

Внедрение Screenshot- тестирования дизайн-системы

Максим Теймуров



Обо мне

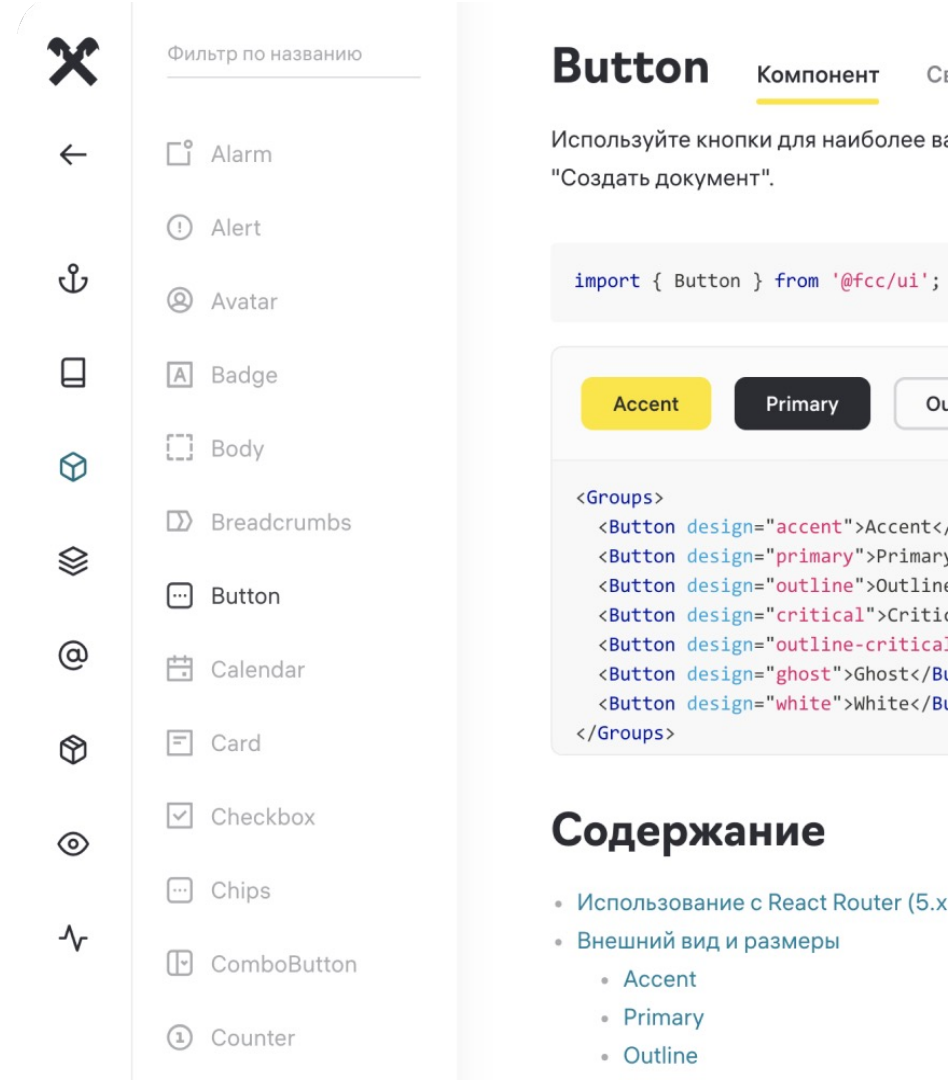
- 7+ лет занимаюсь Android-разработкой
- 3 года работаю с дизайн-системами
- Community Lead в Raiffeisen Bank
- С прошлого года занимаюсь развитием дизайн-системы в Raiffeisen Bank

О чем расскажу

- Что такое скриншот тесты
- Какие Android-библиотеки есть на рынке
- Почему мы остановились на Shot
- Дам советы по написанию тестов для дизайн-системы
 - Figma API
 - Матрица трассировки
- Расскажу как настроить CI

Что такое дизайн-система(ДС)?

Дизайн система - набор компонентов, правил их использования и инструментов разработки для создания продукта с единым визуальным стилем.



