Этапы тестирования API

Каждый тест состоит из тестовых шагов. Это отдельные атомарные действия, которые тест должен выполнять в каждом потоке тестирования API. Для каждого запроса API тест должен будет выполнить следующие действия:

1. **Проверьте корректность кода состояния HTTP**. Например, создание ресурса должно возвращать 201 CREATED, а запрещенные запросы должны возвращать 403 FORBIDDEN и т. Д.
2. **Проверьте полезную нагрузку ответа**. Проверьте правильность тела JSON, имен, типов и значений полей ответа, в том числе в ответах на ошибочные запросы.
3. **Проверьте заголовки ответа.** Заголовки HTTP-сервера влияют как на безопасность, так и на производительность.
4. **Проверьте правильность состояния приложения**. Это необязательно и применяется в основном к ручному тестированию или когда пользовательский интерфейс или другой интерфейс можно легко проверить.
5. **Проверьте базовую работоспособность**. Если операция была завершена успешно, но заняла неоправданно много времени, тест не пройден.

Категории тестовых сценариев

* Основные позитивные тесты (прохождение успешного сценария по умолчанию)
* Расширенное позитивное тестирование с дополнительными параметрами
* Негативное тестирование с валидными входными данными
* Негативное тестирование с недопустимыми входными данными
* Деструктивное тестирование
* Тесты безопасности, авторизации и доступности (которые выходят за рамки этой статьи)

**Тестирование успешного сценария по умолчанию** проверяет базовую функциональность и критерии приемки API.

Следующая группа тестов - это **негативное тестирование**, при котором мы ожидаем, что приложение будет корректно обрабатывать проблемные сценарии как с валидным вводом пользователя (например, попытка добавить существующее имя пользователя), так и с недопустимым вводом пользователя (попытка добавить имя пользователя, которое имеет значение null).

**Деструктивное тестирование** - это более глубокая форма негативного тестирования, когда мы намеренно пытаемся сломать API, чтобы проверить его надежность (например, отправляя заведомо большое тело запроса в попытке переполнить систему).

Тестовые потоки

1. **Изолированное тестирование запросов** - выполнение одного запроса API и соответствующая проверка ответа. Такие базовые тесты - это минимальные строительные блоки, с которых мы должны начинать. И нет смысла продолжать тестирование, если эти тесты упадут.
2. **Многоступенчатый рабочий поток с несколькими запросами** - тестирование серии запросов, которые являются обычными действиями пользователя, поскольку одни запросы могут зависеть от других. Например, мы выполняем запрос POST, который создает ресурс и возвращает автоматически сгенерированный идентификатор в своем ответе. Затем мы используем этот идентификатор, чтобы проверить, присутствует ли этот ресурс в списке элементов, полученных запросом GET. Затем мы используем PATCH для обновления новых данных и снова вызываем запрос GET для проверки этого обновления. И в завершении, мы УДАЛЯЕМ этот ресурс и снова используем GET, чтобы убедиться, что записи больше нет.
3. **Комбинированные тесты API и тесты веб-интерфейса** - это в основном относится к ручному тестированию, при котором мы хотим обеспечить целостность данных и согласованность между пользовательским интерфейсом и API.

Мы выполняем запросы через API и проверяем действия через пользовательский интерфейс веб-приложения и наоборот. Цель этих потоков проверки целостности состоит в том, чтобы гарантировать, что, хотя на ресурсы влияют различные механизмы, система по-прежнему поддерживает ожидаемую целостность и согласованный поток данных.