Mobile Testing

Мобильное приложение — это программное обеспечение, которое можно загрузить на смартфон или планшет. Всё началось с простых игр на ещё кнопочных телефонах, но современные продукты могут закрывать почти любые потребности: оплатить налоги, записаться к врачу, найти вакансии по запросу или пару на вечер, заказать еду и забронировать отель.

Виды мобильных приложений

По целям бизнеса

Внутреннее пользование

- Сервисы, оптимизирующие работу сотрудников внутри компании: корпоративные социальные сети и мессенджеры, виртуальный офис, облачные хранилища данных и пр.

Как инструмент маркетинга

- Программы лояльности: агрегаторы скидок, системы бонусных карт и кэшбэка для постоянных клиентов.
- Онлайн-сервисы бизнеса: программы для записи к врачу, бронирование туров, отелей и прочее с возможностью проводить банковские транзакции.

В этих случаях приложение используется как инструмент для реализации маркетинговой стратегии — его интерфейс позволяет отправлять пуши и напоминания, побуждающие аудиторию воспользоваться скидкой, купить товар, забронировать жилье или записаться к врачу.

По назначению

- Игровые

Задачи на логику, гонки, квесты, викторины, шутеры, детские, взрослые и семейные игры — возможности современной разработки почти безграничны и позволяют создать как простую, так и сложную многоступенчатую систему виртуальной реальности.

- Для e-commerce и сферы услуг

Тут собираем всё, что помогает компании охватить как можно больше потенциальной аудитории: программы для заказа такси, записи на приём к косметологу, покупка билетов в кинотеатр и пр.

- Контентные

Помогают пользователям быстро получать актуальный контент: новости из изданий и газет, блоги с полезными статьями о психологии, биржи с курсом валют и стоимостью акций, задания по языковым курсам.

- Соцсети

Сервисы, которые помогают общаться через смартфон и планшет: ВКонтакте, Instagram, Facebook, Gmail и пр.

По особенностям работы

- **Нативные**

Это проекты, которые созданы под конкретную платформу, написаны на её родном языке и предоставляют все доступные возможности смартфона: камеру, список контактов, GPS, данные о здоровье и режиме сна и т.д. Сервисы под Android чаще всего пишутся на Java, для iOS — Swift или objective-C.

Плюсы: высокая скорость и производительность, возможность реализовать максимальный набор функций, понятный интерфейс, способность работать без интернета, надёжный уровень безопасности, поддержка от маркетплейсов.

Минусы: высокая стоимость и долгий процесс разработки, дорогое техобслуживание.

- **Мобильные веб-приложения**

Его можно назвать сайтом, адаптированным под любой телефон. Веб-сервис можно установить как закладку в браузере и использовать вне зависимости от платформы, не скачивая на телефон и не тратя память.

Плюсы: простой и недорогой процесс создания, не нужно проходить модерацию и публиковаться в каждом маркетплейсе, лёгкий доступ для пользователей.

Минусы: для работы нужно подключение к интернету, ограниченный интерфейс, низкая производительность и скорость, нельзя отправлять push-уведомления.

- **Кроссплатформенные (гибридные)**

Универсальные сервисы, которое создаются сразу под две платформы: iOS и Android и сочетают особенности веб и мобильных приложений.

Плюсы: низкая стоимость и высокая скорость выпуска, кроссплатформенность, автономное обновление.

Минусы: скорость ниже, чем у нативных, некорректная работа в случае плохого интернета, ограниченные возможности визуала.

Как работают мобильные приложения

Чем отличаются от веб-сайтов

Мобильные сервисы сложнее и дороже создавать, но позволяют качественное взаимодействовать с пользователями — затраты на них быстро окупаются и помогают формировать лояльную аудиторию.

- Уведомления. Через приложение можно отправлять push-уведомления и напоминания, даже если человек не открывает сервис, выполнять функции в фоновом режиме и без подключения к интернету.
- Оперативная обратная связь с компанией через чат и техподдержку.
- Индивидуальный сервис. Есть возможность использовать геолокацию, биологический ритм человека, данные об интересах и запросах в поисковиках, чтобы предложить индивидуальный сервис: вызвать машину к дому, создать рацион питания и режим тренировок, предложить необходимые анализы с учётом истории болезней или подобрать подходящую пару по интересам.
- Удобнее пользоваться: интерфейс приложения адаптирован под действия пользователей и имеет понятную структуру с кнопками.
- Подробная аналитика. С помощью статистики в приложении можно анализировать поведение целевой аудитории, составлять более подробный портрет клиента и подбирать эффективные маркетинговые стратегии.

Архитектура

Тут есть два основных блока: фронтенд и бэкенд-части. Они действуют как сплит-система и взаимодействуют друг с другом, передавая информацию и обеспечивая бесперебойную работу продукта.

Back-end часть не видна пользователям, но именно на ней держится вся логика сайта, обрабатываются данные и отправляются реакции. Бэкенд-разработчики обеспечивают корректное функционирование интерфейса, заставляют каждую кнопку переносить человека на нужную страницу, совершать оплату через банковские системы и собирать данные.

Front-end обеспечивает внешний вид интерфейса, с которым взаимодействуют пользователи. Это дизайн страниц, кнопок, пуш-уведомлений и других графических элементов, карта путешествия пользователя и взаимодействие с функциями.

Особенности тестирования мобильных приложений

Разработка мобильных приложений отличается от десктопных. Поэтому и в процессе тестирования инженер обязан провести проверки, обусловленные природой программных продуктов. Например, необходимо провести проверку установки обновлений.

