



## **Tópicos especiais I:** testes e automação para dispositivos móveis

Pós graduação em Testes Ágeis  
08 e 09 de julho de 2022

Prof. Maria Clara Bezerra

[www.github.com/clarabez](https://www.github.com/clarabez)

# Como serão organizadas nossas aulas?



Vamos nos conhecer e falar sobre a organização das aulas :)

# Como serão organizadas nossas aulas?



Vamos nos conhecer e falar sobre a organização das aulas :)



Vamos falar sobre contexto de testes para dispositivos móveis.

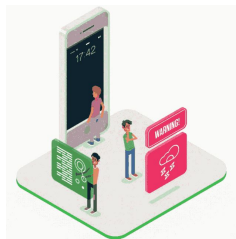
# Como serão organizadas nossas aulas?



Vamos nos **conhecer** e falar sobre a organização das aulas :)

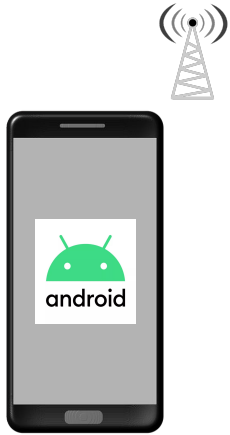


Vamos falar sobre **contexto** de testes para dispositivos móveis.



Vamos falar sobre **automação** e conhecer *frameworks* e ferramentas que nos ajudam nisso.

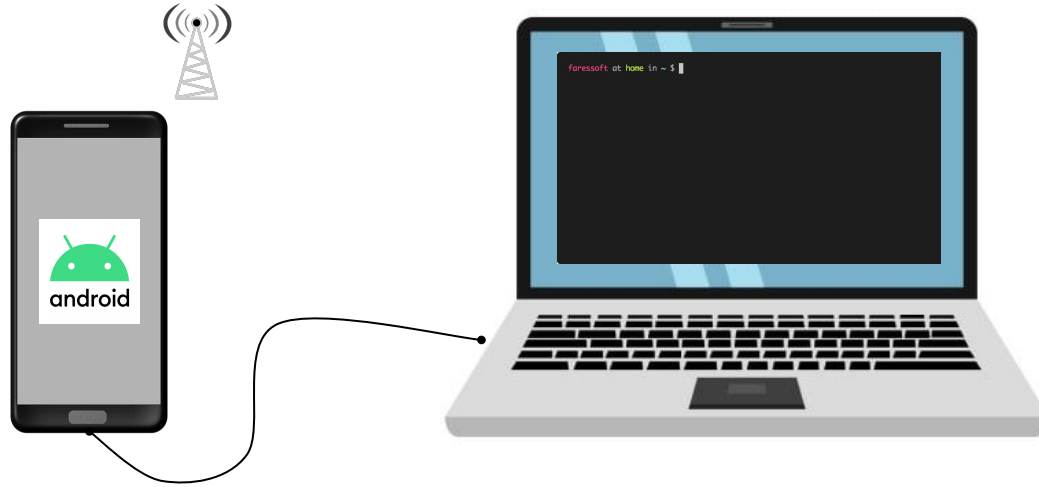
# O que iremos aprender



## Estratégia para testes mobile:

- Fatores externos
- Tipos de redes
- Plataformas, tipos de aplicações

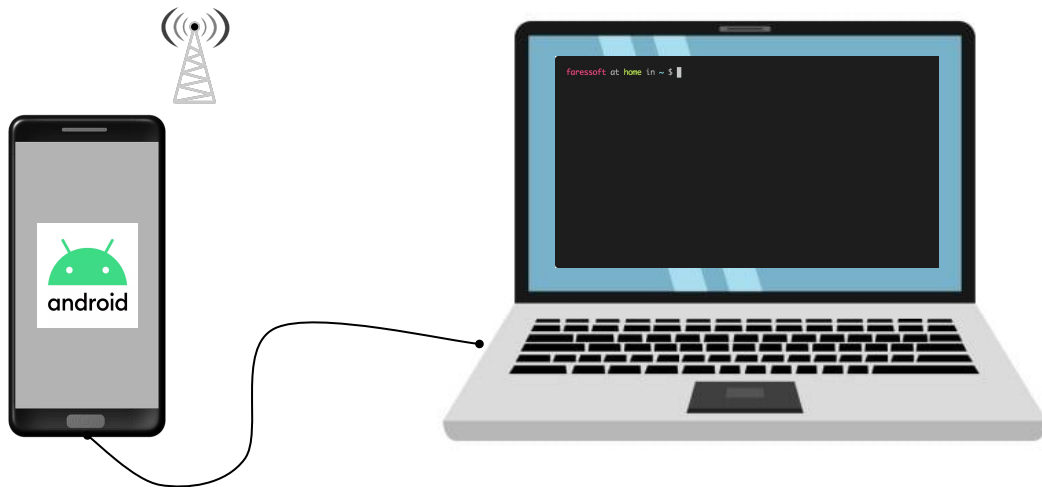
# O que iremos aprender



## Estratégia para testes mobile:

- Fatores externos
- Tipos de redes
- Plataformas, tipos de aplicações

# O que iremos aprender



## Estratégia para testes mobile:

- Fatores externos
- Tipos de redes
- Plataformas, tipos de aplicações

## Setup de ambiente:

- Android SDK
- Download de ferramentas

## Emulando um dispositivo:

- Android Studio (AVD)
- Genymotion

## Interagindo com o dispositivo:

- Comandos ADB
- Coleta de logs
- *Monkey testing*

## Automação de aplicações:

- Espresso
- UIAutomator
- Appium

# Hora de baixar algumas ferramentas:

android  
studio



<https://developer.android.com/studio>

**Finalidade:** Emulador, Espresso, UIAutomator

**Prós:** Todo ecossistema de desenvolvimento Android

**Contras:** Muito pesado

 appium

<https://appium.io/downloads.html>

**Finalidade:** Mapeamento e automação

**Prós:** Leve, fácil de usar, essencial para nossos exercícios

**Contras:** Nada :)

 IntelliJ IDEA

<https://www.jetbrains.com/pt-br/idea/download/>

**Finalidade:** Alternativa ao Android Studio

**Prós:** Menos pesado que o AS

**Contras:** Ainda assim é pesado

oo  
GENYMOTION  
By Genymobile

<https://www.genymotion.com/download/>

**Finalidade:** Emulador

**Prós:** Bem mais leve que o AS, super rápido, mais prático que o AS

**Contras:** Ainda precisa do SDK instalado. Não tem demais recursos como UIAuto ou Espresso



# Hora de baixar algumas ferramentas:

android  
studio



[esse ou IntelliJ]

<https://developer.android.com/studio>

**Finalidade:** Emulador, Espresso, UIAutomator

**Prós:** Todo ecossistema de desenvolvimento Android

**Contras:** Muito pesado



IntelliJ IDEA

[esse ou AS]

<https://www.jetbrains.com/pt-br/idea/download/>

**Finalidade:** Alternativa ao Android Studio

**Prós:** Menos pesado que o AS

**Contras:** Ainda assim é pesado



[obrigatório]

<https://appium.io/downloads.html>

**Finalidade:** Mapeamento e automação

**Prós:** Leve, fácil de usar, essencial para nossos exercícios

**Contras:** Nada :)

GENYMOTION

By Genymobile

[alternativa]

<https://www.genymotion.com/download/>

**Finalidade:** Emulador

**Prós:** Bem mais leve que o AS, super rápido, mais prático que o AS

**Contras:** Ainda precisa do SDK instalado. Não tem demais recursos como UIAuto ou Espresso

# Setup de ambiente

Temos que adicionar variáveis de ambiente para usarmos o ADB:

## Windows:

Meu computador > Propriedades > Propriedades avançadas do sistema > Avançado > Variáveis de ambiente.

Selecione a variável PATH e adicione a ela o caminho das ferramentas do SDK, que ficam em “Android/platform-tools”.

