

Android App Development: Survival Guide

MobileTech Conference and Summit 2019

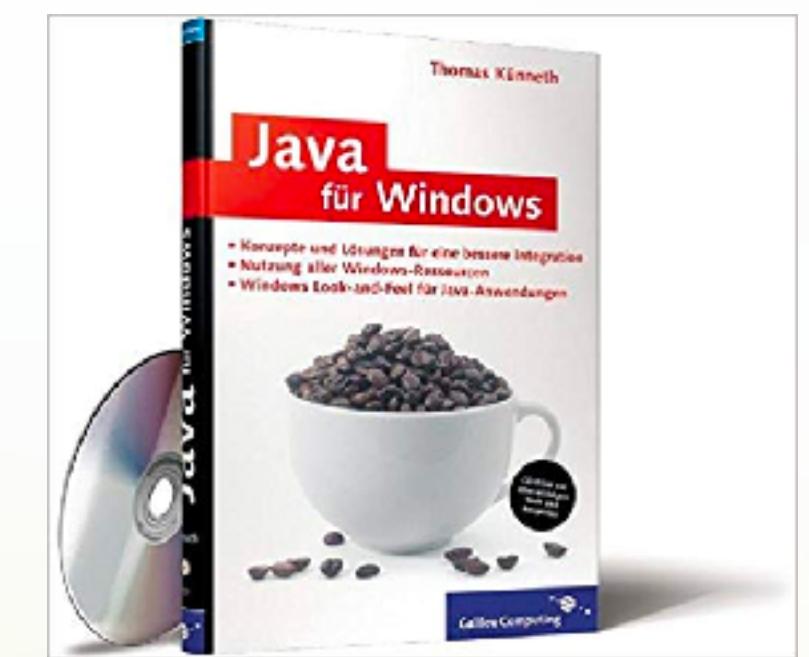
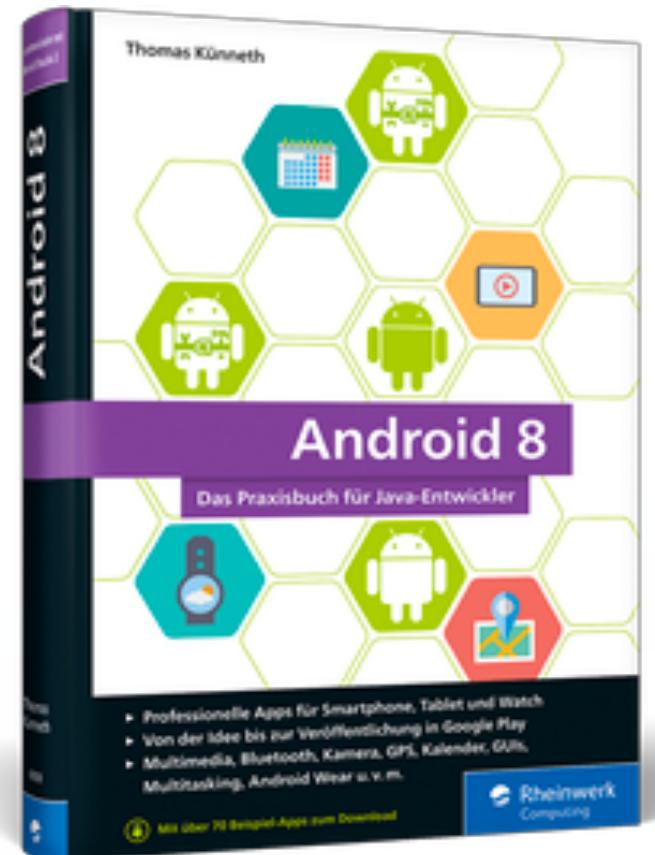
Thomas Künneth
MATHEMA Software GmbH

Die App ist fertig

- Release-Party war ein voller Erfolg
- App ist im Store
- Werbung ist geschaltet
- Kolleginnen und Kollegen haben schon das nächste Projekt begonnen

Die App-Erlöse können kommen

1. Übergreifende Themen
2. Effizient, effektiv und automatisiert testen
3. Debuggen, profilieren und Abstürze analysieren
4. Zielgerade

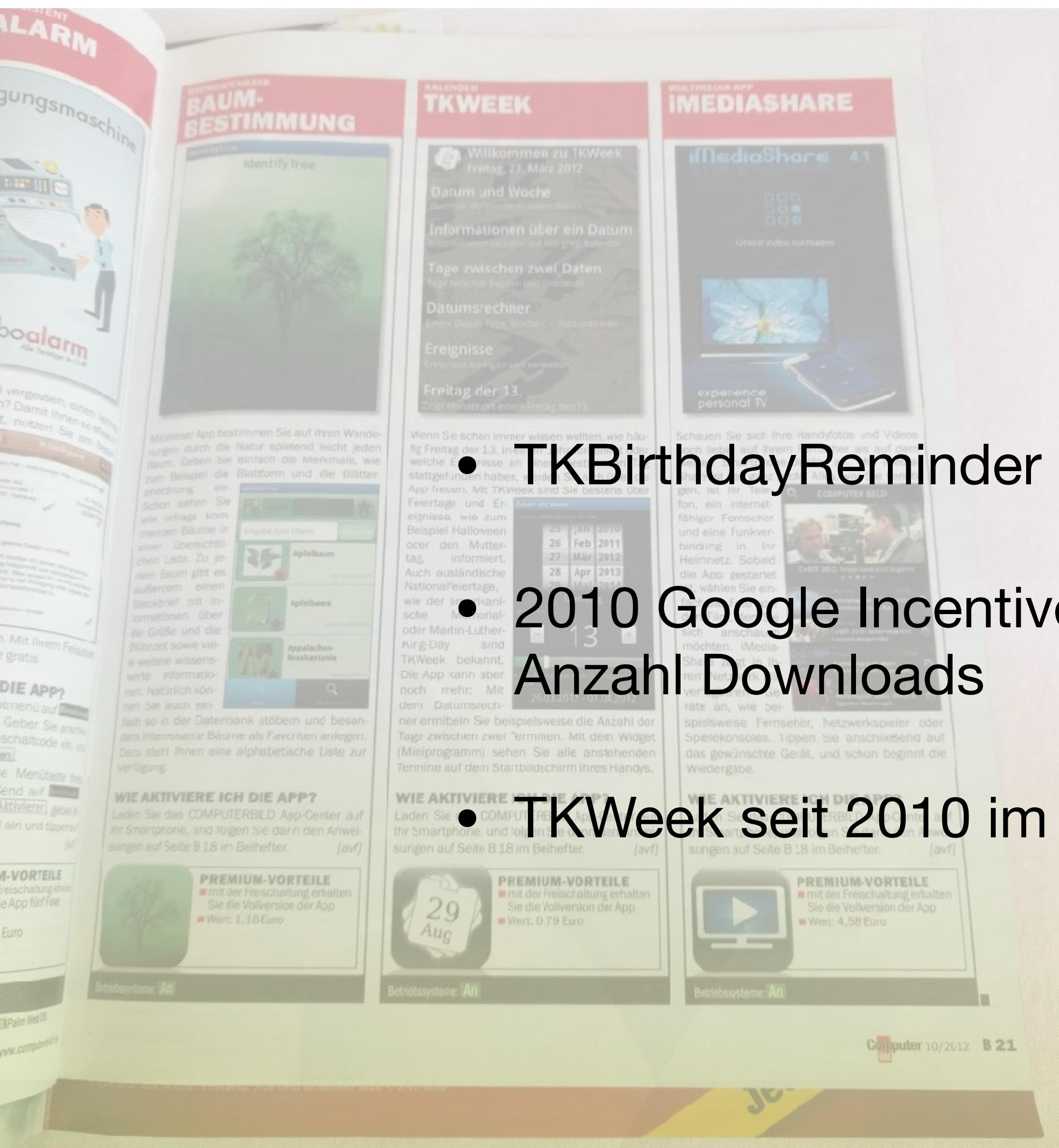


www.mathema.de

thomas.kuenneth@mathema.de

 @tkuenneth

github.com/tkuenneth/android_app_survival_guide



- TKBirthdayReminder seit 2009 im Store
- 2010 Google Incentive für positive Bewertungen und Anzahl Downloads
- TKWeek seit 2010 im Store

Amazon Appstore for x www.amazon.com/gp/feature.html?docId=1000620171&ref=sv_mas_1

Just in time for Mother's Day
Free 1-Day Shipping on Select Gifts

amazon Prime

Shop by Department

Search Appstore for Android

Appstore for Android | Best Seller | Deals | New Releases | Test Drive Apps | Shop Andriod Phones | Your Apps & Devices | Get Started | Help

Buy Select Paid Apps Get \$1 for MP3s [Learn more](#)

Amazon Appstore Deals Enjoy great savings and special deals

Today's free app of the day
TKWeek \$0.99 FRFF 12:52:51 left [Get app](#)

Schauen Sie sich Ihre Handysotos und Videos noch lieber auf Ihrem Computer an, auf dem Sie stattgefunden haben. Wenn Sie möchten, ist Ihr Telefon, ein internet-fähiger Fernseher und eine Funkverbindung in Ihr Heimnetz. Sobald die App gestartet ist, wählen Sie einfach aus, welche Geräte Sie sich anschließen möchten. iMediaShare verzerrt Ihre Fotos und Videos nicht, sondern zeigt sie Ihnen in ihrer natürlichen Größe. Durch das Anpassen der Farben und des Kontrastes können Sie Ihre Bilder so anpassen, wie Sie möchten.

TKWeek
by Thomas Kuennen
★★★★★ (11)

Get your all-in-one, planner extravaganza. Featuring an impressive array of functional capabilities, TKWeek is an in-depth calendar, date book, and week calculator. Find the first and last date of a week, provide detailed date information, maintain important dates, and calculate time between dates.

I tell your friends: [Facebook](#) [Twitter](#)

50% or More Off Editor's Picks

Through May 13, select apps and games are 50% off or more. Save on:

- Biscuit's Birthday: \$0.99
- The Lost City: \$0.99
- iCookbook: The Complete Cooking Solution: \$4.99
- FoodTracker Pro: \$1.99
- The Mystery of the Crimson Manor: \$0.49

[See more](#)

iCookbook: The Complete Cooking Solution Save on iCookbook: The Complete Cooking Solution, which includes more than 2,000 hand-selected recipes. \$4.99 (list price \$9.99) [Learn more](#)

- 1. Übergreifende Themen**
- 2. Effizient, effektiv und automatisiert testen**
- 3. Debuggen, profilieren und Abstürze analysieren**
- 4. Zielgerade**

**Was bedeutet
„Die App“?**

- One Man-Show oder Teamarbeit?
- Art der App (Spiel, Hilfsmittel, Unternehmens-App, ...)
- Das eigentliche Produkt oder „nur“ Mittel zum Zweck?
- Neu oder schon (länger) am Markt?
- Zielgruppe

- Läuft die App ausschließlich auf dem Gerät oder wird Verbindung zu Backend-Services aufgenommen?
- Falls Backend...
 - Nur eigene Services, oder auch von Drittanbietern?
 - Wer betreibt das Backend?

Wozu diese Fragen?

- Das Backend
- Begleitende Webseite
- Kundensupport

Das Backend

- Entwicklung und Test u. U. aufwendiger als die Programmierung der App
- Betrieb beinhaltet Herausforderungen
 - Monitoring
 - Deployments (z. B. von Functions)
 - Skalierung

Begleitende Website

- Impressum
- Kontaktmöglichkeit (E-Mail, Telefon, Formular)
- Auf DSGVO-Konformität achten
 - Datenschutzerklärung
 - Genutzte Drittanbieter-Services

Kundensupport

- Gibt es eine Kontakt-E-Mail-Adresse, Telefonnummer oder ein Kontaktformular?
- FAQs, Dokumentation oder How Tos (z. B. Videos)?
- Präsenz in Sozialen Medien?

Gewichtung dieser Aspekte
abhängig von App und Team-
Größe

Aber darüber nachdenken
müssen Sie!

Wie wird die App von den Nutzern angenommen?

- Download-Zahlen können in der Play Console eingesehen werden
- Einnahmen durch Käufe und ggf. In App-Werbung ebenfalls via Play Console
- Rezensionen und Testberichte z. B. in Zeitschriften sind schwerer zu beobachten

Feedback im Play Store

- Bewertungen kontrollieren ist wichtig!!!
- Indikator für Akzeptanz
- Bei Apps mit vielen Nutzern gutes Frühwarnsystem

- Auch 4 und 5 Sterne-Bewertungen lesen
- Werden Fragen gestellt oder Anregungen gegeben?
- Auch gute Bewertungen können auf Fehler hinweisen

- Zeitnah antworten
- Signalisieren, dass Kontaktaufnahme verstanden wurde
- Demonstrieren Sie Wertschätzung

sehen, ob jmd etwas an seine Pinnwand gepostet hat?! Stellt das bitte wieder rein!

Außerdem bekomme ich ständig Benachrichtigungen über Me...

[Vollständige Rezension](#)



DeliKa

★ ★ ★ ★ ★ 2. März 2019



:

166

Gruppen zu besuchen, macht keinen Spaß mehr, da hier jetzt scheinbar standardmäßig die beliebtesten Beiträge oben sind. Wer das Neueste sehen will, muss jedes Mal (!) umstellen. Sonst bleiben uralte Beiträge, die zufällig nur viele Gefällt mir Klicks haben permanent oben, neue sieht man erst später....

[Vollständige Rezension](#)



Mirco P

★ ★ ★ ★ ★ 28. Februar 2019



:

9

Ganz ehrlich ihr wollt mich doch verarschen! Ich verwende Facebook nur als Mittel um Spiel erfolge zu speichern, so das ich das dann sowohl auf Android als auch ios speichern kann. Jetzt kommt die wunderbare Authentifizierung in der ein code zur angegebenen nummer geschickt werden soll. Diese Nummer...

