

Основные инструменты тестирования: эмуляторы, симуляторы и мобильные фермы

Процесс тестирования мобильных приложений



Оглавление

| | |
|---|----|
| На этом уроке | 3 |
| Термины, используемые в лекции | 3 |
| Эмуляторы | 4 |
| Симуляторы | 4 |
| Отличие симуляторов и эмуляторов | 5 |
| Мобильные фермы | 6 |
| AWS Device Farm | 6 |
| Начало использования | 8 |
| Запуск автоматических тестов | 12 |
| Samsung Remote Test Lab | 16 |
| Firebase Test Lab | 27 |
| Microsoft Visual Studio App Center Test | 29 |
| Выводы | 30 |
| Домашнее задание | 30 |
| Используемая литература | 30 |

На этом уроке

Итак, сегодня мы продолжим изучать основные инструменты и сервисы для тестирования мобильных приложений.

На уроке мы с вами разберём:

- Что такое эмуляторы
- Что такое симуляторы
- В чём разница между эмуляторами и симуляторами
- Что такое мобильные фермы
- Примеры работы с популярными мобильными фермами

В зависимости от размера команды, компании и продукта мы можем проводить тестирования на:

- Реальных девайсах
- Эмуляторах и симуляторах
- Мобильных фермах

Тестирование на реальных устройствах и выбор необходимого набора устройств для тестирования мы уже разбирали. Напомню: какие бы дополнительные инструменты и сервисы ни использовались, тестирование хотя бы на минимальном наборе реальных устройств необходимо. Ведь в итоге пользователи нашего приложения будут запускать его именно на реальных устройствах.

Термины, используемые в лекции

Эмулятор — это программное обеспечение, которое имитирует аппаратное и программное обеспечение целевого устройства на вашем компьютере.

Симулятор — это программное обеспечение, которое помогает вашему компьютеру запускать определённые программы, созданные для другой операционной системы.

Ферма устройств — это группа, состоящая из множества устройств (телефонов, компьютеров или любого другого оборудования), арендованных или находящихся в собственности.

Эмуляторы

Эмулятор — это программное обеспечение, которое имитирует аппаратное и программное обеспечение целевого устройства на вашем компьютере. Они делают это путём преобразования ISA (архитектуры набора инструкций) целевого устройства в ту, которая используется компьютером, который мы используем для проведения тестирования.

Архитектура набора команд (англ. instruction set architecture, ISA) — часть архитектуры компьютера, определяющая программируемую часть ядра микропроцессора.

Более простыми словами, ISA — это набор инструкций, написанных на машинном языке каждым из семейств процессоров. Они используются для создания собственной конфигурации устройства, отражающей функциональность и поведение устройства. Когда мы пользуемся разными версиями компьютеров, у каждого из устройств есть свой код, который отвечает за то, как работает устройство с определёнными характеристиками. Например, есть **Наборы команд** на базе архитектуры Intel, **Наборы команд** на базе архитектуры AMD.

Каждый конкретный процессор имеет свою микроархитектуру и реализует архитектуру уровня набора команд. Например, корпорация Intel разработала три типа ISA, которые реализованы для разных процессоров.

То есть эмуляторы воссоздают все основные компоненты устройства, в том числе процессор, память и устройства ввода/вывода.

Пример эмулятора — Android Emulator. Работу эмулятора в Android Studio мы проходили в одном из предыдущих курсов. Напомню, работая с Android Studio, мы выбирали тестовое устройство, настраивали его конфигурации и скачивали образ устройства себе на компьютер. Таким образом, воссоздаются все основные компоненты реального устройства.

Для Android тоже есть эмуляторы. Пример популярного эмулятора — BlueStacks.

Симуляторы

Симулятор — это программное обеспечение, которое помогает вашему компьютеру запускать определённые программы, созданные для другой операционной системы. В основном они предназначены для устройств iPhone и iPad.

Симуляторы, в отличие от эмуляторов, не имитируют аппаратное обеспечение. То есть нельзя исследовать определённые функции, такие как использование батареи, прерывание сотовой связи и прочее, при использовании симуляторов для тестирования.

Пример симулятора — iOS Simulator. Мы можем использовать iOS Simulator, чтобы посмотреть, как UI будет выглядеть на определённом устройстве.

При этом есть очень серьёзное ограничение: для использования iOS Simulator требуется Mac. Поэтому iOS Simulator не очень часто используется для тестирования именно тестировщиками. Отсутствие имитации аппаратного обеспечения и необходимость наличия Mac делают его не самым целесообразным.

Отличие симуляторов и эмуляторов

Немного углубимся в разницу между эмулятором и симулятором:

| | Эмулятор | Симулятор |
|-----------------------|---|--|
| Что имитирует? | Аппаратное обеспечение, программное обеспечение и операционная система мобильных устройств | Внутреннее поведение мобильного устройства |
| Кто разрабатывает? | Производители устройств | Производители устройств и сторонние разработчики |
| Подходит для отладки? | Больше подходит для отладки | В меньшей степени подходит для отладки |
| Производительность | Довольно медленно работают (эмуляция реального оборудования обычно заставляет программное обеспечение работать медленнее, чем изначально) | Быстрее эмулятора |
| Пример | Android SDK | Apple iOS Simulator |

Мобильные фермы

Мы разобрались с эмуляторами и симуляторами, теперь посмотрим на основные мобильные фермы. Для начала, дадим определение мобильным фермам:

Ферма устройств — это группа, состоящая из множества устройств (телефонов, компьютеров или любого другого оборудования), арендованных или находящихся в собственности. Существуют фермы виртуальных или реальных устройств. С помощью этих ферм мы можем тестировать наше программное обеспечение на различном оборудовании. Покупка большого количества смартфонов для целей тестирования требует больших затрат как в денежных, так и в человеческих ресурсах, необходимых для настройки и управления такой системой. Естественная альтернатива — арендовать их по мере необходимости через облачные платформы.

Благодаря фермам устройств ошибки, которые было бы невозможно воспроизвести без определённой версии ОС или телефона определённой марки, можно воспроизвести, даже если у нас в команде нет необходимого устройства. Всё, что вам нужно сделать, — это перейти на платформу фермы устройств, выбрать версию ОС, загрузить приложение и начать тестирование.

Представим следующую ситуацию. Многие клиенты с устройствами определённой марки жалуются, что наше приложение не работает, но ни у кого в технической команде нет такой же марки. Более того, ошибка, на которую жалуются пользователи, не была обнаружена в наборе автоматизированных тестов. Как воспроизвести эту ошибку и как предотвратить её повторение?

Чтобы воспроизвести ошибку, загрузим приложение на одну из ферм устройств и запустим физическое устройство с той же маркой и версией ОС, которые используются клиентами. С помощью ручного тестирования попытаемся определить, когда возникает ошибка. После воспроизведения проблемы можно даже загрузить набор автотестов на ферму устройств и запускать его при каждом прогоне регрессионных тестов.

Таким образом, мы можем легко расширять список устройств, на которых проходят тесты, и допустить меньше ошибок в продакшене.

AWS Device Farm

Сайт [AWS Device Farm | Тестирование мобильных и веб-приложений | Amazon Web Services](#)

AWS Device Farm — это сервис тестирования для разработчиков мобильных приложений, делающих приложения Android, iOS или Amazon Fire OS.

Fire OS — это операционная система, разработанная Amazon на базе Android для планшетов и телефонов собственного производства.

Это очень широко используемая и хорошая по характеристикам ферма мобильных устройств для тестирования. Перечислим её особенности:

- Она позволяет нам тестировать одновременно на нескольких устройствах, чтобы ускорить выполнение наших наборов тестов.
- Ферма также собирает логи и записывает видео для более полноценной отладки и понимания причин возможных дефектов.
- AWS Device Farm предоставляет большое количество устройств с возможностью выбора памяти, процессора и прочего. Это помогает нам настраивать реальные среды, устанавливая местоположение, параметры сети, язык на устройстве.
- Также мы можем настроить интеграцию с разными сервисами (например, Jenkins, позволяющий настраивать установку новых билдов на устройства автоматически).
- Это платный сервис. При этом есть возможность даже настроить частную лабораторию, которая будет сделана только под нас (это позволит избежать очередей и задержек при использовании удалённых устройств). Но у него есть бесплатный период, так что вы можете попробовать его самостоятельно для ознакомления.
- Этот сервис также можно использовать для запуска автотестов. Работает со следующими инструментами: можно использовать Appium (iOS + Android), Espresso и Robotium для Android, а также UIAutomation и XCTest для iOS.

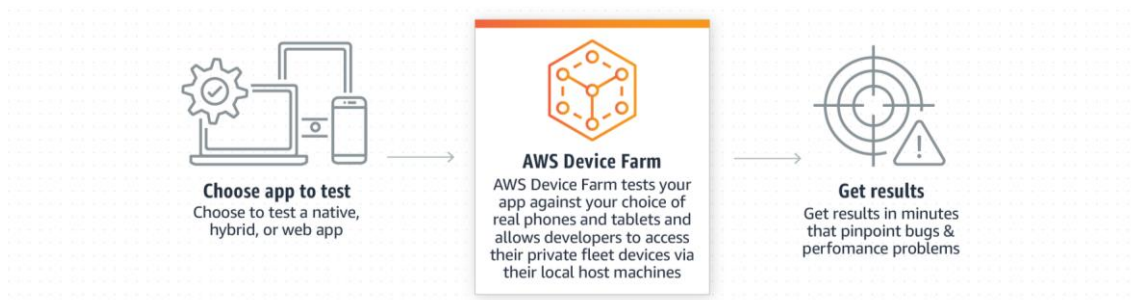
Рассмотрим его немного подробнее.

