

Чек-лист тестирования мобильных приложений

В данном чек-листе указан перечень только общих проверок. Чек-лист для тестирования мобильных приложений состоит из восьми разделов:

Функциональное тестирование

В данном пункте нам важно убедиться, что наш продукт соответствует нужной функциональной спецификации, упомянутой в документации по разработке.

Что проверяем?

1. Установка/удаление/накатка версий
2. Запуск приложения (отображение Splash Screen) Splash Screen — экран загрузки.
3. Работоспособность основного функционала приложения:
 - 3.1 Авторизация (по номеру телефона/через соц. сети/e-mail)
 - 3.2 Регистрация (по номеру телефона/через соц. сети/e-mail)
 - 3.3 Онбординг новых пользователей (Онбординг (англ. onboarding — «введение, вхождение») — процесс адаптации клиента к новому продукту. Маленькие всплывающие подсказки (tooltips) и окна (pop-up) сопровождают пользователей на сайте, рассказывают о разделах или рекомендуют совершить определённые действия.)
 - 3.4 Валидация обязательных полей
 - 3.5 Навигация между разделами приложения
 - 3.6 Редактирование данных в профиле пользователя
 - 3.7 Проверка оплаты
 - 3.8 Тестирование фильтров
 - 3.9 Бонусы
4. Корректное отображение ошибок
5. Работа с файлами (отправка/получение/просмотр)
6. Тестирование тайм-аутов (Тайм-аут в телекоммуникациях и связанных с ними технических науках (в том числе компьютерных сетях и программировании), **время ожидания** — параметр, связанный с совершением события и предназначен для оценки его завершения за определенное время. Тайм-аут позволяет более эффективно использовать ограниченные ресурсы, не требуя дополнительного взаимодействия с объектом, заинтересованным в)потреблении этих ресурсов. Основная идея в том, что в ситуациях, когда система должна ждать чего-то, но не ждать неопределенное время, ожидание будет прервано после истечения тайм-аута. Это основано на предположении, что дальнейшее ожидание бесполезно, и некоторых других действий не требуется.)
7. Тестирование заглушек (не соединения с интернетом/нет, например, товаров и т.д)
8. Тестирование pop-up, алертов (Pop Up — это всплывающее окно, которое появляется поверх страницы сайта. **Алерт** — оповещение)
9. Тестирование WebView (**WebView** — это тип просмотра веб-страниц, во время которого они отображаются в определенном приложении Android без необходимости открывать ссылки во внешнем браузере. Существуют различные типы приложений WebView. К примеру можно выделить следующие: Просмотр объявлений. Программы для ведения блога, которые отображают веб-страницу в собственном приложении. Некоторые торговые приложения (например, Reliance Digital) работают на принципах WebView. Расширенный поиск с помощью технологий WebView. Аутентификация API на веб-сайте. Банковские приложения часто визуализируют выписки с помощью WebView. Далее в статье будет приведен пример сценария атаки, во время которой WebView используется для отображения банковских счетов. Уязвимости на основе WebView Варианты использования этой технологии достаточно гибкие, но в целом из-за плохой реализации WebView существуют следующие уязвимости: Сниффинг учетных данных с открытым текстом Неверная обработка ошибок SSL).
10. Скролл/свайп элементов
11. Тестирование PUSH уведомлений
12. Сворачивание/разворачивание приложения
13. Разные типы подключений (сотовая связь/Wi-Fi)
14. Ориентация экрана (альбомная/портретная)
15. Темная/светлая темы
16. Реклама в приложении
17. Шаринг контента в соц. сети (**Шаринг** (Share, Like, поделиться и т.д.) — способ распространение понравившегося **контента**. Например, в **социальной сети** «ВКонтакте», нажав на «сердечко», мы рассказываем своим «друзьям», что нам понравилось)
18. Работа приложения в фоне
19. Пагинация страниц Пагинация (pagination, пейджинг, листинг) происходит от слов «page» и «navigation» и в буквальном смысле означает «постраничную навигацию». Вывод массива данных (например товаров в категории интернет-магазина) с разбиением на несколько страниц. Пагинация — это механизм, который делит контент на отдельные страницы. Если, проскролив вниз страницы, вы видите ряд цифр — это и есть пагинация сайта или приложения.
20. Политики конфиденциальности и прочие ссылки на документы

Тестирование совместимости

Тестирование совместимости используется, чтобы убедиться, что ваше приложение совместимо с другими версиями ОС, различными оболочками и сторонними сервисами, а также аппаратным обеспечением устройства.