Разработчики операционных систем постоянно улучшают платформы и делают их более безопасными и производительными. Это формирует и новые требования к мобильным приложениям. Пользователь не должен испытывать каких-либо сложностей в процессе обновления. А если пользователь не устанавливает новые версии вовремя? Как на это отреагирует приложение? На эти вопросы тестировщик ищет ответы.

Тестирование помогает выявить реакцию приложения на непредсказуемые пользовательские действия. Представьте, разблокированный гаджет оказался в кармане или сумке, и приложение должно корректно справляться с набором хаотичных и бессвязных действий.

Ещё один вид проверок – оценка качества различного вида соединений. Такое тестирование проходит в лабораторных условиях, где возможно воссоздать максимально реалистичные условия связи. Этот вид проверки демонстрирует, как приложение будет вести себя в нестандартных ситуациях, например, когда сигнал Wi-Fi едва уловим.

Команда на проекте может состоять целиком из тестировщиков мобильных приложений или оставаться комбинированной. Это зависит от особенностей тестируемого продукта. В аутсорсинговой компании может быть всего несколько специалистов по оценке качества мобильного ПО или большой отдел. В этом случае начинающий специалист может построить карьеру от джуниора до лида. Продуктовые компании также могут иметь собственное QA-подразделение.

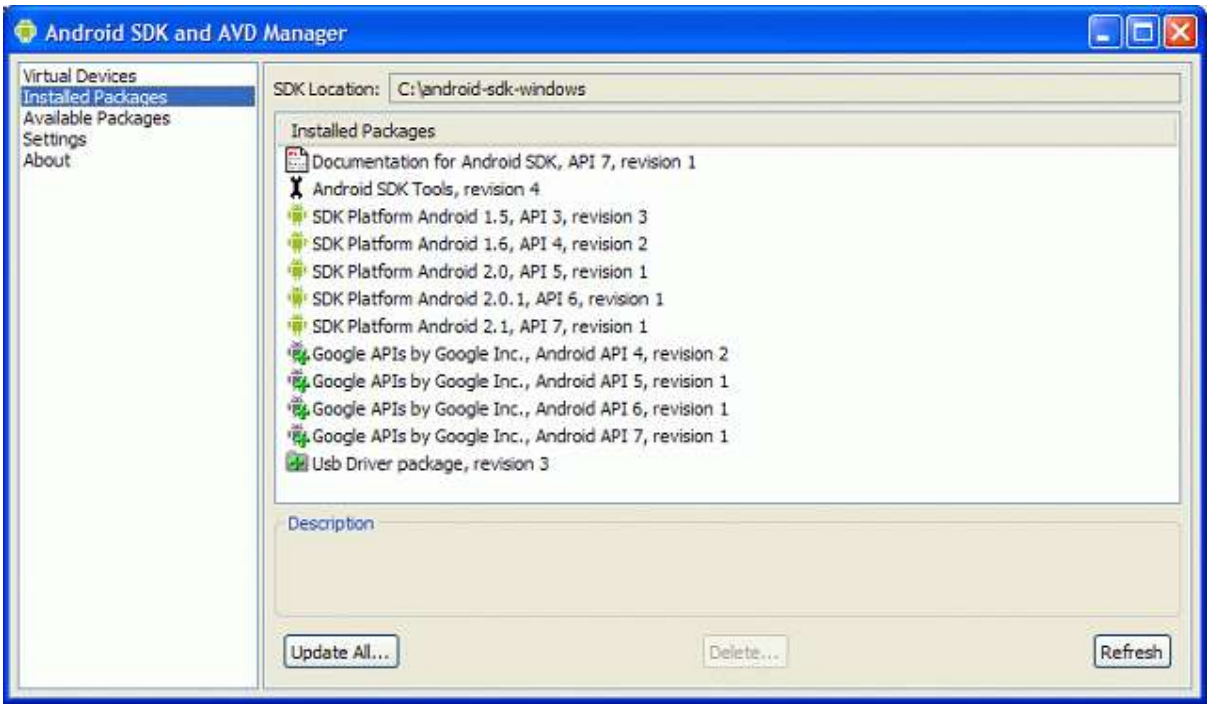
Инструменты для тестирования мобильных приложений

Инструментарий QA-инженера достаточно богат: эмуляторы, сервисы бета-тестирования, программы для сбора статистических данных и прочее. Рассмотрим их подробнее.

Эмуляторы

Эмуляторы – программы, которые имитируют поведение других устройств. Главное преимущество эмуляторов в том, что они помогают протестировать сложные сценарии, которые не рекомендуется проводить на настоящих мобильных телефонах (если тесты могут вывести устройство из строя).

Сегодня можно найти эмуляторы для всех наиболее распространенных ОС. Например, Android SDK позволяет запускать отладку и тестирование исходного кода. К тому же, результат можно получать в режиме реального времени.



