

# Screenshot Testing in Jetpack Compose

*Lights and Shadows*



# Olmo Gallegos

**Senior Android Dev @ Klisst**

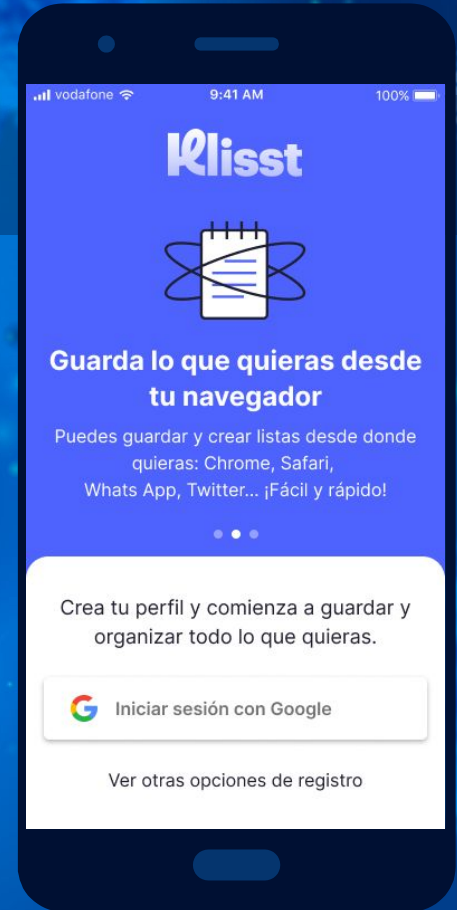
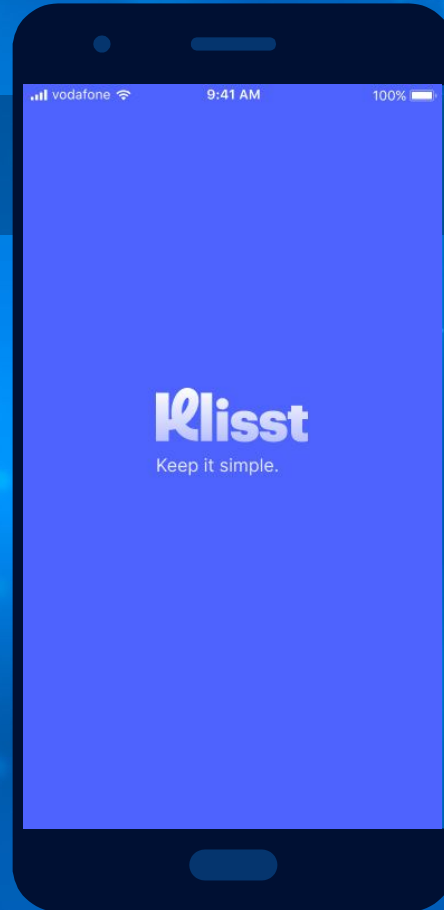
*Formerly Amovens, Devengo, Telefónica I+D, and others*

OSS enthusiast & maintainer. I love to give talks

You can find me at [@olmoDev](https://twitter.com/olmoDev)