Raiffeisen Bank



40+ Android
разработчиков



6+ приложений

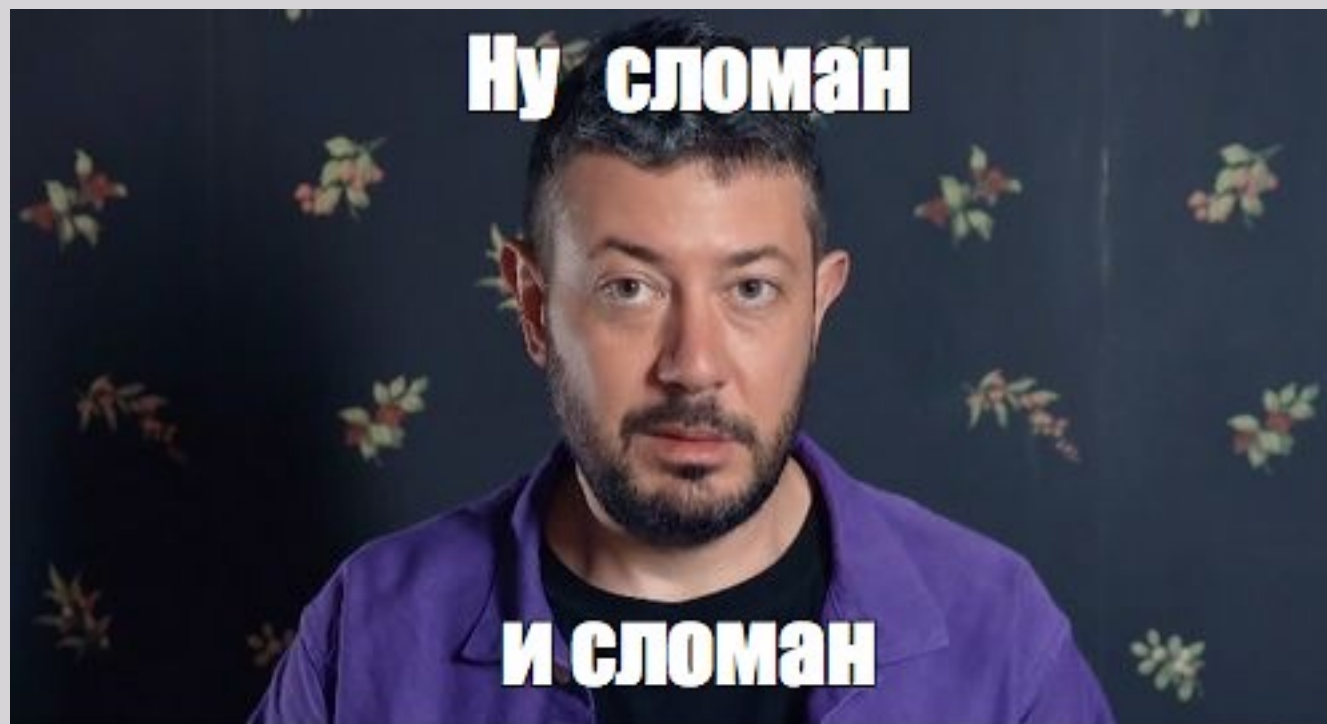


50+ компонентов
на View и Compose

Много разработчиков
работает над библиотекой,
которую используют разные
продукты...

Что может пойти не так?

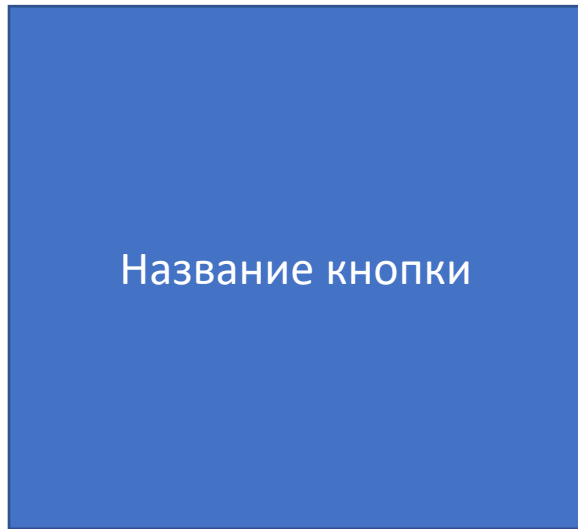
UI все время пытается сломаться



Что делать?

- ручное тестирование
- unit-тесты
- ui-тесты
- скриншот-тесты

Скриншот-тесты



Эталонный
СНИМОК



Разница



Свежая сборка

Какие требования к библиотеке?

