qaLearning

Топ вопросов по **Soft skills** с собеседований на роль QA engineer

С ОТВЕТАМИ

Sift skills

Какое вы видите решение для тестирования продукта с большим количеством фичей, задач, функций или сложным функционалом

Первое, на что надо обратить внимание - это планирование задач. Необходимо разработать test map (карту тестирования, план тестирования), определить с приоритетом задач, определить сроки. Согласно плану необходимо будет создать чек-листы либо тест-кейсы. Но самое главное - это четкое планирование с грамотным раставлением приоритетов.

Как вы сможете быть уверены, что тесты покрывают все необходимые сценарии?

Для таких целей необходимо использовать тестовые метрики. Необходимо линковать (привязывать) все тесты к соответствующим задачам в JIRA, например.
Если мы заводим баги, то необходимо линковать их к задачам, к которым они относятся.

Как вы проверяете кейсы, связанные с тестированием безопасности. Приведите пример.

Как в таковом тестировании безопасности я не участвовал (-а), но я часто проверяю валидацию не только через UI, но и путем отправки API запроса с невалидными данными. Как показывает практика, часто валидация реализована на стороне клиента, но падает при оправке невалидных данных обходным путем.

Что вы будете делать, если по какой-то причине вам не хватает ресурсов выполнить задачу в срок, например, ваш коллега-тестировщик заболел перед релизом и не закончил свои задачи.

Первым делом надо донести до менеджмента, что не хватает ресурсов. Дальше есть несколько вариантов:
1) Запросить дополнительного тестировщика. Но это не всегда работающий вариант, т.к. новому человеку нужно будет время, чтобы вникнуть в продукт, а у нас его нет.
2) Предложить вариант поработать овер-тайм.
3) Пересмотреть приоритет задач и, возможно, исключить из релиза задачи, которые имеют низкий или средний приоритет.

- [ ] **Что вы будете делать, если вам придется работать под давлением или в сжатые сроки?**

Надо всегда оставаться спокойным (-ой) и организованным (-ой).
Надо иметь четкий план и придерживаться его.
Надо правильно расставить приоритеты для задач и работать над ними исходя из расставленных приоритетов.
Самое главное, не забывать, что тестировщик отвечает за качество продукта.

- [ ] **Как убедиться, что приложение подходит для использования людьми с ограниченными возможностями.**

Обычно компании нанимают специальных тестировщиков (специальные команды), которые занимаются такого рода тестами.
Со своей стороны, я могу убедиться, что приложение имеет responsive дизайн, что при увеличении размера элементов, они продолжают корректно и читабельно отображаться, что все графические элементы имеют альтернативные надписи.
Возможно, убедиться, что видео имеют субтитры.

- [ ] **Что вы будете делать, если столкнетесь со сложной технической проблемой (багом)**

Я проведу исследование, попробую разобраться в проблеме, понять, что происходит. Проверю API запросы, которые уходят, и что мы получаем в ответ. Проверю тело запросов и ответов. Посмотрю, что на самом деле сохраняется в базе даных. Проверю логи. Попрошу помощи у коллег-разработчиков при необходимости.

- [ ] **Что вы будете делать, если на проекте надо использовать инструмент или технологию, с которой вы раньше не работали?**

Начну её изучать. Поищу какие-нибудь видео, изучу внимательно официальный гайд, как правило там есть много полезной информации.
Спрошу у коллег, у которых есть опыт работы с данным инструментом, совет с чего начать или попрошу показать мне принципы работы тулзы.

Как вы гарантируете, что ваш процесс тестирования правильно задокументирован и доведен до сведения заинтересованных сторон?

Необходимо убедиться, что были созданы следующие артефакты:
Тест-план (документ с планом тестирования, в котором описывается объем тестирования, подход к тестированию, используемые инструменты и среды тестирования, а также любые риски или предположения)
Тест-кейсы (создаются в инструменте управления тестами или в электронной таблице, в зависимости от потребностей проекта. Каждый тест-кейс должен быть четко задокументирован, включая ожидаемый результат, используемые тестовые данные и любые пред- или пост-условия)
Отчет о тестировании (в него документируют результаты тестов)
Общение - необходимо регулярно общаться с командой разработчиков и заинтересованными сторонами, чтобы убедиться, что все в курсе хода тестирования и любых возникающих проблем.

Что можете рассказать про ваш опыт на предыдущем месте работы? Чем гордитесь? Чего достигли?

Заранее составьте ответ на этот вопрос, сделайте ваш рассказ последовательным и четким. Не путайтесь!
Заранее продумайте, чем бы вы могли гордиться. Если вам нечем гордиться, придумайте! Можно, например, рассказать, что вы организовали тестовый процесс на проекте. Или начали вести подробную документацию, создали onboarding guide, по которому всегда проводят новичков при знакомстве с проектом.
Внедрили, например, Zephyr Scale, для создания тест-кейсов и тестовых циклов.

- [ ] **Вы командный игрок?**

Примеры ответов:

- Да, в моем предыдущем проекте я был частью команды, которая успешно сотрудничала для достижения поставленных целей и справилась с вызовами проекта
- Да, я демонстрирую командный подход в своей работе. Например, я активно взаимодействую с другими членами команды, обмениваюсь знаниями и опытом, и готов взять на себя ответственность за достижение целей проекта

- [ ] **Были ли конфликтные ситуации на предыдущем месте работы? Как вы выходили из них?**

Пример ответа:
Мои правила такие: надо всегда оставаться спокойной и вежливой. На моем последнем проекте у меня был конфликт с разработчиком, которому не нравилось моё оформление багов. Я спокойно и вежливо попросила его объяснить, что ему именно не нравится. Мы рассмотрели один баг, как пример, и выяснили, какую информацию ожидает видеть в баге разработчик. На этом конфликт был исчерпан.