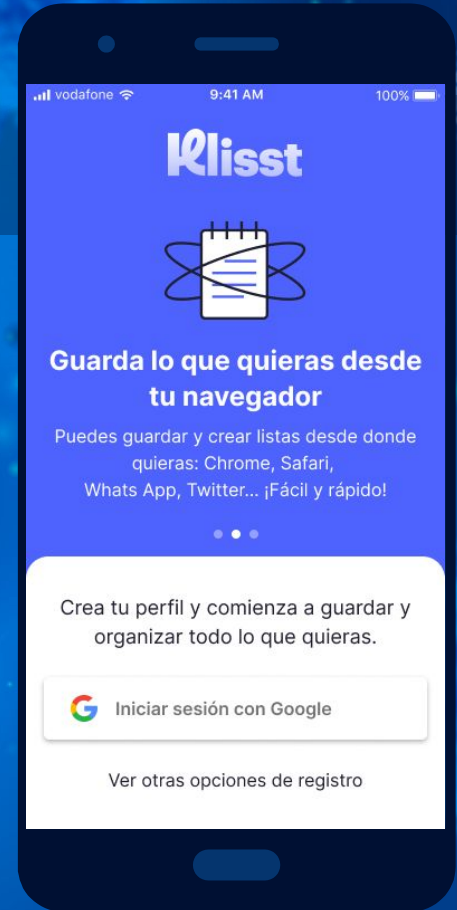
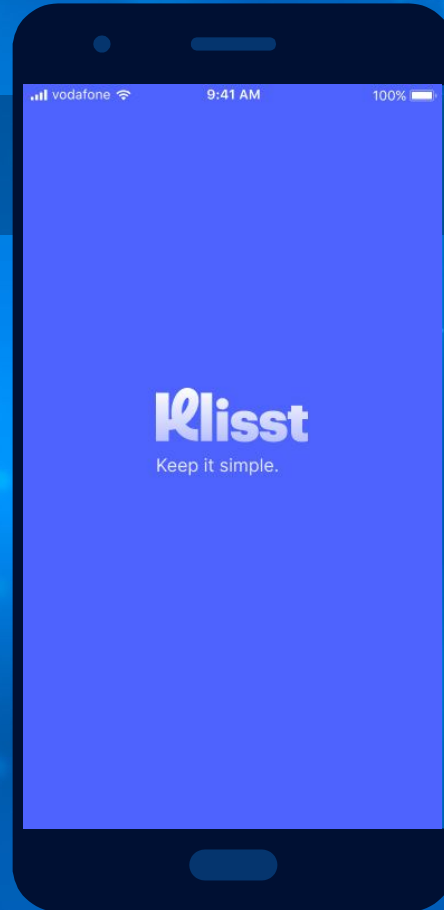
# Klisst: Brief intro

- ▶ Create Lists
- ▶ Add items to Lists
- ▶ Save items to your Lists
- ▶ Share Lists



# Klisst: Brief intro

- ▶ Create Lists
- ▶ Add items to Lists
- ▶ Save items to your Lists
- ▶ Share Lists
- ▶ **New!** Sort items



# Structure

## 1.- Screenshot Testing

Definition and  
differences with other  
Testing techniques

## 2.- Our approach

The way we test our  
code.

## 3.- Lights and shadows

What we like most, what  
you should have in mind.

# 1. Screenshot Testing

A different way of Testing our code



# Screenshot Testing vs Unit Testing

Write a Test

*Unit Test*

Execute

0

1

Failure / Success

# Screenshot Testing vs Unit Testing

## *Unit Test*

Write a Test

0

Execute

1

Failure / Success

## *Screenshot Test*

Write a Test

0

?

?



# Screenshot Testing vs Unit Testing

## *Unit Test*

Write a Test

0

Execute

1

Failure / Success

## *Screenshot Test*

Write a Test

0

Record

1

Compare

?

# Screenshot Testing vs Unit Testing

## *Unit Test*

Write a Test

0

Execute

1

Failure / Success

## *Screenshot Test*

Write a Test

0

Record

1

Develop

2

Compare

3

# Screenshot Testing vs Unit Testing

Write a Test

0

Record

1

[commit]

Develop

2

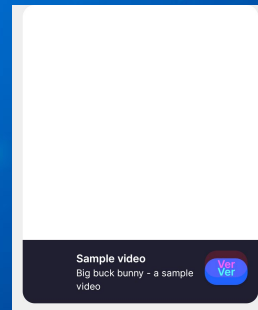
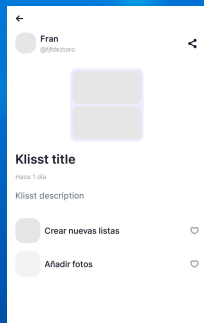
Compare