Сервисы для бета-тестирования

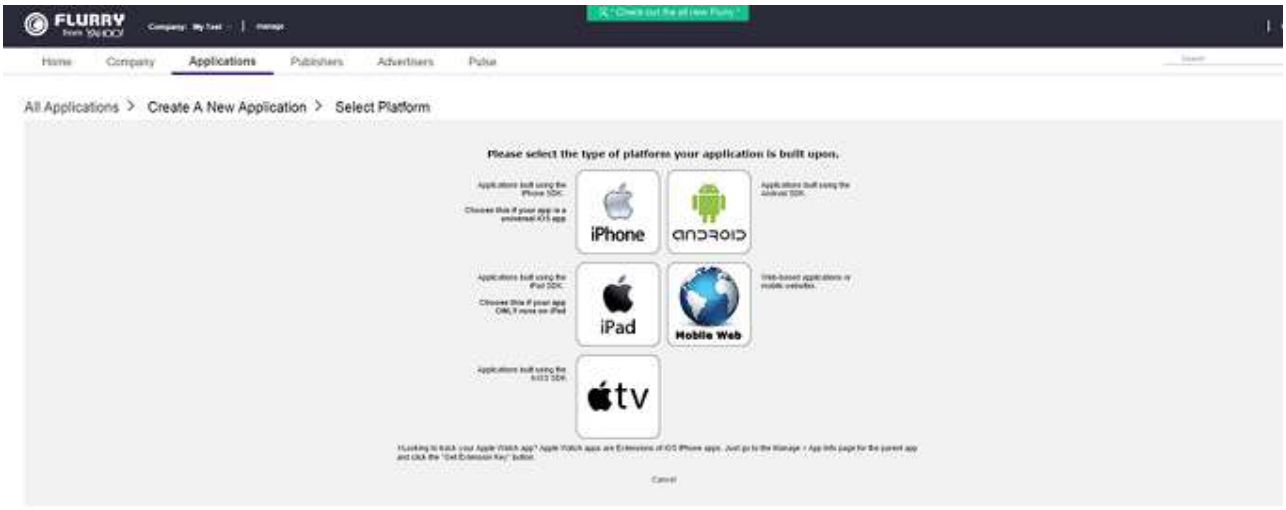
Напоминаем, что бета-тестирование – активная работа с почти готовой версией продукта для нахождения всех возможных ошибок и их дальнейшего устранения.

The Beta Family – бесплатный сервис, позволяющий завести аккаунт и загрузить бета-версию приложения. После этого можно отправить приглашение протестировать ПО и затем проанализировать полученные результаты.

Сбор статистики

Статистическая информация о том, как пользователи работают с приложением, будет не лишней. Вы узнаете, какие функции привлекают пользователей больше всего, какие ошибки они совершают.

Также вы получите статистику о том, какие версии ОС чаще всего встречаются у пользователей приложения, представленность вашего приложения на географической карте. Получить такие данные можно, используя решения по сбору информации. Примерами наиболее распространенных бесплатных систем являются Google Analytics, Flurry, BugSence, Apsalar.



Для эффективного тестирования программных продуктов многие компании комплектуют собственный парк мобильных устройств. Это большая коллекция самых разнообразных гаджетов от умных часов до планшетов и электронных книг на базе различных ОС. Чем больше парк устройств, тем больше дефектов ПО можно выявить и сделать мобильное приложение более качественным и надёжным.

Частые дефекты в тестировании мобильных приложений

Говоря о тестировании мобильных приложений, можно выделить ряд специфических багов.

Остановка работы приложения

Подобное случается, когда в приложении есть неустранённые дефекты. Как только приложение внезапно прекращает свою работу, пользователь получает уведомление об ошибке. Но не всегда перезагрузка или очистка памяти устройства помогает.

Иногда нужно прибегнуть и к более радикальным методам. Например, полная очистка внутренней памяти, переустановка приложения или даже сброс к заводским настройкам. В этом случае пользовательские данные стираются без возможности последующего восстановления.

Ошибки в поддержке разных версий ОС, браузеров и устройств

Это часто случается с гаджетами, чьи платформы уже не обновляются разработчиками. Дефекты обнаруживаются при выполнении кросс-платформенного и кросс-браузерного тестирования. Некоторые ошибки возможно исправить, но избавиться от наиболее серьёзных багов можно лишь после смены гаджета.

Рекомендуется проводить тестирование на различных операционных системах (Windows, Android, macOS, Linux) и в разных браузерах (Chrome, IE, Opera, Firefox, Safari, Konqueror и другие). Однако объем тестирования, конечно, зависит от требований заказчика.

Сбои при отображении на экранах разного разрешения

Такие ошибки возникают, когда приложение не оптимизировано для работы на устройствах с разным разрешением экрана. Также элементы интерфейса могут деформироваться при смене положения смартфона в пространстве.

Дефекты в локализации приложения

Поскольку сейчас мобильные приложения в основном предназначены для использования на международном рынке, тестированию локализации уделяют много внимания.

К процессу локализации можно отнести не только перевод текста на другой язык. Здесь важно учитывать и присущие каждой конкретной культуре особенности перевода. Например, корректная передача аббревиатур, имён, валют, форматов даты и времени. Даже цвет может иметь значение. Например, в западной культуре традиционным цветом траура считается чёрный, а вот в Индии, Японии и Корее его место занимает белый.

Локализационные ошибки возникают очень часто и нередко они отпугивают пользователей, снижают успех приложения на конкретном рынке.

Требования к разработке приложений

Разработка мобильного приложения состоит из множества этапов: оформление, идеи, разработка стратегии, работа над дизайном, непосредственно разработка, выход на рынок и мониторинг ситуации.

Существует ряд ключевых требований для обеспечения успеха любого приложения для мобильных телефонов независимо от платформы, на которой оно будет развернуто.