Краткая схема работы для удалённого доступа к девайсам:



Как мы видим, мы выбираем устройства. Далее AWS Device Farm показывает нам экран устройства прямо в браузере, и мы можем взаимодействовать с этим устройством.

Схема для запуска автотестов:



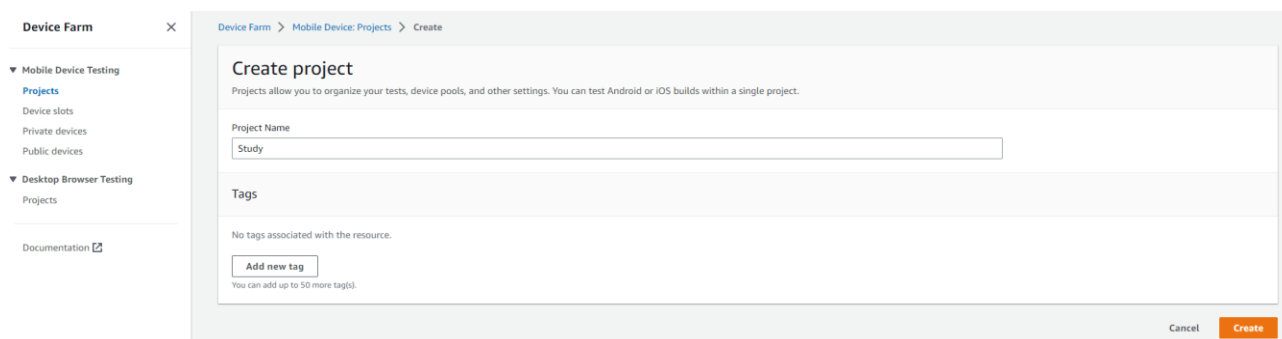
Выбираем приложение, далее выбираем девайсы и запускаем скрипты автотестов. AWS Device Farm прогоняет автотесты на выбранных устройствах, и на выходе мы получаем полный детализированный отчёт. В отчёте мы сможем видеть пройденные и упавшие тесты, запись логов и экрана для воспроизведения проблем и простой отладки.

Начало использования

Чтобы начать пользоваться сервисом, нам в первую очередь нужен аккаунт. Переходим на сайт по ссылке выше и создаём аккаунт. Там обычная процедура, ничего сложного.

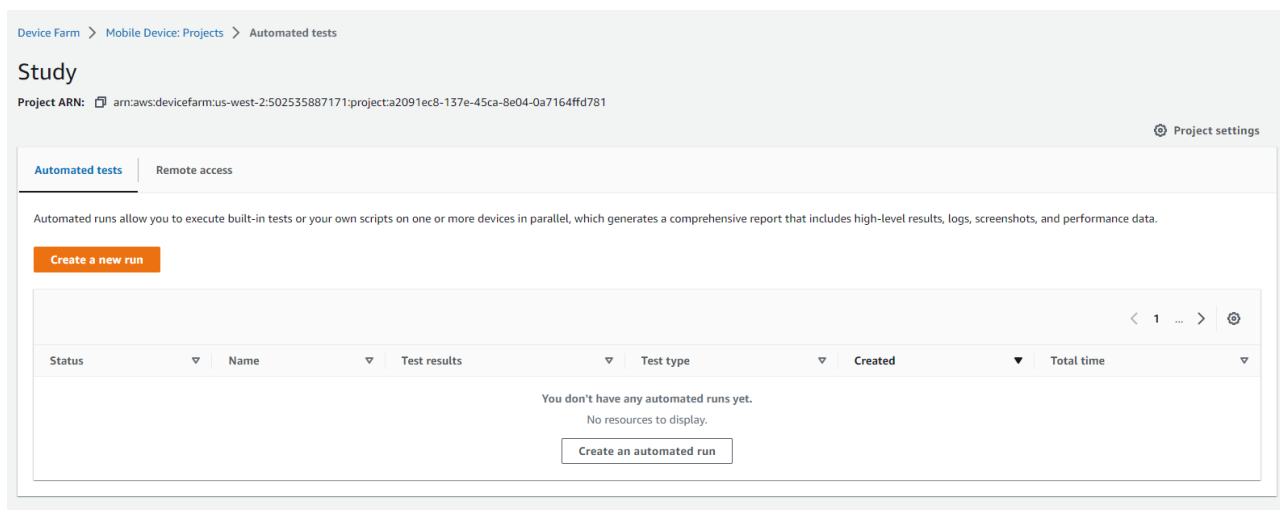
Шаг 1. Создаём проект.

Шаг 2. Переходим в Projects, создаём проект. Вносим название, при необходимости добавляем теги и сохраняем.



Это и будет нашим основным пространством, где будут храниться запуски, отчёты и прочее.

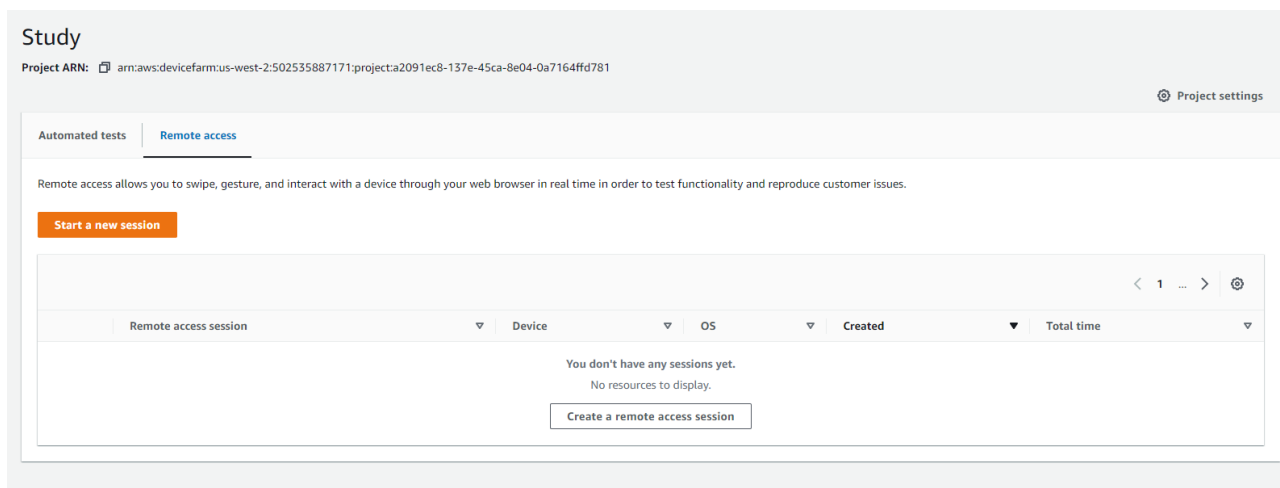
На скрине с проектом мы видим пустой проект и две вкладки: автотесты и удалённый доступ.



А также есть кнопка **Create a run** — создать запуск. По поводу запусков: в AWS Device Farm под запуском подразумевается запуск конкретной сборки приложения с определённым набором тестов для запуска на определённом наборе устройств.

Проекты в AWS Device Farm не различаются по операционной системе, приложению или типу тестов. То есть мы можем запустить один тестовый набор для Android и один для iOS в одном проекте или даже протестировать два разных приложения. Но поскольку нет ограничений на количество проектов, которые мы можем создать, всегда лучше различать операционную систему, приложение или и то и другое.

Шаг 3. Чтобы создать удалённую сессию, переходим на вкладку **Удалённый доступ** и создаём новую сессию.



Шаг 4. На этом шаге мы можем выбрать девайс для тестирования.

Create a new remote session

Choose a device

Select a device for an interactive session. Interested in unlimited, unmetered testing? [Purchase device slots](#)

☐ Show available devices only
(Note: When a device is 'AVAILABLE', your session will start in under a minute.)

Find by name, status, platform, os, form factor, fleetType, or label

< 1 2 3 >

| | Name | Status | Platform | OS | Form factor | Instance Id | Labels |
|-----------------------|--------------------|-----------|----------|----|-------------|-------------|--------|
| <input type="radio"/> | Google Pixel 6a | Available | Android | 13 | Phone | - | - |
| <input type="radio"/> | Google Pixel 7 Pro | Available | Android | 13 | Phone | - | - |
| <input type="radio"/> | Google Pixel 7 | Available | Android | 13 | Phone | - | - |

Девайсы сортируются по названию, статусу («Доступен» или «Занят», советую брать устройства со статусом «Свободен» для быстрого запуска устройства без ожидания в очереди), платформе (iOS и Android), версии ОС, форм фактору (телефон или планшет).

Шаг 5. Вносим название для сессии (советую давать понятные названия) и при желании добавляем теги (удобно в дальнейшем по ним фильтровать).

Name a remote session

Give a name to the remote access session.

Session name
Provide a name for your session. By default, we will use the device name.

Tags

No tags associated with the resource.

Add new tag
You can add up to 50 more tag(s).

Cancel Confirm and start session

Шаг 6. Дальше нужно немного подождать

Device Requested

X

Please wait while we find an available device.

Waiting for Google Pixel 6a

Pending...