Что проверяем?

1. Корректное отображение гео
2. Информации об операциях (чеки и т.д.)
3. Различные способы оплаты (Google Pay, Apple Pay)
4. Тестирование датчиков (освещенности, температуры устройства, гироскоп и т.д.)
5. Тестирование прерываний (входящий звонок/смс/push/будильник/режим «Не беспокоить» и т.д.) Здесь рассматривается, как приложение реагирует на прерывание и возвращается к своему предыдущему состоянию.
6. Подключение внешних устройств (карта памяти/наушники и т.д.)

Тестирование безопасности

Данная проверка нацелена на поиск недостатков и пробелов с точки зрения безопасности приложения.

Что проверяем?

1. Тестирование разрешений (доступ к камере/микрофону/галерее/и т.д.)
2. Данные пользователя (пароли) не передаются в открытом виде
3. В полях, с вводом пароля и подтверждением пароля, данные скрываются астерисками

Тестирование локализации и глобализации

Тестирование интернационализации/глобализации приложения включает тестирование приложения для различных местоположений, форматов дат, чисел и валют, а также замену фактических строк псевдостроками. Тестирование локализации включает тестирование приложения с локализованными строками, изображениями и рабочими процессами для определенного региона.

Что проверяем?

1. Все элементы в приложении переведены на соответствующий язык
2. Тексты защиты внутри приложения и пользователь в настройках приложения может выставить необходимый язык
3. Тексты зависят от языка в системных настройках
4. Тексты приходят с сервера
5. Корректное отображение форматов дат (ГОД — МЕСЯЦ — ДЕНЬ или ДЕНЬ — МЕСЯЦ — ГОД.)
6. Корректное отображение времени в зависимости от часового пояса

Тестирование удобства использования

Тестирование удобства использования помогает удостовериться в простоте и эффективности использования продукта пользователем, с целью достижения поставленных целей. Иными словами, это не что иное, как тестирование дружелюбности приложения для пользователя.

Что проверяем?

1. Корректное отображение элементов на устройствах с различными разрешениями экранов
2. Все шрифты соответствуют требованиям
3. Все тексты правильно выровнены
4. Все сообщения об ошибках верные, без орфографических и грамматических ошибок
5. Корректные заголовки экранов
6. В поисковых строках присутствуют плейсхолдеры
7. Неактивные элементы отображаются серым
8. Ссылки на документы ведут на соответствующий раздел на сайте
9. Анимация между переходами
10. Корректный возврат на предыдущий экран
11. Поддерживаются основные жесты при работе с сенсорными экранами (swipe back и т.д.)
12. Пиксель-перфект

Стрессовое тестирование

Стрессовое тестирование направлено на определение эффективности производительности приложения в условиях повышенной нагрузки. Стресс-тест в этом контексте ориентирован только на мобильные устройства.

Что проверяем?

1. Высокая загрузка центрального процессора
2. Нехватка памяти
3. Загрузка батареи
4. Отказы
5. Низкая пропускная способность сети
6. Большое количество взаимодействий пользователя с приложением (для этого может понадобиться имитация реальных условий состояния сети)

Кросс-платформенное тестирование

Важный вид тестирования, который необходимо проводить для понимания того, будет ли должным образом отображаться тестируемый продукт на различных платформах, используемых целевой аудиторией.

Что проверяем?

— Работоспособность приложения на различных устройствах разных производителей

Тестирование производительности

Если пользователь устанавливает приложение, и оно не отображается достаточно быстро (например, в течение трех секунд), оно может быть удалено в пользу другого приложения. Аспекты потребления времени и ресурсов являются важными факторами успеха для приложения, и для измерения этих аспектов проводится тестирование производительности.