- Установка, удаление и обновление приложения на устройстве должны быть простыми.
- Приложение должно удовлетворять потребности пользователя привлекательным, уникальным и элегантным способом.
- Приложение должно быть многофункциональным, оставаясь пригодным для использования как начинающими, так и опытными пользователями.
- Приложение должно быть понятным для пользователей, которые получили доступ к одной и той же информации другими путями, например через веб-сайт.
- Основные функции должны быть легко доступными.
- Приложение должно иметь общий внешний вид с другими родными приложениями на телефоне в соответствии со стандартами целевой платформы и стилями.
- Приложение должно быть стабильным, масштабируемым, удобным и адаптивным.
- Приложение должно использовать возможности платформы, делая взаимодействие с пользователем более привлекательным.

Проектирование приложений для платформы

Android-требования:

1. **Размер и разрешение экрана.**
   Чтобы классифицировать устройства по типу экрана, платформа Android определяет две характеристики для каждого устройства: размер экрана (физические размеры экрана) и разрешение экрана (физическая плотность пикселей на экране, или dpi (количество точек на дюйм). Чтобы упростить все типы конфигураций экрана, система Android обобщает их на группы, которые упрощают их настройку. При разработке приложения проектировщик должен определить подходящие размер и разрешение экрана.

По умолчанию приложение совместимо со всеми размерами и разрешениями экрана, потому что система Android делает соответствующие корректировки для компоновки пользовательского интерфейса и ресурсов изображения. Однако для определенных значений разрешения и размеров экрана необходимо создавать специализированные компоновки и предоставлять специализированные изображения, используя альтернативные ресурсы компоновки и точно декларируя в своем мани фесте, какие размеры экрана поддерживаются вашим приложением.

2. **Конфигурации ввода.**
   Многие устройства предоставляют различные типы пользовательских механизмов ввода, такие как аппаратная клавиатура, трекбол или пятипозиционная навигационная панель. Если вашему приложению требуется определенный вид аппаратного обеспечения для ввода данных, необходимо объявить его в файле AndroidManifest.xml и помнить, что на сайте Google Play Store ваше приложение не будет отображаться на устройствах, которым не хватает этой функции. Однако для приложения редко требуется определенная конфигурация ввода.

3. **Особенности устройства**.
   Существует множество аппаратных и программных функций, которые могут или не могут существовать на данном устройстве на базе Android, таком как камера, датчик освещенности, механизм Bluetooth, определенная версия OpenGL или качество сенсорного экрана. Вы никогда не должны предполагать, что определенная функция доступна на всех устройствах на базе Android (кроме доступности стандартной библиотеки Android).

4. **Каналы данных и форматы каналов.**
   Нецелесообразно напрямую взаимодействовать с любым источником данных, предоставленным третьей стороной. Например, для прямой связи вашего мобильного приложения с базой данных на вашем сервере было бы неплохо использовать драйвер JDBC третьего типа. Обычным подходом

является сбор данных из нескольких источников в потенциально нескольких форматах посредством промежуточного программного обеспечения, которое затем передает данные в приложение через ряд API-интерфейсов веб-служб RESTful в виде потоков данных JSON.

Как правило, данные предоставляются в таких форматах, как XML, SOAP, или в виде какого-либо другого представления, связанного с XML. Форматы наподобие SOAP являются тяжеловесными, и поэтому передача данных с вспомогательных серверов в этом формате значительно увеличивает время разработки, поскольку ответственность за преобразование этих данных во что-то более управляемое возлагается на приложение или на объект, находящийся на сервере промежуточного программ много обеспечения.

5. Уменьшение объема исходных данных с помощью серверного промежуточного программного обеспечения также помогает разорвать зависимость между приложением и данными. Такая зависимость имеет тот недостаток, что если по какой-либо причине характер данных изменится или они не смогут быть восстановлены, то приложение может выйти из строя и стать непригодным, и для учета таких изменений может потребоваться его повторная публикация. Уменьшение объема данных с помощью серверного промежуточного программного обеспечения гарантирует, что приложение будет продолжать работать, хотя, возможно, ограниченным образом, независимо от того, существуют ли исходные данные. Связь между приложением и предварительно обработанными данными сохраняется.

6. Усовершенствованное приложение для Android имеет два типа меню, предоставляемые платформой Android, в зависимости от обстоятельств.

- Командное меню содержит основные функции, которые применяются глобально к текущей активности или запускают связанную с ним активность. Командное меню обычно появляется, когда пользователь нажимает аппаратную кнопку, часто обозначаемую Menu, или программную кнопку меню на панели действий Action Bar (вертикальный стек из трех точек).

- Контекстное меню содержит вспомогательные функции для выбранного в данный момент элемента. Контекстное меню обычно вызывается пользователем, выполняющим длительное нажатие (нажатие и удержание) на элементе. Как и в меню команд, выбранная операция может выполняться либо в текущей, либо в другой активности. Контекстное меню предназначено для любых команд, которые применяются к текущему выбору. Команды в контекстном меню, которые появляются при длительном нажатии на элемент, должны дублироваться в активности, которая запускается при обычном нажатии на этот элемент.

Материальные компоненты — это интерактивные строительные блоки для создания пользовательского интерфейса.

- Панели приложений: внизу
- Панели приложений: вверху
- Нижняя навигация
- Кнопки
- Кнопки: плавающая кнопка действия
- Карты
- Флажки
- Фишки
- Выбор даты
- Диалоги
- Разделители
- Меню
- Ящик навигации
- Навигационный рельс
- Индикаторы прогресса
- Радио-кнопки
- Листы: нижний
- Слайдеры
- Переключатели
- Вкладки
- Текстовые поля
- Сборщики времени

iOS - требования:

1. **Однотипность моделей iPhone.**
   Приложение должно быть адаптировано под разные размеры экранов и при этом выглядеть безупречно. Пользователю неприятно взаимодействовать с приложением, в котором один элемент «наплывает» на другой, текст обрезается, а изображения занимают почти всё экранное место.