Por exemplo, se sua pasta está em Downloads, adicione o caminho “C:\Download\Android\platform-tools”

## Linux:

```
# instalar
```

```
sudo apt install android-tools-adb android-tools-fastboot
```

```
# variáveis de ambiente
```

```
export PATH=${PATH}:/root/android-sdk-linux/tools
```

```
export PATH=${PATH}:/root/android-sdk-linux/platform-tools
```

# Setup de ambiente

Temos que adicionar variáveis de ambiente para usarmos o ADB:

Mac:

```
brew install android-platform-tools
```

Agora é só testar no terminal:

```
adb devices
```

```
→ projetos-github adb devices
List of devices attached
192.168.56.101:5555    device
→ projetos-github
```

# Setup de ambiente

Para saber se temos o ambiente pronto e configurado, é só abrir o terminal e digitar o comando:

**adb devices**

Caso dê erro, é porque precisamos adicionar ao sistema a variável de ambiente indicando onde estão os recursos do Android, que ficam na pasta **/Android/platform-tools** do download que fizemos. Podemos também configurar diretamente no Android Studio.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DE
(venv) → cloud-user adb devices
zsh: command not found: adb
(venv) → cloud-user
```

Precisa configurar o ADB

```
→ projetos-github adb devices
List of devices attached
emulator-5554    device
```

ADB configurado com sucesso

Uma alternativa é abrir o **adb** diretamente da pasta **/Android/platform-tools**.

# Um pouco sobre mim



Maria Clara Bezerra

QE @ Red Hat



Mestre em Ciência da Computação - UFPE - CIn

Graduação em Sistemas de Informação - UPE (Caruaru)

Certificações: BSTQB/CTFL + Oracle Cloud Infrastructure (OCI)

[www.github.com/clarabez](https://github.com/clarabez) || <https://agiletesters.com.br/>

Agora um pouco sobre vocês! :)



Preencher esse formulário:

<https://forms.gle/P4f4Sxs7Bu8Q7gtU9>

(vou colar o link no chat)

# Como serão computadas as notas

- Não se preocupem em fazer atividades para ter NOTAS. O objetivo é o **aprendizado**.
- **Participem** da aula, pode ser interagindo no chat, comentando uma situação vivenciada por você, apresentando um exercício.
- Todas as atividades podem ser feitas por **trio**.
- Neste final de semana teremos 1 plano de teste, e talvez um exercício em **Appium** (se sobrar tempo).
- Vocês podem me enviar a atividade final em **até 15 dias após a aula**. Ou seja, a partir deste final de semana até 23/07.



# Como serão computadas as notas

- As atividades devem ser enviadas para mim por e-mail: [clarinhab@gmail.com](mailto:clarinhab@gmail.com)
- Não esqueçam de, ao envio da atividade, sinalizar a **composição do trio**.
- Podem se ajudar entre si, o importante é que vocês aprendam de forma leve, **divertida e colaborativa**.
- Pode utilizar a linguagem de programação **que você quiser**. O mesmo para framework. A única exigência é utilizar Appium e comandos ADB - até porque são o foco da aula e recursos direcionados aos problemas que vamos estudar.
- Recomendo que deixem os projetos de vocês no GitHub, assim vocês já usam o material da aula para portfólio.





Como vocês definem o que são **testes** para **mobile**?

Como vocês definem o que são **testes** para **mobile**?

Quais os **desafios** que vcs imaginam para este contexto?

# Como vocês definem o que são **testes** para **mobile**?

Quais os **desafios** que vcs imaginam para este contexto?

Quais as **oportunidades** que vcs imaginam para este contexto?

# Marcas e Plataformas



SAMSUNG

NOKIA



# Marcas e Plataformas



Habitantes no planeta: 7.9 bilhões

Linhas telefônicas ativas no mundo: 8.05 bilhões



Habitantes no Brasil: +216M

Linhas telefônicas ativas no Brasil: +258M

[Mobile Operating System Market Share Worldwide](#) - GlobalStats - Abril/2021

[Mundo tem 8.05 bilhões de linhas de celular](#) - Editora Abril, abr/2021

[Estatísticas de Celulares no Brasil](#) - TELECO - Abril/2021

# Marcas e Plataformas



Habitantes no planeta: 7.9 bilhões

Linhas telefônicas ativas no mundo: 8.05 bilhões



Habitantes no Brasil: +216M

Linhas telefônicas ativas no Brasil: +258M



android

**71,59%**



**27,68%**

[Mobile Operating System Market Share Worldwide](#) - GlobalStats - Abril/2021

[Mundo tem 8.05 bilhões de linhas de celular](#) - Editora Abril, abr/2021