- [ ] **Были ли конфликтные ситуации на предыдущем месте работы? Как вы выходили из них?**

Пример ответа:
Мои правила такие: надо всегда оставаться спокойной и вежливой. На моем последнем проекте у меня был конфликт с разработчиком, которому не нравилось моё оформление багов. Я спокойно и вежливо попросила его объяснить, что ему именно не нравится. Мы рассмотрели один баг, как пример, и выяснили, какую информацию ожидает видеть в баге разработчик. На этом конфликт был исчерпан.

Какой у вас опыт работы в Agile и что вы об этом знаете

Agile - это гибкая методология разработки программного обеспечения, которая ставит акцент на гибкость, сотрудничество и быструю адаптацию к изменениям. Agile подразумевает итеративный подход к разработке, разбивая проект на короткие циклы разработки, называемые спринтами.
Некоторые ключевые концепции Agile:

- Спринт: Это короткий временной интервал (обычно от 1 до 4 недель), в течение которого команда разрабатывает определенный набор функциональности или достигает конкретных целей.
- Backlog продукта: Это список требований и функциональностей, которые должны быть реализованы в проекте. Он может быть изменен и приоритизирован в течение разработки.
- Планирование спринта: Команда выбирает элементы из backlog'а продукта, которые будут реализованы в следующем спринте, и планирует, как их достичь.
- Daily Stand-up: Ежедневное короткое совещание, на котором члены команды обмениваются информацией о проделанной работе, планах на день и препятствиях, с которыми они сталкиваются.
- Ретроспектива: После завершения спринта команда проводит ретроспективное собрание, на котором анализируются прошлый спринт, выявляются улучшения и принимаются решения для будущего развития.
- Гибкость: Agile способствует быстрой адаптации к изменениям в требованиях и среде разработки, позволяя команде гибко реагировать и вносить изменения в проект.

- [ ] **Что вы будете делать, если столкнетесь с багом, который тяжело воспроизвести.**

Сделаю все, что от меня завит, чтобы понять его и попытаться воспроизвести.
Свяжусь с коллегой, который этот баг завел, попробую понять, какой шаг в описании он упустил, попробую вместе с коллегой воспроизвести баг.
Изучу внимательно логи, возможно там будет полезная информация.
Попробую на различных окружениях и различных настройках.

- [ ] **Какой алгоритм интеграции автотестов в CI/CD pipeline.**

1 Для начала надо написать тесты. например, при помощи Selenium.
2 Затем надо настроить тестовое окружение.
3 Затем надо настроить автоматический запуск тестов, например, при помощи Jenkins.
4 Надо настроить автоматический запуск тестов с каждым билдом.
5 Необходимо определить стратегию автотестов.
6 Далее можно будет мониторить результаты и решать возникающие проблемы.

- [ ] **Как определить тест-кейсы, которые подойдут для автоматизации**

Для автоматизации подходят

- часто повторяющие тесты (например, логин)
- тесты, которые требуют много времени на выполнение
- тесты, которые проверяют критический и важный функционал

Как вы гарантируете, что ваши автотесты расширяемы и легко поддерживаемые

Автотесты должны быть модульными с возможность использовать один и тот же тест много раз. Каждый тест должен иметь возможность запускаться независимо от других тестов. Такой подход гарантирует, что тесты можно поддерживать и масштабировать с течением времени.
Необходимо испольовать систему контроля версий, например Git, для управления кодом автоматических тестов. Это позволяет отслеживать изменения в тестовом коде с течением времени, при необходимости возвращаться к предыдущим версиям кода.
Необходимо использовать интегрировать тесты в CI/CD
Внедрите надлежащую обработку ошибок и отчетность, чтобы в случае сбоя теста можно было легко определить основную причину сбоя и быстро исправить ее.
Необходимо вести надлежащую документацию: подробно документировать автоматизированные сценарии тестирования, их назначение и ожидаемые результаты. Эта документация помогает поддерживать и масштабировать автоматические тесты с течением времени, поскольку новые члены команды могут быстро понять цель тестов и способы их выполнения.

Как можно определить эффективность автотестов. Какие метрики вы используете для этого?

Можно выделить следующие основные метрики:

- Тестовое покрытие - Test Coverage (показывает процент тестов, охваченных автоматизацией)
- Время выполнения теста - Test Execution Time (измеряет сколько времени требуется для запуска тестов. Используется для выявления медленных и ненадежных тестов, которые нуждаются в оптимизации)
- Частота неудачных тестов - Test Failure Rate (измеряет процент неудачных тестов. Используется для выявления тенденций и закономерностей в сбоях тестов, а также для повышения общей стабильности тестов)
- Скорость обнаружения дефектов- Defect Detection Rat (измеряет количество дефектов, обнаруженных автотестами. Используется для отслеживания эффективности автоматизации при выявлении дефектов и выявлении областей, требующих улучшения)
- Экономия затрат - Cost Savings (измеряет сумму денег, сэкономленную за счет использования автоматизации вместо ручного тестирования)

Какой у вас опыт внедрения и соблюдения лучших практик и стандартов автоматизации тестирования, таких как BDD или TDD?

Я работал (-а) над проектами, в которых мы использовали Cucumber для написания сценариев в стиле BDD. Мы сотрудничали с заинтересованными сторонами бизнеса, чтобы создать feature файлы, описывающие ожидаемое поведение приложения. Я реализовывал(-а) определения шагов и использовал(-а) их для автоматизации тестов. Этот подход помог нам убедиться, что тесты соответствуют бизнес-требованиям, и обеспечили четкое общее понимание того, что должно делать приложение.