[Vollständige Rezension](#)



Alexander Baurfelder

★ ★ ★ ★ ★ 28. Februar 2019



:

32

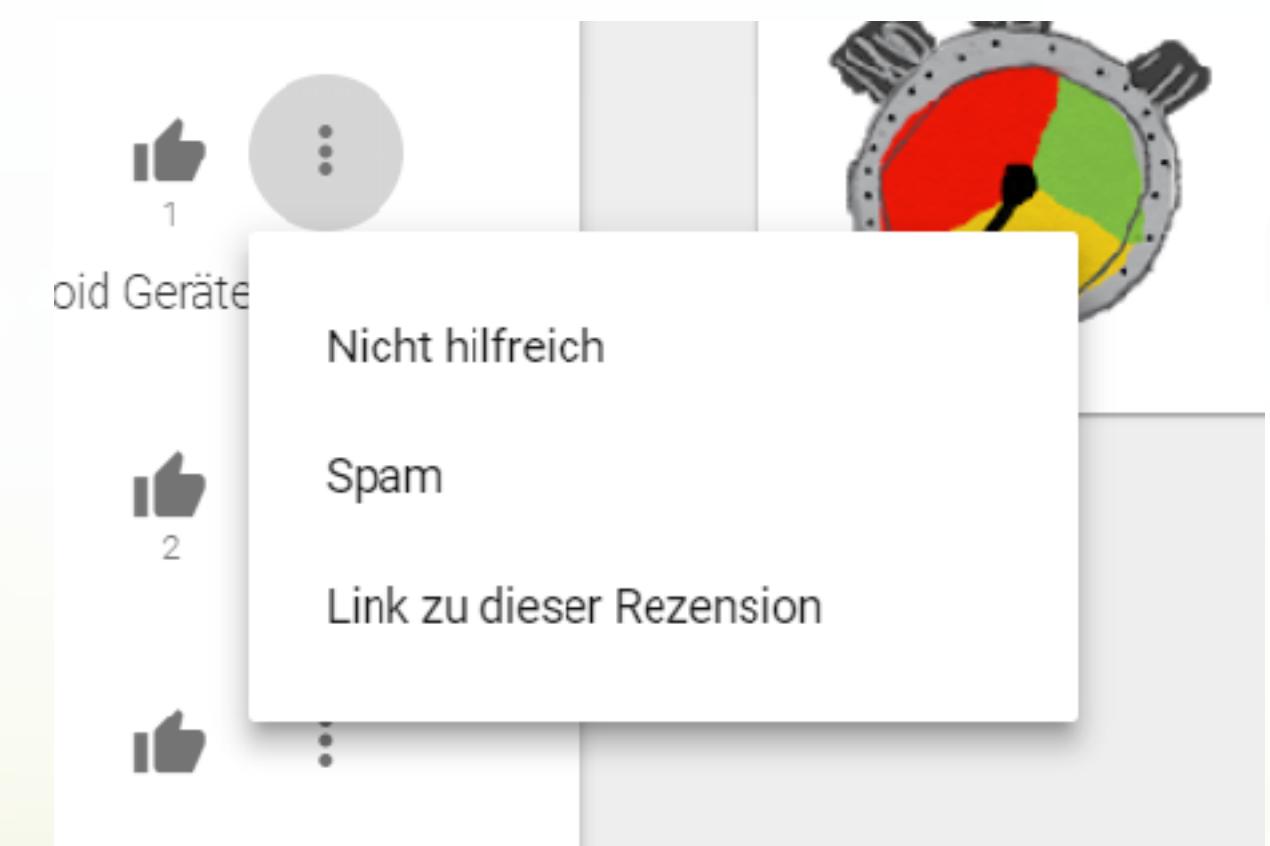
Egal ob ich nun auf Neuigkeiten gehe, oder die Hauptmeldungen anklicke.. Sind identisch. Ich erhalte Benachrichtigungen von Dingen, die teilweise schon 1 Std vorher gepostet werden. Ganz toll, wenn man eine Gruppe hat, wo verkauft wir und ich die Beiträge erst freischalten muss. Die Krönung war, ein...

[Vollständige Rezension](#)

Don't Panic

- Bewertung auf verwertbare, konstruktive Hinweise prüfen
- Ggf. auf Workarounds hinweisen
- Ggf. neue Version in Aussicht stellen
- In jedem Fall Unterstützung anbieten (z. B. über Support-Kanäle)

- Bei provozierenden, sich im Ton vergreifenden Bewertungen sachlich antworten
- Bei Google melden



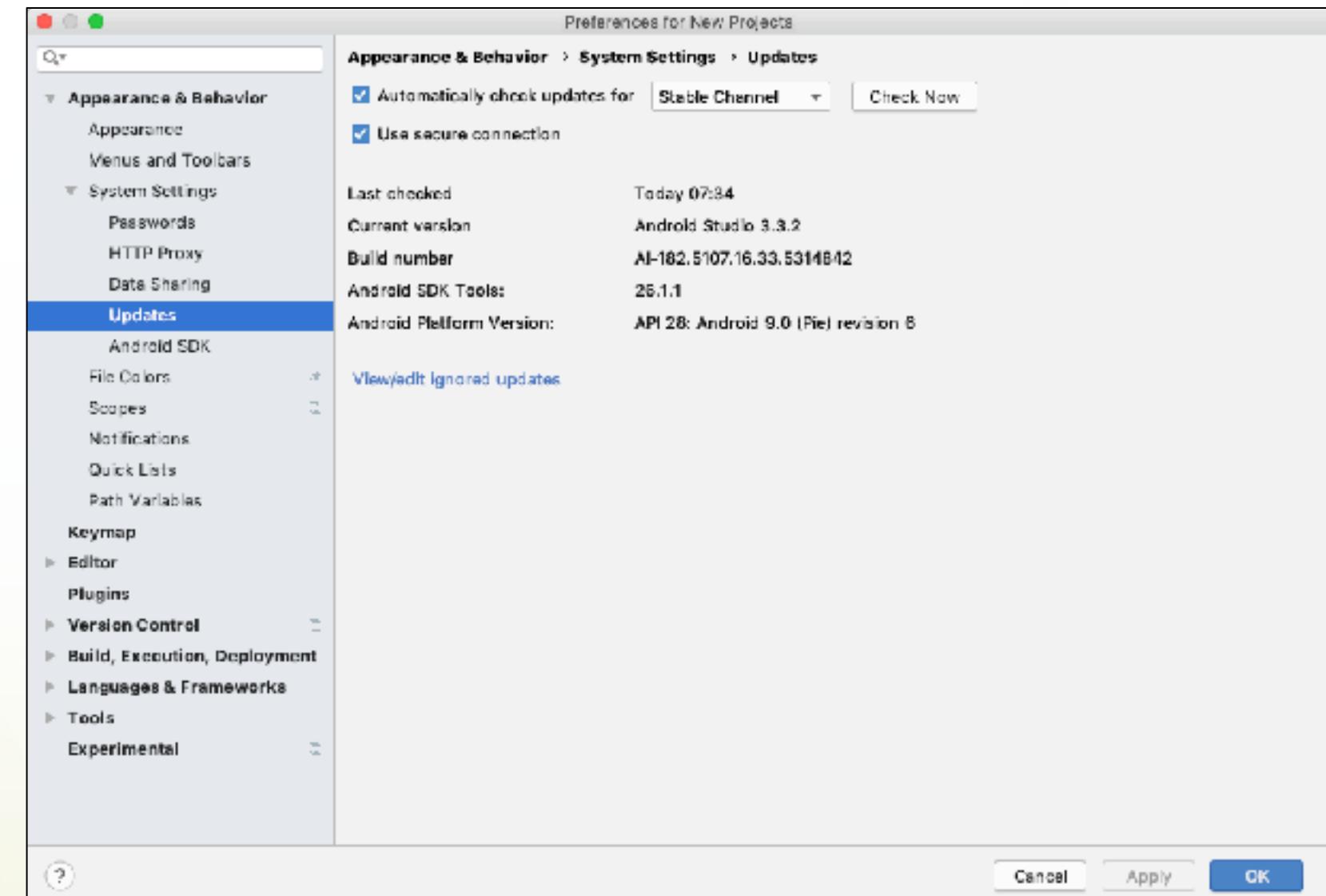
Schnell reagieren

- Nutzer erwarten schnelle Reaktionen
- Praxis zeigt: niedrige Frustschwelle
 - Apps werden schnell deinstalliert
 - Kaum zeitnäher erneuter Download
 - Ziel: Um jeden Preis Deinstallationen vermeiden

Wie kann ich
schnell
reagieren?

- Professionell entwickeln
- Professionell entwickeln
- Professionell entwickeln

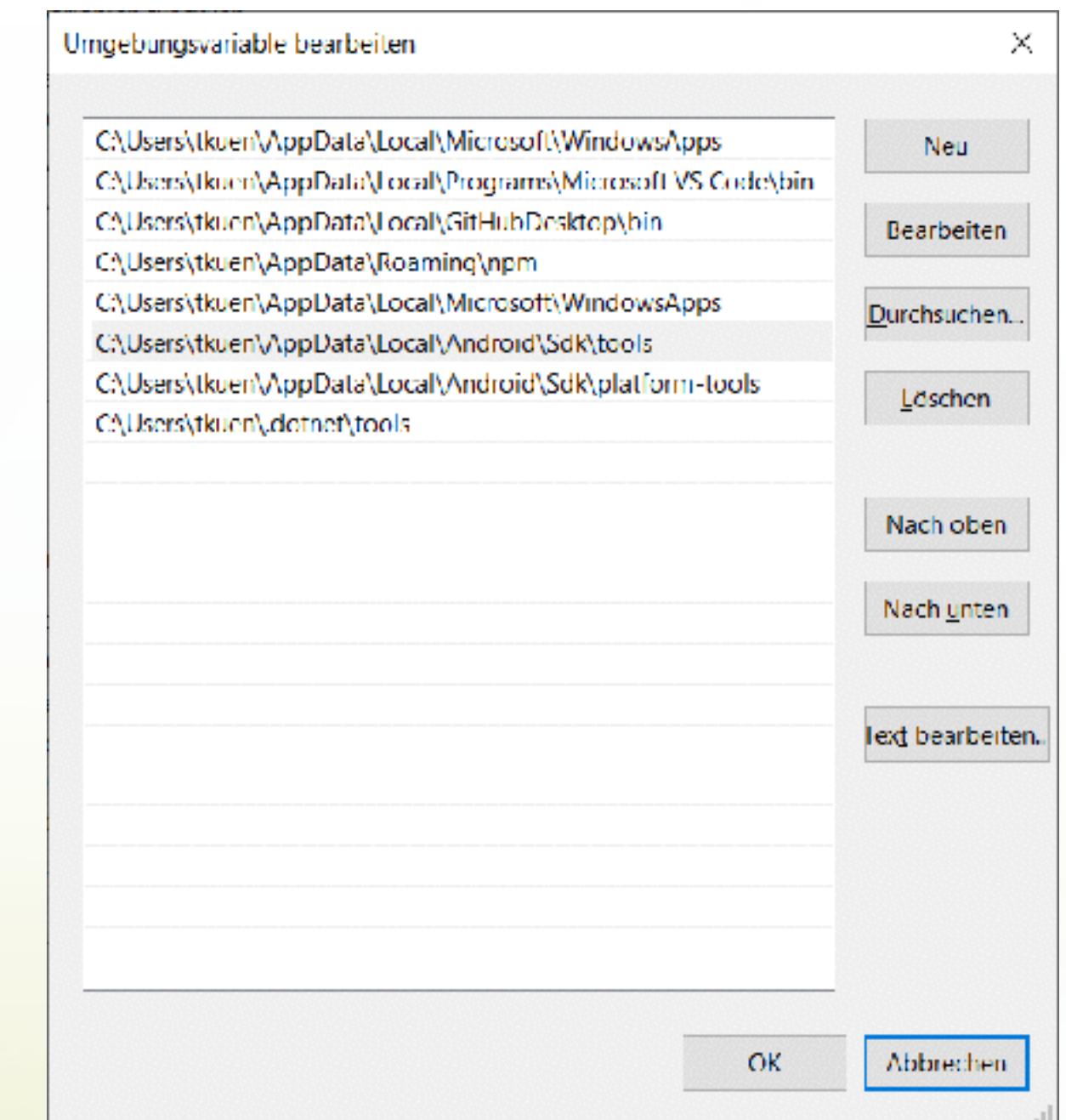
- Entwicklungsumgebung richtig aufsetzen und aktuell halten
- Programmiersprache, Plattform und Werkzeuge kennen und konsequent nutzen
- Sich selbst auf dem Laufenden halten



- Android Developers Blog
<https://android-developers.googleblog.com/>
- Android in sozialen Medien folgen z. B. @AndroidDev
(Twitter)
- Android Developers auf YouTube
<https://www.youtube.com/user/androiddevelopers/>

- Kostet viel Zeit
- Aber: Im Android Ökosystem gibt es oft Neues
- Je früher Sie davon hören, desto einfacher wird es für Sie

- Android Studio und das Android SDK sind schnell eingerichtet
- Kommandozeilentools `emulator` und `adb` auch wichtig
- SDK-Verzeichnisse `tools` und `platform-tools` dem Systempfad hinzufügen
- Auch klassische Java-Tools (`jarsigner`, `keytool`, ...) hinzufügen



- Wichtige Tools Kommandozeilenparameter einprägen
- Griff zur Shell schneller als Klicken in der IDE

- Alle Artefakte sauber versionieren
- Regelmäßig Backups anfertigen, um z. B. nach Hardwareausfällen schnell wieder loslegen zu können

- Prüfen, ob eigene Build-Infrastruktur nötig
- Jenkins kann nicht nur App bauen/testen/prüfen, sondern auch automatisiert in den Play Store hochladen
- Einarbeitungs- und Admin-Aufwand nicht unterschätzen (kein Oh Klasse, wollte ich schon immer mal probieren)

- Nutzer sind vom Hotfix des Hotfixes generiert
- Keine Schnellschuss-Versionen
- Keine Release-Salven
- Jedes Release intensiv testen
- Tracks in der Play Console nutzen