[Estatísticas de Celulares no Brasil](#) - TELECO - Abril/2021

# Fragmentação de Plataformas

## Android OS market share

Android OS version	Market share ▾	Change in the last 30 days
11 (Android 11)	34.5 %	No change
10 (Android 10)	22.9 %	↓ 3%
9.0 (Pie)	12.5 %	↓ 4%
8.0-8.1 (Oreo)	9.3 %	↓ 2%
7.0-7.1 (Nougat)	5.1 %	↓ 7%
6.0 (Marshmallow)	2.7 %	↓ 6%
5.0-5.1 (Lollipop)	2.3 %	↓ 8%
4.4 (KitKat)	0.7 %	↓ 7%
4.1-4.3 (Jelly Bean)	0.3 %	↓ 3%
4.0 (Ice Cream Sandwich)	0.0 %	↓ 8%

## iOS and iPadOS usage

As measured by devices that transacted on the App Store on January 11, 2022.

### iPhone

72% of all devices introduced in the last four years use iOS 15.



iOS 15

● 72% iOS 15

● 26% iOS 14

● 2% Earlier

63% of all devices use iOS 15.



iOS 15

● 63% iOS 15

● 30% iOS 14

● 7% Earlier

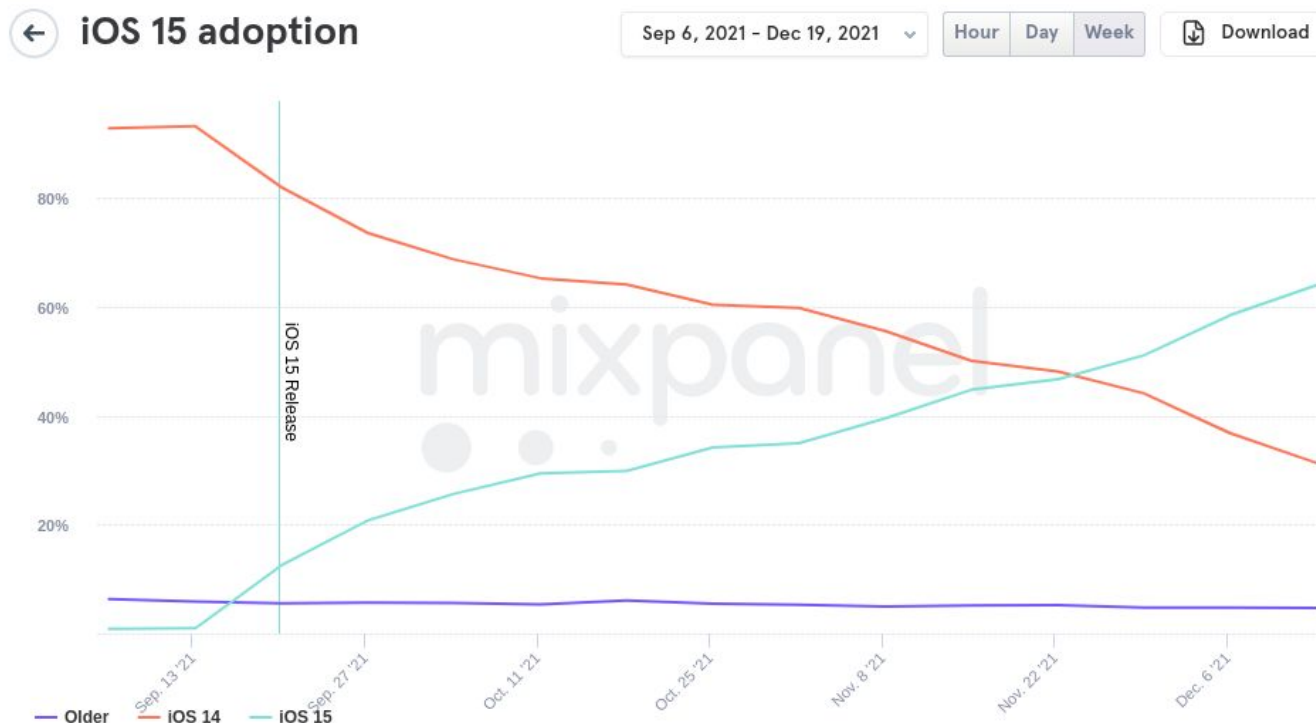


android  
71,59%



iOS  
27,68%

# Fragmentação de Plataformas



Apple iOS  
27,68%

THIS REPORT WAS GENERATED FROM 3,100,907,106,104 RECORDS. • TIME/DATE IN UTC

[iOS 15 adoption](#) - Mixpanel, May/2022

[iOS usage](#) - developer.apple, May/2022



# Fragmentação da Plataforma

COMO TESTAR A MESMA APLICAÇÃO NUM AMBIENTE DIVERSO?



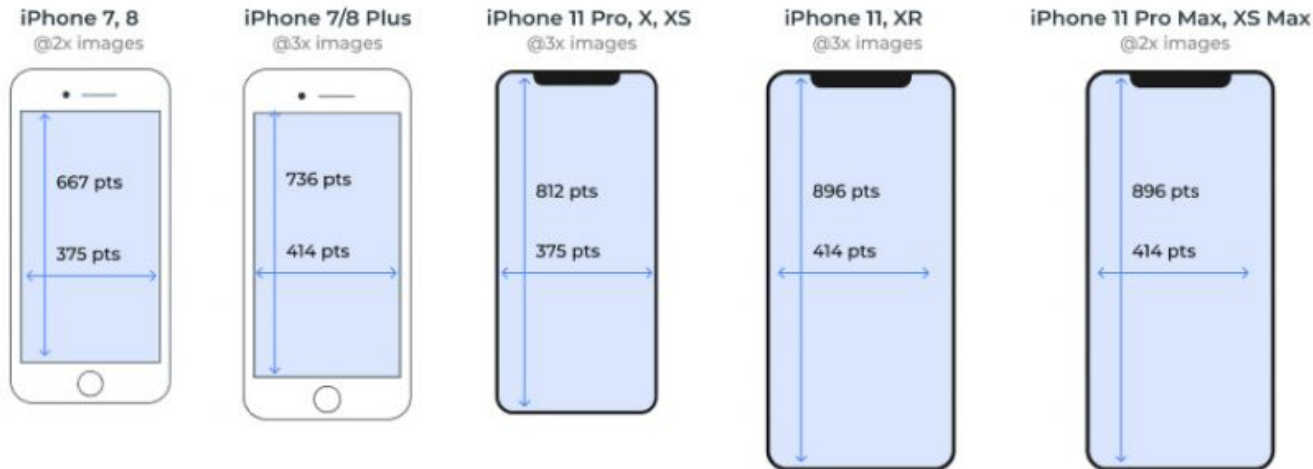
AUTOMAÇÃO



C.E.S.A.R  
school



# Diversidade de telas

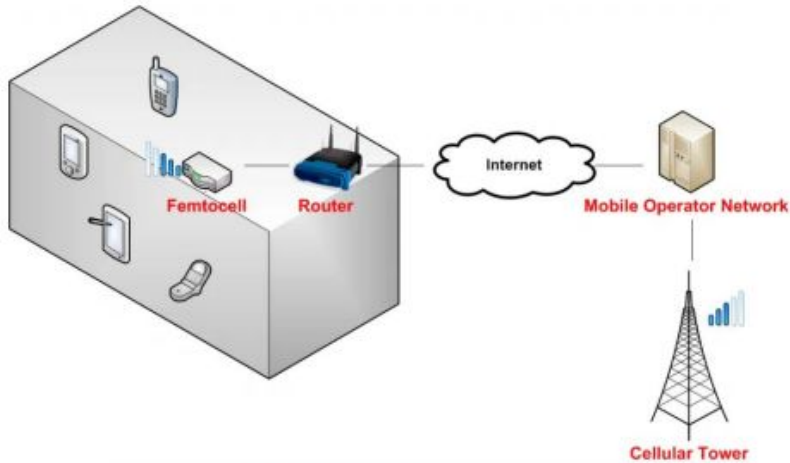


- Tamanho x Resolução
- Formatos
- Orientação (portrait / landscape)

# Redes



2G, 3G, 4G, 5G, WIFI, BT



FEMTOCELL

femtocell\* é com 'm' mesmo :)

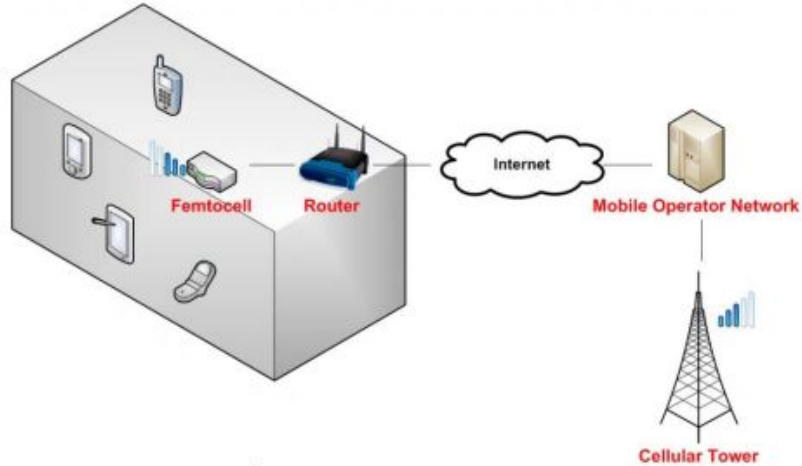


????

# Redes



2G, 3G, 4G, 5G, WIFI, BT



FEMTOCELL

femtocell\* é com 'm' mesmo :)



GAMBIARRAS :)

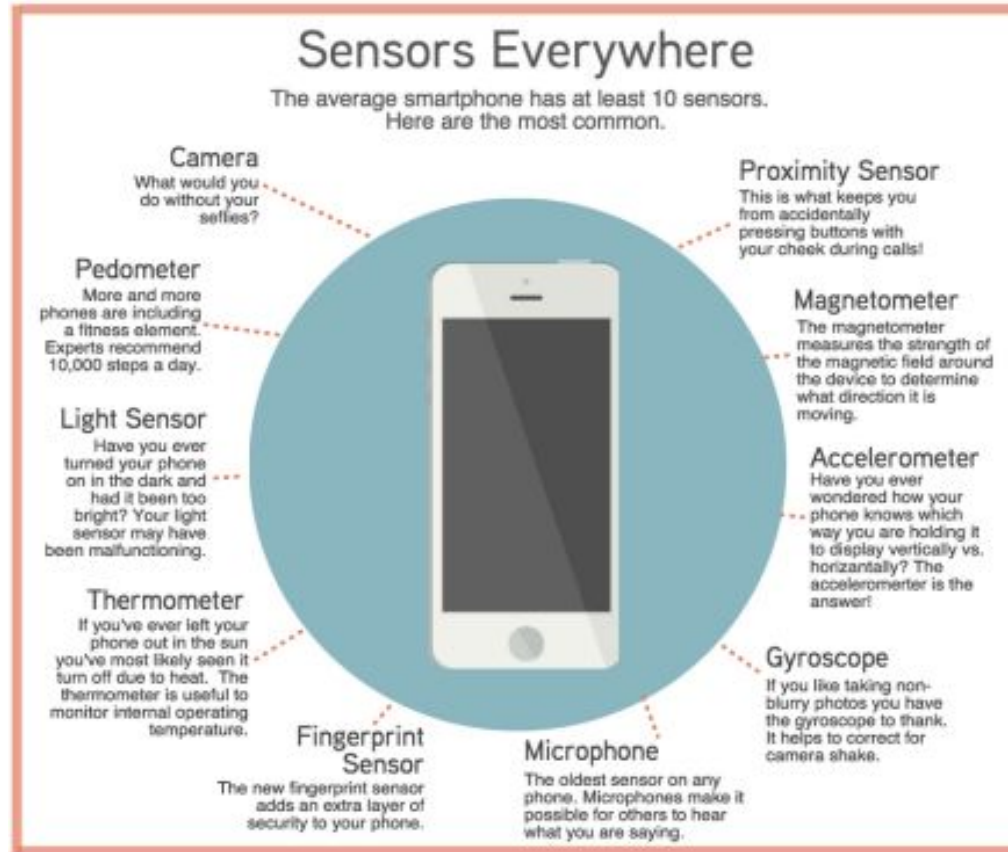
# Recursos externos



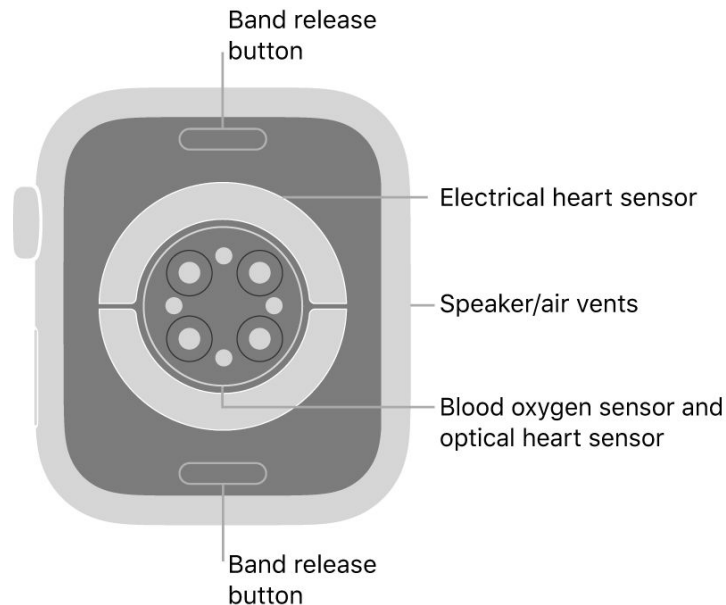
# Recursos internos



# Recursos internos



# Recursos internos



GPS

Acelerômetro

Sensor de oxigenação no sangue

Sensor de cardíaco elétrico (ECG)

Giroscópio (com detecção de queda)

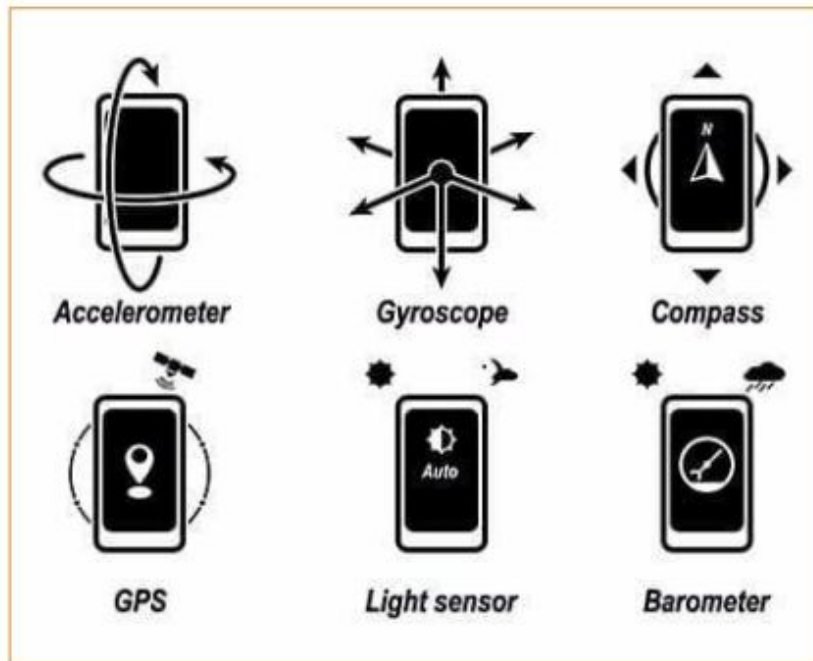
Luz

Ruído



# Recursos internos

## SENSORES



# Recursos internos

Virou assinatura da Motorola



Para abrir a câmera



Para ligar a lanterna

# Gestos na tela



# Tipos de aplicações



App Nativo



Web Application



Hybrid App

- TRANSPORTE: UBER, 99, CARPOOL, CITTAMOBIL ...
- BANCOS: BB, BRADESCO, ITAÚ, INTER, NUBANK ...
- NAMORO: TINDER, HAPPEN ...
- MAPAS: GOOGLE MAPS, WAZE ...
- EDUCAÇÃO: DUOLINGO, UDEMY, GOOGLE CLASSROOM ...
- LAZER: CLASH ROYALE, CANDY CRUSH ...

# Multiplataforma



- FUNCIONALIDADES
- CONSTRUÇÃO
  - TIPOGRAFIA
  - COMPONENTES
- INTERAÇÃO
  - BOTÕES FÍSICOS VS DIGITAIS
  - PADRÕES DE NAVEGAÇÃO
  - GESTOS

# Integração



- COM OUTROS APLICATIVOS
- COM RECURSOS DO DISPOSITIVO
- COM FUNÇÕES DOS DISPOSITIVO

# Automação



# Automação mobile



Robot Framework

Cross-platform



UI TESTING FOR ANDROID  
**espresso**



uiautomator

Aplicações nativas

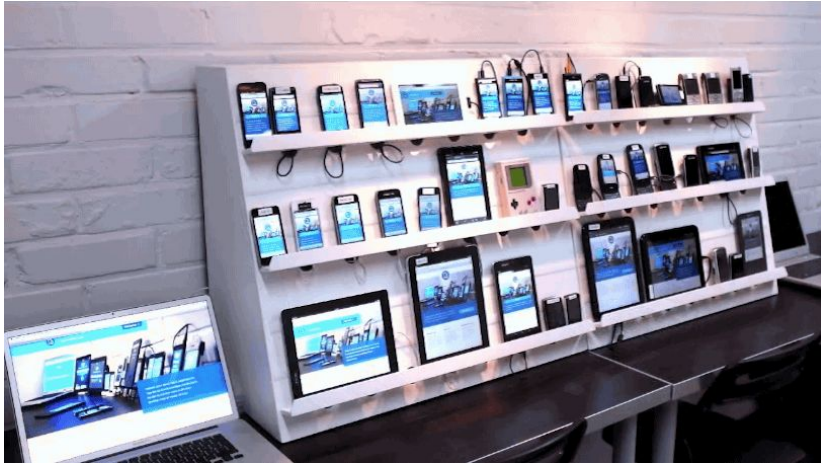


# Como começar um ambiente de automação do zero?



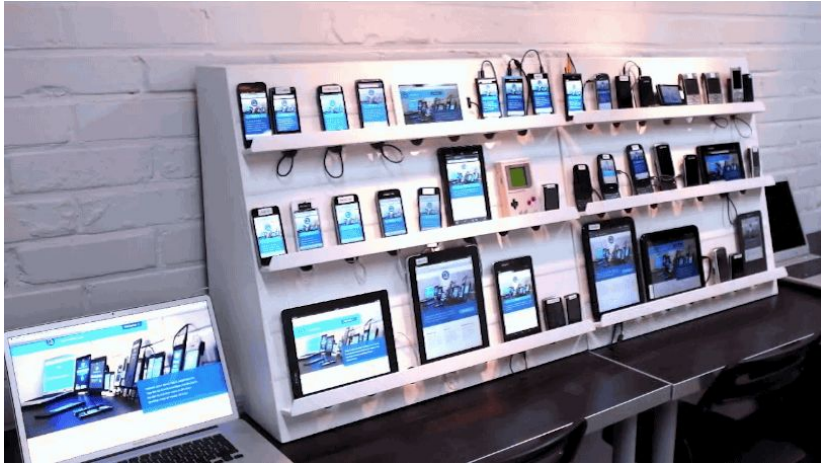
- Quais *frameworks* fazem sentido pro contexto?
- Dispositivos reais ou emulados?
- Apenas UI? API?

# Tipos de dispositivos



Dispositivos reais

# Tipos de dispositivos



Dispositivos reais



Dispositivos emulados

# Tipos de dispositivos

Como podemos emular um dispositivo?

android  
studio



Genymotion  
Desktop



Dispositivos emulados

# Tipos de dispositivos

Como podemos emular um dispositivo?

android  
studio



Genymotion  
Desktop



Dispositivos emulados

Vamos aprender a emular um dispositivo usando as duas ferramentas :)

# Como começar um ambiente de automação do zero?



- Quais *frameworks* fazem sentido pro contexto?
- Dispositivos reais ou emulados?
- Apenas UI? API?

Em um time?

# Sustentabilidade de automação para mobile

PLANEJAR E GERIR IMPACTOS DE MUDANÇAS

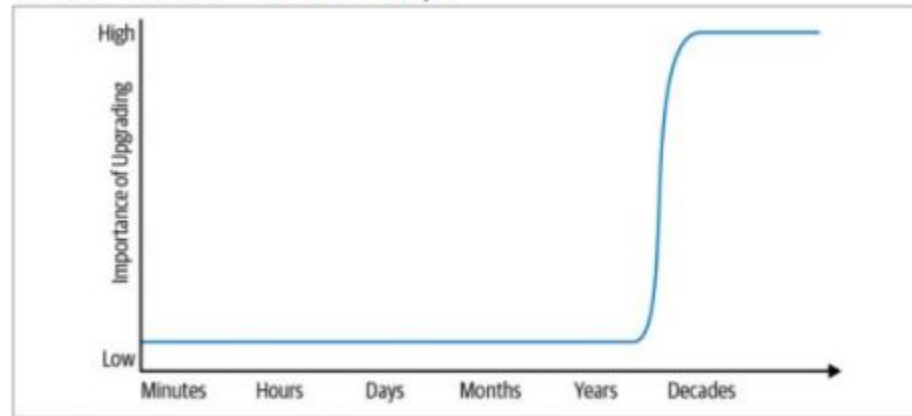


Figure 1-1. Life span and the importance of upgrades

E agora? Vamos definir novamente?



Como vocês definem o que são **testes** para **mobile**?

Quais os **desafios** que vcs imaginam para este contexto?

Quais as **oportunidades** que vcs imaginam para este contexto?



# Alguns erros para evitarmos em testes para mobile



❌ TENTAR TESTAR TUDO

❌ MOBILE NÃO É WEB

❌ NÃO CONSIDERE APENAS UI

❌ NÃO DESCONSIDERE TESTES DE UPDATE/INSTALAÇÃO

❌ NÃO TENHA TENTADO AUTOMATIZAR TUDO