Cancel request

If you would like to keep working while you wait for your session to start, you can continue in a [new tab](#)

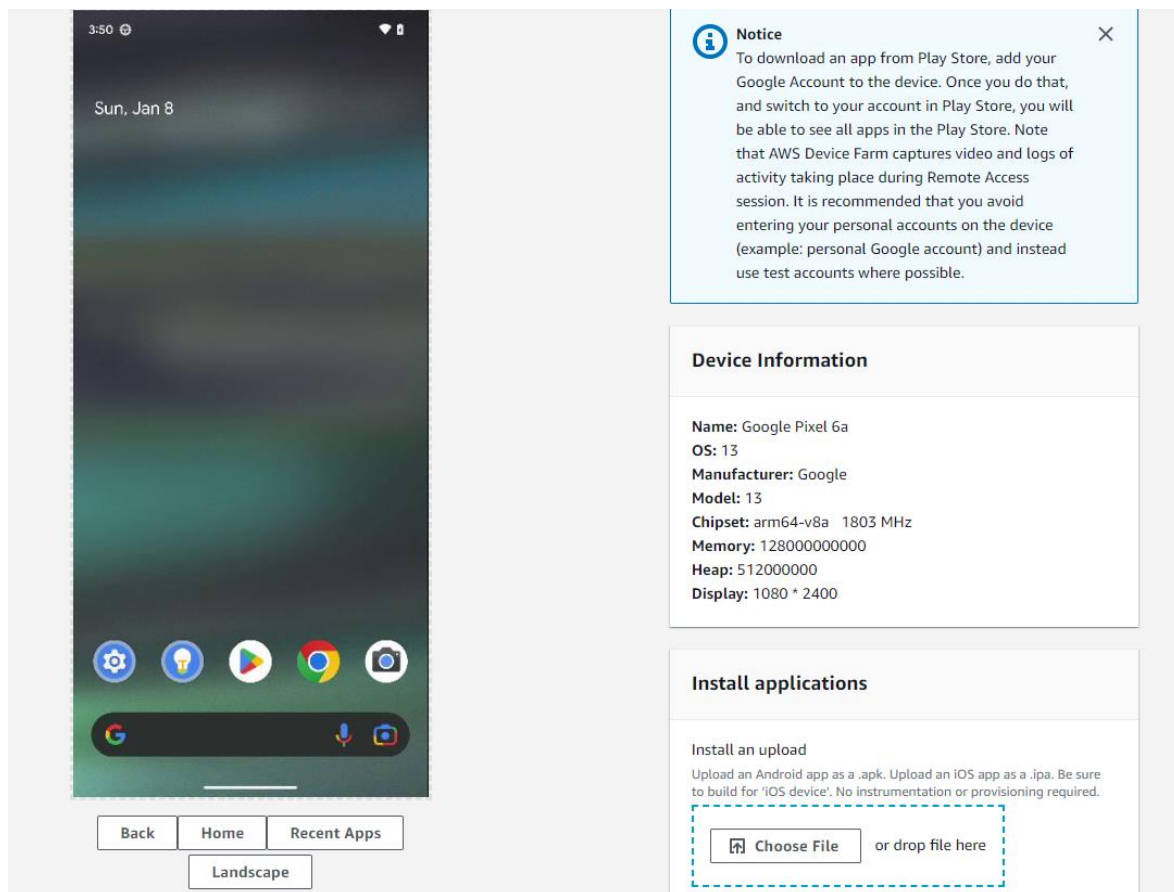
Device Requested, connecting to device ✕

Connecting to Google Pixel 6a



If you would like to keep working while you wait for your session to start, you can continue in a [new tab](#)

В итоге у нас открывается окно, где мы видим экран нашего устройства.



Обратите внимание, внизу есть обозначения кнопок и выбор портретной и альбомной ориентации.

Мы можем устанавливать приложения из сторов, установить напрямую APK или IPA (установочные файлы для Android и iOS). Если мы используем какой-то из сервисов дистрибуции (которые мы проходили на одном из прошлых уроков), логинимся и устанавливаем приложение. То есть, как видите, всё довольно просто.

Шаг 7. Для завершения сессии нажимаем Stop Session (обязательно не забываем нажать эту кнопку, чтобы таймер перестал работать). После завершения сессии нас перебросит на страницу с итогами сессии.

Device Farm > Mobile Device: Projects > Remote access > Session

Google Pixel 6a

Session ARN: arm:aws:devicefarm:us-west-2:502535887171:session:a2091ec8-137e-45ca-8e04-0a7164ffd781/7156ce56-f6e5-47d2-807f-3ebf036a4...

Details Files Tags

Passed

Session name: Google Pixel 6a
Created: Sun, 08 Jan 2023 11:49:01 GMT
Total minutes: 00:05:28
OS: 13

Download logs

Logs

Find logs by source, PID, level, Tag or Message.

Search by Source to filter **Harness** or **Device** Logs.

| Source | Time | PID | Level | Tag | Message |
|---------|---------|------|-------|-----|---|
| Harness | 00:00.0 | 1706 | Info | - | Starting 00000 with device 25041JEGR09297 |
| Harness | 00:00.0 | 1706 | Info | - | Using test content version 0.1.0 |
| Harness | 00:00.0 | 1706 | Info | - | Using image version None |
| Harness | 00:00.1 | 1706 | Info | - | Using kpc version None |
| Harness | 00:00.1 | 1706 | Info | - | Using kpl version None |
| Harness | 00:00.1 | 1706 | Info | - | Using ktc version None |

Мы видим историю взаимодействия с устройством и можем скачать файлы: логи или видео.

Google Pixel 6a

Session ARN: arm:aws:devicefarm:us-west-2:502535887171:session:a2091ec8-137e-45ca-8e04-0a7164ffd781/7156ce56-f6e5-47d2-807f-3ebf036a4...

Details **Files** Tags

Files

[Video](#)
[Logcat](#)
[TCP dump log](#)

Шаг 8. Когда мы вернёмся в проект, мы увидим сессию в истории.

Запуск автоматических тестов

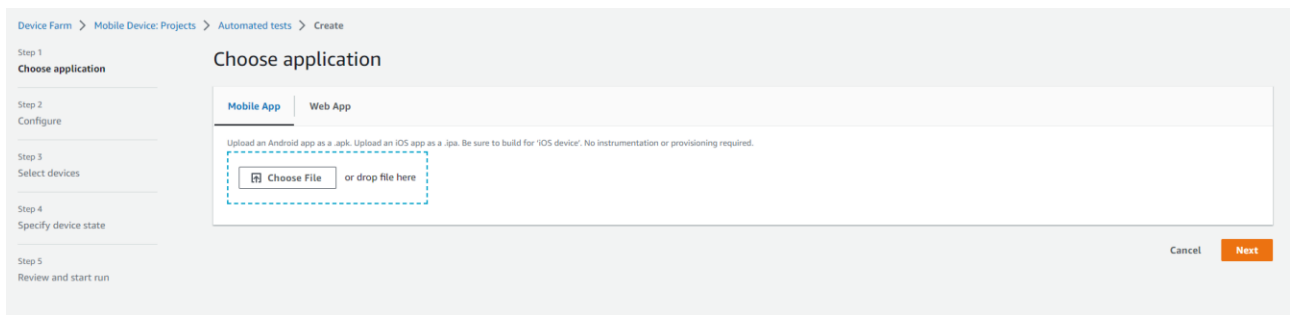
Для ознакомления рассмотрим, как происходит процесс запуска автоматических тестов.

Шаг 1. Возвращаемся в проект.

Шаг 2. Переходим на вкладку **Automated**.

Шаг 3. Создаём запуск (**Create a run**).

Здесь прописаны все шаги, которые необходимо выполнить для запуска тестов.



Шаг 4. Загружаем приложение. Сразу после загрузки мы увидим основную информацию о приложении.



Шаг 5. Теперь мы можем настроить наши тесты. Доступно много вариантов, но мы можем выбрать только один для каждого запуска. Можно выбрать внешние фреймворки, такие как Appium или Cabalash. Если тестов нет, то выбираем **Built-in: Fuzz** или **Built-in: Explorer**. В этом случае будут произведены случайные действия в приложении. Цель такого тестирования состоит в том, чтобы загрузить приложение большими объёмами случайных данных, которые могут привести к его сбою, чтобы обнаружить любые ошибки или лазейки. Никаких дополнительных файлов не требуется.

Configure

Setup test framework

Select the test type you would like to use. If you do not have any scripts, select 'Built-in: Fuzz' or 'Built-in: Explorer' and we will fuzz test or explore your app.

Built-in: Fuzz

No tests? No problem. We'll fuzz test your app by sending random events to it with no scripts required.

Event count
The number of events between 1 and 10000 that the UI Fuzz test should perform.

10

Event throttle
The time in ms between 0 and 1000 that the UI fuzz test should wait between events.

5

Randomizer seed
A seed to use for randomizing the UI fuzz test. Using the same seed value between tests ensures identical event sequences.

Enter a randomizer seed

► Advanced Configuration (optional)

Cancel Previous Next

Шаг 6. Выбираем устройства.

Select devices

Select mobile devices to test

Select from one of the available device pools or create a new device pool.

Device pool Top Devices or Create device pool

100% Compatibility
Your app is compatible with 2/2 devices in the selected pool.