Что проверяем?

1. Время загрузки приложения
2. Обработка запросов
3. Кэширование данных
4. Потребление ресурсов приложением (например расход заряда батареи)

Мобильное тестирование: полное руководство

Александр Шитиков, руководитель отдела разработки AGIMA.mobile, прокомментировал перевод статьи о шести основных типах тестирования мобильных приложений

Данная статья будет полезна всем, кто вовлечен в процесс разработки как со стороны агентства, так и со стороны клиента: менеджерам, проектировщикам, разработчикам, тестировщикам. В ней рассматриваются основные особенности тестирования мобильных приложений. В своих комментариях я более подробно раскрою нюансы по каждому типу тестирования, а также приведу несколько примеров использования. Вам подойдет один из представленных ниже типов в зависимости от того, какую цель вы преследуете:

- функциональное тестирование;
- тестирование производительности;
- тестирование безопасности;
- тестирование удобства пользования (юзабилити-тестирование);
- тестирование совместимости;
- тестирование на восстановление.

Функциональное тестирование

Функциональное тестирование мобильных приложений обычно охватывает тестирование взаимодействия с пользователем, а также тестирование транзакций. Важные для этого вида тестирования факторы:

1. Тип приложения, определяемый его бизнес-функциональностью (банкинг, игровая индустрия, социальные сети, образование).
2. Целевая аудитория (пользователь, компания, образовательная среда).
3. Канал, по которому распространяется приложение (например, App Store, Google Play или раздача напрямую).
4. Говоря простым языком, мы проверяем, выполняет ли приложение ожидаемые функции, которые обычно описаны в спецификации или продиктованы бизнес-процессами.
5. Поэтому функциональное тестирование может проводиться на основе требований. В этом случае формируются тестовые случаи (Test cases), для их создания используется техническое задание на основе бизнес-процессов. После этого создают, так называемые случаи использования (Use cases). Они описывают сценарии ежедневного или постоянного использования приложения.

Основные сценарии функциональных тестов:

1. Проверить корректность работы обязательных полей.
2. Убедиться, что обязательные поля отображаются на экране не так, как необязательные.
3. Убедиться, что работа приложения во время запуска/выхода удовлетворяет основным требованиям.
4. Убедиться, что приложение переходит в фоновый режим в случае входящего звонка. Для этого вам понадобится еще один телефон.
5. Проверить, может ли телефон хранить, принимать и отправлять SMS-сообщения во время работы приложения. Для этого вам понадобится другой телефон, с которого можно отправить сообщение на тестируемое устройство с уже запущенным приложением.

6. Убедиться, что устройство работает в многозадачном режиме, когда это необходимо.
7. Проверить, как функционируют необходимые опции для работы с социальными сетями — Поделиться, Публикация, Навигация.
8. Убедиться, что приложение поддерживает платежные операции через системы оплаты Visa, Mastercard, PayPal и др.
9. Проверить адекватность работы сценариев прокрутки страницы.
10. Проверить, присутствует ли надлежащая навигация между важными модулями приложения.
11. Убедиться, что количество ошибок округления минимально.
12. Проверить наличие сообщений об ошибках, например, сообщения «Ошибка сети. Пожалуйста, попробуйте позже» в случае некорректной работы сети.
13. Убедиться, что установленное приложение не препятствует нормальной работе других приложений и не съедает их память.
14. Проверить, способно ли приложение вернуться в то состояние, в котором оно находилось перед приостановкой (например, жесткая перезагрузка или системный сбой).
15. Установка приложения должна проходить без значительных ошибок при условии, что устройство соответствует системным требованиям.
16. Убедиться, что автоматический запуск приложения работает корректно.
17. Проверить, как приложение работает на всех устройствах поколений 2G, 3G и 4G.
18. Выполнить регрессивное тестирование для выявления новых программных ошибок в существующих и уже модифицированных областях системы. Дополнительное проведение всех предыдущих тестов для проверки поведения программы после изменений.
19. Убедиться, что существует доступное руководство пользователя.