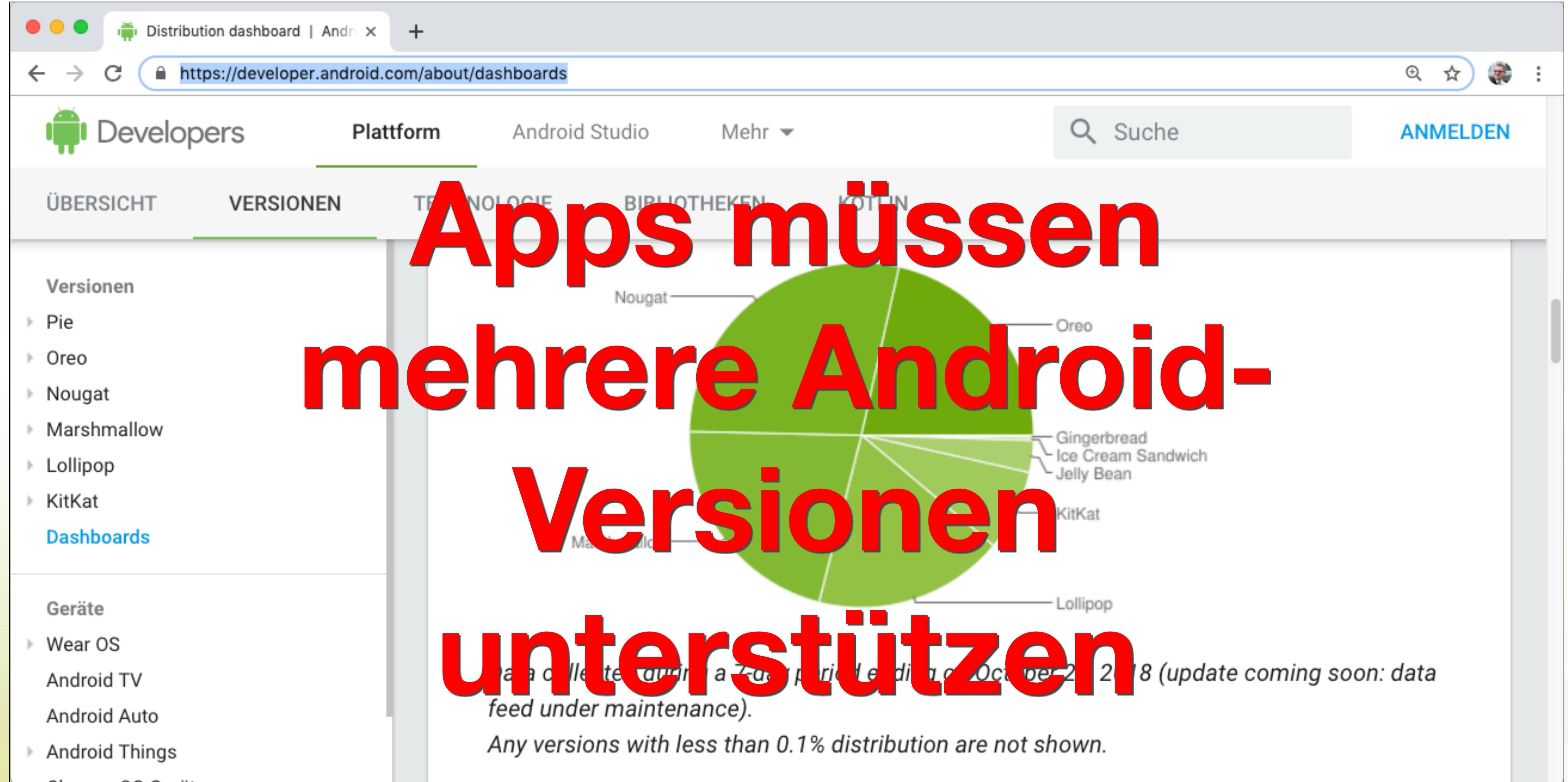
Frühzeitig reagieren

- Änderungen durch neue Android-Versionen
- Änderungen durch neue Bibliotheksversionen
- Aktualisierung der Play Store-Richtlinien

- Lange war Google lax bzgl. Aktualisierung von Apps
...das hat sich geändert...
- Ab August 2019 müssen neue Apps mindestens API-Level 28 unterstützen
- Ab November 2019 müssen aktualisierte Apps mindestens API-Level 28 unterstützen
- Ähnliches galt 2018 für API-Level 26

- Fragmente in Android 3
- Geändertes Berechtigungsmodell in Android 6
- Einschränkungen bei der Hintergrundverarbeitung ab Android 8
- Wahrscheinlich Änderungen bei den Berechtigungen in Android 10

- Eine halbwegs aktuelle App zu aktualisieren ist einfach
- Eine alte App auf den neuesten Stand zu bringen ist aufwendig und bedarf sorgfältiger Planung, weil sich in der Zwischenzeit viele Änderungen angesammelt haben



Android-Versionen

- Interessant für Nutzer
 - Was kann die neue Version?
 - Kommt sie für mein Smartphone?
- Besteht aus typischen 1-, 2-, oder 3-segmentigen Zahlen
- Name einer Süßspeise

API-Level

- Für Entwickler interessant
- Identifiziert die Plattform aus API-Sicht
- Mehrere API-Level pro Version möglich
- App gibt API-Level an 3 Stellen an

minSdkVersion

- Wird in Manifestdatei bzw. build.gradle eingetragen
- Niedrigster möglicher API-Level
- Auf Geräten mit niedrigeren API-Level kann die App nicht installiert werden
- Standard-Wert: 1

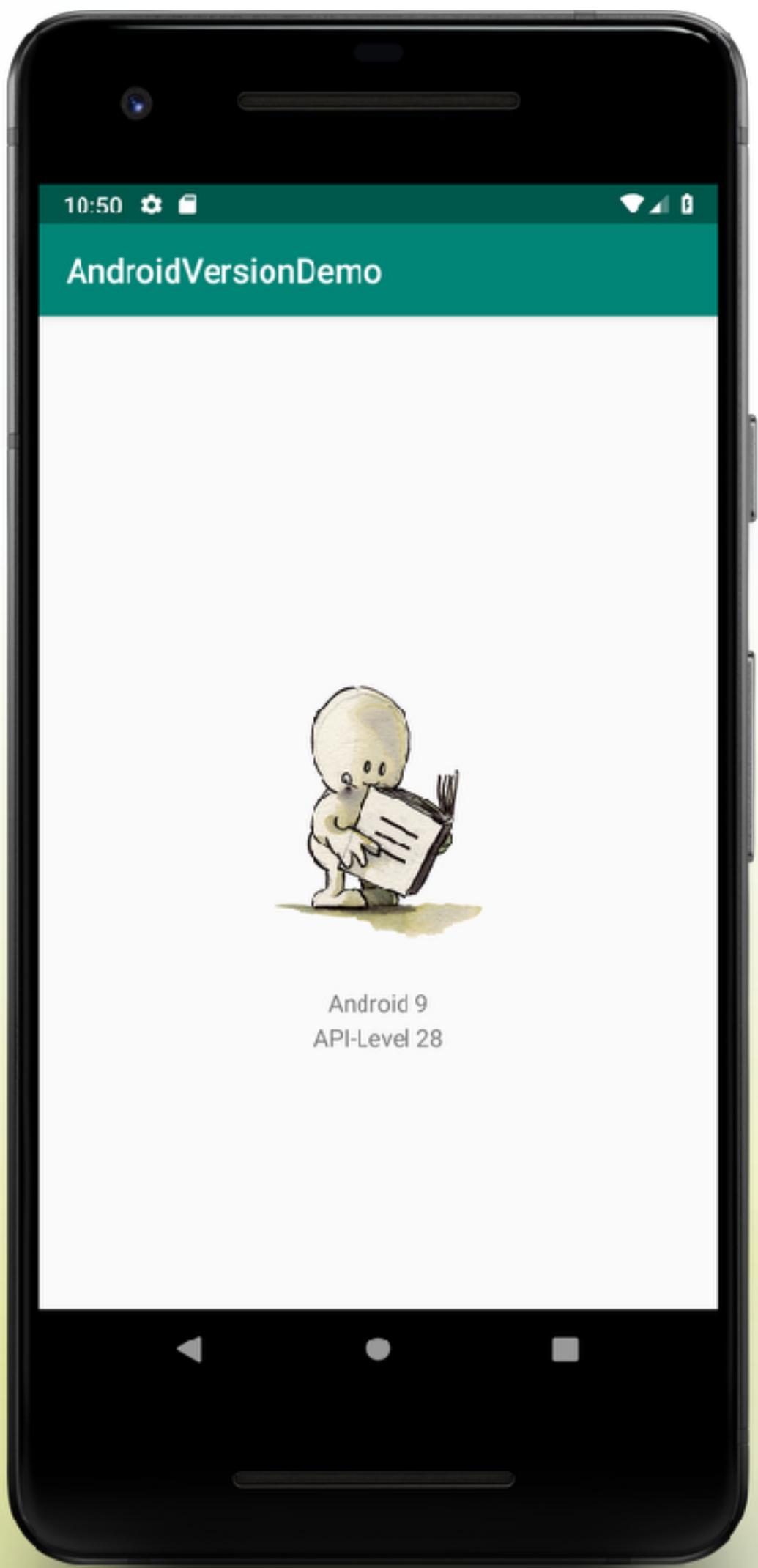
targetSdkVersion

- Wird in Manifestdatei bzw. build.gradle eingetragen
- Gibt den API-Level an, gegen den die App optimiert und getestet wurde
 - Es werden keine Kompatibilitätsfunktionen aktiviert
 - Die App muss aber unter älteren Android-Versionen lauffähig sein (bis minSdkVersion)
- Standard-Wert: minSdkVersion

compileSdkVersion

- Wird in build.gradle eingetragen
- Gibt an, mit welchem API-Level Gradle die App bauen soll
- Viele setzen den Wert auf den aktuellen API-Level

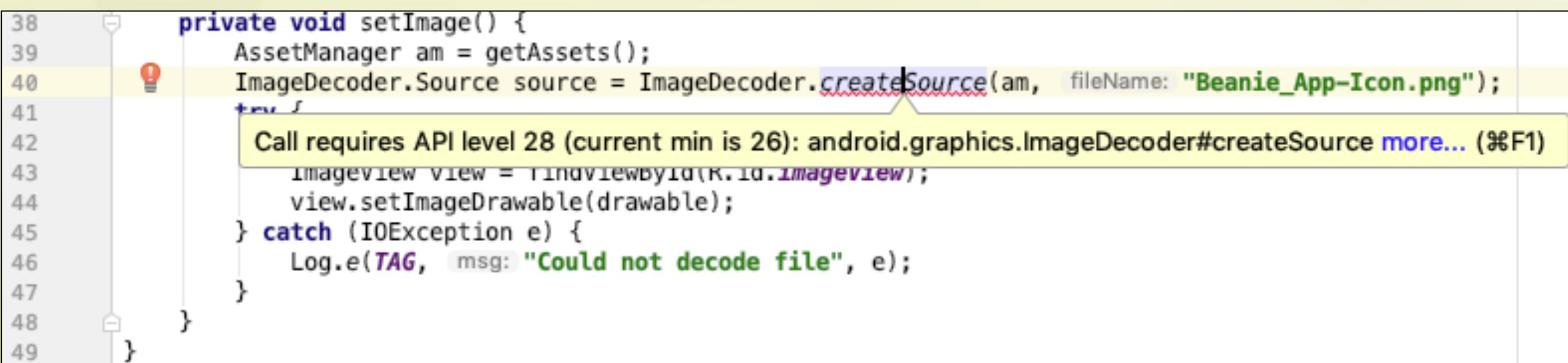
Beispiel: AndroidVersionDemo



**Heruntersetzen der
compileSdkVersion hilft, zu
neue APIs zu entdecken**

Versionsweichen im Code

- Zu niedrige compileSdkVersion provoziert Fehler
(hier müssen Sie tätig werden!)
- Alternative: minSdkVersion setzen und Weichen und Annotationen implementieren



The screenshot shows a code editor in Android Studio with the following Java code:

```
38     private void setImage() {
39         AssetManager am = getAssets();
40         ImageDecoder.Source source = ImageDecoder.createSource(am, fileName: "Beanie_App-Icon.png");
41         try {
42             ImageView view = findViewById(R.id.imageView);
43             view.setImageDrawable(drawable);
44         } catch (IOException e) {
45             Log.e(TAG, msg: "Could not decode file", e);
46         }
47     }
48 }
49 }
```

An error tooltip is displayed over the line `ImageDecoder.createSource(am, fileName: "Beanie_App-Icon.png");`. The tooltip text is: "Call requires API level 28 (current min is 26): android.graphics.ImageDecoder#createSource more... (⌘F1)". This indicates that the `createSource` method was introduced in API level 28, but the current minimum SDK version is 26.