2. **Новые версии операционной системы и поддержка.**
   Новые версии операционной системы iOS выходят раз в год. Каждое обновление — это тренды, полезные технологии и увеличение производительности. Люди охотно обновляют операционку и ожидают, что у приложений появится новая функциональность. Но «автоматически» этого не происходит. Мобильные приложения нужно адаптировать под новые версии.

Это помогает:

а) избегать конфликтов между системой и приложением, ведь некоторые функции могут перестать работать;

б) оправдывать пользовательские ожидания.

3. **Совместимость с айпадами.**
   Надо изначально решить, предусмотрено ли открытие вашего приложения на разных платформах (iOS, iPadOS, macOS). Если да, то при разработке и проектировании нужно учесть массу особенностей, например компоновку интерфейса приложений на размерах экрана 960×640 пикселей, а не только на размерах iPhone.

Без отдельной проработки экранов под другие девайсы приложение на них будет открываться некорректно: возможно, элементы будут накладываться друг на друга, а текст переноситься неправильно.

Вот основные принципы iOS-интерфейса, которые нужно соблюдать при разработке приложений:

- Эстетическая целостность — дизайн не противоречит предназначению приложения. Мы не можем добавить в приложение, которое выполняет серьёзную задачу, милую анимацию: Apple считает это неуместным.
- Последовательность — навигация в приложении очевидна: используются известные значки, стандартные стили текста и единая терминология, а приложение реагирует на действия людей так, как они ожидают.
- Прямое действие — приложение однозначно реагирует на повороты устройства и жесты пользователя.
- Обратная связь — приложение даёт пользователю обратную связь в ответ на его действия: интерактивные элементы выделяются при нажатии, индикаторы прогресса сообщают о состоянии длительных операций, а анимация и звук помогают прояснить результаты действий.
- Аналогичность — люди быстрее понимают приложение, если могут взаимодействовать с ним по аналогии с физическим опытом: «смахивать» экраны, «перетаскивать» элементы, «прокручивать» страницы.
- Пользовательский контроль — В iOS все контролируют люди, а не искусственный интеллект. Приложение может предупредить или уведомить об ошибке, но никогда не будет принимать решения за пользователя.

• Дизайн для iOS

Отображать. iPhone имеет дисплей среднего размера с высоким разрешением. Эргономика. Люди обычно держат свой iPhone в одной или обеих руках, когда взаимодействуют с ним, переключаясь между альбомной и портретной ориентацией по мере необходимости.

Входы. Жесты Multi-Touch , экранная клавиатура и голосовое управление позволяют выполнять действия и выполнять важные задачи, находясь в пути. Кроме того, люди часто хотят, чтобы приложения использовали их местоположение и данные, поступающие от акселерометра и гироскопа устройства , и они также могут участвовать в пространственных взаимодействиях .

Взаимодействие приложений. Иногда люди тратят всего минуту или две на проверку событий или обновлений в социальных сетях, отслеживание данных или отправку сообщений. В другое время люди могут провести час или больше, просматривая веб-страницы, играя в игры или наслаждаясь мультимедиа. Люди обычно имеют несколько приложений, открытых одновременно, и им нравится часто переключаться между ними.

Особенности системы. iOS предоставляет несколько функций, которые помогают людям взаимодействовать с системой и их приложениями привычными и последовательными способами.

- Виджеты
- Быстрые действия на главном экране
- Прожектор
- Ярлыки
- Просмотры активности

- ## Дизайн для iPadOS

Отображать. iPad имеет большой дисплей с высоким разрешением.

Эргономика. Люди часто держат свой iPad во время его использования, но они также могут положить его на поверхность или поставить на подставку. Размещение устройства по-разному может изменить расстояние просмотра, хотя люди обычно находятся в пределах 3 футов от устройства, когда они взаимодействуют с ним.

Входы. Люди могут взаимодействовать с iPad, используя жесты Multi-Touch и экранную клавиатуру , подключенную клавиатуру или указывающее устройство , Apple Pencil или голос , и они часто сочетают несколько режимов ввода.

Взаимодействие приложений. Иногда люди выполняют несколько быстрых действий на своем iPad. В другое время они проводят часы, погруженные в игры, мультимедиа, создание контента или задачи по повышению производительности. Люди часто имеют несколько приложений, открытых одновременно, и им нравится просматривать более одного приложения на экране одновременно и использовать возможности между приложениями, такие как перетаскивание.

Особенности системы. iPadOS предоставляет несколько функций, которые помогают людям взаимодействовать с системой и их приложениями привычными и последовательными способами.

- Многозадачность
- Виджеты
- Перетащите

- ## Проектирование для macOS

Приступая к разработке приложения или игры для macOS, начните с понимания основных характеристик устройства и шаблонов, отличающих macOS. Использование этих характеристик и шаблонов для информирования ваших дизайнерских решений может помочь вам создать приложение или игру, которые оценят пользователи Mac.

Отображать. Mac обычно имеет большой дисплей с высоким разрешением, и люди могут расширить свое рабочее пространство, подключив дополнительные дисплеи, включая свой iPad.