Find devices

| Compatible | Device | OS | Reason | Instance Id | Status |
|------------|-------------------------------|----|--------|-------------|------------------|
| ✓ | Samsung Galaxy S20 (Unlocked) | 10 | - | - | Highly available |
| ✓ | Samsung Galaxy Tab S7 | 11 | - | - | Highly available |

Cancel Previous Next

Обратите внимание, здесь сразу идёт проверка совместимости устройства и загруженного приложения. А ещё мы можем создать свой набор устройств (Device pool). Это очень удобная вещь: мы один раз создаём набор устройств, на которых будут прогоняться тесты, и в дальнейшем не надо будет выбирать их заново (просто выбираем набор из списка готовых наборов).

Шаг 7. На следующем шаге мы можем указать более точные настройки для устройств. Посмотрим, что здесь есть.

Specify device state

Install additional software

Specify additional software to install on the mobile device.

Add extra data

Upload a .zip file to be extracted before your app is tested.

or drop file here

Install other apps

Upload an Android app as a .apk. No instrumentation or provisioning required.

or drop file here

Location and network settings

Specify settings to simulate real-world scenarios and device configurations.

Set radio states

Control the states of various radios before your app's first launch.

- ☒ WiFi
- ☐ Bluetooth
- ☒ GPS
- ☒ NFC

Device location

By setting the latitude and longitude of the device you can test location-specific behavior.

| Latitude | Longitude |
|--------------------------------------|--|
| <input type="text" value="47.6204"/> | <input type="text" value="-122.3491"/> |

Device locale

Locales allow you to specify the language and region of the device.

Network profile

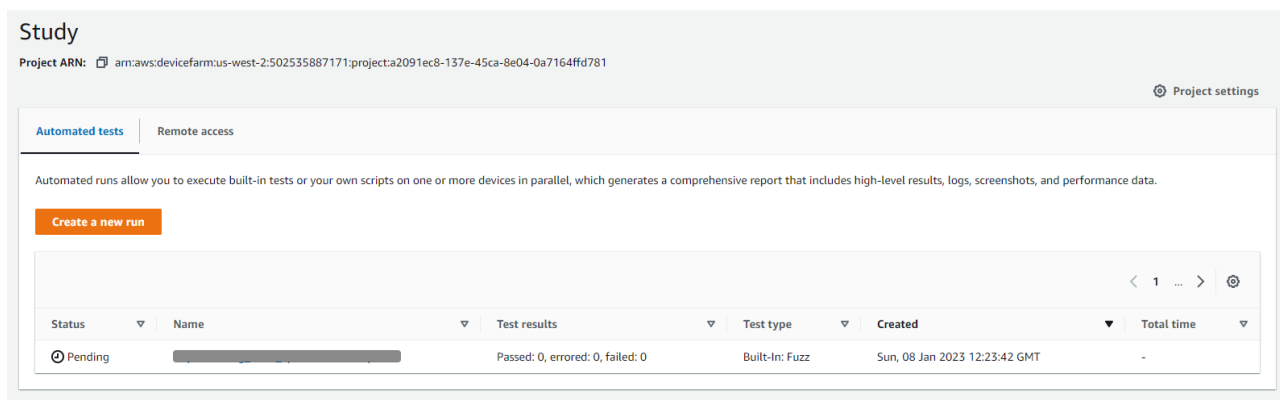
Select a pre-defined network profile or create a new one by clicking the button on the right.

Шаг 8. Мы можем загружать данные на устройство (полезно, когда нам нужно воссоздать тест с заполненной памятью телефона). Можно установить дополнительные приложения. Также настраиваем необходимость датчиков Wi-Fi, Bluetooth, GPS, NFC.

Шаг 9. Указываем локацию, язык устройства, тип сети.

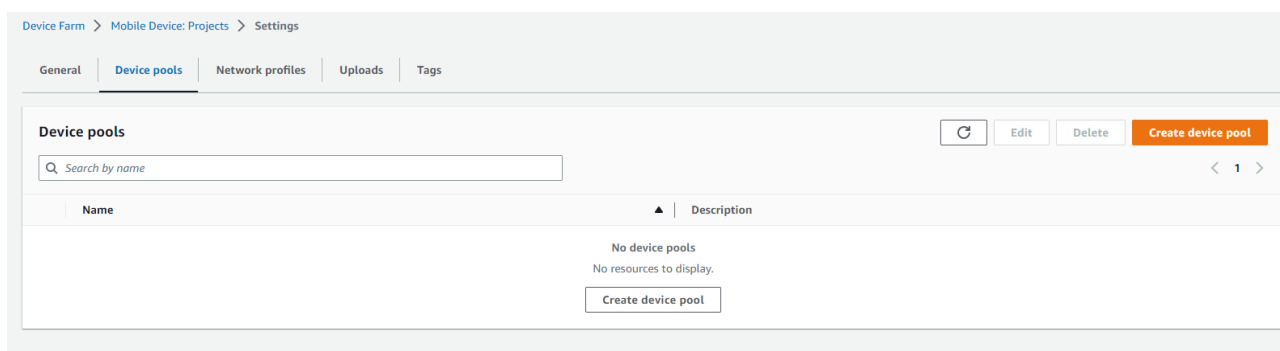
Шаг 10. Дальше просматриваем заполненные конфигурации и запускаем тесты.

После запуска нас вернёт на страницу проекта, где мы сможем увидеть статус теста и дождаться его прохождения.



В результате получается красивый отчёт по проведённым тестам. Также мы сможем найти детализацию по каждому тесту и повторить запуск тестов.

В завершение немного пройдемся по настройкам проекта.



Мы можем создать или редактировать наборы устройств (например, все Samsung или все устройства на iOS 13, либо, наоборот, набор максимально разнообразных устройств). Можем создавать или редактировать профили сети (например, медленное соединение) и просматривать все приложения, которые мы загрузили в проект, а также добавить новую сборку.

Следующей рассмотрим довольно простую ферму от Samsung.

Samsung Remote Test Lab

Ссылка: [Remote Test Lab | Samsung Developers](https://developer.samsung.com/remote-test-lab/)

Здесь название говорит само за себя. С помощью этого инструмента мы можем удалённо использовать устройства Samsung. Их устройства довольно популярны, и производитель сам даёт разработчикам возможность проверить, как будет вести себя разрабатываемое приложение на этих устройствах.

Remote Test Lab

Use the Remote Test Lab service to test your applications on a real device.

Get Started

Featured Devices

Don't have a Samsung device to test your app? No Problem! Test your apps on the latest Samsung Galaxy devices in our Remote Test Lab.



Galaxy Z Fold4



Galaxy Z Flip4



Galaxy S22



Galaxy Tab S8



Galaxy Tab 7+

Сервис бесплатный. Нужна только регистрация. В лаборатории представлены популярные модели телефонов, планшетов и даже часов. Большой плюс этого сервиса — возможность протестировать приложение на новых устройствах ещё до того, как они поступают в продажу на рынке. Это даёт возможность выпустить необходимые обновления к моменту, когда наши пользователи перейдут на новые телефоны. Во время использования можно снимать логи и делать запись экрана.

Так как инструмент довольно полезный, и вы можете им довольно часто пользоваться, тоже пошагово пройдемся по процессу.

Сразу обращаю внимание: для использования устройств нам нужны баллы (их можно будет увидеть в личном кабинете). Но их можно получать каждый день, чего достаточно для полноценной работы с сервисом.

My Credits 403

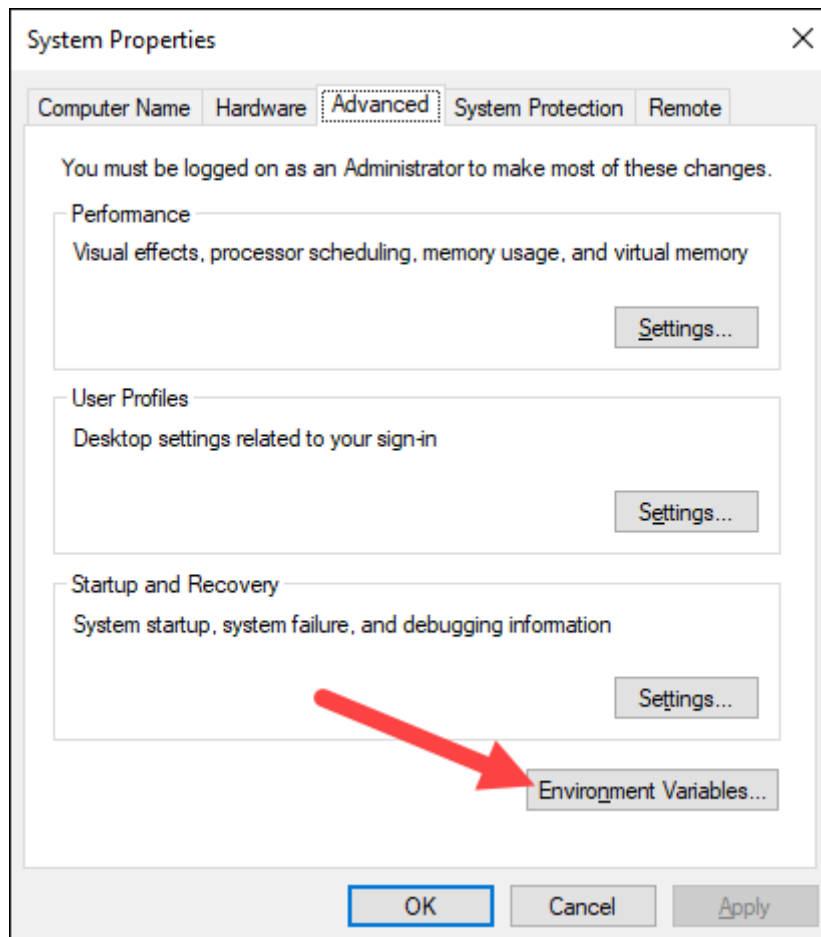
В первую очередь необходимо установить Java, для этого нужно:

Шаг 1. Скачать с сайта Windows Installer [Java Downloads | Oracle](#).