- Делать скриншоты
- Сравнение скриншотов с эталонными и создание отчета об этом
- Работа с View и Compose без использования AndroidView и ComposeView
- Возможность отключить/пропустить анимацию без доработки компонентов

Обзор библиотек

UI Automator

✓ Умеет делать скриншоты

Как сделать скриншот:

```
val screenshotFile = File(  
    InstrumentationRegistry.getTargetContext().filesDir,  
    "filename"  
)  
val device = UiDevice.getInstance(  
    InstrumentationRegistry.getInstrumentation()  
)  
device.takeScreenshot(screenshotFile)
```

Kaspresso

- ✓ Делает screenshot из коробки
- ✓ Работа с View и Compose без использования AndroidView и ComposeView (experimental)

Как сделать скриншот:

```
captureScreenshot("ScreenshotName")
```

Espresso

- ✓ Умеет делать скриншоты
- ✓ Работа с View и Compose без использования AndroidView и ComposeView
- ✓ **Возможность отключить/пропустить анимацию без доработки компонентов**

Как сделать скриншот:

```
val view = activity.getWindow().getDecorView().getRootView()  
val bitmap = Bitmap.createBitmap(view.getDrawingCache())  
bitmap.compress(Bitmap.CompressFormat.PNG, 90, out)
```

Screenshot Tests for Android (Facebook)

- ✓ Делает screenshot из коробки
- ✓ Сравнение скриншотов с эталонными и создание отчета об этом
- ~~☐ Работа с View и Compose без использования AndroidView и ComposeView~~

Как сделать скриншот:

```
val view = View(InstrumentationRegistry.getInstrumentation().context)
Screenshot.snap(view).setName("ScreenshotName").record()
```

Paparazzi

- ✓ Делает screenshot из коробки
- ✓ Сравнение скриншотов с эталонными и создание отчета об этом
- ✓ Работа с View и Compose без использования AndroidView и ComposeView
- Не нужно настраивать и запускать эмуляторы
- ~~☐ Возможность отключить/пропустить анимацию без доработки компонентов (ComposeTestRule – не поддерживается)~~

Как сделать скриншот:

```
val paparazzi = Paparazzi(deviceConfig = PIXEL_5, theme = ...)
paparazzi.snapshot(view)
paparazzi.snapshot { ComposeFunction() }
```


Shot

- ✓ Делает screenshot из коробки
- ✓ Сравнение скриншотов с эталонными и создание отчета об этом
- ✓ Работа с View и Compose без использования AndroidView и ComposeView
- ✓ Возможность отключить/пропустить анимацию без доработки КОМПОНЕНТОВ

Как сделать скриншот:

@get:Rule

```
val rule = createAndroidComposeRule<TestActivity>()  
rule.setContent { ComposeFunction() }  
compareScreenshot(rule)
```

Dropshots – Герой не моего романа

- ✓ Делает screenshot из коробки
- ✓ Сравнение screenshot`ов с эталонными и создание отчета об этом
- ✓ Работа с View и Compose без использования AndroidView и ComposeView
- ✓ Возможность отключить/пропустить анимацию без доработки КОМПОНЕНТОВ

Как сделать скриншот:

```
val activityScenarioRule = ActivityScenarioRule(TestActivity::class.java)
activityScenarioRule.scenario.onActivity {
    Dropshots().assertSnapshot(it, "ScreenshotName")
}
```

Сравнение

Библиотека	Скриншоты	Сравнение/Отчет	View/Compose	Animation off
UI Automator	Да	Нет	-	-
Kaspresso	Да	Нет	Experimental	-
Espresso	Да	Нет	Да	Да
Facebook	Да	Да	Нет	Да
Paparazzi	Да	Да	Да	Нет
Dropshots	Да	Да	Да	Да
Shot	Да	Да	Да	Да

Настройка,
написание тестов
и запуск на Shot

Как настроить проект для запуска?