Эргономика. Люди обычно используют Mac, когда они стационарны, часто кладя устройство на стол или стол. В типичном случае использования расстояние просмотра может варьироваться от 1 до 3 футов.

Входы. Люди ожидают ввода данных и управления интерфейсом, используя любую комбинацию режимов ввода, таких как физические клавиатуры , указывающие устройства , игровые контроллеры и голос .

Взаимодействие приложений. Взаимодействие может длиться от нескольких минут выполнения некоторых быстрых задач до нескольких часов глубокой концентрации. У людей часто одновременно открыто несколько приложений, и они ожидают плавного перехода между активным и неактивным состояниями при переключении с одного приложения на другое.

Особенности системы. macOS предоставляет несколько функций, которые помогают людям взаимодействовать с системой и их приложениями привычными и последовательными способами.

- Строка меню
- Искатель
- Управление полетами
- Док

- ## Дизайн для watchOS

Приступая к разработке приложения для Apple Watch, начните с понимания следующих основных характеристик устройства и шаблонов, отличающих работу watchOS. Использование этих характеристик и шаблонов для информирования ваших дизайнерских решений может помочь вам создать приложение, которое оценят пользователи Apple Watch.

Отображать. Небольшой дисплей Apple Watch помещается на запястье, обеспечивая легкое чтение и высокое разрешение.

Эргономика. Поскольку люди носят Apple Watch, они обычно находятся на расстоянии не более фута от дисплея, когда поднимают запястье, чтобы посмотреть на него, и взаимодействуют с устройством противоположной рукой. Кроме того, дисплей Always On позволяет людям просматривать информацию на циферблате, когда они опускают запястье.

Входы. Использование стандартных жестов Multi-Touch , таких как касание, смахивание и перетаскивание, позволяет пользователям управлять своим взаимодействием, даже когда они находятся в движении. Поворот цифровой короны придает дополнительную точность и обратную связь прокручиваемым интерфейсам, нажатие кнопки « Действие» инициирует важное действие, не глядя на экран, а использование ярлыков Siri может помочь людям быстро и легко

выполнять свои рутинные задачи. Люди также ценят использование данных, которые могут предоставить функции устройства, такие как GPS, датчики кислорода в крови и функции сердца, высотомер, акселерометр и гироскоп.

Взаимодействие приложений. Люди часто поглядывают на дисплей Always On много раз в течение дня, совершая целенаправленные взаимодействия с приложением, каждое из которых может длиться менее минуты. Люди часто используют связанные с приложением watchOS возможности, такие как расширения, уведомления и взаимодействие с Siri, больше, чем само приложение.

Особенности системы. watchOS предоставляет несколько функций, которые помогают людям взаимодействовать с системой и их приложениями знакомым и последовательным образом.

- Осложнения
- Уведомления
- Всегда включен
- Лица

Основные методы для управления приложением

- Кнопка действия
- Apple Pencil и Scribble
- Цифровая корона
- Фокус и выбор
- Игровые контроллеры
- Гироскоп и акселерометр
- Клавиатуры
- Указывая устройства
- Пульты
- Пространственные взаимодействия
- Сенсорная панель
- Жесты сенсорного экрана

Основные шаблоны приложения

- Доступ к личным данным
- Сотрудничество и обмен
- Перетащите
- Ввод данных
- Обратная связь
- Управление файлами
- Полноэкранный режим

- Запуск
- Приложения для просмотра в реальном времени
- Загрузка
- Управление учетными записями
- Управление уведомлениями
- Модальность
- Многозадачность
- Оказание помощи
- Онбординг
- Воспроизведение аудио
- Игра тактильных ощущений
- Воспроизведение видео
- Печать
- Рейтинги и обзоры
- Идет поиск
- Настройки
- Отменить и повторить
- Тренировки

Компоненты приложения

Планировка и организация

- Коробки
- Коллекции
- Просмотры столбцов
- Контроль раскрытия информации
- Этикетки
- Списки и таблицы
- Блокировки
- Контурные представления
- Разделить просмотры
- Просмотры вкладок

Меню и действия

- Просмотры активности
- Кнопки
- Контекстные меню
- Док-меню

- Редактировать меню
- Меню
- Всплывающие кнопки
- Выпадающие кнопки
- Панели инструментов

Навигация и поиск

- Панели навигации
- Элементы управления путем
- Поля поиска
- Боковые панели
- Панели вкладок
- Поля токенов

Презентация

- Листы действий
- Оповещения
- Элементы управления страницей
- Панели
- Поповеры (всплывающие окна)
- Просмотры прокрутки
- Листы
- Окна

Выбор и ввод

- Цветовые колодцы
- Комбинированные поля
- Просмотры ввода цифр
- Лунки изображения
- Экранные клавиатуры
- Сборщики
- Сегментированные элементы управления
- Слайдеры
- Степперы
- Переключает

Статус

- Кольца активности
- Индикаторы уровня
- Индикаторы прогресса

Системный опыт

- Осложнения
- Быстрые действия на главном экране
- Строка меню
- Уведомления
- Строки состояния
- Верхняя полка
- Виджеты

Устройства ввода

- Кнопка действия
- Apple Pencil и Scribble
- Цифровая корона
- Фокус и выбор
- Игровые контроллеры
- Гироскоп и акселерометр
- Клавиатуры
- Указывая устройства
- Пульты
- Пространственные взаимодействия
- Сенсорная панель
- Жесты сенсорного экрана