Часто система обладает большим количеством функций, и не всегда есть возможность проверить их все. Поэтому перед началом функционального тестирования обычно приоритезируют те или иные тест-кейсы и юз-кейсы, в соответствии с расставленными приоритетами распределяют время и затем уделяют внимание наиболее важным. Выделить какие-то стандартные сценарии для функциональных тестов довольно сложно из-за разнообразия приложений, но можно выделить часто встречающиеся модули, составить для них тест-кейсы и в дальнейшем использовать их, модифицируя под конкретные требования.

Для каждой функции необходимо проверять как позитивные сценарии, так и негативные. Сценарий считается позитивным, если в итоге пользователь достигает своей цели (создает item, отправляет сообщение и т.д.). Негативный, соответственно, наоборот — на каком-то из шагов происходит ошибка, и цель не может быть достигнута.

К примеру, рассмотрим логин/логаут и создание контакта (раздела, пользователя, или любого другого item). Стандартный логин/логаут может включать в себя опции:

- регистрация: с логином и паролем, без пароля, через соц.сети и т.д.;
- авторизация: с логином и паролем, через соц.сети и т.д.;
- восстановление пароля;
- выход из системы: самостоятельный, по истечению сессии и т.д.

Позитивные сценарии:

- Регистрация в приложении доступна всеми описанными в ТЗ способами.
- Можно зарегистрироваться, заполнив только обязательные поля.
- Можно зарегистрироваться, заполнив полностью все поля.
- После регистрации можно авторизоваться в приложении. При этом введенные данные корректно сохранены в профиле (e-mail, пароль, личная информация и т.д.).
- Зарегистрировавшись на одном устройстве, можно авторизоваться на другом — данные корректно сохраняются на сервере и доступны.
- Выход из системы работает корректно.
- Восстановление пароля работает корректно.

Негативные сценарии (самое очевидное):

- Повторная регистрация на один и тот же e-mail, с одним и тем же логином недоступна.
- Регистрация без заполнения обязательных полей недоступна.
- Регистрация, если все поля оставлены пустыми, недоступна.
- Регистрация, если формат введенных данных не соответствует требованиям, недоступна.
- Авторизация с пустыми полями недоступна.
- Авторизация с неправильным/удаленным/заблокированным логином недоступна.
- Авторизация с неправильным паролем недоступна.

Создание контакта. Логично предположить, что если пользователь создает контакт, то должна быть возможность его просмотреть, отредактировать и удалить. Это базовый набор функций, которыми может обладать item.

Позитивные сценарии:

- Создание, изменение, просмотр и удаление контактов доступны.
- Создание контакта с минимальным набором данных доступно.
- Создание контакта с максимальным набором данных доступно.
- При создании корректно обрабатываются все описанные в ТЗ типы данных.
- После создания контакт доступен для просмотра.
- Изменение учитывает обязательные поля/данные/элементы. Сохранить контакт без них недоступно.
- После удаления контакт больше не доступен.

Негативные сценарии:

- Создание двух одинаковых контактов недоступно (это может быть и позитивным сценарием).
- Создание контакта с отсутствующими обязательными элементами/данными недоступно.

Сюда же, к функциональному тестированию, отнесу проверку пользовательского интерфейса:

1. Проверка экранов на совпадение с макетами.
2. Проверка работы «нативных» жестов: свайп, мультитач и т.д. — приложение должно реагировать на них определенным образом.
3. Проверка состояний элементов: кнопки изменяют цвет, если нажаты; списки сворачиваются и разворачиваются и т.д.
4. Проверка локализации, если таковая заявлена в приложении. При этом важно уделить внимание верстке — многие названия на других языках гораздо длиннее, чем на английском или на русском.

Тестирование производительности

Также известно как нагрузочное тестирование. Это автоматизированное тестирование, которое имитирует работу определенного количества пользователей какого-либо общего ресурса.

Основные задачи:

1. Определить количество пользователей, которые могут одновременно работать с приложением.
2. Проверить, как ведет себя приложение при увеличении интенсивности выполнения каких-либо операций.
3. Проверить работоспособность приложения при многочасовом использовании на средней нагрузке.
4. Проверить поведение приложения в стресс-условиях.
5. Проверить работу в условиях «разросшейся» базы данных — насколько быстро выполняются запросы.