3

verification

```
@Test
fun rendersKlisstDetailForAnyKlisstWithTwoPhotos() {
    renderScreen(KlisstMother.anyKlisstWithTwoPhotos)

    compareScreenshot(composeRule)
}
```



## 2. The Klisst way

Our approach to Testing and Android architecture

“

```
implementation "androidx.compose.ui:ui:$compose_version"
implementation "androidx.activity:activity-compose:1.3.1"
implementation "androidx.navigation:navigation-fragment-ktx:$navigation_version"

implementation "org.jetbrains.kotlinx:kotlinx-coroutines-core:$coroutines_version"

implementation "com.google.dagger:hilt-android:$dagger_hilt_version"
implementation "androidx.hilt:hilt-common:$dagger_hilt_lifecycle_version"
implementation "androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha03"
kapt "androidx.hilt:hilt-compiler:$dagger_hilt_lifecycle_version"

implementation "io.arrow-kt:arrow-generic:$arrow_version"
implementation "io.swagger:swagger-annotations:1.6.2"
implementation "com.google.firebase:firebase-auth-ktx"
```


“

```
testImplementation "io.mockk:mockk:1.10.6"
testImplementation "com.squareup.okhttp3:mockwebserver:5.0.0-alpha.2"
testImplementation "org.robolectric:robolectric:4.5.1"
testImplementation "androidx.test.espresso:espresso-core:3.3.0"
testImplementation "com.schibsted.spain:barista:3.4.0"
testImplementation "org.jetbrains.kotlinx:kotlinx-coroutines-test:1.4.3"
testImplementation "com.google.dagger:hilt-android-testing:$dagger_hilt_version"
testImplementation "androidx.navigation:navigation-testing:$navigation_version"

androidTestImplementation "androidx.test.espresso:espresso-core:3.3.0"
androidTestImplementation "org.jetbrains.kotlinx:kotlinx-coroutines-test:$coroutines_version"
androidTestImplementation "com.google.dagger:hilt-android-testing:$dagger_hilt_version"
androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_version"
androidTestImplementation "androidx.navigation:navigation-testing:$navigation_version"
```



README.md

 **Shot** Build, lint, and test gradle

Shot is an Android project you can use to write screenshot for your apps in a simple and friendly way.







### What is this?


Shot is a Gradle plugin and a core android library thought to run screenshot tests for Android. This project provides a handy interface named `ScreenshotTest` and a ready to use `ShotTestRunner` you can use in order to write tests like these:

```
class GreetingScreenshotTest : ScreenshotTest {  
  
    // If you are using regular Android views  
  
    @Test  
    fun theActivityIsShownProperly() {  
        val mainActivity = startMainActivity();  
  
        compareScreenshot(activity);  
    }  
  
    // If you are using Jetpack Compose  
  
    @Test  
    fun rendersGreetingMessageForTheSpecifiedPerson() {  
        composeRule.setContent { Greeting(greeting) }  
  
        compareScreenshot(composeRule)  
    }  
}
```

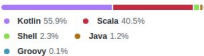
Since Shot 5.0.0 we provide screenshot testing support for **Jetpack Compose**. If you are testing your components using Shot we strongly recommend you to configure your emulator using the gpu mode `swiftshader_indirect`. This will help you to avoid rendering issues when verifying your screenshots from any CI environment.

### Screenshots comparison

Test name	Original screenshot	New screenshot	Diff
Test class: com.karumi.screenshot.MainActivityTest Test name: showNotShowAvengersBadgeAndGetSuperheroNameOfTheAvengersTeam			
Test class: com.karumi.screenshot.MainActivityTest Test name: showAvengersBadgeAndGetSuperheroNameOfTheAvengersTeam			