build.gradle

- buildscript { dependencies { classpath 'com.karumi:shot:<LATEST_RELEASE>' } }
- android { defaultConfig { testInstrumentationRunner "com.karumi.shot.ShotTestRunner" } }

app/build.gradle

- apply plugin: 'shot'

androidTest/AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package=" your.app.id.test"  
    android:sharedUserId="your.app.id.uid">/>
```

Как настроить проект для запуска?

Если вы используете Shot для тестирования своей android-библиотеки:

- `android { defaultConfig { testApplicationId "<MY_TEST_APPLICATION_ID>" }`

Если вы используете AGP 7. X

- `shot { applicationId = "com.myapp" }`

Для Android 9 (уровень API 28)

- `adb shell settings put global hidden_api_policy_pre_p_apps 1`
- `adb shell settings put global hidden_api_policy_p_apps 1`

Для Android 10 (уровень API 29) или выше

- `adb shell settings put global hidden_api_policy 1`

Как написать первый тест? View

```
class ViewComponentTest : ScreenshotTest {  
  
    @get:Rule  
    var activityScenarioRule = activityScenarioRule<TestViewActivity>()  
  
    private lateinit var text: TextView  
  
    @Before  
    fun setup() {  
        val activity = activityScenarioRule.scenario.waitForActivity()  
        runOnUiThread {  
            val text = TextView(activity).apply {  
                setText("Test string")  
            }  
            activity.container.addView(text)  
        }  
    }  
}
```

Как написать первый тест? View

```
class ComponentViewTest : ScreenshotTest {  
  
    private lateinit var text: TextView  
  
    ...  
  
    @Test  
    fun viewTest() {  
        compareScreenshot(  
            view = text,  
            name = "ScreenshotName",  
            widthInPx = 200.dp,  
            heightInPx = 200.dp  
        )  
    }  
}
```


Как написать первый тест? Compose

```
@RunWith(AndroidJUnit4::class)
class ComposeComponentTest : ScreenshotTest {

    @get:Rule
    val rule = createAndroidComposeRule<TestComposeActivity>()

    @Test
    fun composeTest() {
        rule.setContent { ComposeFunction() }
        compareScreenshot(rule)
    }
}
```

Запуск

Как сделать эталонные снимки:









- `./gradlew <Flavor><BuildType>ExecuteScreenshotTests –PreCORD`
- `./gradlew executeScreenshotTests –PreCORD`

Верификация новых скриншотов:

- `./gradlew <Flavor><BuildType>ExecuteScreenshotTests`
- `./gradlew executeScreenshotTests`

Отчет

Screenshots comparison

Test name	Original screenshot	New screenshot	Diff
Test class: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest Test name: style Screenshot name: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest_Ghost_s_loading_enabled			
Test class: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest Test name: style Screenshot name: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest_Ghost_s_loading			
Test class: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest Test name: style Screenshot name: ru.raiffeisen.viennakit.test.view.ButtonScreenshotTest_Ghost_s_enabled	Съешь еще		

Как сделать тестирование
еще проще?

Отключить анимацию для View и Compose

View

```
testOptions {  
    animationsDisabled = true  
}
```

Compose(androidx.test.espresso:espresso-core)

`@RequiresApi`(Build.VERSION_CODES.O) // MinApi 26