Основная цель этого вида тестирования — убедиться в том, что приложение работает приемлемо при определенных требованиях производительности: доступ большому числу пользователей, устранение важного элемента инфраструктуры, как, например, сервера базы данных, и др.

Основные сценарии тестирования производительности мобильных приложений:

1. Определить, работает ли приложение одинаково в разных условиях загрузки сети.
2. Выяснить, способно ли текущее покрытие сети обеспечить работу приложения на различных уровнях пользовательской нагрузки.
3. Выяснить, обеспечивает ли существующая клиент-серверная конфигурация оптимальную производительность.
4. Найти различные узкие места приложения и инфраструктуры, которые снижают производительность приложения.
5. Проверить, соответствует ли требованиям время реакции приложения.
6. Оценить способность продукта и/или аппаратного обеспечения справляться с планируемыми объемами нагрузки.
7. Оценить время, в течение которого аккумулятор может поддерживать работу приложения в условиях планируемых объемов нагрузки.
8. Проверить работу приложения в случаях перехода из Wi-Fi-сети в мобильную 2G/3G-сеть и наоборот.
9. Проверить, что каждый из уровней памяти процессора работает оптимально.

10. Убедиться в том, что потребление батареи и утечка памяти не выходят за пределы нормы, а работа различных ресурсов и сервисов, таких как GPS-навигация или камера, соответствует требованиям.
11. Проверить стойкость приложения в условиях жесткой пользовательской нагрузки.
12. Проверить эффективность сети в условиях, когда устройство находится в движении.
13. Проверить производительность приложения, если оно работает в условиях непостоянного подключения к интернету.

Тестирование безопасности

Это стратегия тестирования, используемая для проверки безопасности системы, а также для анализа рисков, связанных с обеспечением целостного подхода к защите приложения, атак хакеров, вирусов, несанкционированного доступа к конфиденциальным данным. Основная цель этого типа тестирования — обеспечить безопасность сети и данных приложения.

Ниже приведены ключевые действия для проверки безопасности мобильного приложения.

1. Убедиться в том, что данные пользователей приложения — логины, пароли, номера банковских карт — защищены от сетевых атак автоматизированных систем и не могут быть найдены путем подбора.
2. Удостовериться в том, что приложение не дает доступ к секретному контенту или функциональности без надлежащей аутентификации.
3. Убедиться в том, что система безопасности приложения требует надежного пароля и не позволяет взломщику завладеть паролями других пользователей.
4. Убедиться в том, что время таймаута сессии адекватно для приложения.
5. Найти динамические зависимости и принять меры для защиты этих уязвимых участков от взломщиков.
6. Защитить приложение от атак типа SQL-injection.
7. Найти случаи неуправляемого кода и устранить его последствия.
8. Удостовериться в том, что срок действия сертификатов не истек, вне зависимости от того, использует приложение Certificate Pinnig или нет.
9. Защитить приложение и сеть от DoS-атак.
10. Проанализировать требования хранения и проверки данных.
11. Обеспечить управление сеансами для защиты информации от неавторизованных пользователей.
12. Проверить все криптографические коды и, если необходимо, исправить ошибки.
13. Удостовериться в том, что бизнес-логика приложения защищена и не подвержена атакам извне.
14. Проанализировать взаимодействие файлов системы, выявить и скорректировать уязвимые места.
15. Проверить обработчики протокола (например, не пытаются ли перенастроить целевую страницу по умолчанию, используя вредоносные плавающие фреймы).
16. Защитить приложение от вредоносных атак на клиентов.
17. Защитить систему от вредоносных внедрений в момент работы программы.
18. Предотвратить возможные вредоносные последствия кэширования файлов.
19. Предотвратить ненадежное хранение данных в кэш-памяти клавиатуры устройства.
20. Предотвратить возможные вредоносные действия файлов cookie.
21. Обеспечить регулярный контроль безопасности данных.
22. Изучить пользовательские файлы и предотвратить их возможное вредоносное влияние.
23. Обезопасить систему от случаев переполнения буфера или нарушения целостности информации в памяти.
24. Сделать анализ различных потоков данных и защитить систему от их возможного вредоносного влияния.

