**Android Studio**

**Что такое Android Studio?**

Android Studio – это официальная интегрированная среда разработки (IDE) для платформы Android.

Она предоставляет полный набор инструментов, необходимых для создания, тестирования и отладки приложений для Android. По сути, это “рабочее место” для разработчиков, где они пишут код, компилируют его, запускают приложения на эмуляторе или реальных устройствах, а также имеют доступ к различным вспомогательным инструментам.

**Для каких задач тестировщик может использовать свою работу Android Studio?**

Как QA Manual Tester, я могу эффективно использовать Android Studio для выполнения ряда важных задач, которые напрямую влияют на качество тестируемого приложения:

1. **Установка и запуск приложений:**
   * **Установка APK/APPX файлов:** Я могу легко установить сборку приложения (APK для Android) на эмулятор или реальное устройство, используя Android Studio. Это быстрее и надежнее, чем ручная установка через файловый менеджер.
   * **Запуск на эмуляторах:** Android Studio позволяет создавать и настраивать различные эмуляторы с разными версиями Android, размерами экрана и аппаратными характеристиками. Это критически важно для тестирования на широком спектре устройств без необходимости иметь их физически. Я могу протестировать, как приложение ведет себя на разных платформах и разрешениях.
2. **Просмотр логов (Logcat):**
   * **Отслеживание ошибок и исключений:** Logcat – это мощный инструмент в Android Studio, который отображает системные сообщения и сообщения, сгенерированные самим приложением. Я могу отслеживать ошибки, исключения, предупреждения и информационные сообщения, которые возникают во время работы приложения. Это помогает мне находить причину проблем, которые не всегда очевидны пользователю, и предоставлять более детальные отчеты для разработчиков.
   * **Анализ поведения приложения:** Я могу добавлять свои собственные логи в код (при наличии такой возможности или с помощью разработчиков) для отслеживания конкретных сценариев и ветвей выполнения.
3. **Инспекция UI-элементов (Layout Inspector):**
   * **Проверка структуры интерфейса:** Layout Inspector позволяет визуально исследовать иерархию UI-элементов на экране. Я могу посмотреть, какие View (кнопки, текстовые поля, изображения и т.д.) присутствуют, как они расположены, какие у них размеры, отступы и выравнивание.
   * **Поиск проблем с версткой:** Это помогает выявлять проблемы с корректным отображением интерфейса на разных устройствах, несоответствия макетам, а также находить скрытые или перекрывающиеся элементы.
4. **Отладка (Debugging) – на базовом уровне:**
   * **Установка точек остановок (Breakpoints):** Хотя я не пишу код, я могу, в некоторых случаях, с помощью разработчиков, установить точки остановки в определенных участках кода. Это позволит мне остановить выполнение приложения в нужный момент и изучить состояние переменных, что помогает глубже понять, почему происходит тот или иной баг.
   * **Пошаговое выполнение:** Я могу проследить выполнение кода шаг за шагом, чтобы понять логику работы приложения и то, как оно реагирует на мои действия.
5. **Просмотр и управление данными приложения:**
   * **Database Inspector:** Если приложение использует базы данных (например, SQLite), Database Inspector позволяет просматривать содержимое базы данных, выполнять запросы и даже вносить изменения. Это полезно для проверки корректности сохранения и получения данных.
   * **File Explorer:** Я могу просматривать файловую систему устройства или эмулятора, чтобы найти файлы, которые создает или использует приложение (например, файлы настроек, кэш, изображения).
6. **Просмотр трафика сети (Network Inspector):**
   * **Мониторинг сетевых запросов:** Network Inspector позволяет отслеживать все сетевые запросы, которые отправляет приложение, а также ответы от сервера. Я могу видеть URL, методы запросов, заголовки, тело запроса и ответа, а также время выполнения.
   * **Выявление проблем с API:** Это незаменимо при тестировании интеграции с бэкендом, для поиска ошибок в API-запросах, некорректных ответов, проблем с авторизацией и таймаутов.
7. **Работа с профилировщиками (CPU, Memory, Network profilers):**
   * **Выявление узких мест:** Профилировщики помогают обнаружить потенциальные проблемы с производительностью – например, чрезмерное потребление памяти, высокую загрузку процессора или неэффективное использование сети. Хотя детальный анализ требует знания кода, я могу заметить аномальные всплески потребления ресурсов и сообщить о них.
8. **Проверка работоспособности различных компонентов:**
   * **Intent Filter Tester:** Позволяет проверить, как приложение реагирует на различные типы входящих интентов (сообщений между компонентами Android), что важно для тестирования глубоких ссылок и интеграции с другими приложениями.

Таким образом, Android Studio – это не просто инструмент для разработчиков, но и мощный помощник для QA-специалиста, позволяющий проводить более глубокое, точное и эффективное тестирование мобильных приложений. Это позволяет мне выходить за рамки “черного ящика” и более осмысленно подходить к поиску и документированию дефектов.

**Как мы можем читать логи в Android Studio?**

Чтение логов в Android Studio – это одна из ключевых навыков QA-тестировщика, так как логи являются окном в происходящее внутри приложения. Вот как мы можем это делать:

**Основной инструмент: Logcat**

Logcat – это главная панель в Android Studio, предназначенная для просмотра логов. Она показывает поток сообщений от операционной системы Android и самого приложения.