❌ NÃO DESCONSIDERE TESTES DE SEGURANÇA

❌ NÃO DESCONSIDERE SEUS USUÁRIOS

❌ NÃO CONSIDERE APENAS UM TIPO DE PRODUTO

# Algumas boas práticas em testes para mobile



- ✓ CONHECER USUÁRIOS E USUÁRIAS
- ✓ CONSIDERAR INTERRUPÇÕES E NÍVEIS DE BATERIA
- ✓ CRIE GRUPOS DE PRODUTOS
- ✓ CONSIDERAR SENSORES
- ✓ CONSIDERAR VARIAÇÕES DE REDE
- ✓ CONSIDERAR RESTRIÇÕES DE PERMISSÃO
- ✓ VARIAÇÕES DE IDIOMA
- ✓ USABILIDADE COM MULTIUSUÁRIOS

#tbt

## Samsung finalmente explica como o Galaxy Note 7 virou uma “bomba”

Sim, a culpa é mesmo da bateria

Por Emerson Alecrim  
23/01/2017 às 15:12

NEWS



# Novas tendências

## DISPOSITIVOS DE TELA DOBRÁVEL



# Tipos de testes para dispositivos móveis

## PODEMOS E DEVEMOS IR ALÉM DOS TESTES FUNCIONAIS

- ✓ TESTE DE COMPATIBILIDADE
- ✓ TESTE DE INSTALAÇÃO
- ✓ TESTE DE INTERRUPÇÃO
- ✓ TESTE DE LOCALIZAÇÃO
- ✓ TESTE DE PERFORMANCE
- ✓ TESTE DE CONFORMIDADE



# Atividade #1



## Planejamento e design de testes para dispositivos móveis.

Tipo: individual, dupla ou trio

Entregável: Documento do plano (vamos apresentar aqui na aula mas vcs me enviam via e-mail também).

### Descrição da atividade:

- Acessar a [Play Store](#) e escolher um aplicativo da lista de top free (vcs decidem qual);
- Com base no que vimos, elabore um plano de testes para seu aplicativo;
- Escolha 1 ou 2 funcionalidades (pode considerar as principais);
- Proponha 1 cenário de teste (use a criatividade);
- Para os seus testes, vc optaria por dispositivo real ou emulado (lembre-se da cobertura de testes)?
- Pensando na sua estratégia de testes, quais riscos na escolha do tipo de dispositivo?
- Quais tipos de testes podemos ter no seu planejamento? Funcional, stress, carga, usabilidade, etc.
- Dentre os cenários elaborados, quais vc automatizaria? Quais vc não automatizaria?



# Atividade #1



- ✓ PLANO DE TESTES É O DOCUMENTO CENTRAL DE TESTES
- ✓ CONSIDERE: O QUE TEM QUE SER FEITO ANTES DE INICIAR A EXECUÇÃO, RECURSOS, VERSÕES DE SOFTWARE, RISCOS E COMO SUPERÁ-LOS.
- ✓ IDENTIFIQUE CASOS DE TESTE QUE VC INDICARIA PARA AUTOMAÇÃO.
- ✓ SOMOS CRIATIVOS PARA MOBILE. LEMBRE-SE DE SENSORES, RECURSOS INTERNOS E EXTERNOS, NOTIFICAÇÕES, MODO AVIÃO, ATUALIZAÇÃO DO SO, CHAMADAS.

# Atividade #1



Agora vcs podem apresentar as propostas para a turma toda! :)



# Comandos ADB

Android Debug Bridge



É uma ferramenta de **linha de comando**, nativa do Android SDK, que permite a **comunicação** entre a pessoa desenvolvedora e o **dispositivo** (real ou emulado) conectado.



<https://github.com/clarabez/comandosadb>

# Comandos ADB

Android Debug Bridge



É uma ferramenta de linha de comando, nativa do Android SDK, que permite a **comunicação** entre a pessoa desenvolvedora e o **dispositivo** (real ou emulado) conectado.

## O que podemos fazer com ADB?

- Instalar/desinstalar aplicativos;
- Mudar configurações internas;
- Habilitar/desabilitar funções de conexões;
- Reiniciar/desligar dispositivos;
- Capturar logs;
- Consultar qualquer informação sobre o dispositivo.



<https://github.com/clarabez/comandosadb>

# Comandos ADB

Android Debug Bridge



É uma ferramenta de linha de comando, nativa do Android SDK, que permite a comunicação entre a pessoa desenvolvedora e o dispositivo (real ou emulado) conectado.

## O que podemos fazer com ADB?

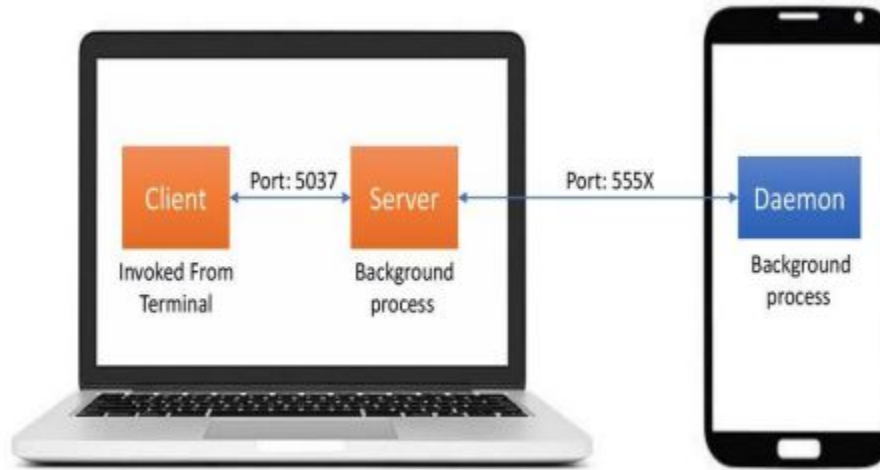
- Instalar/desinstalar aplicativos;
- Mudar configurações internas;
- Habilitar/desabilitar funções de conexões;
- Reiniciar/desligar dispositivos;
- Capturar logs;
- Consultar qualquer informação sobre o dispositivo.



<https://github.com/clarabez/comandosadb>

# Comandos ADB

Android Debug Bridge



Arquitetura cliente-servidor

# Comandos ADB

Android Debug Bridge



- ✓ SÃO SIMPLES E SUPER RÁPIDOS
- ✓ BASTA UMA CLI (COMMAND LINE INTERFACE)
- ✓ PODEM AUTOMATIZAR ATIVIDADES DE ROTINA
- ✓ FERRAMENTA OFICIAL DO ANDROID

# Comandos ADB

Android Debug Bridge



- ✓ SÃO SIMPLES E SUPER RÁPIDOS
- ✓ BASTA UMA CLI (COMMAND LINE INTERFACE)
- ✓ PODEM AUTOMATIZAR ATIVIDADES DE ROTINA
- ✓ FERRAMENTA OFICIAL DO ANDROID

Vamos tentar?

# Vamos instanciar um dispositivo emulado?



- ✓ ABRIR ANDROID STUDIO
- ✓ ANDROID AVD
- ✓ E VAMOS PRA LÁ QUE FICA MELHOR DE EXPLICAR :)



# Comandos ADB

Android Debug Bridge



## Listar dispositivos conectados:

```
adb devices
```

## Reiniciar dispositivo:

```
adb reboot
```

## Instalar uma aplicação (apk):

```
adb install -r nome_do_app.apk
```

## Listar pacotes:

```
adb shell pm list package
```

## Gerar log:

```
adb logcat
```

## Gerar um bugreport

```
adb bugreport
```

## Passar string:

```
adb shell input text 'texto'
```

## Desbloquear tela:

```
adb shell input keyevent 66
```

## Bloquear a tela:

```
adb shell input keyevent 26
```

## Obter informações do aparelho:

```
adb shell getprop
```





# Comandos ADB

Android Debug Bridge



# Monkey testing



# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de estresse nos aplicativos em desenvolvimento.

# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de **estresse** nos aplicativos em **desenvolvimento**.

Sintaxe básica:

```
$ adb shell monkey [options]  
<event-count>
```

# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de **estresse** nos aplicativos em **desenvolvimento**.

Sintaxe básica:

```
$ adb shell monkey [options]  
<event-count>
```

# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de **estresse** nos aplicativos em **desenvolvimento**.

Sintaxe básica:

```
$ adb shell monkey [options]  
<event-count>
```

# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de **estresse** nos aplicativos em **desenvolvimento**.

Sintaxe básica:

```
$ adb shell monkey [options]  
<event-count>
```

Opções de depuração  
(podemos ver a lista no link abaixo)

# Monkey testing



O Monkey é uma ferramenta de linha de comando (CLI) executado no dispositivo (emulado ou real) e que gera fluxos pseudoaleatórios de eventos de usuário, toques ou gestos, bem como vários eventos do sistema;

Pode ser utilizado para testes de **estresse** nos aplicativos em **desenvolvimento**.

Sintaxe básica:

```
$ adb shell monkey [options]  
<event-count>
```

Número de eventos



# Monkey testing



Vamos testar:

```
$ adb shell monkey 500 -v
```

Agora especificando um pacote:

```
$ adb shell pm list package |grep espresso
```

```
$ adb shell monkey -p com.example.android.testing.espresso.BasicSample -v 5
```

# Monkey testing



Vamos testar com o número do seed. Observe que na saída temos o seguinte campo:

```
# adb shell monkey -v 100
bash arg: --hprof
bash arg: -v
bash arg: 100
args: [--hprof, -v, 100]
arg: "--hprof"
arg: "-v"
arg: "100"
arg="--hprof"          mCurArgData="null"          mNextArg=1
argwas="--hprof" nextarg="-v"
:Monkey: seed=1655868414131 count=100
```

Copie e cole esse valor e adicione ao seu comando anterior:

```
$ adb shell monkey -s 1655868414131 500
```

<https://developer.android.com/studio/test/other-testing-tools/monkey>

# Automação mobile



Robot Framework

Cross-platform



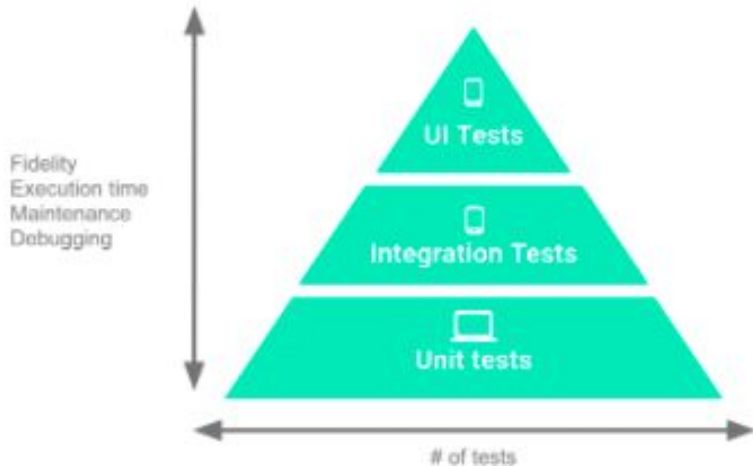
UI TESTING FOR ANDROID  
**espresso**



uiautomator

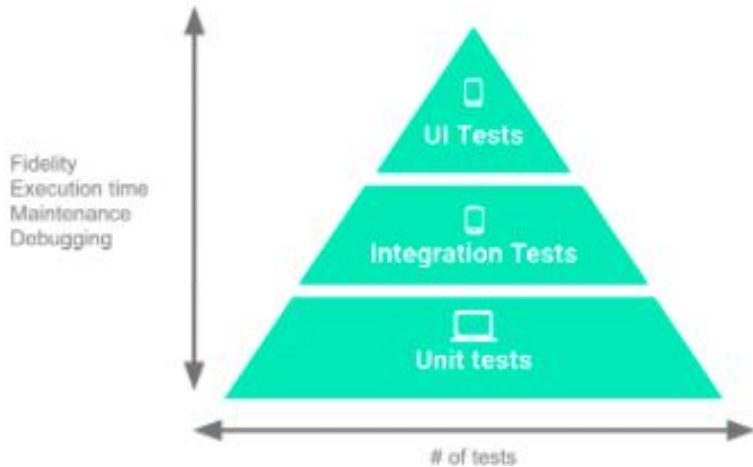
Aplicações nativas

# Lembram dessa imagem?

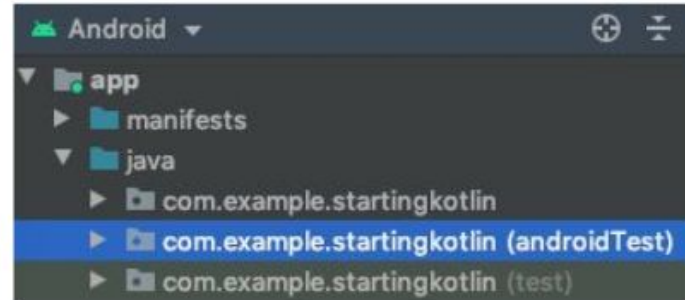


"Embora a proporção de testes para cada categoria possa variar de acordo com os casos de uso do app, geralmente recomendamos a seguinte divisão entre as categorias: **70% pequenos**, **20% médios** e **10% grandes**." (Google)

# Lembram dessa imagem?

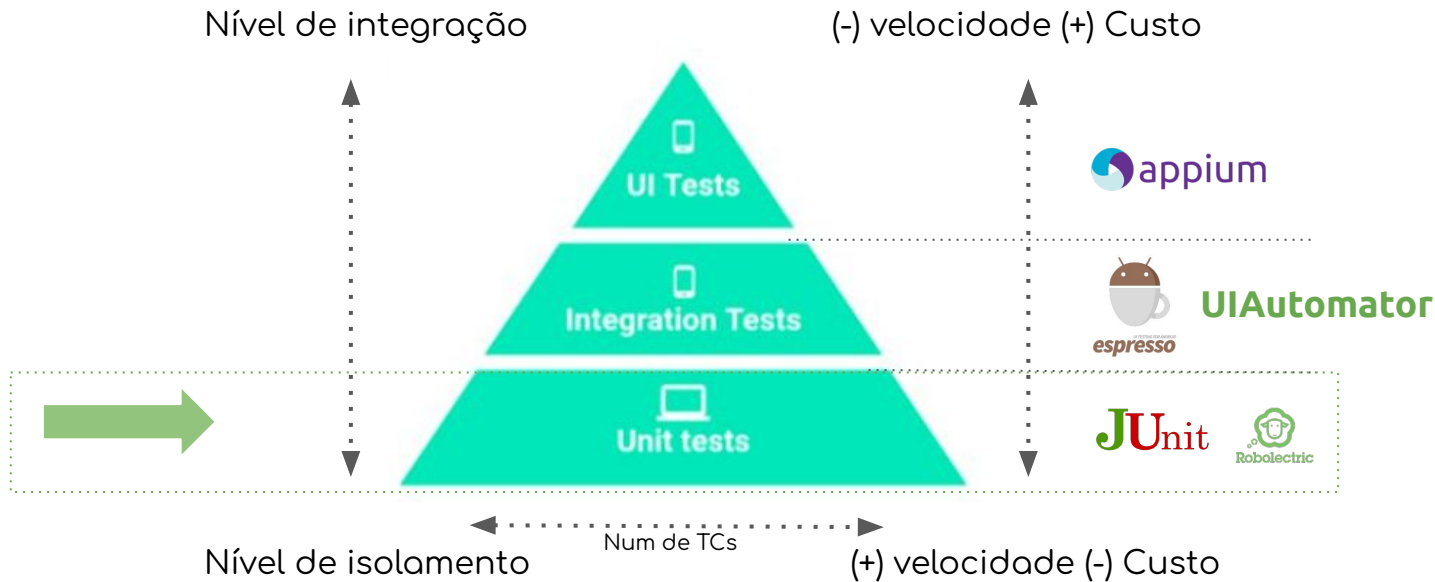


"Embora a proporção de testes para cada categoria possa variar de acordo com os casos de uso do app, geralmente recomendamos a seguinte divisão entre as categorias: 70% pequenos, 20% médios e 10% grandes." (Google)



androidTest = instrumentados  
test = não instrumentados

# Pirâmide de testes para mobile



# Roboelectric



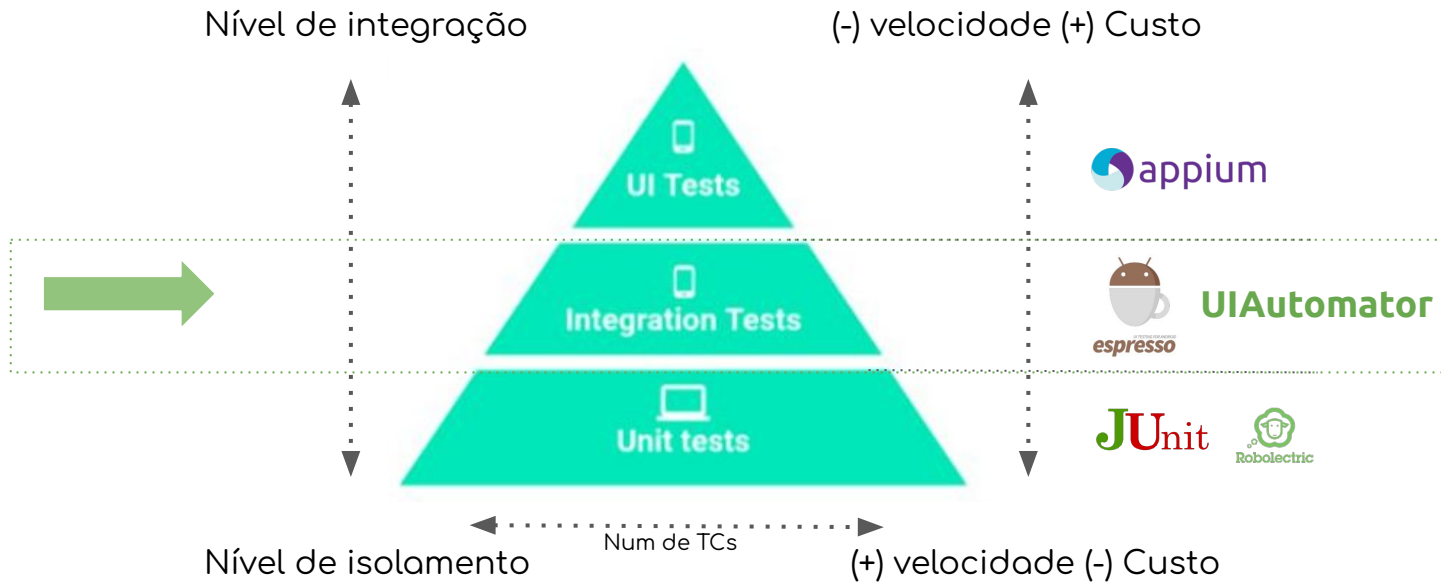
Framework de testes que permite que recursos específicos do Android sejam simulados.

Viu agora a diferença entre **emulado** e **simulado**?

Faz testes que dependem de classes específicas do Android sem precisar carregar um dispositivo.

<http://roboelectric.org/>

# Pirâmide de testes para mobile







# Espresso



Ferramenta de automação de testes de UI feitas para pessoas desenvolvedoras com mais familiaridade com o código JAVA/Kotlin. Caixa preta ou branca.

É um recurso que faz parte do Android SDK que baixamos;

O que difere o Espresso das outras ferramentas é que ele consegue inicializar uma tela diretamente sem passar por todo o fluxo até chegar na tela.

É composto por 3 componentes: **ViewAction**, **ViewMatcher** e **ViewAssertion**.



# Espresso



Ferramenta de automação de testes de UI feitas para pessoas desenvolvedoras com mais familiaridade com o código JAVA/Kotlin. Caixa preta ou branca.

É um recurso que faz parte do Android SDK que baixamos;

O que difere o Espresso das outras ferramentas é que ele consegue inicializar uma tela diretamente sem passar por todo o fluxo até chegar na tela.

É composto por 3 componentes: `ViewAction`, `ViewMatcher` e `ViewAssertion`.

## Bora testar!

<https://developer.android.com/training/testing/espresso/basics>



# Espresso



Clonar o projeto (vamos usar também em outros momentos da aula):