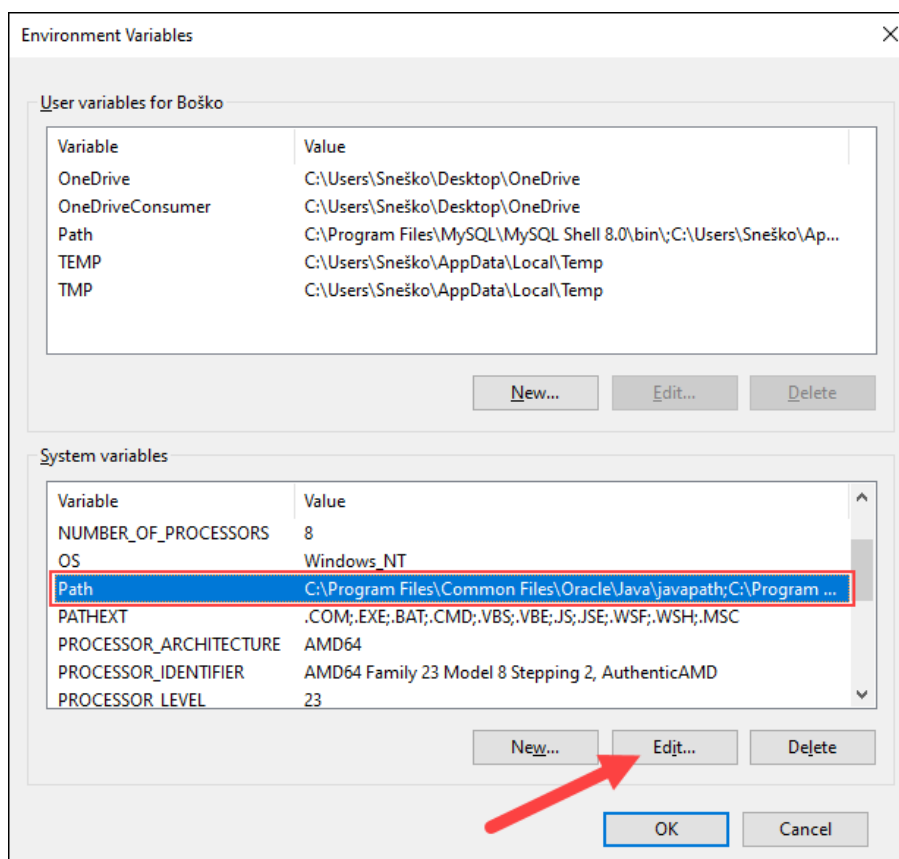
Шаг 2. Запустить установочный файл.

Шаг 3. После установки в настройках найти **Edit the system environment variables**.

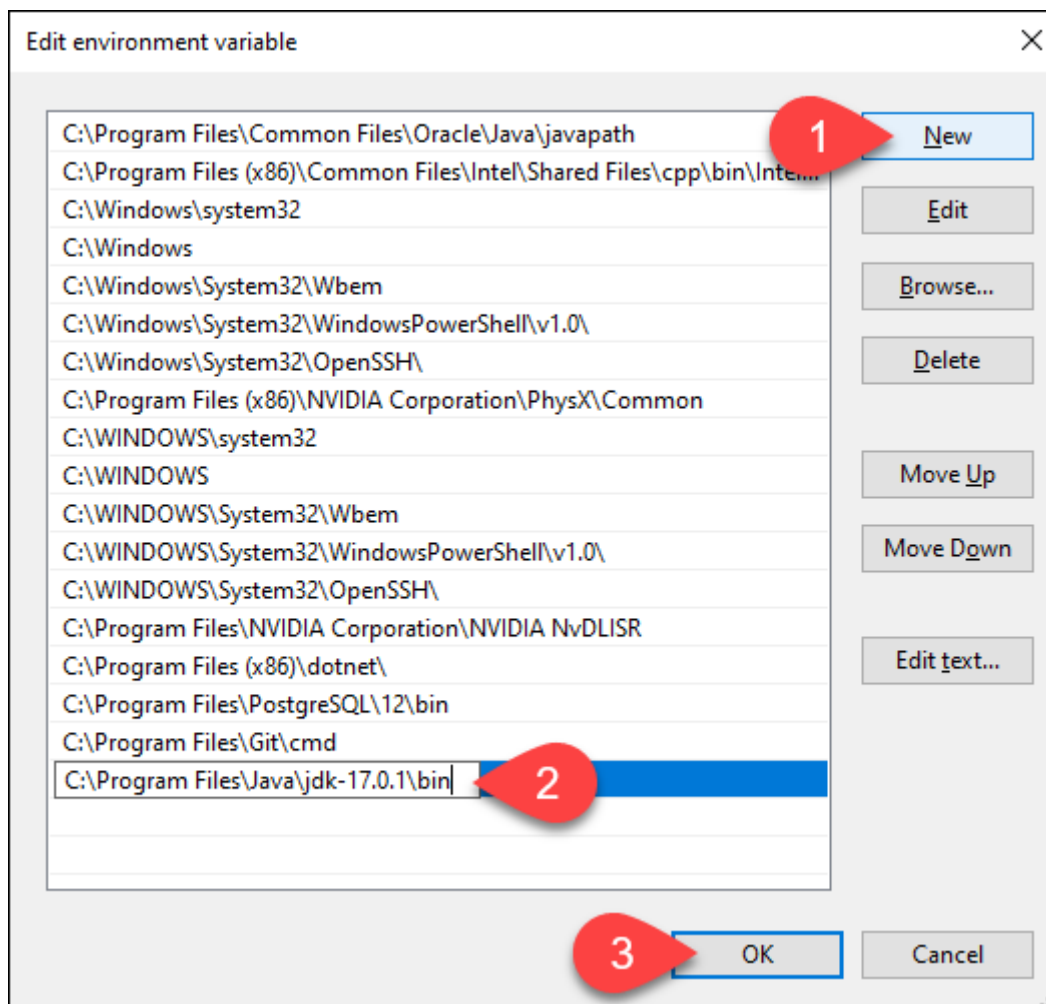
Шаг 4. Перейти во вкладку **Advanced** и выбрать **Environment Variables**.



Шаг 5. Нажать **Edit**.



Шаг 6. Нажать **New** и прописать путь до Java bin.

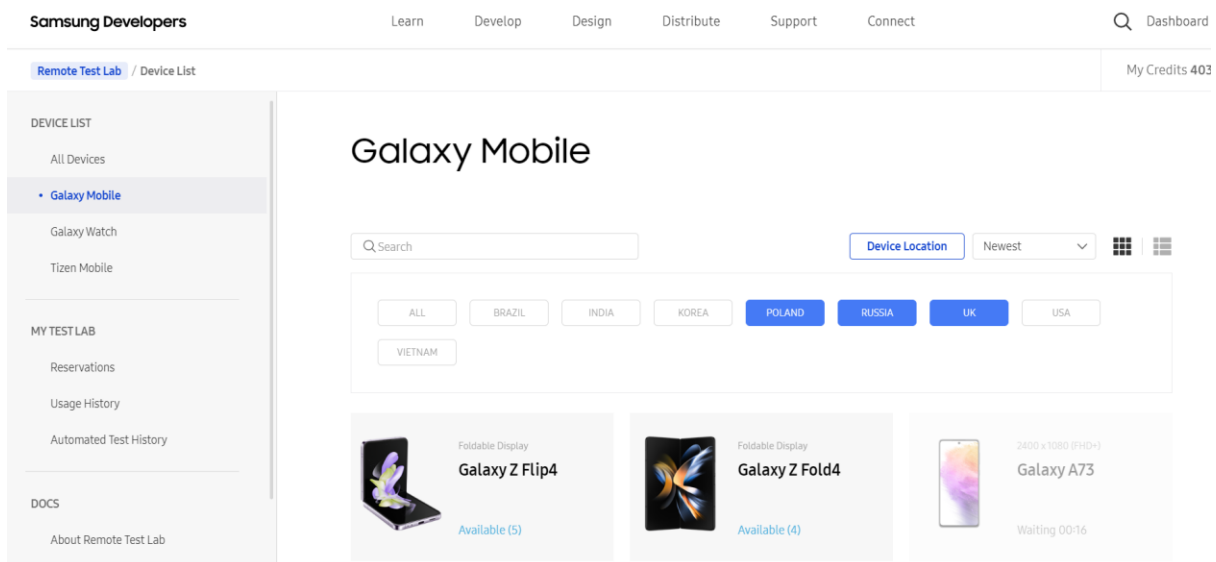


Сервис работает путём удалённого доступа к физическим устройствам через приложения Java, работающие на вашем компьютере.

Шаг 7. Разобраться, как он работает:

1. Обязательно зарегистрироваться в сервисе.
2. После нажать **Get Started**, тогда нам откроется страница с доступными устройствами.


Обратите внимание: справа идёт разделение на категории устройств (Galaxy Mobile, Galaxy Watch, Tizen Mobile).