  
+ 23 contributors

### Languages



Kotlin 55.9%	Scala 40.5%
Shell 2.3%	Java 1.2%
Groovy 0.1%	

# Klisst Arch / Jetpack Compose

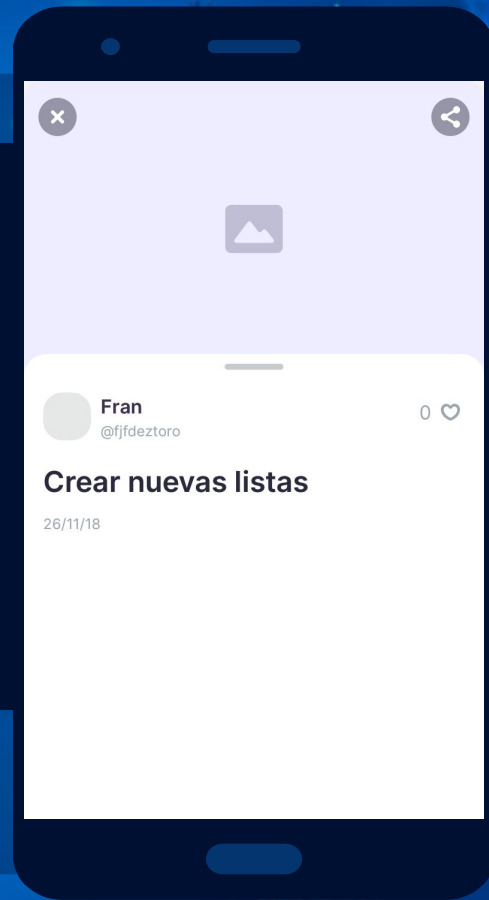
- ▶ Kotlin MVVM + coroutines + Flow + Compose
- ▶ Unidirectional Data Flow
- ▶ Use-Case Driven
- ▶ Designed for Testability (Testable code)
- ▶ 1000+ Tests in total
- ▶ [Karumi/KataSuperHeroesCompose](#)

“

```
@Test
fun rendersAnyItemWithoutDescriptionAndUrl() {
    val item = KlisstItemMother.anyKlisstItemWithoutDescriptionAndUrl
    val klistt = givenAKlisttWithAnUpdatedItem(item)

    renderScreen(klistt, item)

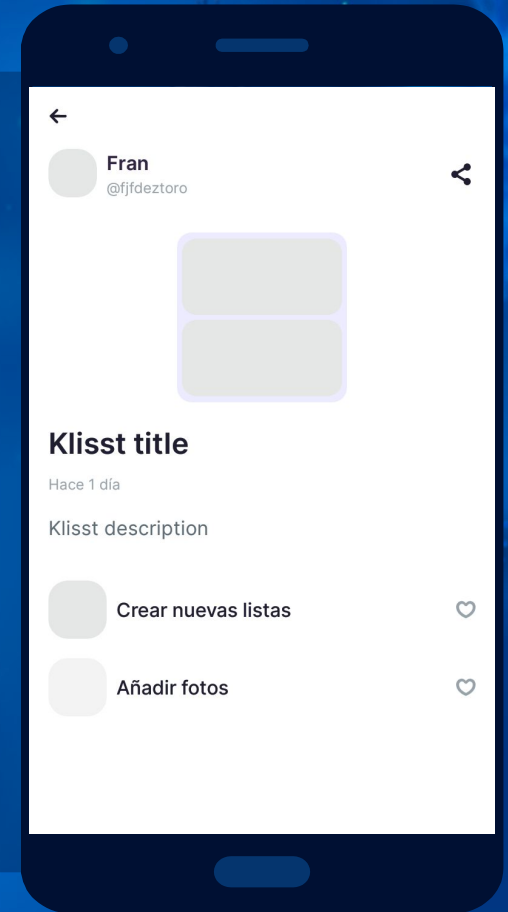
    compareScreenshot(composeRule)
}
```



“

```
@Test
fun rendersKlisstDetailForAnyKlisstWithTwoPhotos() {
    renderScreen(KlisstMother.anyKlisstWithTwoPhotos)

    compareScreenshot(composeRule)
}
```