- Machen den Code schwerer lesbar
- Blähen ihn auf
- Schwierig zu testen

**Wenn möglich,
Support-Bibliotheken
nutzen**

- Rüsten neue Funktionen für ältere Plattformen nach
- Gibt es schon lange
- Lange Zeit Wirrwarr bei Namensgebung und Versionierung
 - Nicht klar, für welchen API-Level sie gedacht waren
 - Versionierung sehr uneinheitlich
 - Java-Paketnamen uneinheitlich
- Deshalb Reboot mit AndroidX und Jetpack



Quelle: <https://developer.android.com/images/jetpack/jetpack-hero.svg>

Jetpack

- Sammlung von Software-Komponenten und Best Practices
- Beides kann die Entwicklung beschleunigen und Boilerplate Code vermeiden

Foundation

- AppCompat
- Android KTX
- Multidex
- Test

Behavior

- Download manager
- Media & playback
- Notifications
- Permissions
- Preferences
- Sharing
- Slices

Architecture

- Data Binding
- Lifecycles
- LiveData
- Navigation
- Paging
- Room
- ViewModel
- WorkManager

UI

- Animation & transitions
- Wear OS by Google
- Auto
- Emoji
- Fragment
- Layout
- Palette
- TV

**Bestehende Support-
Bibliotheken werden
in Jetpack weiter
entwickelt**

AndroidX

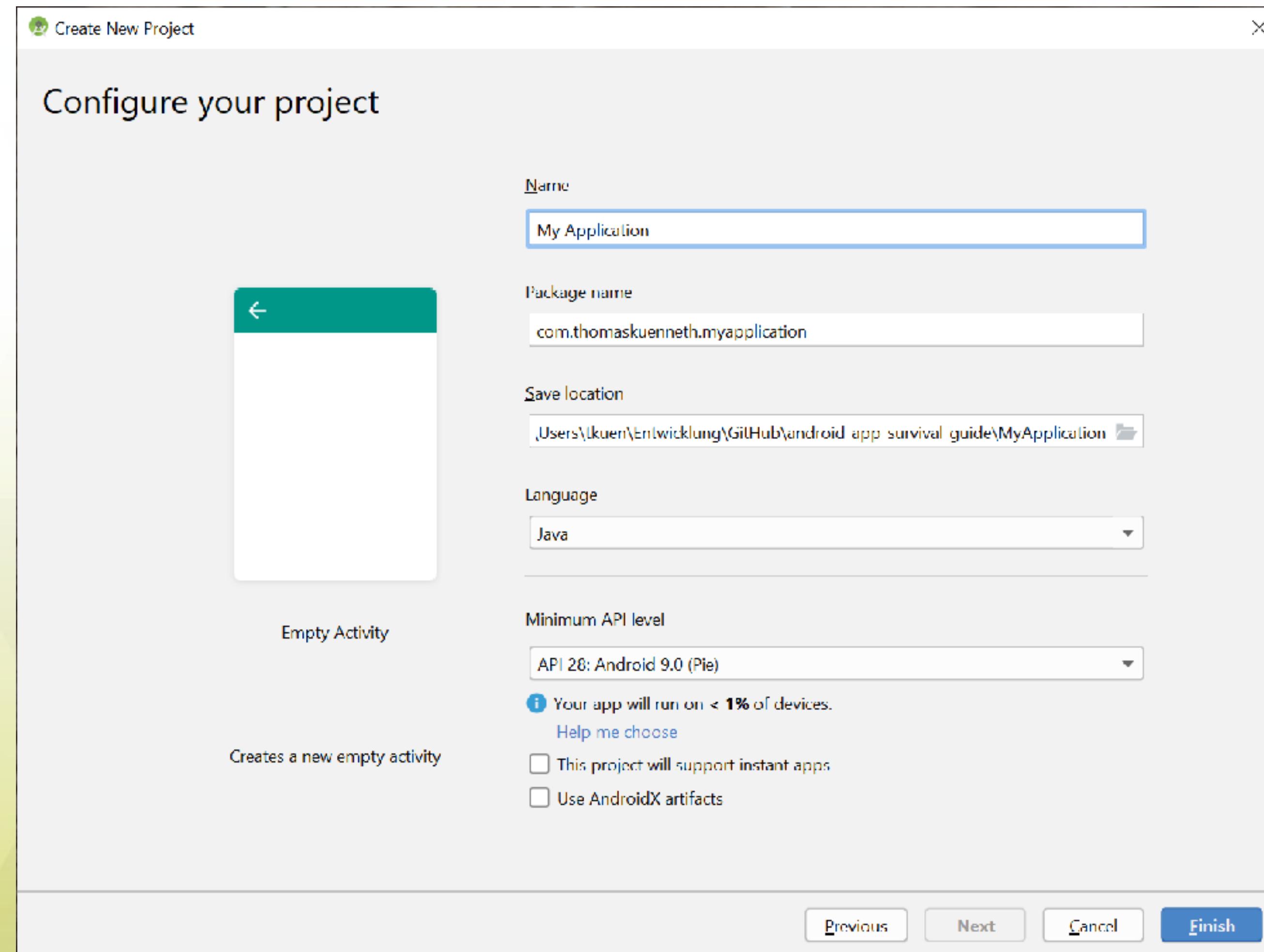
- Bietet Rückwärtskompatibilität über mehrere Plattform-Versionen
- Wird wie die ursprüngliche Support Bibliothek unabhängig von der Plattform verteilt
- Ersetzt die Support Bibliothek

- Konsistenter Namensraum: androidx.*
- Pakete der Support Bibliothek werden auf den neuen Namensraum gemappt
- AndroidX-Pakete werden unabhängig voneinander gepflegt und aktualisiert
- Folgen einer strikten semantischen Versionierung, beginnend mit 1.0.0

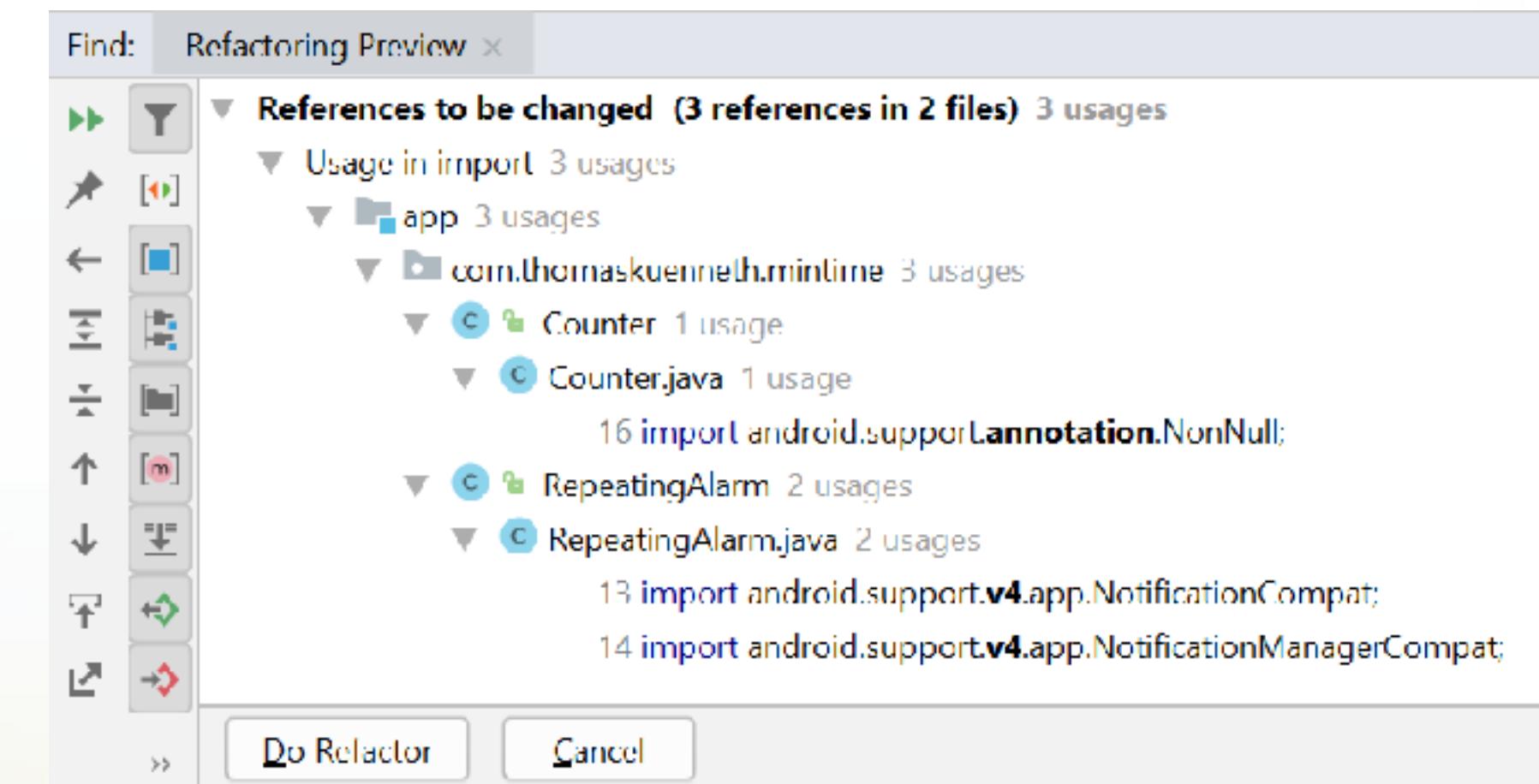
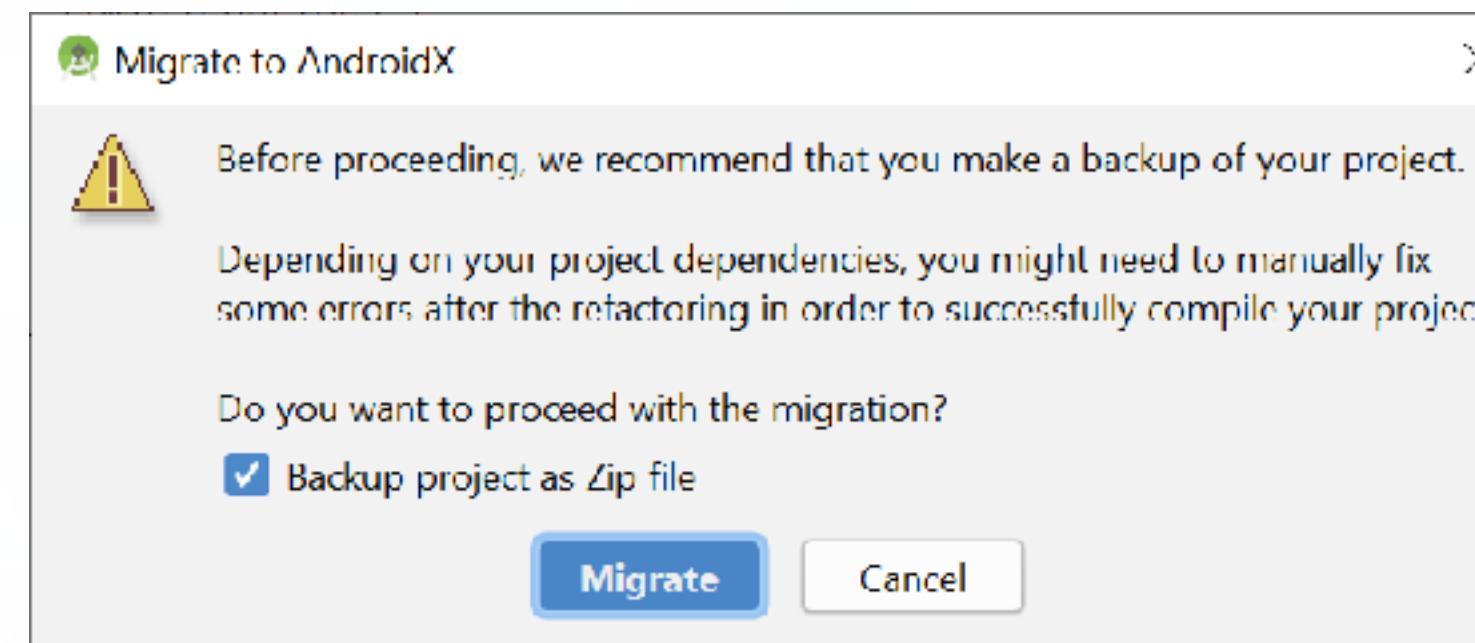
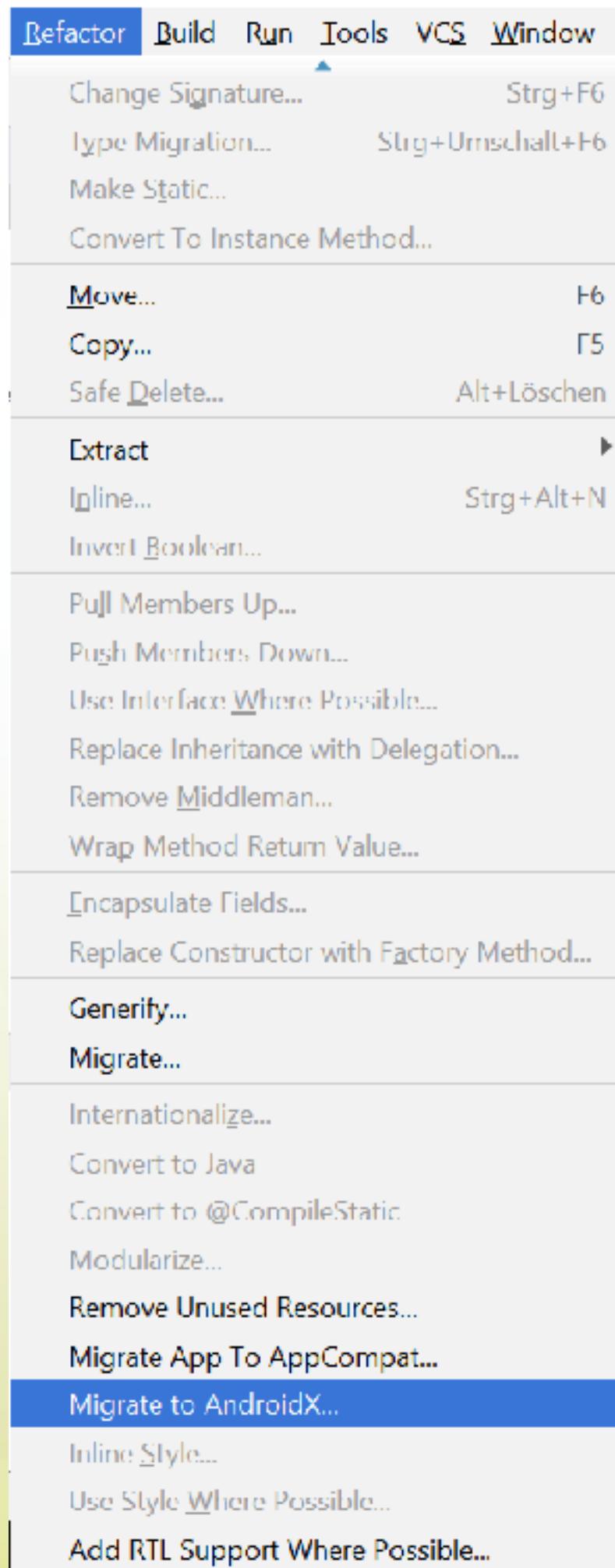
AndroidX verwenden

- compileSdkVersion auf mindestens API-Level 28 setzen
- In gradle.properties:
 - android.useAndroidX: Das Android Plugin verwendet AndroidX anstelle der Support Bibliothek
 - android.enableJetifier: Das Android Plugin migriert vorhandene Drittanbieter-Bibliotheken nach AndroidX, indem die Bibliotheken umgeschrieben werden

Beim Anlegen eines neuen Projekts



Vorhandene Projekte aktualisieren



- Derzeit werden Referenzen in *build.gradle* **nicht aktualisiert**
- Deshalb
implementation "com.android.support:appcompat-v7:28.0.0"
ändern in
implementation "androidx.appcompat:appcompat:1.0.2"
- Mappings:
<https://developer.android.com/jetpack/androidx/migrate>

Steigen Sie auf AndroidX um

App Bundles

- Google propagiert seit einiger Zeit so genannte App Bundles
- Finden beim Upload in die Play Console Verwendung
- Nutzer erhalten weiterhin .apk-Dateien
 - Auf ein Gerät optimiert
 - Deshalb (viel) kleiner

App-Signatur - TKBirthdayRem X TKBirthdayReminder – Apps be X +

https://play.google.com/apps/publish/?account=4902678604945597505#KeyManagementPlace:p=com.thomaskuenneth.android.birthday&appid=4974905993589... 🔍 ⭐ 🚫 :

Deobfuscation-Dateien

Entwickler-Tools

Release-Verwaltung

Release-Dashboard

App-Releases

Android Instant-Apps

Artefaktebibliothek

Gerätekatalog

App-Signatur

Pre-Launch-Bericht

App-Präsenz im Play Store

Store-Eintrag

Tests für Store-Einträge

App-Signatur

TKBirthdayReminder
Veröffentlicht

App-Signaturschlüssel von Google verwalten und schützen lassen

Wenn du das [Android App Bundle](#) nutzen und von den Vorteilen von Dynamic Delivery in Google Play profitieren möchtest, musst du Google Play deinen App-Signaturschlüssel verwalten lassen. Ein weiterer Vorteil hierbei ist, dass der Signaturschlüssel durch dieselbe Sicherheitsinfrastruktur geschützt wird, die auch für die Sicherheit unserer Google-Dienste sorgt.