Мы можем отфильтровать устройства по названию и расположению сервера. Советую не выбирать серверы, которые находятся слишком далеко от вас — слишком большое расстояние от сервера до вас может влиять на корректную работу сервиса.

У каждого из устройств отображается либо доступное количество, либо время, через которое устройство освободится.

Шаг 8. Выбрать устройство.



Location

-- Location --

OS Version

-- OS Version --

Duration

Reserve for 30 minutes

Available Device

10

2 Credit Required to Start

Start

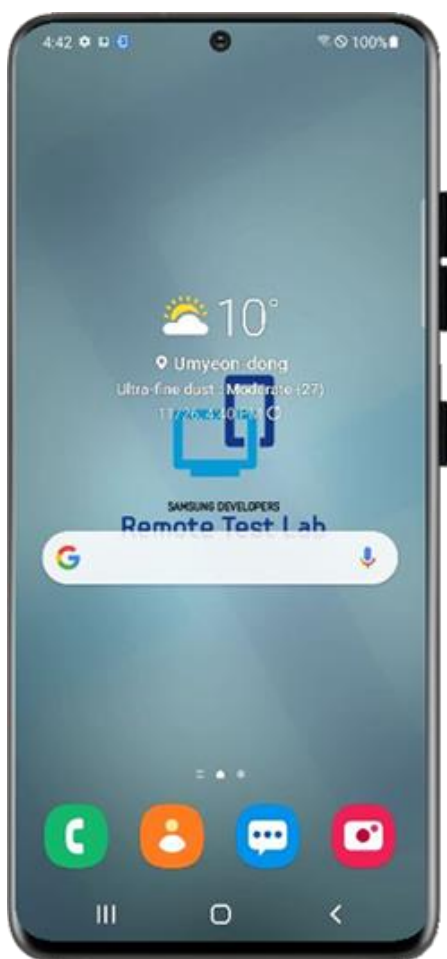
Шаг 9. В зависимости от количества доступных устройств у нас будет возможность выбрать расположение сервера и версию Android (очень удобно, что у устройств есть

несколько вариантов с разными версиями Android — полезно, когда нужно воспроизвести какой-нибудь дефект по сценарию от пользователей).

Шаг 10. Выбрать время, на которое бронируем устройство.

Шаг 11. После нажатия **Start** нам на компьютер скачается файл, который необходимо установить (если загрузка не происходит, нужно перезапустить браузер или просто подождать).

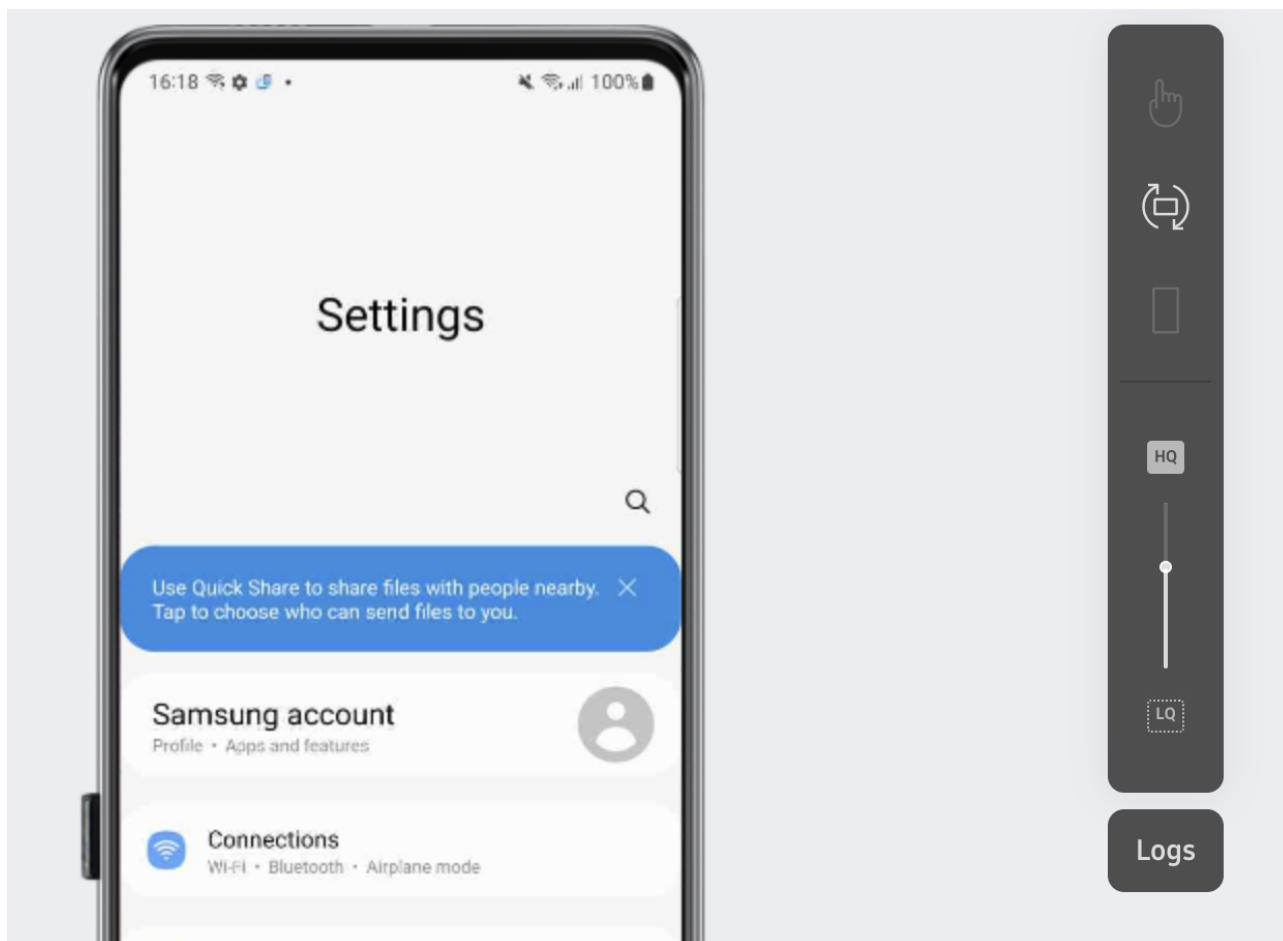
Шаг 12. Запустить загруженный файл, и в новом окне браузера мы увидим экран устройства.



Итак, перед нами удалённое устройство. Посмотрим, как с ним работать. Начнём с методов ввода. Нам доступны следующие:

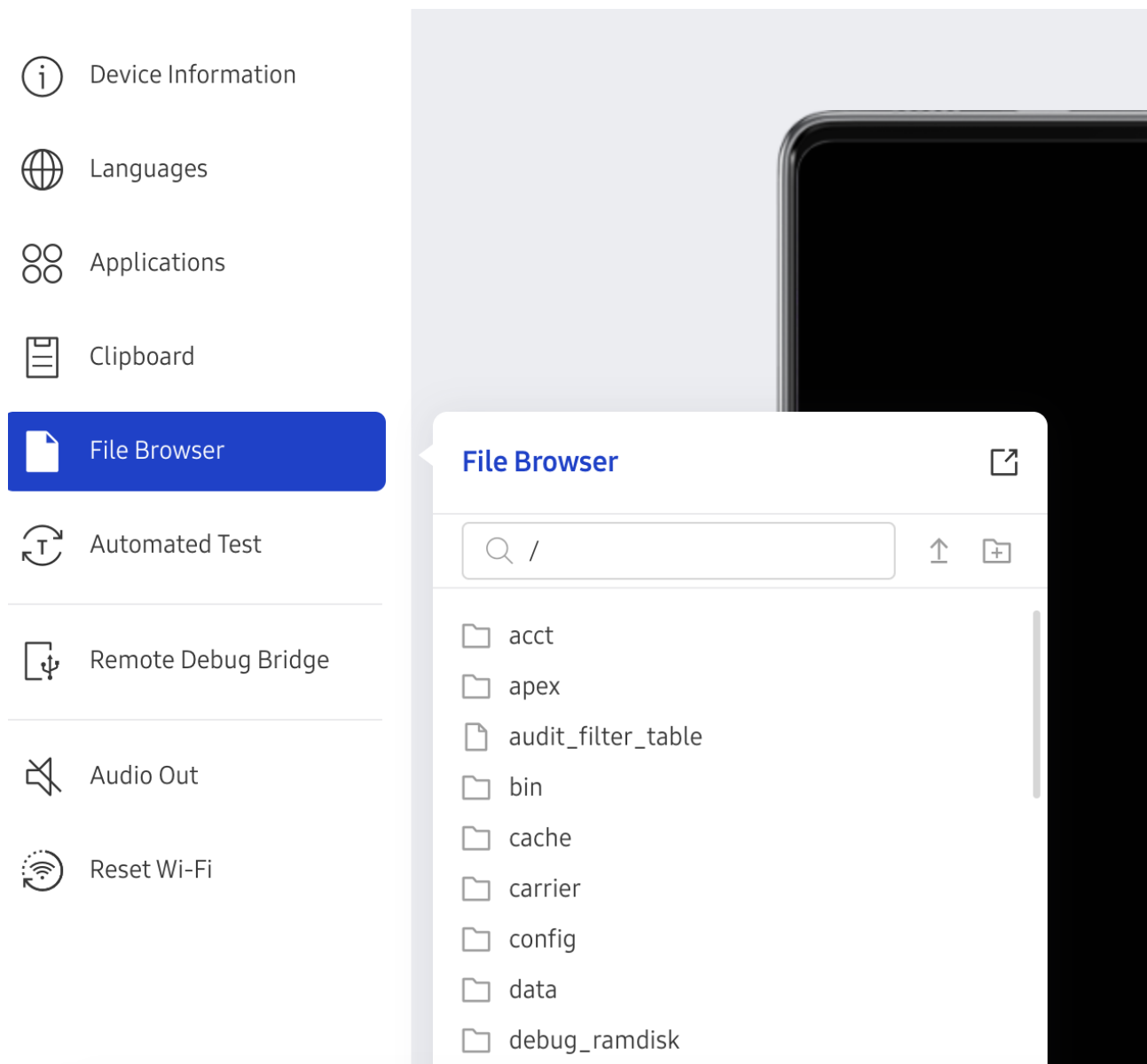
- **Тап, свайп.** Всё просто, с помощью мышки мы можем делать тапы, в том числе длинные, и свайпать, например, для листания ленты.
- **Мультитач.** Чтобы имитировать мультитач, нам нужно зажать Shift и быстро кликнуть мышкой в те области экрана, где должны быть нажатия.
- **Аппаратные кнопки** расположены по краям экрана с устройством.