Технологии

- AirPlay
- Всегда включен
- Клипы приложений
- Apple Pay
- Дополненная реальность
- CareKit
- CarPlay
- Игровой центр

- HealthKit
- HomeKit
- iCloud
- Покупки в приложении
- Живые фотографии
- Mac Катализатор
- Машинное обучение
- Карты
- Сообщения для бизнеса
- NFC
- Редактирование фотографий
- Исследовательский комплект
- SharePlay
- ShazamKit
- Войти через Apple
- Сири
- Нажмите, чтобы оплатить на iPhone
- Бумажник

Windows Phone - требования:

Возможности программного обеспечения: приложение получает доступ к идентификатору телефона, камере, картам, медиа-библиотекам, сетям, связи ближнего действия (NFC), push-уведомлениям, датчикам, кошельку, веб-браузеру и другим возможностям программного обеспечения. Windows Phone предоставляет доступ к этим возможностям, и перед покупкой приложения пользователям сообщается, какие возможности используются.

Требования к оборудованию: функции телефона, которые приложение должно иметь для правильной работы. Операционная система использует список, чтобы запретить установку приложений на телефоны, которые не соответствуют этим требованиям.

Функциональные возможности: приложение запрашивает больше памяти во время выполнения, если она доступна.
Приложение должно включать соответствующие возможности и требования к оборудованию. Приложение не помечает возможности и требования, которые ему не нужны. Пользователи видят возможности и требования приложения при установке приложения и могут решить не продолжать установку, если увидят, что приложение использует возможности, которые они не хотят включать, или которые не связаны с назначением приложения.

Параметры нефункционального тестирования

Non Functional Testing Parameters

1) Безопасность:

Параметр определяет, как система защищена от преднамеренных и внезапных атак из внутренних и внешних источников. Это проверено с помощью Security Testing .

2) Надежность:

Степень, в которой любая программная система непрерывно выполняет указанные функции без сбоев. Это проверено проверкой надежности

3) Живучесть:

Параметр проверяет, что система программного обеспечения продолжает функционировать, и восстанавливается в случае сбоя системы. Это проверено тестированием восстановления

4) Наличие:

Параметр определяет степень, в которой пользователь может зависеть от системы во время ее работы. Это проверено тестированием стабильности.

5) Удобство использования:

Легкость, с которой пользователь может учиться, работать, готовить входные и выходные данные посредством взаимодействия с системой. Это проверено юзабилити-тестированием

6) Масштабируемость:

Термин относится к степени, в которой любое программное приложение может расширить свои вычислительные мощности, чтобы удовлетворить увеличение спроса. Это проверено Scalability Testing

7) Совместимость:

Этот нефункциональный параметр проверяет взаимодействие программной системы с другими программными системами. Это проверено тестированием совместимости

8) Эффективность:

Степень, в которой любая программная система может обрабатывать емкость, количество и время отклика.

9) Гибкость:

Термин относится к легкости, с которой приложение может работать в различных аппаратных и программных конфигурациях. Как минимум оперативной памяти, требования к процессору.

10) Портативность:

Гибкость программного обеспечения для перехода от его текущей аппаратной или программной среды.

11) Возможность повторного использования:

Это относится к части системы программного обеспечения, которая может быть преобразована для использования в другом приложении.

Основные нефункциональные типы тестирования

1. **Тестирование прерываний**

Прерывание — это смена фокуса с одного приложения на другое, которое требует немедленной реакции.

Пример: оформляешь перевод в мобильном банке, появляется поп-ап входящего звонка. Поп-ап вызова создает прерывание. После завершения прерывания прерванный сценарий должен быть продолжен с момента остановки.

Почему важно тестировать прерывания:
Прерывания часто становятся причиной того, что пользователь не может завершить сценарий, прерванный другим приложением. И часто это причина блокеров.

Примеры:
Аппаратные

- Разрядка аккумулятора
- Низкий заряд аккумулятора
- Перезагрузка
- Отключение
- Блокировка-разблокировка
- Горячая установка сим/флеш-карты
- Подключение/отключение зарядки
- Нехватка памяти

Сетевые

- Потеря/восстановление сети
- Смена типа сети
- Входящий, исходящий звонок
- Удаленное управление. Например, с помощью Apple Watch можно запустить таймер фото, соответственно фокус переключается на приложение "Камера"
- Передача и получение информации с помощью периферийных устройств
  Например, во время работы с приложением на iOS по AirDrop прилетает запрос принять фотографию

Системные

- Пуши
  От других приложений, системные (будильник, таймер, напоминание)
- Диалоговые окна
  Предложение обновить ОС
- Смена фокуса на другое приложение
- Смена ориентации экрана

- Вызов нативных приложений
  В приложении тапаешь на ссылку - открывается встроенный браузер внутри приложения. Или кликаешь написать на эл.почту - открывается нативная приложение "почта"

2. Тестирование безопасности и конфиденциальности

Тестирование безопасности затрагивает две широкие области, вызывающие озабоченность:

- Конфиденциальность, целостность и доступность программного обеспечения и данных, обрабатываемых программным обеспечением. В эту область входят все функции и функциональные возможности, предназначенные для устранения угроз, как описано в модели угроз.

- Отсутствие проблем, которые могут привести к уязвимостям в системе безопасности. Например, переполнение буфера в коде, который анализирует данные, может быть использовано способами, которые имеют последствия для безопасности.

Начало тестирования безопасности происходи сразу после написания кода. Этот этап тестирования требует одного полного прохождения теста после этапа проверки, поскольку потенциальные проблемы и уязвимости могут измениться во время разработки.