**Как открыть Logcat:**

1. **Через меню:** View -> Tool Windows -> Logcat
2. **С помощью сочетания клавиш:** Обычно это Alt + 6 (на Windows/Linux) или Cmd + 6 (на macOS).
3. **Кнопка внизу окна:** в нижней части окна Android Studio есть панель инструментов “Event Log” или “Debug”, и там часто есть иконка, похожая на свиток или чат, которая открывает Logcat.

**Основные элементы панели Logcat:**

* **Устройство/Эмулятор:** Выпадающий список вверху панели Logcat. Здесь вы выбираете, с какого устройства или эмулятора вы хотите получать логи. Крайне важно выбрать правильное устройство, если у вас их несколько подключено.
* **Процесс приложения:** Ниже устройства вы можете выбрать конкретный процесс вашего приложения. Это удобно, когда на устройстве запущено много приложений, чтобы отфильтровать логи только для нужного.
* **Поиск (Search):** Поле для ввода ключевых слов, по которым вы хотите отфильтровать логи. Это очень мощный инструмент, чтобы найти конкретные сообщения или ошибки.
* **Фильтры (Filters):**
  + **Уровень логов (Log Level):** Выпадающий список, позволяющий фильтровать сообщения по их важности:
    - **Verbose (V):** Наиболее подробный уровень, для отладочной информации.
    - **Debug (D):** Отладочная информация, полезная разработчикам.
    - **Info (I):** Информационные сообщения, показывающие нормальное выполнение программы.
    - **Warn (W):** Предупреждения о потенциальных проблемах, которые не являются критическими.
    - **Error (E):** Сообщения об ошибках, которые привели к сбою или некорректному поведению.
    - **Assert (A):** Критические ошибки, которые обычно приводят к аварийному завершению приложения.
  + **Создание собственных фильтров:** Вы можете создавать пользовательские фильтры, комбинируя ключевые слова и уровни логов. Нажмите на + рядом с выпадающим списком фильтров. Это очень полезно для регулярных тестов.
* **Список сообщений:** Основная область, где отображаются сами логи. Каждая строка обычно содержит:
  + **Временную метку (Timestamp):** Когда было сгенерировано сообщение.
  + **PID (Process ID):** Идентификатор процесса.
  + **TID (Thread ID):** Идентификатор потока.
  + **Уровень лога:** Короткое обозначение (V, D, I, W, E, A).
  + **Тег (Tag):** Имя класса или компонента, который сгенерировал лог.
  + **Сообщение:** Собственно, текст лога.
* **Кнопки управления:**
  + **Очистить (Clear Logcat):** Удаляет все текущие логи из панели.
  + **Запись логов (Save Logcat):** Позволяет сохранить текущий лог в файл (текстовый или специальный формат .txt).
  + **Автоскроллинг (Auto Scroll to End):** Если включен, новые сообщения будут автоматически добавляться внизу.

**Как использовать Logcat при тестировании:**

1. **Запуск приложения:** После запуска приложения на эмуляторе или устройстве, откройте Logcat.
2. **Фильтрация:**
   * Сначала выберите нужное устройство и процесс.
   * Для начала можно установить уровень логов на Info или Debug, чтобы видеть более полную картину.
   * Если вы ищете конкретный баг, используйте поле поиска с ключевыми словами, которые могут быть связаны с проблемой (например, имя функции, сообщение об ошибке, название экрана).
3. **Выполнение действий:** Выполняйте действия в приложении, которые вы тестируете. Обращайте внимание на сообщения, появляющиеся в Logcat, особенно если вы видите сообщения с уровнем Error или Warn.
4. **Анализ сообщений:**
   * **Сообщения об ошибках (Error):** Если вы видите сообщение с E, внимательно прочитайте тег, сообщение и, если возможно, стектрейс (последовательность вызовов функций, приведших к ошибке). Это даст вам понять, где именно произошла проблема.
   * **Сообщения о предупреждениях (Warn):** Могут указывать на потенциальные проблемы, которые могут проявиться в будущем или при определенных условиях.
   * **Информационные сообщения (Info):** Полезны для отслеживания нормального течения событий. Например, “User logged in successfully”, “Data saved to database”.
5. **Сохранение логов:** Если вы обнаружили интересный или критический баг, обязательно сохраните логи. Это драгоценная информация для разработчиков. Вы можете сохранить весь поток логов или отфильтровать их по нужным ключевым словам перед сохранением.
6. **Создание пользовательских фильтров:** Если вы часто тестируете определенный функционал, который генерирует специфические логи, создайте фильтр с нужными ключевыми словами и уровнем важности. Это сэкономит вам много времени.

**Пример использования:**

Предположим, при нажатии на кнопку “Сохранить” в приложении ничего не происходит, и нет видимых ошибок на экране.

1. Я открываю Logcat, выбираю нужное устройство и процесс.
2. Я устанавливаю уровень логов на Debug и Info.
3. Я ищу в поле поиска по ключевым словам, связанным с сохранением (например, “save”, “save\_data”, “database”, “save\_button\_click”).
4. Я выполняю действие “нажать кнопку Сохранить”.
5. Я просматриваю логи, появляющиеся после нажатия. Возможно, я увижу сообщение вида: E/Database: Error inserting record: null value или W/Network: Failed to send data to server, response code 404. Это сразу даст мне направление для дальнейшего расследования.

Понимание Logcat – это фундамент для эффективного QA-тестирования Android-приложений.

Начало формы

Конец формы