- **Ввод с клавиатуры.** Для ввода с клавиатуры можно использовать клавиатуру компьютера, эти значения будут передаваться на удалённое устройство.



Далее мы можем:

- Менять качество отображения экрана (может пригодиться при плохом соединении)
- Менять ориентацию экрана (альбомная и портретная) и менять масштаб размера экрана
- Использовать функции: запись экрана и скриншоты
- Поменять папку, в которую видео и скриншоты будут сохраняться



Иногда бывает полезно в режиме реального времени показать кому-нибудь (например, программисту), что происходит на экране. Для этого:

1. Вы можете отправить приглашение, указав адрес электронной почты.
2. Второму участнику нужно будет перейти по ссылке (но при этом второй участник не сможет управлять устройством, только смотреть, какие манипуляции вы совершаете).

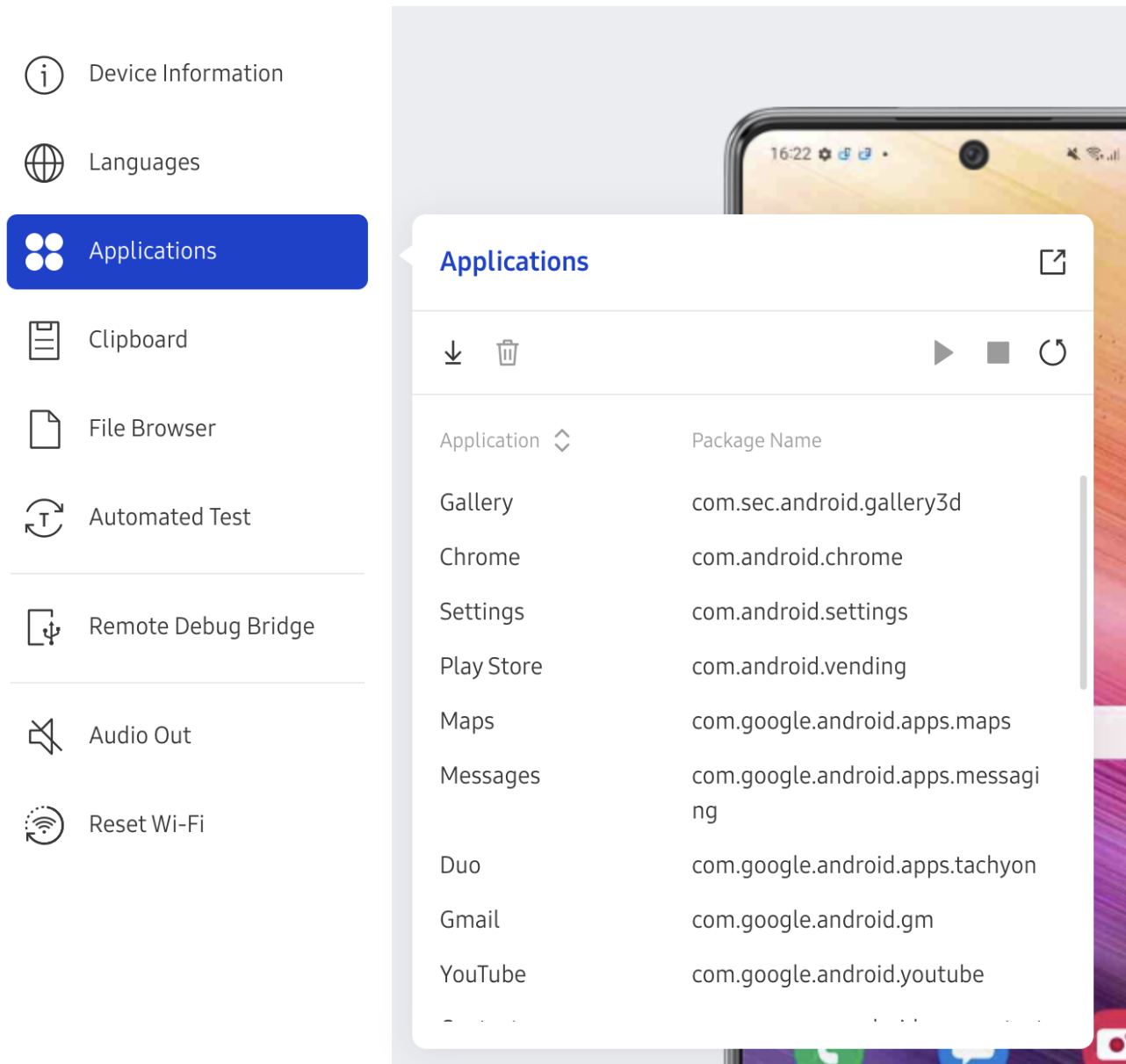
Рассмотрим, как устанавливать приложения на удалённое устройство.

В меню есть варианты взаимодействия с приложениями:

- Установить приложение
- Удалить приложение
- Показать все приложения

- Запуск приложения
- Закрывать приложения

Для установки просто указываем путь к файлу на компьютере.



Во время тестирования полезно снимать логи (в меню с помощью одной функции можно активировать запись логов).


```

date      time      pid      tid      priority  tag      message
2023-01-26 16:23:58.706 665      665      D          io_stats: MemInfo(KB):
total 5440408 free 215480 avail 1468304 actv(f) 422632 inactv(f) 756836 dirty 3424 wp 0 anonpg
1827260 slab 394632
2023-01-26 16:23:58.706 665      665      D          io_stats: Read_top(KB):
system_server(1199) 31068 gle.android.gms(3981) 13204 ch_zygote(19220) 8932
2023-01-26 16:23:58.706 665      665      D          io_stats:
Write_top(KB): system_server(1199) 2540 .gms.persistent(3106) 1904 ch_zygote(19220) 1804
2023-01-26 16:23:58.706 856      856      D          Zygote Forked child process
19678
2023-01-26 16:23:58.707 1199     1628     I          ActivityManager Start proc
19678:com.samsung.android.bbc.bbcagent/1000 for broadcast {com.samsung.android.bbc.bbcagent/
com.samsung.android.bbc.bbcagent.bbccontroller.receiver.PackageEventReceiver}
2023-01-26 16:23:58.709 1199     1234     D          SamsungAlarmManager
Cancel Alarm calling from uid:10227 pid :3106 / OP:PendingIntent{7c39a9a: PendingIntentRecord{2dbc9bf
com.google.android.gms/com.google.android.gms.gcm broadcastIntent}}
2023-01-26 16:23:58.709 1199     1234     V          SamsungAlarmManager
setLocked to kernel - W:403740 / NW:314133, now=312841
2023-01-26 16:23:58.711 943      19319    I          HYPER-HAL
[RequestManager.cpp]releaseLocked(): Released ID : 106645710
2023-01-26 16:23:58.714 1199     1234     D          SamsungAlarmManager
setInexact (T:3/F:0/AC:false) 20230126T172357 now=312846 - CU:10227/CP:3106/OP:PendingIntent{78b23a8:
PendingIntentRecord{2dbc9bf com.google.android.gms/com.google.android.gms.gcm broadcastIntent}}
2023-01-26 16:23:58.714 1199     1234     V          SamsungAlarmManager
setLocked to kernel - W:403740 / NW:314133, now=312846
2023-01-26 16:23:58.718 19678    19678    I          libc SetHeapTaggingLevel:
tag level set to 0
2023-01-26 16:23:58.720 1199     1234     V          SamsungAlarmManager
setLocked to kernel - W:403740 / NW:314133, now=312853
2023-01-26 16:23:58.720 1199     1234     D          SamsungAlarmManager

```

И ещё очень интересная функция — автоповтор действий. Это нельзя назвать полноценной автоматизацией, но тем не менее мы можем записать действия в приложении и поставить их на автоматическое воспроизведение.