# Possibilities that Shot offers

- ▶ Recording a single Test
- ▶ Recording an entire Test class
- ▶ Recording all Screenshot Tests for the entire project
- ▶ The same for verifications

```
./gradlew executeScreenshotTests -Precord  
-Pandroid.testInstrumentationRunnerArguments.class=com.kliss.feature.FeatureScreenScreenshotTest
```

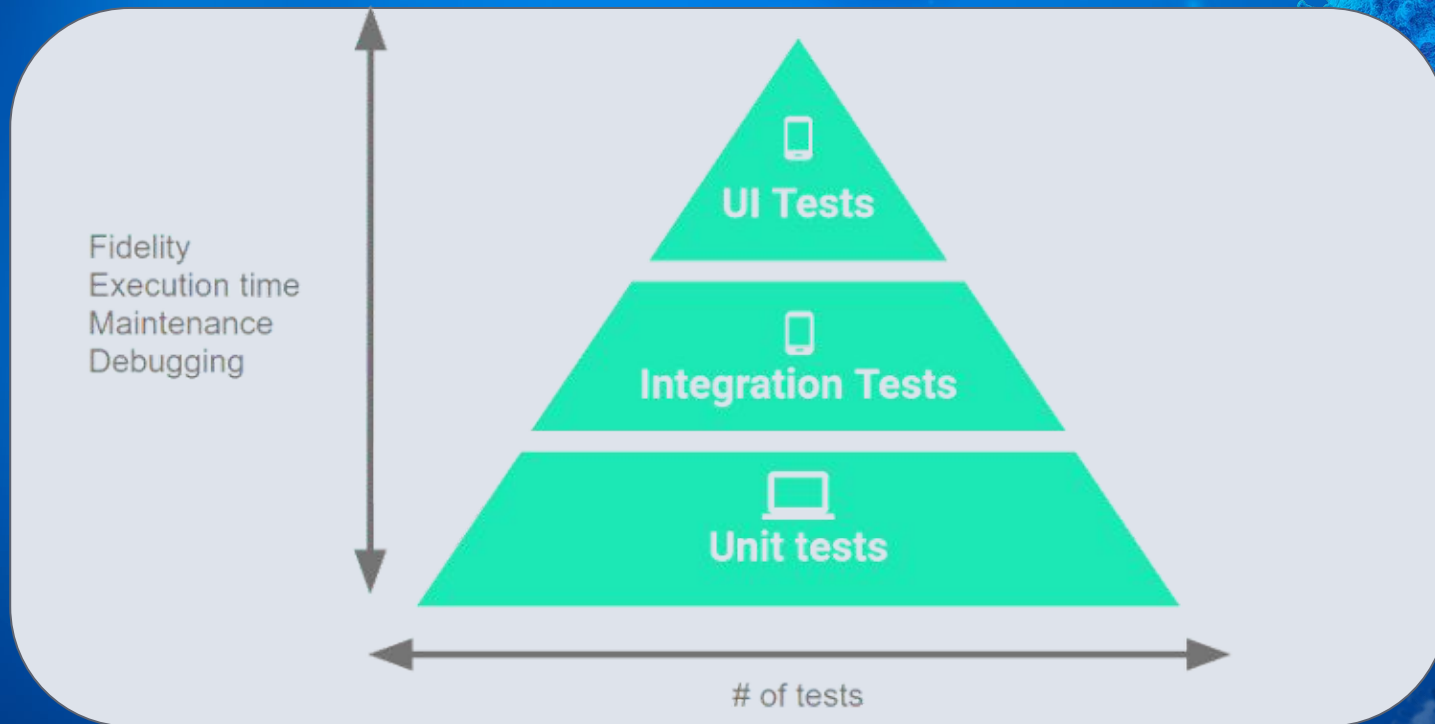
```
./gradlew executeScreenshotTests -Precord  
-Pandroid.testInstrumentationRunnerArguments.class=com.kliss.feature.FeatureScreenScreenshotTest#addsALocationToTheItemWeAreCreating
```