Du kannst die App-Signatur jetzt aktivieren, während du weiterhin APKs mit deinem App-Signaturschlüssel für die Produktion veröffentlichtst. So kannst du dein App Bundle intern oder mit den Alpha- oder Beta-Zielgruppen testen, ohne den aktuellen Releasevorgang zu ändern. [Weitere Informationen zur Funktionsweise der App-Signatur von Google Play](#).

Mit dem Android App Bundle die Größe deines nächsten Release reduzieren

 Die Größe deiner App kann um 27,8 % verringert werden, wenn du das Android App Bundle verwendet. Mithilfe der APKs, die vom App Bundle generiert werden, kannst du kleinere und optimierte Downloads für deine Nutzer veröffentlichen. [So funktioniert's](#) Diese Berechnung basiert auf deinem letzten Produktionsrelease und der Gerät konfiguration XXHDPI ARMv7.

Geschätzte
Größeneinsparungen
27,8 %

Wähle eine der folgenden Optionen aus, um dich jetzt anzumelden

- Einen aus Android Studio exportierten Schlüssel hochladen
- Einen Schlüssel aus einem Java Keystore exportieren und hochladen
- Einen Schlüssel exportieren und hochladen (nicht aus einem Java Keystore)

FERTIGSTELLEN

Zusammenfassung

- Zur App-Pflege gehören eine Reihe von Themen jenseits der klassischen Entwicklung
- Je nach App-Typ können diese sehr aufwendig sein
- Sind für eine positive Wahrnehmung unerlässlich
- Um entspannt mehrere Android-Versionen zu unterstützen, unbedingt auf Jetpack umsteigen
- App Bundles können Downloads signifikant verkleinern

1. Übergreifende Themen
- 2. Effizient, effektiv und automatisiert testen**
3. Debuggen, profilieren und Abstürze analysieren
4. Zielgerade

Was bedeutet eigentlich „Testen“?

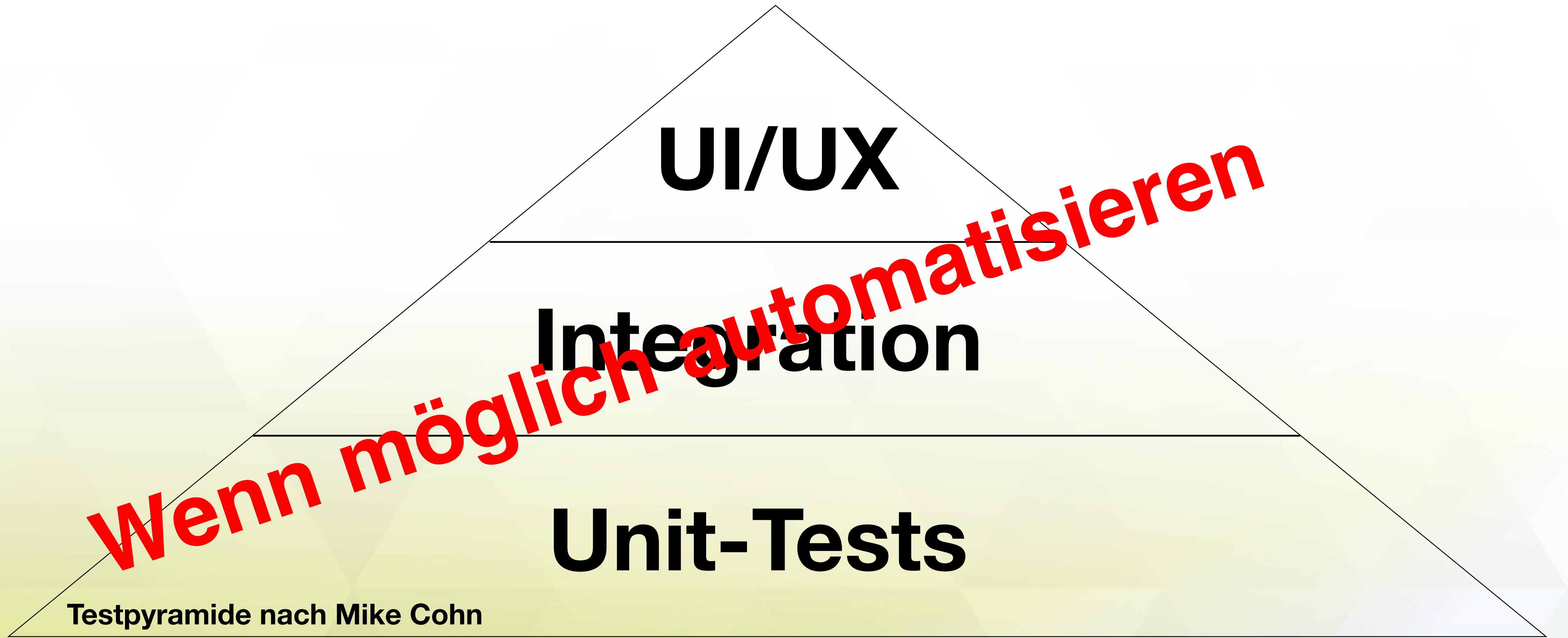
- Warum testen wir?
- Was testen wir?
- Wann testen wir?
- Wie testen wir?

- App soll wie gewünscht funktionieren
- App soll sich auf dem Gerät „gut benehmen“
 - Akku nicht leer saugen
 - Datenvolumen nicht fressen
 - Keine anderen Apps behindern

Testen ist mehr als „testen“

- **Testen**
Prüfen, ob alles richtig funktioniert
- **Debuggen**
Nachvollziehen, wenn etwas nicht richtig funktioniert
- **Profilen**
Das Verhalten zur Laufzeit verstehen

Die Testpyramide



Unit-Tests

- Prüfen die „Geschäftslogik“
- Sind einfach zu erstellen
- So viele wie möglich und sinnvoll

- Haben nichts mit der Plattform oder dem Gerät zu tun
- Prüfen nicht über Schnittstellen hinweg
- Modulexterne Abhängigkeiten werden gemockt

System- und Integrationstests

- Nutzung von Systemressourcen
- Verwendung von Schnittstellen und Bibliotheken
- Werden auf dem Gerät oder Emulator ausgeführt

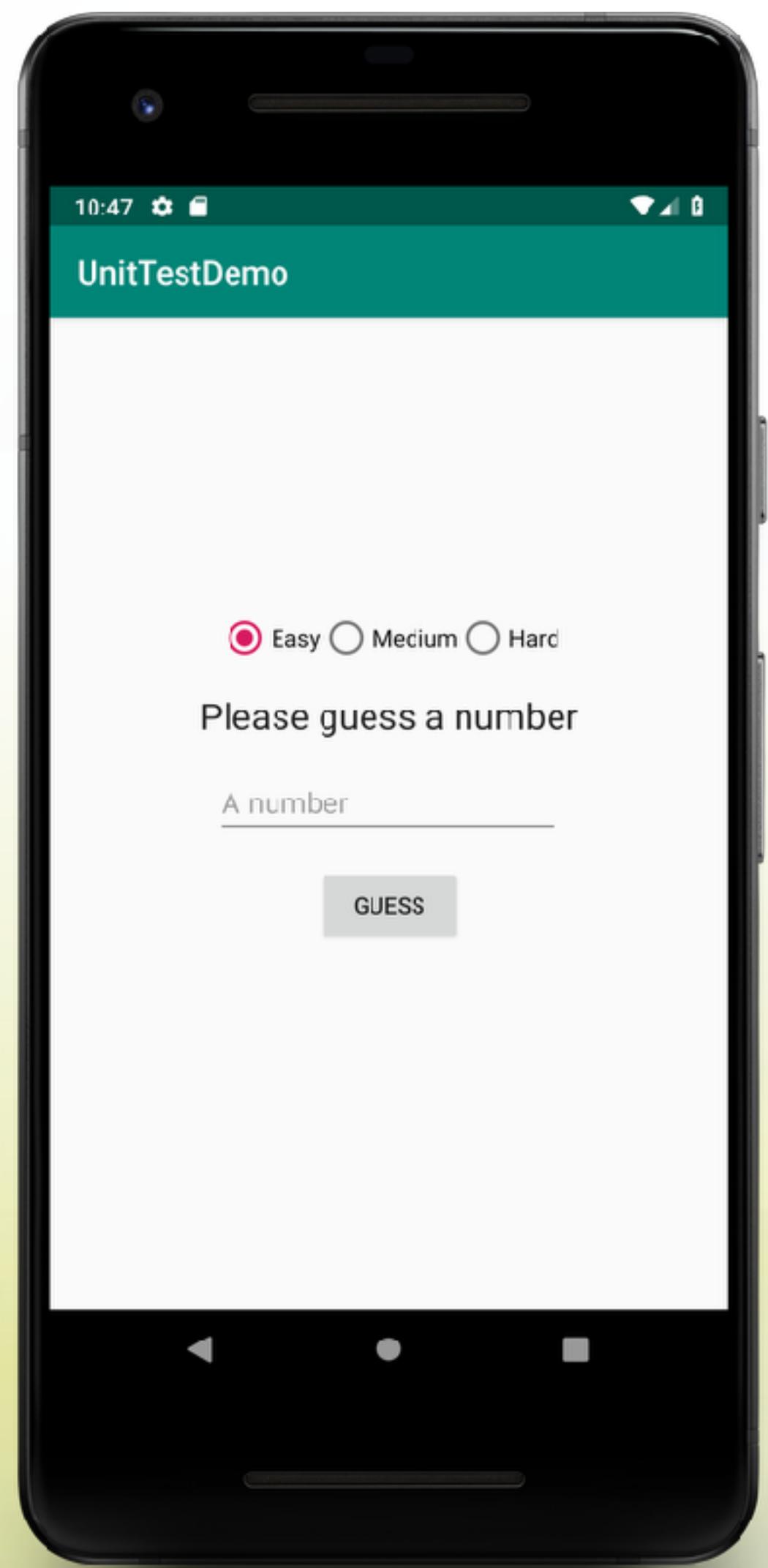
UI/UX

- Stehen in Feldern die erwarteten Werte?
- Funktionieren Interaktionen?
- Klappt die Navigation?

- Reagiert die App flüssig?
- Passt sie sich bei Drehungen an?
- Läuft sie auf unterschiedlichen Geräteklassen?
- Geht sie schonend mit Ressourcen um?
- Passt sie sich dem Benutzer und seine Vorlieben gut an?