Тестирование безопасности — это метод тестирования, позволяющий определить, защищает ли информационная система данные и поддерживает ли функциональность должным образом. Тестирование безопасности не гарантирует полной безопасности системы, но важно включить тестирование безопасности в процесс тестирования.

Тестирование безопасности использует следующие шесть мер для обеспечения защищенной среды:

- Конфиденциальность − защищает от раскрытия информации непреднамеренным получателям.
- Целостность — Это позволяет передавать точную и правильную желаемую информацию от отправителей предполагаемым получателям.
- Аутентификация − Оно проверяет и подтверждает личность пользователя.
- Авторизация − Он определяет права доступа к пользователям и ресурсам.
- Доступность — Это гарантирует готовность информации по требованию.
- Безотказность - гарантирует отсутствие отказа со стороны отправителя или получателя в отправке или получении сообщения.

Примеры:

- Войдите в веб-приложение, используя действительные учетные данные.
- Выйдите из веб-приложения.
- Нажмите кнопку "НАЗАД" в браузере.
- Проверьте, попросят ли вас снова войти в систему или сможете ли вы снова вернуться на страницу входа в систему.

3. Тестирование зависимости от сетей и каналов связей.

Основные виды соединения с Интернетом:
- Передача данных по сотовой связи: 3G, 4G, 5G,
- Wi-Fi
- Mi-Fi - когда точка раздает интернет полученный по сотовой связи

Тестирование позволяет определить, реализованы ли сетевые устройства и сети согласно спецификациям и обеспечивает ли сеть требуемые уровни качества обслуживания.

Основными причинами разработки технологий мобильной связи являются стремление увеличить пропускную способность сотовых сетей в условиях лавинообразного роста объемов трафика данных и необходимость обеспечения широкого набора отраслей, включая промышленное производство, автомобильный транспорт и сельское хозяйство, новыми передовыми возможностями мобильной связи, среди которых — сверхнизкая задержка передачи пакетов и великолепная масштабируемость сети.

Тестирование сетей 5G представляет собой непростую задачу, поскольку в этих сетях объединено множество разных сетевых технологий и решений, среди которых есть ряд новых (для рынка сотовых систем) решений, включая связь в миллиметровом диапазоне длин волн с формированием узких лучей, massive MIMO, виртуализированную инфраструктуру. Эти решения придется испытывать как по отдельности, так и во множестве различных комбинаций для гарантии их совместимости.

Примеры:
- Тестирование при подключеном Wi-Fi, 4G/3G/E/etc
- Разрыв и восстановление сети
- Переключение с одного типа сети к другому
- Оффлайн
- Смена, отключение геопозиции

4. Тестирование установки.

Тестирование установки направленно на проверку успешной инсталляции и настройки, а также обновления или удаления программного обеспечения.

В настоящий момент наиболее распространена установка ПО при помощи инсталляторов (специальных программ, которые сами по себе так же требуют надлежащего тестирования.

В распределенных системах, где приложение разворачивается на уже работающем окружении, простого набора инструкций может быть мало. Для этого, зачастую, пишется план установки (Deployment Plan), включающий не только шаги по инсталляции приложения, но и шаги отката (roll-back) к предыдущей версии, в случае неудачи. Сам по себе план установки также должен пройти процедуру тестирования для избежания проблем при выдаче в реальную эксплуатацию. Особенно это актуально, если установка выполняется на системы, где каждая минута простоя - это потеря репутации и большого количества средств, например: банки, финансовые компании или даже баннерные сети. Поэтому тестирование установки можно назвать одной из важнейших задач по обеспечению качества программного обеспечения.

Именно такой комплексный подход с написанием планов, пошаговой проверкой установки и отката инсталляции, полноправно можно назвать тестированием установки или Installation Testing.

Примеры:

- Установка
- Обновление
- Переустановка
- Удаление

4. Тестирование совместимости с функциями телефона.

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

Примеры:

- Жесты/смахивания
- Расширение/ориентация/размер экрана
- Видео/фото
- GPS
- Отображение на заблокированном экране
- Добавление в календарь/wallet/локальные папки

5. 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.

Важно проводить тестирование производительности последовательно с измерением конкретных показателей. Такой подход даст полное представление о продуктивности системы и укажет на области для дальнейшей доработки

приложения с целью приведения в соответствие с обозначенными бизнес-требованиями.

Примеры:

- Время загрузки приложения
- Обработка запросов
- Кэширование данных
- Потребление ресурсов приложением (например расход заряда батареи)

6. Тестирование локализации.

Тестирование локализации включает тестирование приложения с локализованными строками, изображениями и рабочими процессами для определенного региона.

Примеры:

- Корректное отображение форматов дат (ГОД — МЕСЯЦ — ДЕНЬ или ДЕНЬ —МЕСЯЦ — ГОД.)
- Корректное отображение времени в зависимости от часового пояса
- Все элементы в приложении переведены на соответствующий язык

9. Тестирование конфигурации.

Тестирование конфигурации — это подход к тестированию программного обеспечения, при котором программное приложение тестируется с использованием различных программных и аппаратных комбинаций с целью анализа функциональных требований и определения наилучших конфигураций, в которых программное приложение работает без ошибок или недостатков.

Это тестирование представляет собой метод оценки производительности программы, системы или приложения с учетом различных системных настроек.

Примеры:

- протестировать комбинацию клиента
- сервера
- базы данных