# Unit Tests, UI Tests, Screenshot Tests

- ▶ ViewModel Tests
- ▶ Verify stats tracking, assertions on ViewModel State
- ▶ Quick writing - Maintenance - Flakiness - Bug detection
- ▶ Martin Fowler's "Passive View"
- ▶ Decisions on which is the correct Testing strategy
- ▶ Decisions on which approach to take
- ▶ Idea of Smoke tests - "Confidence testing"



# The Android Testing pyramid



# 3. Challenges

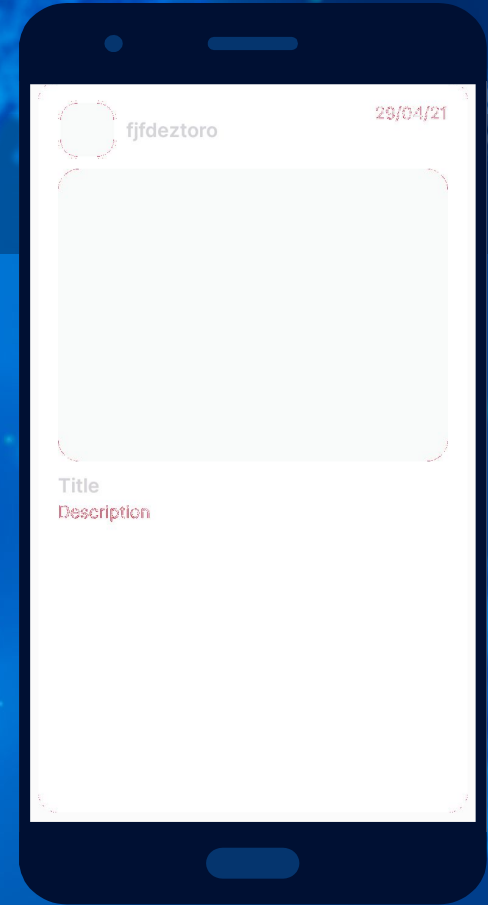
Arise when you work with software quality. Have it in mind

# CI Build time

- ▶ CI Build time > 60m
- ▶ GitHub Actions
- ▶ 1 Dev per Mobile platform
- ▶ Need to optimize / parallelize

# Screenshot tests

- ▶ Mac Intel i7 vs Intel i9 vs Apple M1
- ▶ CI runs in the morning, afternoon
- ▶ Flaky tests
- ▶ Emulators vs Simulators



Typical Diff between Mac and Ubuntu

# Shot test reports

## Screenshots comparison

Test name

Original screenshot

New screenshot

Diff

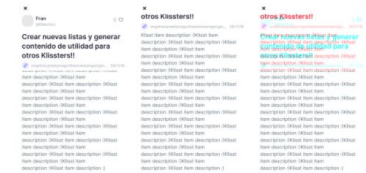
Test class: com.klisst.klisstitem.ui.KlisstItemScreenScreenshotTest

Test name: rendersTheScreenWithAListOfSuggestedKlisstsExpandingTheBottomSheet



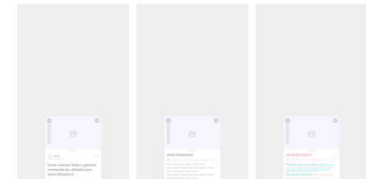
Test class: com.klisst.klisstitem.ui.KlisstItemScreenScreenshotTest