Unit-Tests

Beispiel: UnitTestDemo



Voraussetzungen für gute Tests

- Sauber strukturierter Code
- Geschäftslogik nicht mit UI-Code verwoben

Voraussetzungen für gute Tests

- Sauber strukturierter Code,
weil die Struktur die zu testenden Einheiten offen legt
- Geschäftslogik nicht mit UI-Code verwoben,
weil nur so die Logik lokal testbar ist

- Konsequente, gute Unit-Tests zwingen zu sauberen Strukturen
- Fördern kontinuierliches Refactoring
- Helfen so, Fehler zu finden und zu vermeiden

UnitTestDemo Refactoring

- Erstellen einer Klasse Game mit den Methoden setup () und guess ()
- Erstellen eines enumS Fifficulty
- Klasse und enum in MainActivty nutzen

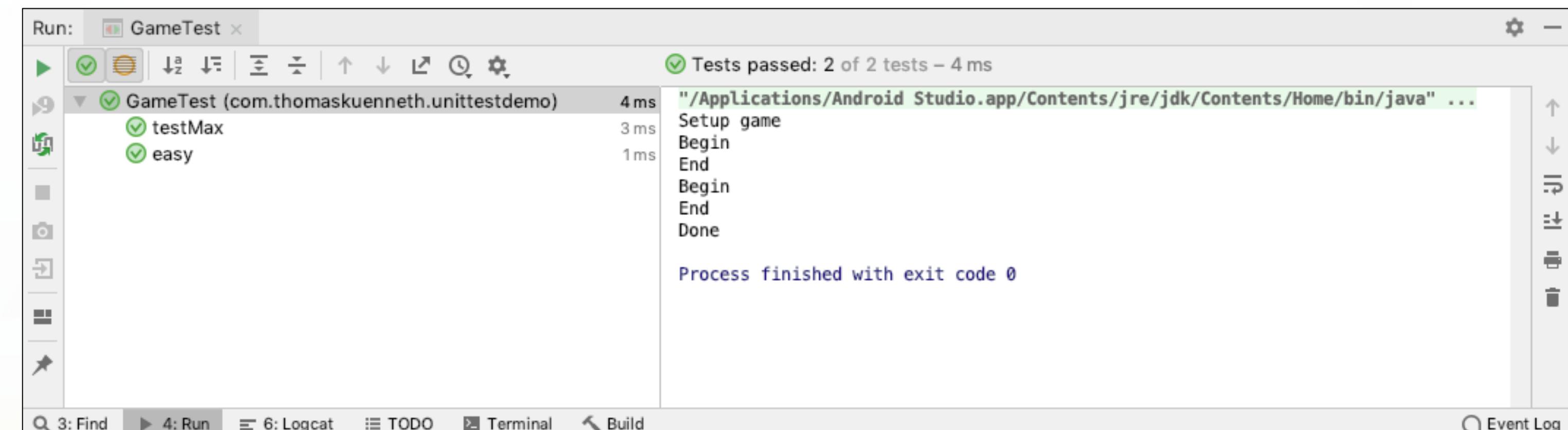
Lokale Unit-Tests in Android Studio

- JUnit 4 in *build.gradle* referenzieren:
`testImplementation 'junit:junit:4.12'`
- Einfach Klassen unter *src/test/java* anlegen
- Testmethoden mit `@org.junit.Test` annotieren

- Um etwas vor einem Test zu erledigen, kann der entsprechende Code mit @Before annotiert werden
- @After führt Code nach einem Test aus
- @BeforeClass und @AfterClass können zum Initialisieren und Aufräumen verwendet werden

Demo: UnitTests

```
10 public class GameTest {  
11  
12     private static Game game;  
13  
14     @BeforeClass  
15     public static void initialize() {  
16         System.out.println("Setup game");  
17         game = new Game();  
18     }  
19  
20     @AfterClass  
21     public static void finished() {  
22         System.out.println("Done");  
23     }  
24  
25     @Before  
26     public void begin() {  
27         System.out.println("Begin");  
28     }  
29  
30     @After  
31     public void end() {  
32         System.out.println("End");  
33     }  
34  
35     @Test  
36     public void testMax() {  
37         Assert.assertTrue( condition: 1 <= game.getMax(Game.Difficulty.EASY));  
38         Assert.assertTrue( condition: 1 <= game.getMax(Game.Difficulty.MEDIUM));  
39         Assert.assertTrue( condition: 1 <= game.getMax(Game.Difficulty.HARD));  
40     }
```



Tipps für Tests

- Tests sollten kurz sein
- Tests müssen nachvollziehbar
- Namen der Testmethoden sollten aussagekräftig sein
- Tests so sorgfältigen wie jeden anderen Code schreiben

Android-spezifische Tests

Instrumentierte Unit-Tests

- Werden auf einem echten Gerät oder dem Emulator ausgeführt
- Werden unter <modul>/src/androidTest/java/ abgelegt
- In build.gradle eintragen:

```
testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"  
...  
androidTestImplementation 'androidx.test:rules:1.1.0-alpha4'  
androidTestImplementation 'androidx.test:runner:1.1.0-alpha4'  
androidTestImplementation 'androidx.test.espresso:espresso-core:  
3.1.0-alpha4'
```

Was ist Espresso?

- Google-Framework für Android-Oberflächentests
- Kompakte, einfach zu verstehende und zu nutzende API
- Tests werden mittels Fluent API definiert

Pakete

- espresso-core
- espresso-web (Unterstützung für die WebView)
- espresso-idling-resource (Synchronisierung mit Hintergrundjobs)
- espresso-contrib (Zulieferungen von Drittanbietern, z. B. Aktionen für DatePicker und RecyclerView)
- espresso-intents (Nutzung von Intents)
- espresso-remote (Multi-Prozess-Unterstützung)

Eine einfache Testklasse

```
1 package com.thomaskuenneth.unittestdemo;
2
3 import androidx.test.filters.LargeTest;
4 import androidx.test.rule.ActivityTestRule;
5 import androidx.test.runner.AndroidJUnit4;
6
7 import org.junit.Rule;
8 import org.junit.Test;
9 import org.junit.runner.RunWith;
10
11 import static androidx.test.espresso.Espresso.onView;
12 import static androidx.test.espresso.action.ViewActions.click;
13 import static androidx.test.espresso.action.ViewActions.typeText;
14 import static androidx.test.espresso.matcher.ViewMatchers.withId;
15
16 @RunWith(AndroidJUnit4.class)
17 @LargeTest
18 public class MainActivityTest {
19
20     // https://developer.android.com/training/testing/espresso/setup
21
22     @Rule
23     public ActivityTestRule<MainActivity> activityRule =
24         new ActivityTestRule<>(MainActivity.class);
25
26     @Test
27     public void test1() {
28         onView(withId(R.id.button_easy)).perform(click());
29         onView(withId(R.id.edittext_guess)).perform(typeText("3"));
30         onView(withId(R.id.button_guess)).perform(click());
31     }
32 }
```

Hamcrest Matcher

- Finden Views auf Basis von Eigenschaften/Bedingungen
 - `withId(...)` matcht auf der Basis von Resource-Ids
 - `withResourceName(...)` auf der Basis von Ressourcen-Namen
 - `withText(...)` matcht TextViews auf Basis ihrer aktuellen Texte

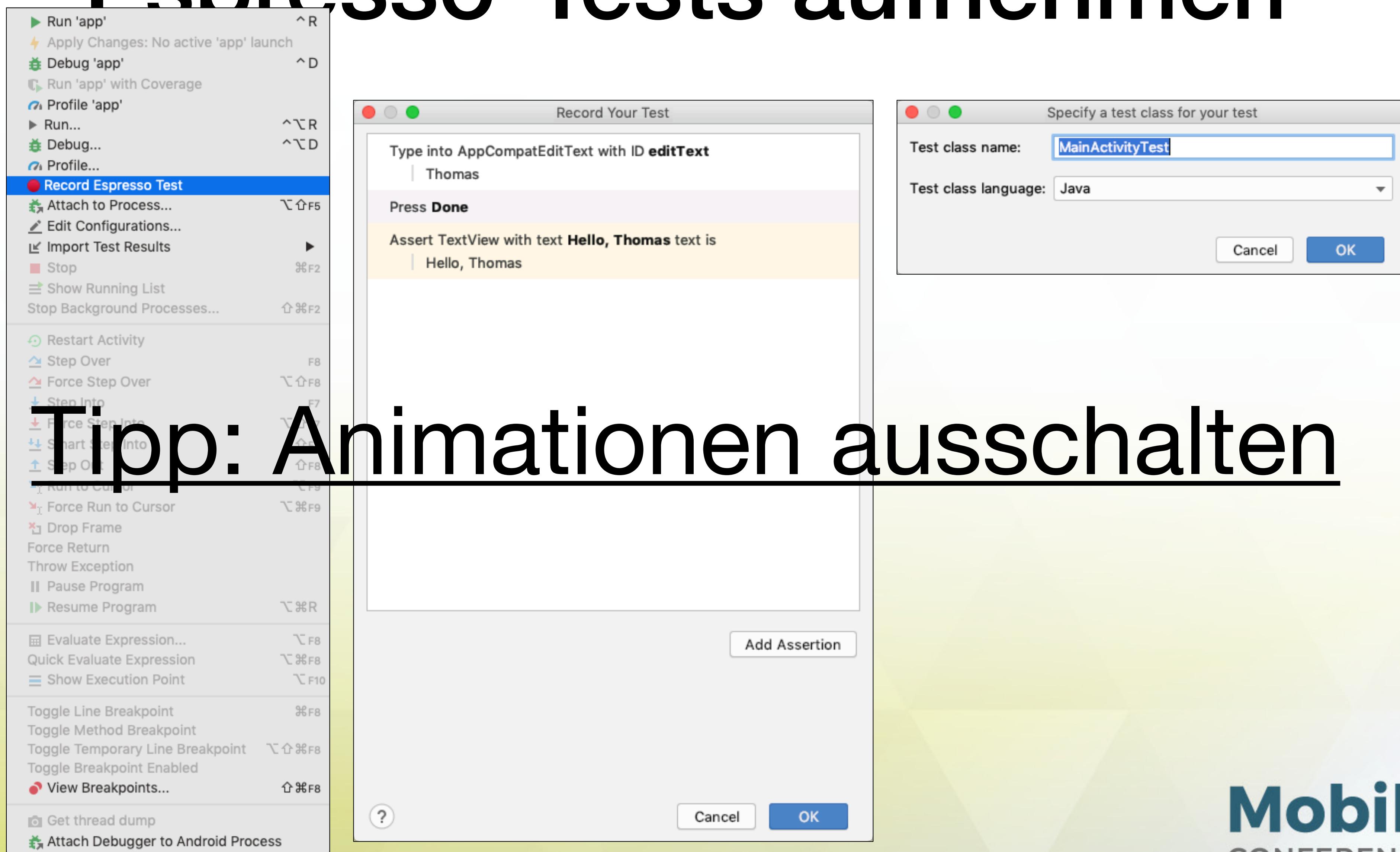
Interaktionen

- ViewInteraction zentrale Zugriffsklasse für Aktionen und Annahmen auf Views
- DataInteraction erlaubt Zugriff auf Daten von AdapterViews
- Instanzen werden von Espresso.onView() und onData() geliefert
- ViewInteraction.perform() führt ViewActions aus

ViewActions

- Häufig verwendete ViewActions werden in der Klasse `androidx.test.espresso.action.ViewActions` definiert
- `clearText()` leert Textfelder
- `swipeLeft()` simuliert eine Wischegeste
- `closeSoftKeyboard()` schließt die Bildschirmtastatur

Espresso-Tests aufnehmen



Tipp: Animationen ausschalten

- Oberflächentests leben mit der UI
- Bei Änderungen der Bedienoberfläche müssen Tests natürlich angepasst werden
- Tipps für gute Tests gelten auch hier

- Tests aktuell halten
- Tests sind nachvollziehbar
- Ein Test – eine Prüfung (keine `testUI()`-Methode)

Weitere Möglichkeiten

- Mockito ist eine Bibliothek zum Erstellen von Mock-Objekten für Unit-Tests
- Mock-Objekte kommen z. B. zum Einsatz, wenn in Unit-Tests auf Schnittstellen zugegriffen werden soll
- Kann auch unter Android verwendet
- Hierzu build.gradle ergänzen:
`testImplementation 'org.mockito:mockito-core:1.10.19'`

```
1 package com.thomaskuenneth.unittestdemo;
2
3 import android.widget.TextView;
4
5 import org.junit.Test;
6 import org.junit.runner.RunWith;
7 import org.mockito.Mock;
8 import org.mockito.runners.MockitoJUnitRunner;
9
10 import static org.hamcrest.CoreMatchers.is;
11 import static org.hamcrest.MatcherAssert.assertThat;
12 import static org.mockito.Mockito.when;
13
14 @RunWith(MockitoJUnitRunner.class)
15 public class MockitoTest {
16
17     private static final CharSequence HELLO_WORLD = "HELLO WORLD";
18
19     @Mock
20     TextView mockTextView;
21
22     @Test
23     public void readStringFromContext_LocalizedString() {
24         when(mockTextView.getText())
25             .thenReturn(HELLO_WORLD);
26         CharSequence result = mockTextView.getText();
27         assertThat(result, is(HELLO_WORLD));
28     }
29 }
```

Robolectric

- Ermöglicht ebenfalls Unit-Tests unter Android
- Tests laufen in der JVM auf dem Entwicklungsrechner
- Die Sandbox schottet Tests von einander ab
- Das simulierte Android-Framework kann nach Belieben konfiguriert werden

Zusammenfassung

- Fachlichkeit konsequent mit Unit-Tests absichern
- Activities mit Espresso-Tests prüfen
- Einsatz weiterer Test-Frameworks abwägen
- Backends natürlich ebenfalls testen

1. Übergreifende Themen
2. Effizient, effektiv und automatisiert testen
- 3. Debuggen, profilieren und Abstürze analysieren**
4. Zielgerade

Debuggen

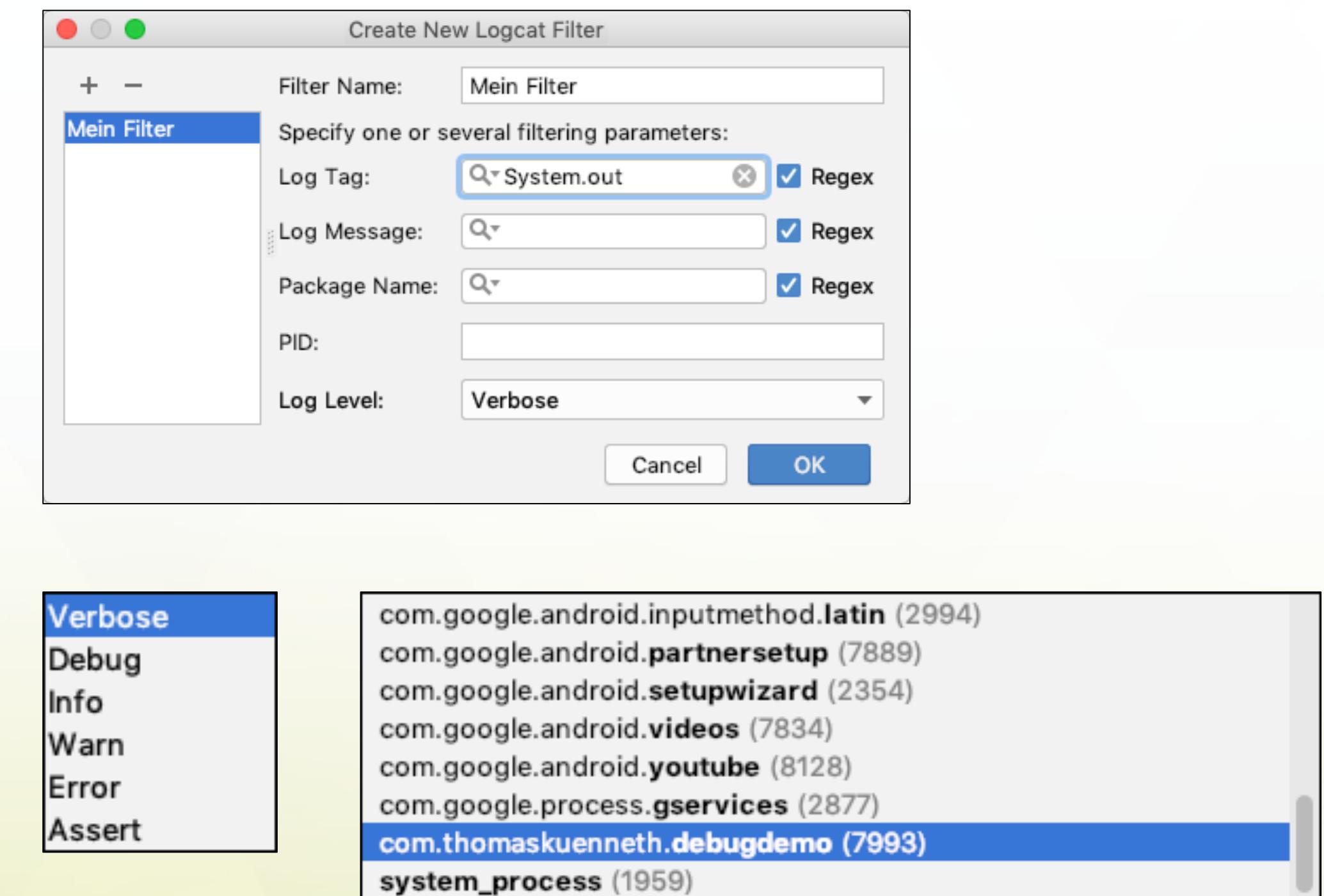
- Verhalten zur Laufzeit anhand des Quelltextes nachvollziehen
- Ziel: einen Fehler finden
- Programmablauf kann angehalten werden
- Inhalte von Variablen kann ausgelesen und geändert werden