Как это выглядит:

- Нажимаем кнопку Record
- Затем проводим действия в приложении
- Смотрим, как у нас всё записалось
- Нажав Play, мы можем построить всё
- Записанные действия можно сохранить в файл и затем загрузить для воспроизведения на других устройствах с таким же разрешением



Test duration:

—

5

+

min.

| Application ↕ | Package Name ↻ |
|---------------|---|
| Chrome | com.android.chrome |
| Play Store | com.android.vending |
| Maps | com.google.android.apps.maps |
| Messages | com.google.android.apps.messaging |
| Duo | com.google.android.apps.tachyon |
| Gmail | com.google.android.gm |
| YouTube | com.google.android.youtube |
| AR Zone | com.samsung.android.arzone |
| SmartThings | com.samsung.android.oneconnect |
| Google | com.google.android.googlequicksearchbox |
| OneDrive | com.microsoft.skydrive |
| Samsung Free | com.samsung.android.app.space |

Time remaining:

Test status:

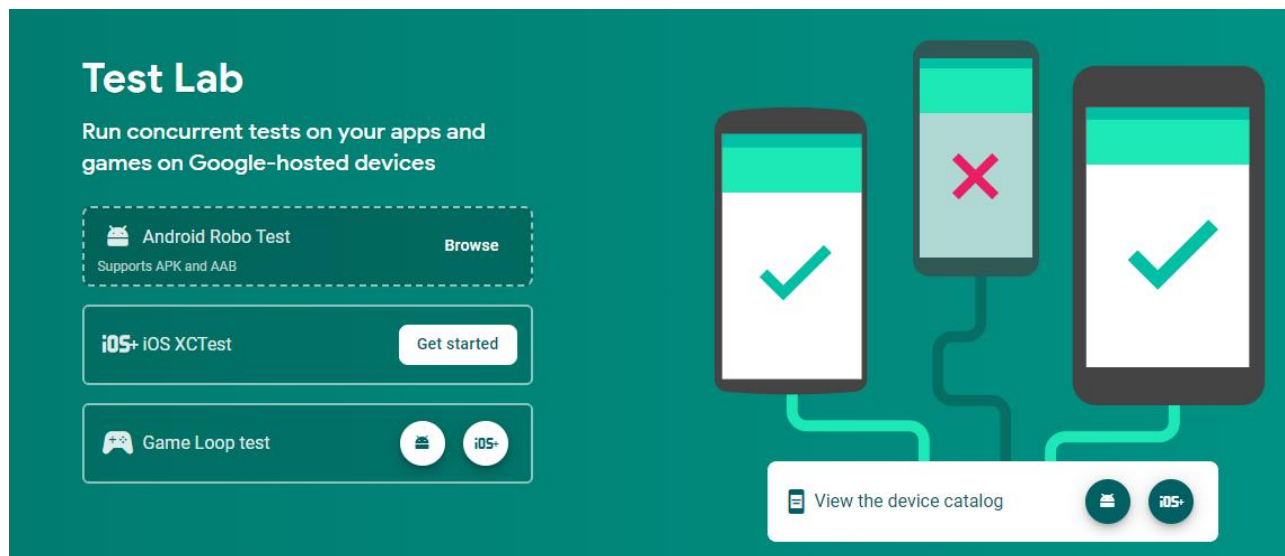
Start

Также мы можем запустить Remote Debug Bridge для полноценной работы с ADB (вы проходили это подробнее на предыдущих уроках).

Следующие фермы мы будем рассматривать не так подробно, но тем не менее пройти по ним, чтобы понять, в каких случаях они могут нам помочь, будет полезно.

Firebase Test Lab

Ссылка: [Firebase Test Lab](https://firebase.google.com/test-lab/)



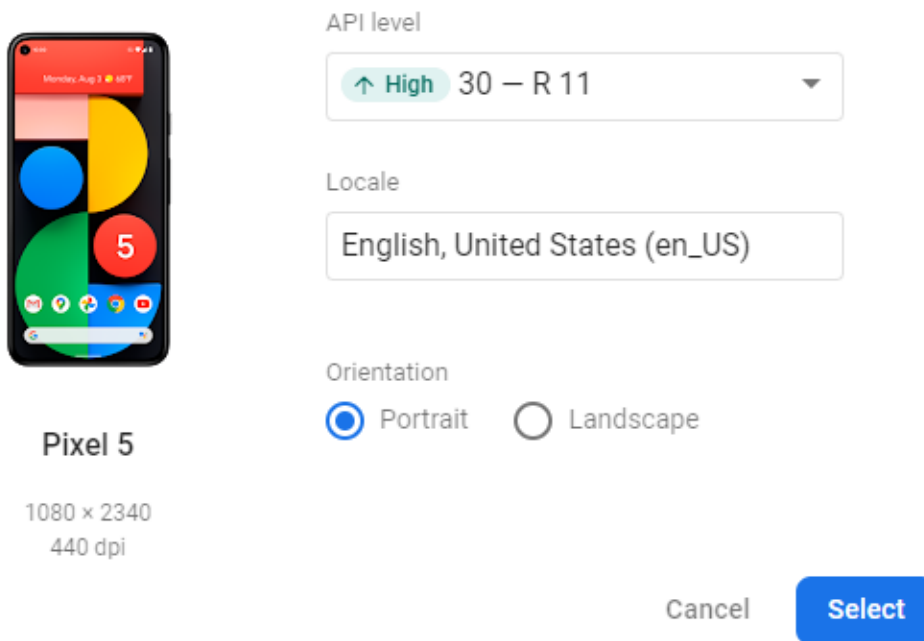
Firebase Test Lab — это облачная инфраструктура для тестирования приложений, которая позволяет вам тестировать ваше приложение на различных устройствах и конфигурациях, чтобы вы могли лучше понять, как оно будет работать в руках реальных пользователей.

Как мы уже знаем, Firebase — это продукт Google. Но, несмотря на это, мы можем использовать лабораторию Firebase Test Lab для тестирования как Android, так и iOS-приложений.

На удалённых устройствах можно настраивать различные конфигурации. Что касается самих устройств, это различные модели на Android и iOS, но при этом количество вариантов не такое большое, как, например, у AWS Device Farm. Сразу отмечу: здесь нет возможности просто запустить удалённое устройство и вручную произвести какие-либо действия. Для тестирования используются автоматические инструменты.

Firebase Test Lab позволяет определить набор различных конфигураций устройств (здесь это называется тестовой матрицей), на которых должен выполняться тест. Для каждого отдельного устройства можно выбрать следующую конфигурацию: версии ОС, языка и ориентации. Всё это настраивается как параметр теста. На скрине показана настройка для Google Pixel 5.

Select device



API level

↑ High 30 — R 11

Locale

English, United States (en_US)

Orientation

☒ Portrait ☐ Landscape

Pixel 5

1080 × 2340
440 dpi

Cancel Select

Как я уже говорила, ручного взаимодействия с устройством тут не подразумевается, поэтому есть два варианта запуска тестов:

1. Предварительно созданные автотесты. Подробнее про фреймворки для автотестов мы будем говорить позже, поэтому просто обозначу, что для Android можно использовать Android Espresso или UI Automator. Для iOS — XCTest. Инструментальные тесты могут выполняться до 45 минут на физических устройствах и до 60 минут на виртуальных устройствах .
2. Роботест для Android. О нём чуть подробнее. Robotest — это инструмент тестирования, интегрированный с Firebase Test Lab. Роботест анализирует структуру пользовательского интерфейса (UI) приложения, а затем методично исследует его, автоматически имитируя действия пользователя. При этом мы можем создавать и редактировать скрипты для роботестов, например, чтобы воспроизвести простые сценарии действий пользователей. Это позволит использовать Robotest для проверки исправлений ошибок и проверки регрессий. После прогона тестов мы можем увидеть логи, скриншоты, видео и информацию о производительности.

В заключение добавлю, что сервис бесплатный, но при этом в бесплатном плане есть ограничения на время тестов в день.

Microsoft Visual Studio App Center Test

Ссылка: [Visual Studio App Center | iOS, Android, Xamarin & React Native | Create Account](#)

С App Center мы уже немного познакомились, когда рассматривали сервисы для дистрибуции мобильных приложений.



Его тестовая лаборатория отличается куда большим пулом устройств.

Он также поддерживает Android и iOS.

Что касается самих тестов — лаборатория используется для запуска автоматических тестов. То есть мы загружаем билд для тестирования, запускаем скрипты для автоматических тестов, настраиваем набор тестовых устройств и запускаем тесты.



Сервис удобен тем, что мы можем хранить тесты и наборы устройств, а также запускать прохождение автотестов каждый раз, когда собран новый билд (при

корректной интеграции с App Center). После прохождения тестов все логи и скриншоты доступны в течение 6 месяцев.

В заключение скажу, что это удобный и быстрый сервис с большим количеством настраиваемых устройств. Но сервис платный, и не каждая компания будет им пользоваться.

Выводы


Сегодня мы:

- Разобрали, что такое эмуляторы и симуляторы, посмотрели разницу между ними.
- Изучили принцип работы мобильных ферм на примере самых популярных сервисов. На следующих уроках мы продолжим изучать полезные инструменты, которые могут пригодиться нам в процессе тестирования.

Домашнее задание

Вы можете попробовать использовать сервис Samsung Remote Test Lab и протестировать приложение. Для этого вам нужно:

1. Установить Java
2. Использовать сервис Samsung Remote Test Lab
3. Создать гугл-аккаунт для тестирования, залогиниться на удалённом устройстве в Google Play
4. Скачать приложение-переводчик
5. Запустить, сделать скриншоты и снять логи

 Задание является необязательным, но крайне полезным для получения практического опыта и подготовки к семинару. Сдавать задание не нужно.

Используемая литература

1. [Run a Robo test | Firebase Test Lab](#)
2. [Архитектура набора команд — Википедия](#)