Test name: rendersAnyItemAndExpandsTheBottomSheet



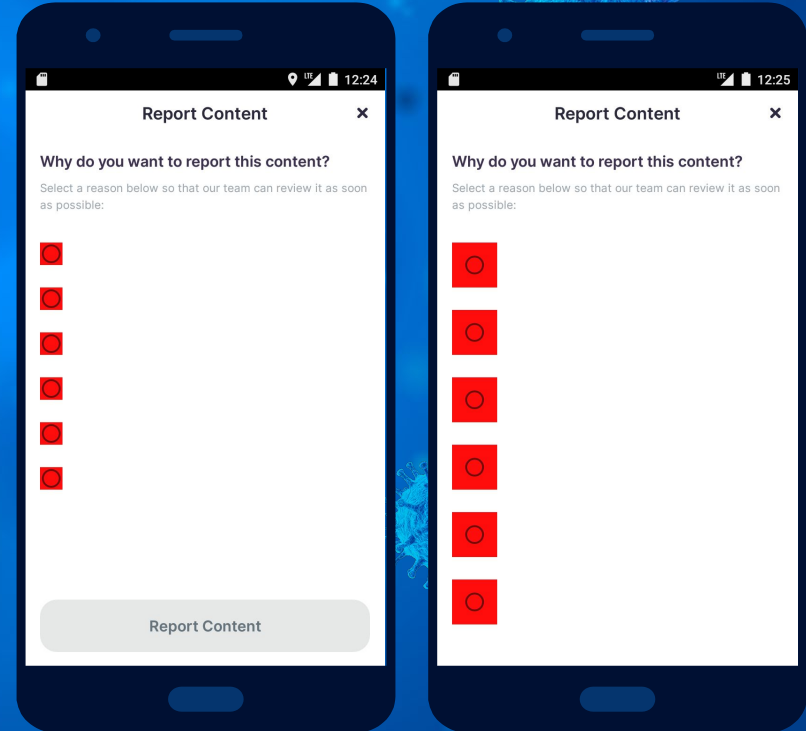
Test class: com.klisst.klisstitem.ui.KlisstItemScreenScreenshotTest

Test name: rendersAnyItemAndExpandsAndLaterCollapsesTheBottomSheet



# Compose APIs and Updates

- ▶ During the beta phase
- ▶ Breaking changes
- ▶ 1.0.5 to 1.1.0 took one week
- ▶ Screenshot tests become essential





“

```
# .github/workflows/build.yml
name: "Build, lint, and test"
on: [pull_request, workflow_dispatch]
env:
  GRADLE_OPTS: "-Dorg.gradle.jvmargs=-Xmx4g . . ."
jobs:
  test:
    name: Build
    runs-on: macos-latest
    steps:
      - uses: actions/checkout@v2
      - name: Set up JDK 1.8
```

# CI running Mac OS X

- ▶ Need to record screenshots in a Mac
- ▶ Physical Mac vs Virtual Mac
- ▶ Possibility of increasing Shot tolerance - Proceed with care
- ▶ Upgrading our CI setup

# 4. Conclusions

From our knowledge

# Conclusions

- ▶ We all want software quality, but software quality is not free
- ▶ We will normally be exposed to fragility
- ▶ Screenshot Testing requires discipline and human effort
  - ▷ When validating the screenshots we record
  - ▷ When reviewing someone else's pull request
  - ▷ When reading CI reports to discover failure reasons
- ▶ Testing components in isolation is slow in Jetpack Compose
- ▶ Crash reports in Jetpack Compose are difficult to understand

# 5. Bonus

Some useful tips and tricks for Android Development



# Some tips and tricks

- ▶ `./gradlew kF detekt == ./gradlew ktlintFormat detekt`
- ▶ `./gradlew eST == ./gradlew executeScreenshotTests`
- ▶ `./gradlew pSB == ./gradlew publishStagingBundle`
- ▶ Android Studio “Double Shift” feature
  - ▷ **CKSST** for **C**reate**K**liss**S**creen**S**creenshot**T**est
  - ▷ **EKVM T** for **E**dit**K**liss**T**View**M**odel**T**est
  - ▷ **KIVM** for **K**liss**I**tem**V**iew**M**odel, etc.