Юзабилити-тестирование

Юзабилити-тестирование проводится для создания быстрых и простых в обращении приложений. Главная цель — обеспечить удобство пользования приложением, создать интуитивный, соответствующий принятым стандартам интерфейс.

Итак, для проведения юзабилити-тестирования следует:

1. Убедиться в том, что кнопки имеют нормальный размер и подходят для крупных пальцев.
2. Поместить кнопки в одной области экрана, чтобы не вызвать замешательства у пользователей.
3. Убедиться в том, что значки и картинки смотрятся естественно в среде приложения.
4. Убедиться в том, что цвет кнопок, выполняющих одну и ту же функцию, совпадает.
5. Убедиться в правильной работе системы уменьшения и увеличения масштаба просмотра.
6. Обеспечить минимальный ввод данных с клавиатуры.
7. Убедиться в наличии возможности возврата или отмены действия в случае нажатия не на ту кнопку.
8. Убедиться в том, что контекстуальные меню не перегружены, так как они предполагают быстрое использование.

9. Убедиться в том, что текст прост, ясен и виден пользователю.
10. Убедиться в том, что короткие предложения и абзацы возможно прочитать.
11. Найти оптимальный размер шрифта.
12. Убедиться в том, что в случае загрузки пользователем больших объемов информации приложение предупреждает о возможных сбоях в его работе из-за этого.
13. Убедиться в том, что завершить работу приложения можно из любого состояния и что оно возобновляет работу в этом же состоянии.
14. Проверить, что все строки отображаются на нужном языке, если в приложении есть опция перевода.
15. Убедиться в том, что компоненты приложения синхронизируются с действиями пользователя.
16. Обеспечить пользователя руководством, которое бы помогло ему понять работу приложения и эффективно им пользоваться.

Юзабилити-тестирование обычно проводится на пользователях, поскольку только люди могут понять субъективные ощущения других людей, вызываемые тем или иным приложением.

Тестирование удобства использования — это метод тестирования, направленный на установление степени удобства использования, обучаемости, понятности и привлекательности для пользователей разрабатываемого продукта в контексте заданных условий

Тестирование удобства пользования дает оценку уровня удобства использования приложения по следующим пунктам:

- Производительность, эффективность (efficiency) — сколько времени и шагов понадобится пользователю для завершения основных задач приложения, например, размещения новости, регистрации, покупки (чем меньше времени и шагов понадобится пользователю, тем лучше).
- Правильность (accuracy) — сколько ошибок сделал пользователь во время работы с приложением?
- Активизация в памяти (recall) — как долго пользователь помнит о том, как пользоваться приложением, после приостановки работы с ним на длительный период времени? (Повторное выполнение операций после перерыва должно проходить быстрее, чем у нового пользователя).
- Эмоциональная реакция (emotional response) — Как пользователь себя чувствует после завершения задачи: растерян, испытал стресс или, наоборот, ему все понравилось? Посоветует ли пользователь систему своим друзьям?

Для улучшения удобства использования полезно следовать двум принципам:

1. «Защита от дурака». Если поле предполагает ввод номера телефона, то стоит ограничить диапазон ввода только цифрами и соответствующим образом сформировать клавиатуру. Аналогично для e-mail и остальных элементов, которые предполагают пользовательский ввод данных.
2. Использовать цикл Демминга (планирование-действие-проверка-корректировка), то есть собирать информацию о дизайне и удобстве использования у существующих пользователей, и на основе их мнений планировать изменения в приложении.

Конфигурационное тестирование

Конфигурационное тестирование проводится для того чтобы обеспечить оптимальную работу приложения на разных устройствах — с учетом их размера, разрешения экрана, версии, аппаратного обеспечения и пр.

