобильного тестирования или тестирования на совместимость.

**3)** Задача  
 Вот три примера задач разного типа и пошаговый метод их решения.

Пример 1: Задача на тестирование новой функциональности

Задача: Протестировать новую кнопку "Экспорт в PDF" в личном кабинете пользователя.

Методика решения: Шаг 1: Анализ требований

Действие: Тестировщик изучает ТЗ: "Кнопка должна находиться на странице 'История заказов'. При нажатии формируется PDF-файл, содержащий таблицу с последними 10 заказами. Файл скачивается в папку 'Загрузки' с именем order\_history.pdf".

Вопросы, которые он задаёт: Что если заказов меньше 10? Что если заказов нет вообще? Какие именно данные должны быть в таблице (номера, даты, суммы)?

Шаг 2: Планирование и проектирование тестов

Действие: На основе анализа создаёт тест-кейсы: Позитивный сценарий: Есть >10 заказов -> нажимаем кнопку -> проверяем, что файл скачался с правильным именем и содержимым. Негативный сценарий 1: Нет ни одного заказа -> нажимаем кнопку -> проверяем, что появляется сообщение "Нет данных для экспорта". Негативный сценарий 2: Ровно 1 заказ -> нажимаем кнопку -> проверяем, что в файле только один заказ.

Шаг 3: Выполнение тестов

Действие: Запускает тестовую сборку, выполняет каждый тест-кейс.

Важно: Проверяет не только функционал, но и расположение кнопки, её внешний вид, доступность.

Шаг 4: Отчётность

Результат: Все тесты прошли успешно. Тестировщик отмечает задачу как "Выполнено" или "Готово к проверке". Если бы нашёл баг, он завёл бы баг-репорт.

Пример 2: Задача на исследовательское тестирование

Задача: Провести исследовательское тестирование процесса восстановления пароля

Методика решения:

Шаг 1: Определение областей и целей

Действие: Тестировщик определяет для себя сферу исследования: форма ввода email, процесс получения и ввода кода, установка нового пароля. Цель — найти уязвимости и неочевидные ошибки.

Шаг 2: Создание мысленной карты (Mind Map)

Действие: Мысленно или на листке бумаги набрасывает ключевые точки для проверки: Email-поле: валидация, несуществующий email, SQL-инъекция. Письмо: приходит ли, корректность ссылки/кода, срок жизни. Новый пароль: требования к сложности, проверка на старый пароль.

Шаг 3: Выполнение "сессий"

Действие: В течение 30-45 минут (сессия) тестировщик активно "атакует" функцию, следуя карте и своей интуиции. Пример действий: Вводит email с пробелами, пытается запросить код 10 раз подряд, вводит просроченный код, копирует код из письма с лишним пробелом.

Шаг 4: Документирование результатов

Результат: Тестировщик не проходит заранее написанные кейсы, а сразу фиксирует найденные баги. Например: "При вводе кода с лишним пробелом в начале система показывает неясную ошибку 'Ошибка сервера 500'".

Пример 3: Задача на верификацию исправления бага

Задача: Верифицировать исправление бага WEB-456: "В корзине неверно считается итоговая сумма при применении промокода".

Методика решения:

Шаг 1: Изучение исходного баг-репорта

Действие: Тестировщик открывает задачу WEB-456 и смотрит: Шаги воспроизведения: "1. Добавить товар за 1000 руб. 2. Применить промокод 'SALE10' (скидка 10%). 3. Перейти в корзину." Ожидаемый результат: Итоговая сумма = 900 руб. Фактический результат: Итоговая сумма = 1000 руб. (скидка не применена).

Шаг 2: Проверка исправления (Верификация)

Действие: Тестировщик берет новую сборку от разработчика и точно повторяет шаги из баг-репорта.

Проверка 1: Сценарий из бага — теперь сумма 900 руб. ✅

Проверка 2: Добавляет второй товар, проверяет, что скидка применяется к общей сумме.

Проверка 3: Пробует другие промокоды.

Шаг 3: Регрессионная проверка

Действие: Тестировщик проверяет, не повлияло ли это исправление на смежный функционал. Например, не сломалась ли обычная корзина без промокодов? Не сломался ли процесс оформления заказа?

Шаг 4: Отчётность

Результат: Если баг исправлен и регрессии нет, тестировщик меняет статус бага на "Закрыт" (Closed). Если проблема осталась или появилась новая — возвращает баг на доработку с комментарием.

Ключевые выводы по методике:

Всегда начинать с анализа. Нельзя тестировать вслепую.

Системный подход: От общего к частному. Сначала позитивные сценарии, потом негативные, потом "адские".

Документировать всё. Что делал, что ожидал, что получил. Это нужно и для баг-репортов, и для тест-кейсов.

Мыслить как пользователь, но действовать как хулиган. Пользователь будет использовать функцию "по инструкции", а тестировщик — пытаться сделать всё наоборот.

Понимать цикл жизни бага. Нашёл -> описал -> передал -> проверил исправление -> закрыл.