Meldungen ausgeben

- Konsolausgaben landen unter Android in LogCat
- Java-Entwickler kennen `System.out.println()` (funktioniert auch unter Android)
- Schöner: `Log.v()`, `Log.d()`, `Log.i()`, `Log.w()` bzw.
`Log.e()`

LogCat filtern

- Auf Basis eines Tags
- Nach Log Level
- Nach Prozessen

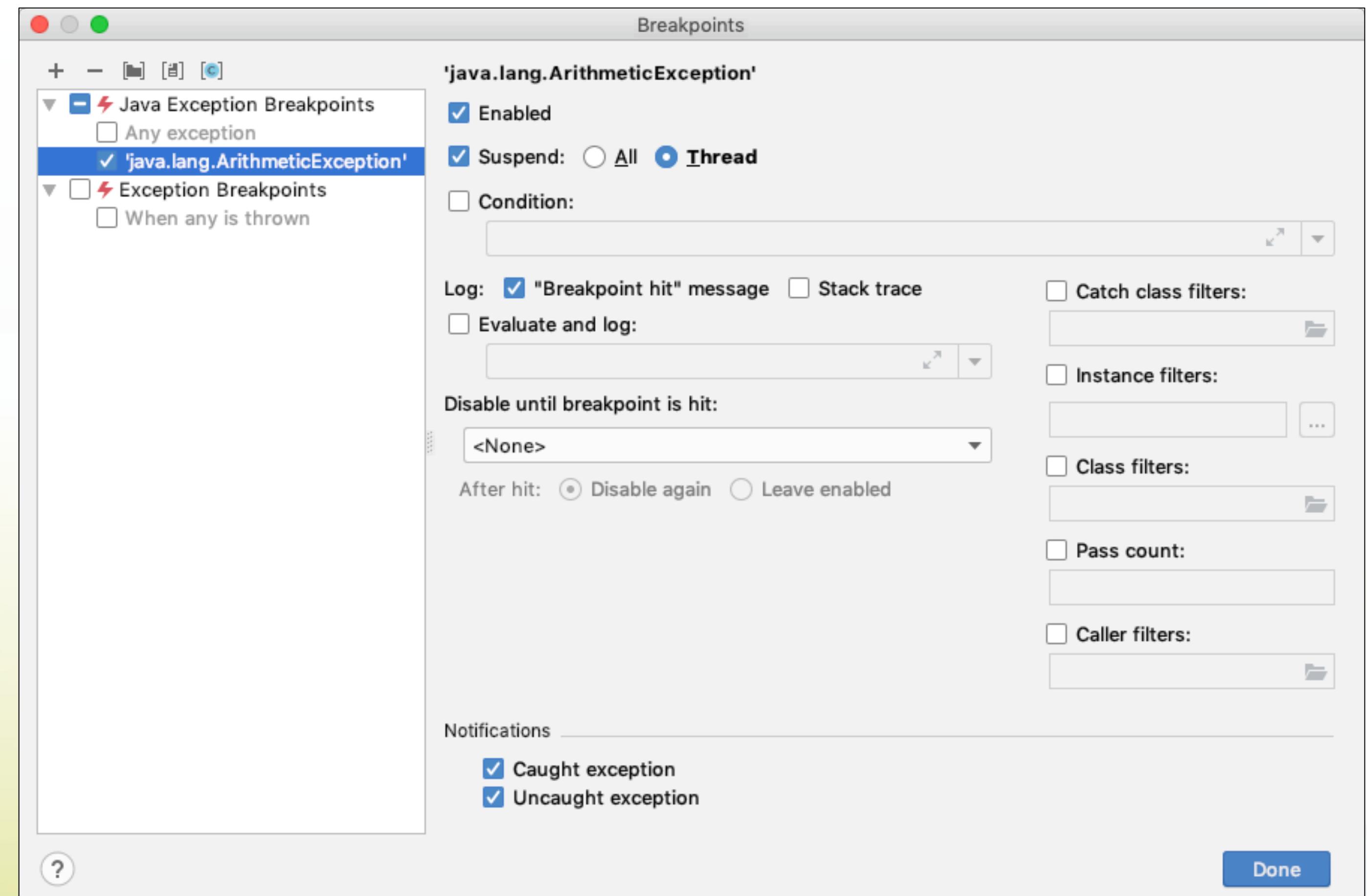
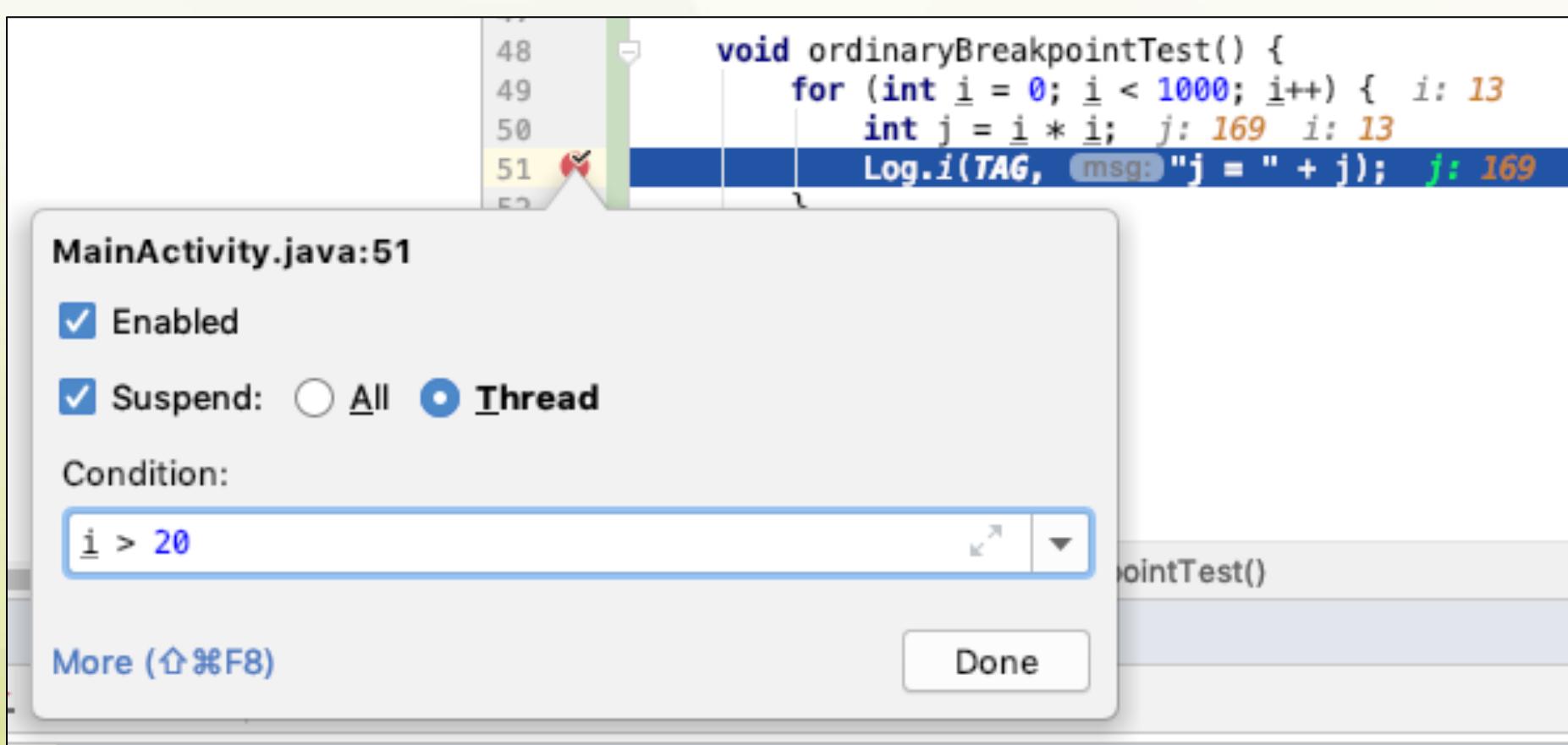


Log-Varianten

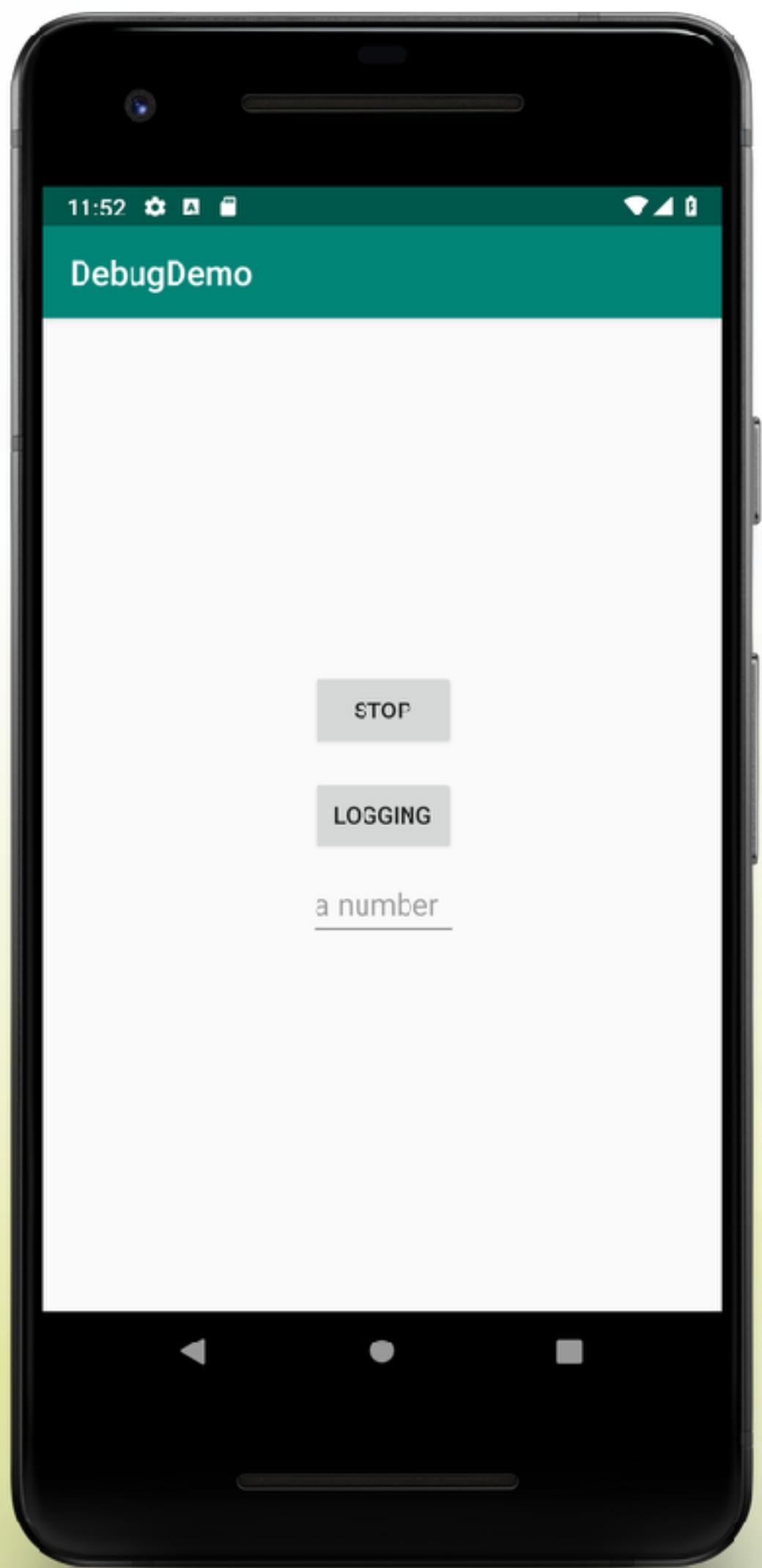
- Anstelle von Methodennamen kann auch über Loglevels selektiert werden: VERBOSE, DEBUG, INFO, WARN, ERROR, ASSERT in Verbindung mit `println()`
- Ob ein bestimmter Level geloggt werden kann, sollte mit `isLoggable()` geprüft werden
- Anekdoten: Lustige Log-Methode `wtf()`
- Beim Logging können Exceptions übergeben werden
- Log Level für Tag setzen: `adb shell setprop log.tag.MainActivity ...`