# cashApp/paparazzi

- ▶ An alternative to Shot
- ▶ No need to have a physical device or emulator
- ▶ Also affected by hardware/software rendering differences

# cashApp/paparazzi

github.com/cashapp/paparazzi/issues/554

## Images slightly different from one computer to another #554

Open borsini opened this issue on Aug 26, 2022 · 3 comments

borsini commented on Aug 26, 2022 · edited

**Description**  
Depending on the machine where snapshots are rendered, there can be subtle differences.  
maybe related to #311 ?

**Steps to Reproduce**

- Generate a screenshot on one computer (in the example below, we use a CustomShape) `./gradlew recordPaparazziVARIANT_NAME --rerun-tasks`
- Generate the same screenshot on another computer
- Find out that there are binary differences on the new screenshot




**Expected behavior**  
Images are exactly (pixel/binary) the same

**Additional information:**  
Both machines are using exact same versions 🍌

- Paparazzi Version: 1.0.0
- OS: MacOS 12.5.1
- Compile SDK: 33
- Gradle Version: 7.3.3
- Android Gradle Plugin Version: 7.2.1
- Java version:

```
openjdk 11.0.16.1 2022-08-12
OpenJDK Runtime Environment Homebrew (build 11.0.16.1+0)
OpenJDK 64-Bit Server VM Homebrew (build 11.0.16.1+0, mixed mode)
```

**Screenshots**

Original screenshot	New screenshot	compare result
		

# cashApp/paparazzi

## Lots of different pixels when verifying text shadows on different OSs #311

Open TWISIErRob opened this issue on Nov 8, 2021 · 4 comments

TWISIErRob commented on Nov 8, 2021 · edited Contributor



Notice that background image rendering is clearly the same, it's only the text shadows.

The two runs are from local machine and GitHub actions.

GHA (readme says Zulu)

```
- uses: actions/setup-java@v1
  with:
    java-version: 11
```

Assignees  
No one assigned  
  
Labels

Local

```
openjdk 11.0.2 2019-01-15
OpenJDK Runtime Environment 18.9 (build 11.0.2+9)
OpenJDK 64-Bit Server VM 18.9 (build 11.0.2+9, mixed mode)
```

or

```
openjdk version "11.0.13" 2021-10-19 LTS
OpenJDK Runtime Environment Zulu11.52+13-CA (build 11.0.13+8-LTS)
OpenJDK 64-Bit Server VM Zulu11.52+13-CA (build 11.0.13+8-LTS, mixed mode)
```

I'm running on Windows, while GHA is Ubuntu.

Feel free to close this, if you think it's not fixable, just wanted to let you know.  
We have a multiplatform team at work, so I guess it might come up for others too.

borsini mentioned this issue on Aug 26, 2022

Images slightly different from one computer to another #554

Open



yschimke commented on Oct 9, 2022 Contributor

Got hit by this as well. Is there any advice on the best combinations across OSX for local development and Linux for CI?  
  
Are newer JDKs better or worse for consistency?



borsini commented on Oct 9, 2022

@yschimke as a temporary solution we ended up generating the screenshots only on the CI, then pushing them in a new commit on our PRs. This ensures that screenshots are always the same.

3

# Credits

Links used for this talk. Special thanks to these free resources:

- ▶ [Simulators vs Emulators](#), [Real device vs simulator](#)
- ▶ [ObjectMother pattern](#)
- ▶ [Parallelization - Mercadona Tech](#)
- ▶ [Martin Fowler's Passive View](#)
- ▶ [cashApp/paparazzi](#)
- ▶ Presentation template was provided by [SlidesCarnival](#)
- ▶ Photographs by [Unsplash](#)



# Thanks for coming!

Questions & Answers

You can find me at  
@olmoDev · olmo@klisst.com





**Diego Freniche**  
**@dfreniche**  
**freniche.com**

# Your mobile app with MongoDB/Realm

with live coding and sample code! 🤖

**Thu, 23rd February 2023**



**Innovation Hub**



MalagaMobile



# Flutter Forward Extended

Flutter Málaga, 2 de Febrero