```
# git clone https://github.com/android/testing-samples/
```

Abrir Android Studio e abrir o projeto do Espresso:

File > Open > testing-samples/ui/espresso/BasicSample/build.gradle

Agora espera carregar o projeto (pode levar uns 5 minutos)

Issues conhecidas:

The project is using an incompatible version (AGP 7.3.0-beta01) of the Android Gradle plugin. Latest supported version is AGP 7.2.1

É só ajustar o build.gradle na mão a versão do AGP:

```
buildscript {  
    //... dependencies {  
        classpath 'com.android.tools.build:gradle:7.2.1' // ...  
    }  
}
```



# Espresso



```
onView(ViewMatcher) .perform(ViewAction) .check(ViewAssertion  
)
```

## User properties:

```
withID(..)  
withText(..)  
withTagValue(..)  
hasLinks(..)
```

## UI properties:

```
isDisplayed()  
isEnabled()  
hasFocus()  
isChecked()
```

## Hierarchy:

```
withParent(Matcher)  
withChild(Matcher)  
hasDescendant(Matcher  
)
```

[ViewMatchers](#). Android Developer, 2022.

[ViewActions](#). Android Developer, 2022.

[ViewAssertion](#). Android Developer, 2022.



# Espresso



```
onView (ViewMatcher) . perform (ViewAction) . check (ViewAssertion  
)
```

## Click/Press:

```
click()  
doubleClick()  
longClick()  
pressBack()
```

## Gestures:

```
scrollTo()  
swipeLeft()  
swipeRight()  
swipeUp()  
swipeDown()
```

## Text:

```
clearText()  
typeText (String)  
replaceText ()
```

[ViewMatchers](#). Android Developer, 2022.  
[ViewActions](#). Android Developer, 2022.  
[ViewAssertion](#). Android Developer, 2022.



# Espresso



```
onView(ViewMatcher) . perform(ViewAction) . check(ViewAssertion  
)
```

## General:

```
matches()  
doesNotExist()
```

## Layout:

```
noEllipsizedText()  
noOverlaps()
```

## Position:

```
isAbove()  
isBelow()  
ifLeftOf()  
isPartialAbove()
```

[ViewMatchers](#). Android Developer, 2022.  
[ViewActions](#). Android Developer, 2022.  
[ViewAssertion](#). Android Developer, 2022.



# Espresso



```
@Test
fun changeText_sameActivity() {

    // Type text and then press the button.
    onView(withId(R.id.editTextUserInput))
        .perform(typeText(StringToBeTyped), closeSoftKeyboard())
    onView(withId(R.id.changeTextBt)).perform(click())

    // Check that the text was changed.
    onView(withId(R.id.textToBeChanged)).check(matches(withText(StringToBeTyped)))
}
```

*onView(): "Public method that creates a ViewInteraction for a given view."*



# Espresso



Vamos praticar um pouco?



1 - Clonar o projeto:

git clone <https://github.com/clarabez/appium-android-app.git>

2 - Buildar o projeto:

File > Open > cursodeappium/build.gradle

Agora espera carregar o projeto (pode levar uns 5 minutos)

3 - Testes existentes com o Espresso:

- Cadastrar pessoa válida;
- Cadastrar com dados inválidos;

4 - Vamos criar os testes que estão vazios?

Temos alguns cenários por lá :)





# Espresso Recorder



- Ferramenta que permite criar testes de UI sem precisar escrever códigos de teste, gravando interações com a tela e convertendo em código (JAVA ou Kotlin);
- Utiliza o Espresso;
- É possível adicionar *asserts*;
- Para usarmos: **Run > Record Espresso Test**;
  
- **Minha opinião:** particularmente tive que fazer alguns ajustes quando tentei usar para ver como que era, e também achei lento.

# UIAutomator



Framework recomendado para **testes funcionais** (caixa preta) de UI em apps nativos (feitos em Android para Android);

É possível, no mesmo teste, percorrer por diferentes aplicações;

Foca em testes de caixa preta (onde não temos detalhes de implementação do app em teste);

Possui um visualizador chamado *uiautomatorviewer* que fica localizado em: `<android-sdk>/tools/bin`

<https://developer.android.com/training/testing/ui-automator>

# UIAutomator



Framework recomendado para **testes funcionais** (caixa preta) de UI em apps nativos (feitos em Android para Android);

É possível, no mesmo teste, percorrer por diferentes aplicações;

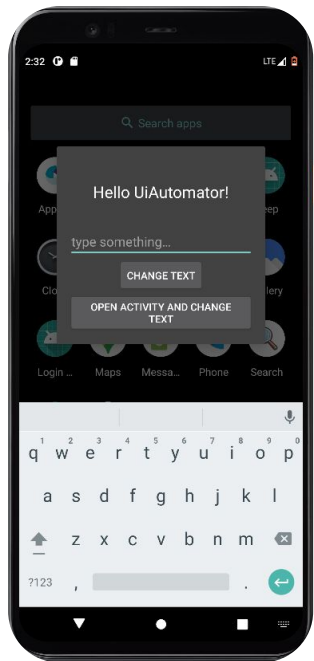
Foca em testes de caixa preta (onde não temos detalhes de implementação do app em teste);

Possui um visualizador chamado *uiautomatorviewer* que fica localizado em: `<android-sdk>/tools/bin`

## Bora testar!

<https://developer.android.com/training/testing/ui-automator>

# UIAutomator



Projeto que clonamos anteriormente:

```
# git clone https://github.com/android/testing-samples/
```

Abrir Android Studio e abrir o projeto de UIAutomator:

File > Open > testing-samples/ui/uiautomator/BasicSample/build.gradle

Agora espera carregar o projeto (pode levar uns 5 minutos)

Issues conhecidas:

The project is using an incompatible version (AGP 7.3.0-beta01) of the Android Gradle plugin. Latest supported version is AGP 7.2.1

É só ajustar o build.gradle na mão a versão do AGP:

```
buildscript {  
    //... dependencies {  
        classpath 'com.android.tools.build:gradle:7.2.1' // ...  
    }  
}
```

# UIAutomator



Agora vamos realizar o mesmo teste de cadastrar uma pessoa no App com o **UIAutomator**?

Assim podemos ver melhor as diferenças e semelhanças entre o Espresso ou o **UIAutomator**.

## Bora testar!

# Diferença entre o Espresso e o UIAutomator



**UIAutomator** consegue testar em escopo bem mais abrangente, vai além da aplicação que estamos desenvolvendo;

Podemos escrever um teste acessando configurações do dispositivo através do menu, acessar outro aplicativo, etc;

Podemos escrever os testes no mesmo projeto do aplicativo em desenvolvimento.



**Espresso** testa UI num escopo mais isolado;

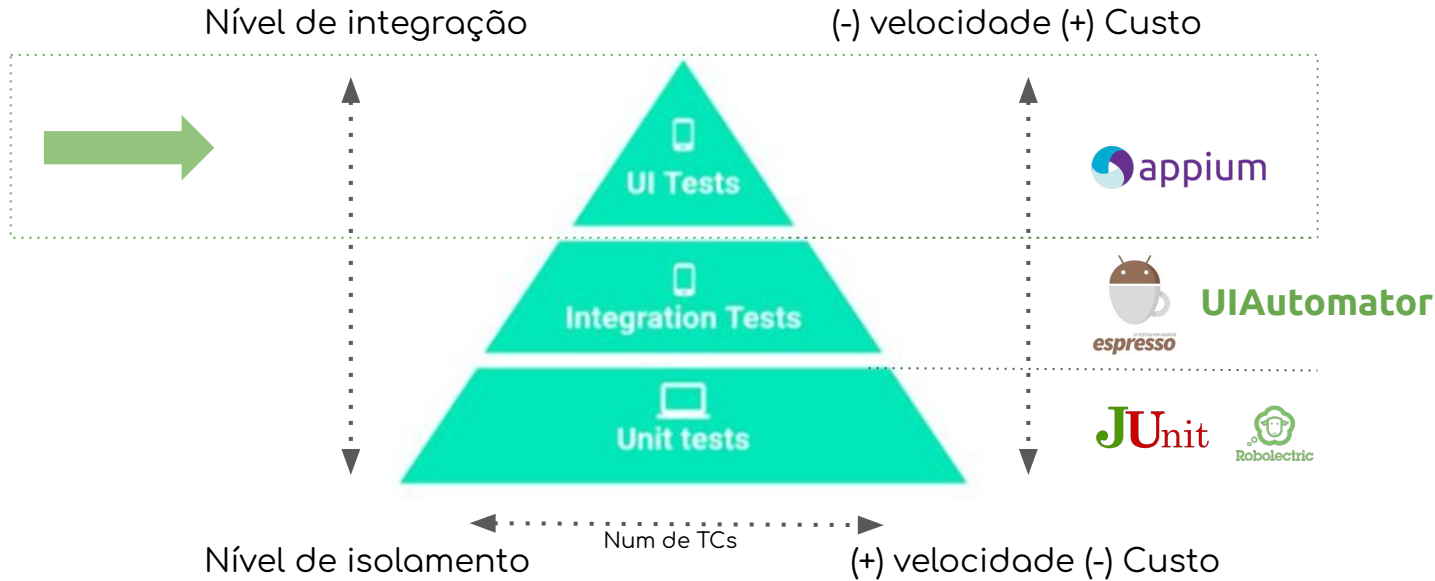
# Diferença entre o Espresso e o UIAutomator



Quem vocês acham que é o melhor?



# Pirâmide de testes para mobile







# Appium



Testes fora do Android Studio;

Pode utilizar outras linguagens que não JAVA/Kotlin;

Seu time de QA é independente do time de desenvolvimento? Appium é uma boa alternativa!

Podemos fazer testes para Android, iOS e web.



# Appium



Testes fora do Android Studio;

Pode utilizar outras linguagens que não JAVA/Kotlin;

Seu time de QA é independente do time de desenvolvimento? Appium é uma boa alternativa!

Podemos fazer testes para Android, iOS e web.

## Vamos conhecer?



# Appium



Appium Desktop (server):

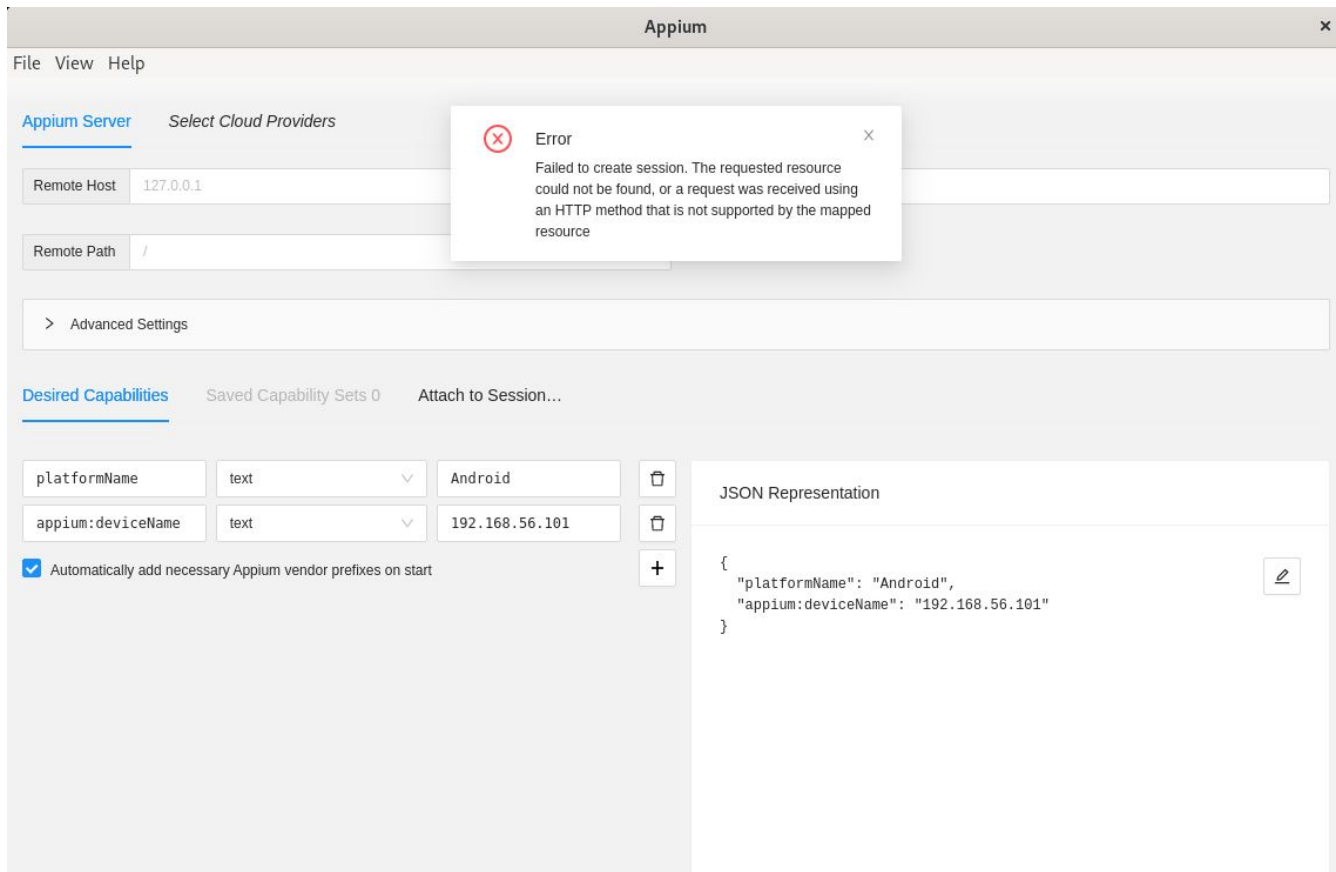
<https://github.com/appium/appium-desktop/releases/>

Appium Inspector - Download:

<https://github.com/appium/appium-inspector/releases>

(Até setembro/2021 essas funções eram integradas, mas foram separadas para melhoria de performance)

# Appium - problemas comuns



The screenshot shows the Appium web interface. At the top, there's a menu bar with 'File', 'View', and 'Help'. Below it, the 'Appium Server' section is active, showing 'Select Cloud Providers'. The 'Remote Host' field is set to '127.0.0.1' and 'Remote Path' is '/'. An error dialog box is displayed in the center, stating: 'Error: Failed to create session. The requested resource could not be found, or a request was received using an HTTP method that is not supported by the mapped resource'. Below the error, there's an 'Advanced Settings' link. The 'Desired Capabilities' section is also visible, showing 'Saved Capability Sets 0' and 'Attach to Session...'. Under 'Desired Capabilities', there are two rows of fields: 'platformName' (text) set to 'Android' and 'appium:deviceName' (text) set to '192.168.56.101'. A checkbox 'Automatically add necessary Appium vendor prefixes on start' is checked. To the right, the 'JSON Representation' section shows the following JSON object: { "platformName": "Android", "appium:deviceName": "192.168.56.101" }

## Solução:

No Appium Inspector preencher os campos:

Remote host: localhost

Remote path: /wd/hub

Remote Port: 4723

Advanced Settings:

[x] Allow Unauthorized Certificates

No Appium server:

Advanced>

Server address: localhost

Port: 4723

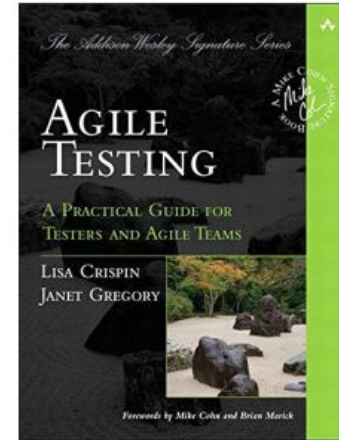
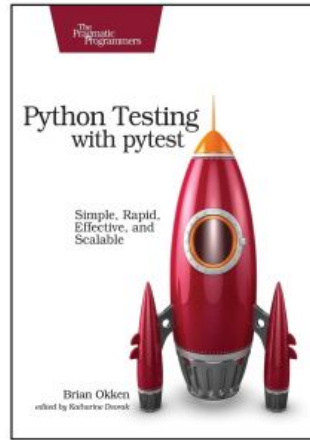
Salvar, preencher capabilities e Start Session.

# Comparando os frameworks

<b>Critério</b>	<b>Appium</b>	<b>Espresso</b>	<b>UIAutomator</b>
<b>Tempo de Execução</b>	Lento	Rápido	Médio
<b>Linguagens suportadas</b>	Python, JAVA, Kotlin, C#, JS, Ruby, Robot	JAVA, Kotlin	JAVA, Kotlin
<b>Tipos de testes</b>	Caixa preta	Caixa Cinza	Caixa Preta
<b>Dificuldade de setup</b>	Alta	Baixa (faz parte do projeto Android)	Baixa (faz parte do projeto Android)

# Referências

- [Meetups of Ministry of Testing - Recife](#)
- [Blog do CESAR School](#)
- [Android Developers](#)
- [Github Maria Clara - Comandos ADB](#)
- [Developer Android - Testing](#)





## **Tópicos especiais I:** testes e automação para dispositivos móveis

Pós graduação em Testes Ágeis  
08 e 09 de julho de 2022

Prof. Maria Clara Bezerra

[www.github.com/clarabez](https://www.github.com/clarabez)