Важнейшие сценарии конфигурационного тестирования:

1. Убедиться в том, что интерфейс приложения соответствует размеру экрана устройства, текст не выходит за рамки дисплея.
2. Убедиться в том, что текст легко читается на любом устройстве.
3. Убедиться в том, что функция вызова/будильника доступна при запущенном приложении, приложение сворачивается или переходит в режим ожидания в случае входящего звонка, а по его завершении возобновляется.

"Это тип тестирования, предназначенный для проверки работоспособности приложения на различных конфигурациях системы. Имеет смысл рассмотреть клиентский уровень конфигурационного тестирования. Сервер у мобильных приложений часто единственный, а клиент устанавливается на большое количество самых разнообразных устройств. Но нужно учесть, что если приложение специфичное — например, игра, для которой выделено несколько серверов, то серверный уровень также становится приоритетным.

На клиентском уровне можно выделить:

- Тип устройства: смартфон, планшет и т.д.
- Конфигурация устройства: количество оперативной памяти, тип процессора, разрешение экрана, емкость аккумулятора и т.д.
- Тип и версия операционной системы. iOS 6, 7; Android 4.2.2 и т.д.
- Тип сети: Wi-Fi, GSM.

Перед проведением конфигурационного тестирования рекомендуется

- Создавать матрицу покрытия (матрица покрытия — это таблица, в которую заносят все возможные конфигурации).
- Проводить приоритезацию конфигураций (на практике, скорее всего, все желаемые конфигурации проверить не получится).
- Шаг за шагом, в соответствии с расставленными приоритетами, проверяют каждую конфигурацию.

Уже на начальном этапе становится очевидно: чем больше требований к работе приложения при различных конфигурациях рабочих станций, тем больше тестов нам необходимо будет провести. В связи с этим рекомендуем по возможности автоматизировать этот процесс, так как именно при конфигурационном тестировании автоматизация реально помогает сэкономить время и ресурсы. Конечно же, автоматизированное тестирование не является панацеей, но в данном случае оно окажется очень эффективным помощником.

Тестирование на восстановление

Тестирование на восстановление проверяет тестируемый продукт с точки зрения способности противостоять и успешно восстанавливаться после возможных сбоев, возникших в связи с ошибками программного обеспечения, отказами оборудования или проблемами связи. Применяется чаще всего в приложениях, которые должны работать 24x7, где каждая минута простоя стоит очень дорого.

1. Проверка восстановления после сбоя системы и сбоя транзакций.
2. Проверка эффективного восстановления приложения после непредвиденных сценариев сбоя.
3. Проверка способности приложения обрабатывать транзакции в условиях сбоя питания (разряженная батарея / некорректное завершение работы приложения).
4. Проверка процесса восстановления данных после перерыва в соединении.

Другие важные области проверки:

1. Тестирование установки (быстрая, соответствующая требованиям установка приложения).
2. Тестирование удаления (быстрое, соответствующее требованиям удаление приложения).
3. Сетевые тест-кейсы (проверка адекватной работы сети в разных условиях загрузки, а также способности сети обеспечить функционирование всех приложений, используемых в ходе тестирования).
4. Проверка наличия нефункциональных клавиш.
5. Проверка экрана загрузки приложения.
6. Проверка возможности ввода с клавиатуры во время сбоев сети.
7. Проверка методов запуска приложения.
8. Проверка наличия эффекта зарядки в случае, если приложение находится в фоновом режиме.
9. Проверка функционирования экономичного режима и режима высокой производительности.
10. Выявление последствий извлечения аккумулятора во время работы приложения.
11. Проверка уровня потребления энергии приложением.
12. Проверка побочных эффектов приложения.

Отдельно укажу **интеграционное тестирование**, в рамках которого проводятся проверки взаимодействия модулей друг с другом, а также приложения с операционной системой и другими приложениями. Сюда попадают все сценарии, связанные с различного рода прерываниями во время работы с приложением:

- произошел телефонный звонок;
- поступило смс;
- на экране появилось уведомление другого приложения;
- пропала сеть или ухудшилось качество связи (попадает сюда, т.к. это обычное событие для мобильных сетей);
- приложение было свернуто в трей;
- приложение переведено в фоновый режим и т.д.

Данные случаи должны быть предусмотрены при разработке и тестировании приложения.