```
fun compareScreenshot(rule: ComposeTestRule, name: String? = null) {  
    rule.waitForIdle()  
    compareScreenshot(rule.onRoot(), name)  
}
```

Понижить чувствительность тестов

build.gradle

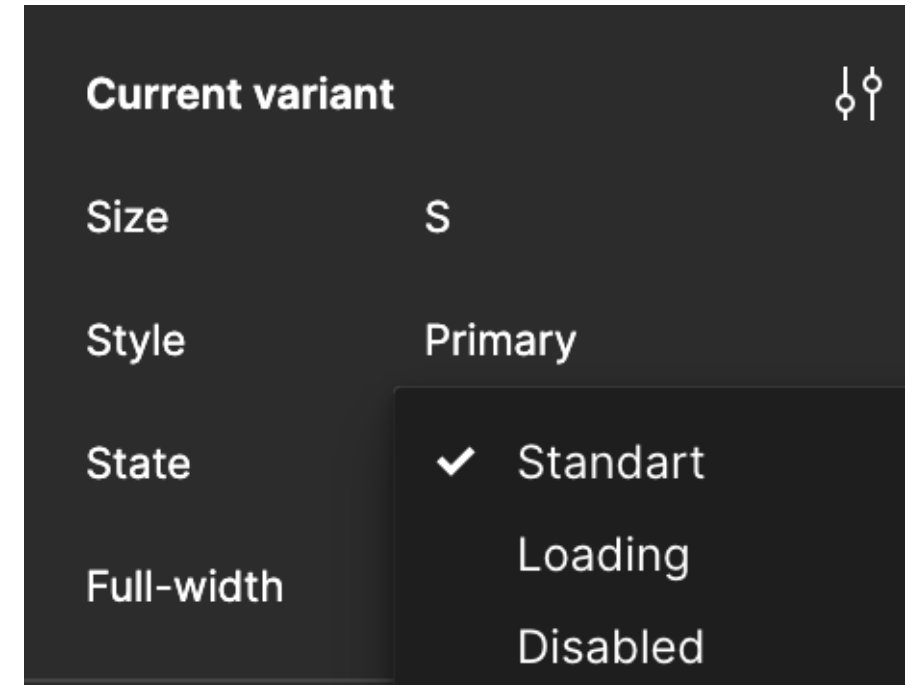
```
shot {  
    applicationId = "your.app.test"  
    showOnlyFailingTestsInReports = true  
    tolerance = 0.5 // 0.5%  
}
```

Figma Api

- Позволяет скачивать в виде json описание токенов и компонентов
- На их основе можно генерировать цвета, размеры, иконки, типографию, стили к компонентам
- Таким образом при изменениях дизайн-системы не затрагивающих структуру компонентов нужно только запустить регенерацию стилей и прогнать тесты

Variants в Figma

- Описывают различные состояния для одного компонента
- Полный набор значений вариантов описывает все возможные состояния компонента
- Чтобы протестировать компонент полностью, нужно протестировать как минимум по разу каждое значение варианта



Пример

Кнопка может быть:

- Разной по форме: круглый, квадратный
- Разной по размеру: маленький, средний, большой



Матрица трассировки (traceability matrix)

Двумерная таблица, содержащая соответствие функциональных требований продукта (functional requirements) и подготовленных тестовых сценариев (test cases)

Построим матрицу для нашего примера

	Круглый	Квадратный	Маленький	Средний	Большой
Тест 1	Да		Да		
Тест 2		Да		Да	
Тест 3		Да			Да

Вышло 3 теста и все значения вариантов протестированы

Мир не идеален – выбери два из трех



Что еще можно протестировать?

- Темная / светлая тема
- Версия Android на эмуляторе
- Размеры экрана и виды устройств (н-р, если ваши компоненты выглядят по-разному для смартфонов, планшетов, телевизоров)

Настройка СИ

«Идеальный» процесс merge-request

- Разработчик обновляет эталонные тесты и отправляет на review
- На CI запускаются скриншот-тесты и отчет прикрепляется к MR
- Дизайнеры или другие разработчики могут оценить результат по получившимся скриншотам
- Review получает одобрение после успешного прохождения всех тестов
- Код вливается в основную ветку, а эталонные скриншоты для этой ветки становятся эталонными для всего проекта

«Идеальный» процесс merge-request

- Разработчик обновляет эталонные тесты и отправляет на review
- На CI запускаются скриншот-тесты и отчет прикрепляется к MR
- **Дизайнеры или другие разработчики могут оценить результат по получившимся скриншотам**
- Review получает одобрение после успешного прохождения всех тестов
- Код вливается в основную ветку, а эталонные скриншоты для этой ветки становятся эталонными для всего проекта

Где хранить скриншоты и отчеты?

- **Начальный уровень** - отчеты прикрепляются к `merge-request`у`.
- **Средний уровень** – файловое хранилище / `artifactory`.
- **Продвинутый уровень** – **Git LFS**. Подключается к текущему Git-проекту, но хранит файлы в отдельном хранилище.

Плюсы:

- Не растет размер основного репозитория
- Скриншоты отображаются прямо в MR GitLab
- Не нужно хранить эталонные скриншоты для каждой ветки где-то снаружи

Минусы:

- Иногда ломается `git changelist` в Android Studio, приходится вызывать `git lfs uninstall && git reset --hard HEAD`

А что в итоге?

- Разобрались как выбирать библиотеку для скриншот-тестирование под свой проект
- Интегрировали библиотеку Shot и настроили прогон тестов на CI
- Экономим время, не тестируя руками UI

Спасибо за
внимание.
Вопросы?

Максим Теймуров

✈ MTeymurov