Breakpoints

- ◆ 1. Java Method Breakpoints
- 2. Java Field Watchpoints
- ⚡ 3. Java Exception Breakpoints
- 4. Kotlin Field Watchpoints
- ⚡ 5. Exception Breakpoints
- ◆ 6. Symbolic Breakpoints



Demo: DebugDemo



Stack Frames

The screenshot shows an Android Studio debugger interface. The top part displays the code for the `Factorial` class:package com.thomaskuenneth.debugdemo;
class Factorial {
 static int factorial(int f) {
 int result = 1;
 System.out.println("0! = " + result);
 for (int i = 1; i <= f; i++) {
 result = i * result;
 System.out.println(i + "!" + " = " + result);
 }
 return result;
 }
 static int factorialRecursive(int n) {
 if (n <= 1) { n: 3
 return 1;
 }
 return factorialRecursive(n - 1) * n;
 }
}The line `if (n <= 1) { n: 3` is highlighted with a red breakpoint. The bottom part of the screenshot shows the debugger's frames view. It lists several stack frames, with the first one being the current frame:

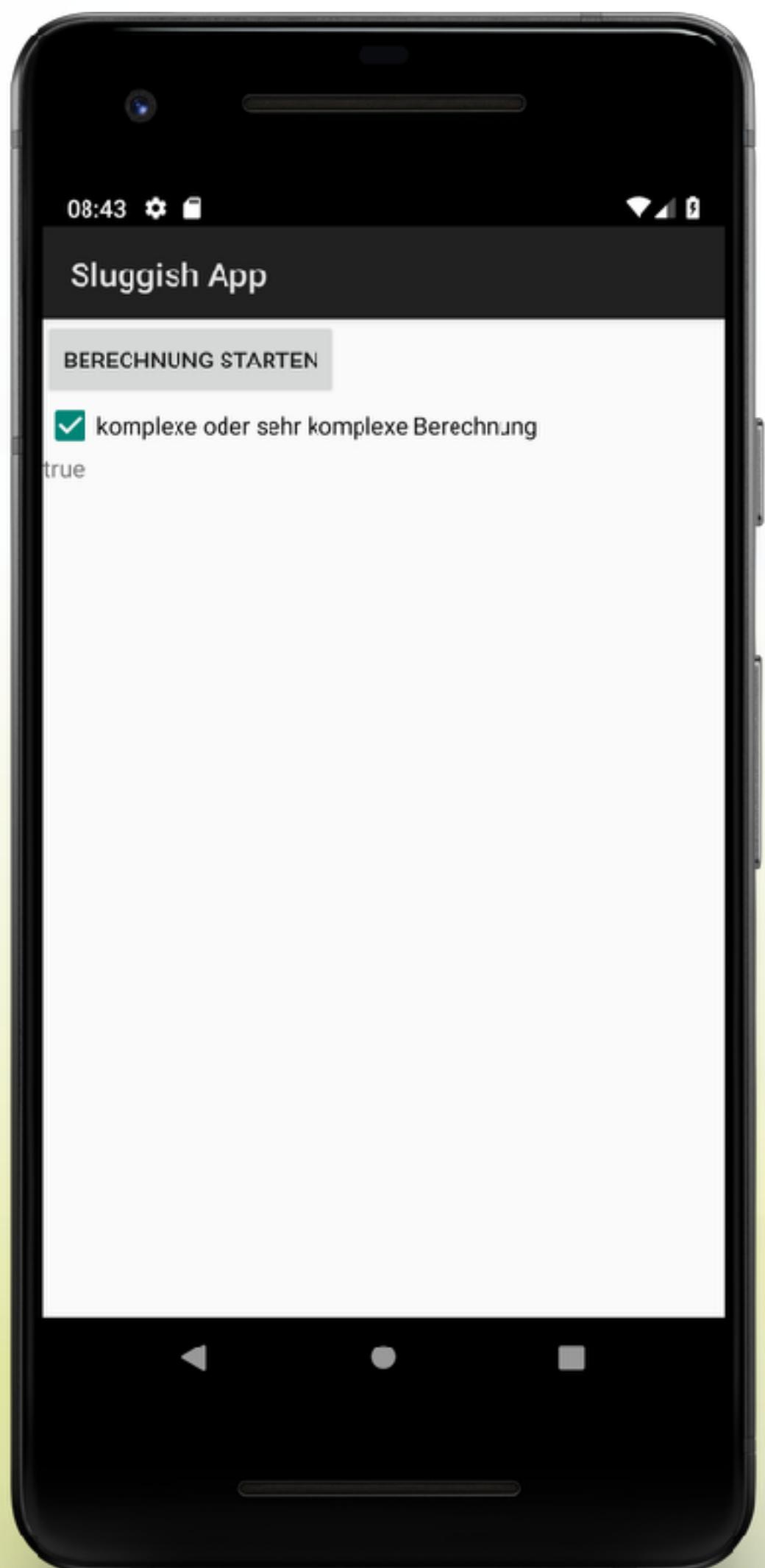
```
"main"@10,688 in group *main...  
factorialRecursive:16, Factorial (com.thomaskuenneth.debugdemo)  
factorialRecursive:19, Factorial (com.thomaskuenneth.debugdemo)  
factorialRecursive:19, Factorial (com.thomaskuenneth.debugdemo)  
lambda$onCreate$0:34, MainActivity (com.thomaskuenneth.debugdemo)
```

The variable `n` is shown with a value of 3. A note indicates that the local variable `i` cannot be found.

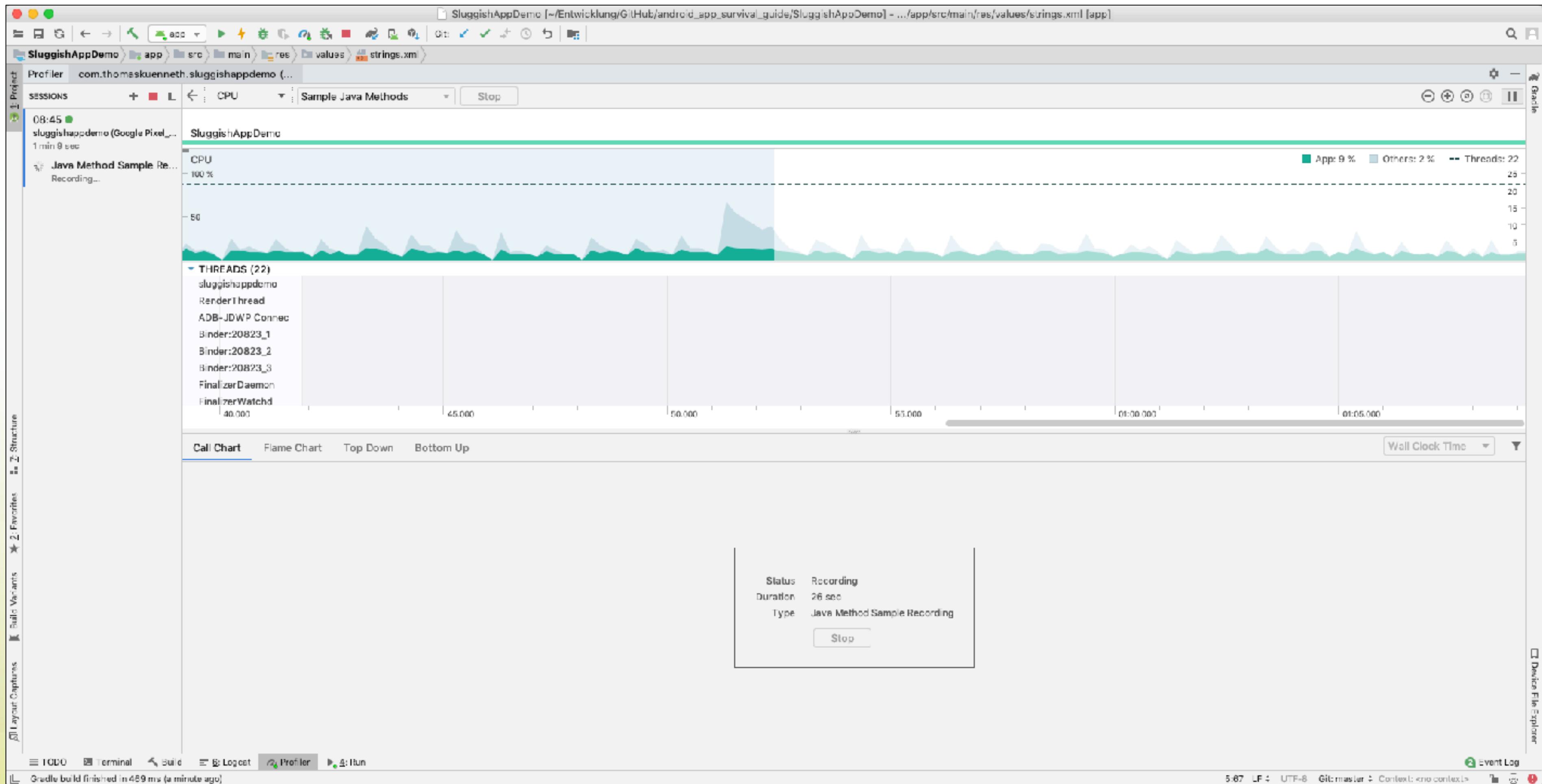
Profilen

- Zur Vorbeugung
- Bei konkreten Problemen
 - stotternde Animationen
 - hoher Stromverbrauch
 - Abstürze
 - Einfrieren (ANR)

Demo: SluggishAppDemo



Demo: Android Profiler



Main thread

- Main (UI) thread für Änderungen der Benutzeroberfläche
- Rechenzeitintensive Aktionen müssen auf andere Threads verteilt werden
- Übliches Vorgehen bei UI-Frameworks

StrictMode

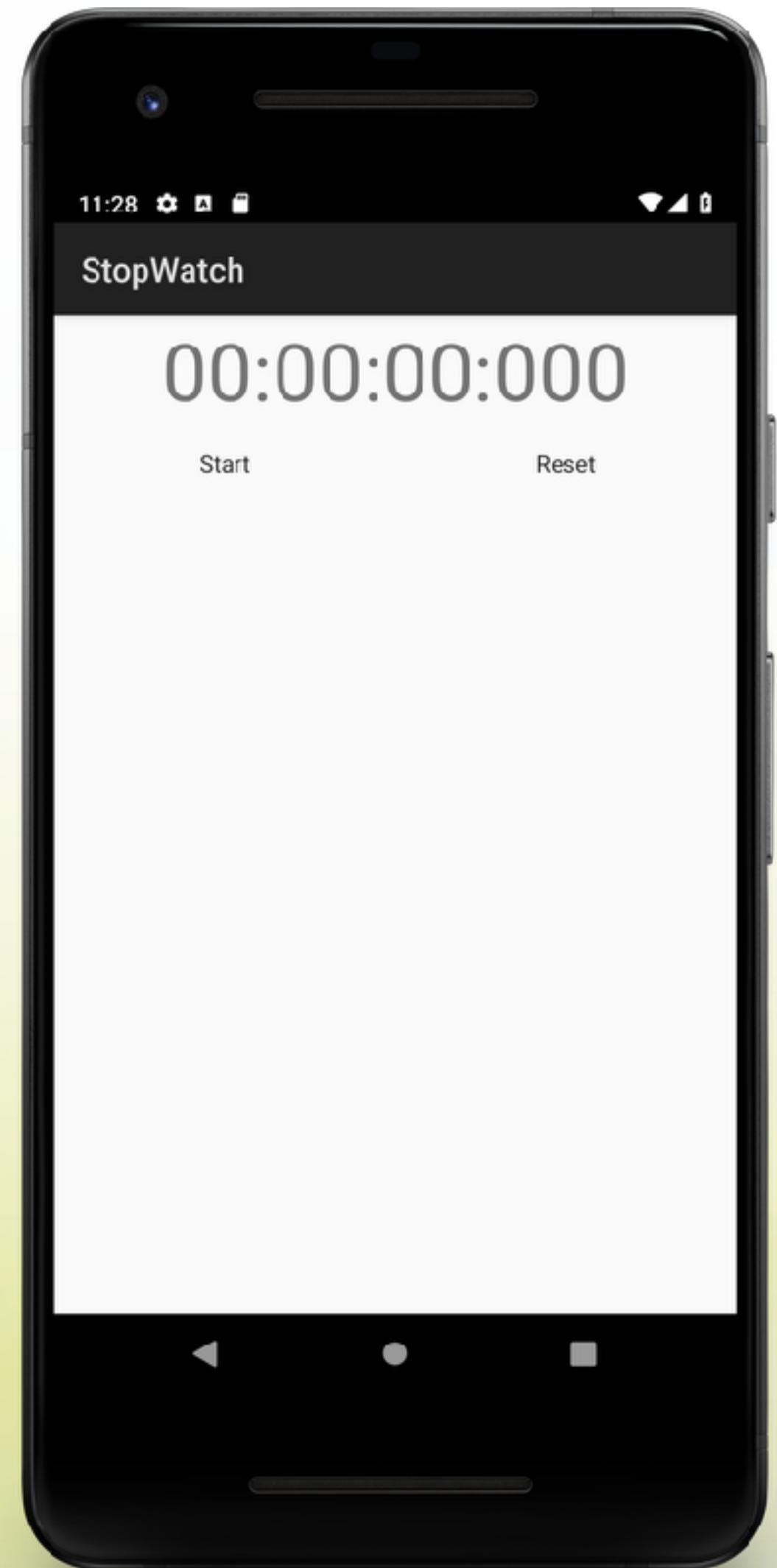
```
StrictMode.setThreadPolicy(new ThreadPolicy.Builder()  
    .detectDiskReads()  
    .detectDiskWrites()  
    .detectNetwork() // or .detectAll() for all detectable problems  
    .penaltyLog()  
    .build());  
  
StrictMode.setVmPolicy(new VmPolicy.Builder()  
    .detectLeakedSqlLiteObjects()  
    .detectLeakedClosableObjects()  
    .penaltyLog()  
    .penaltyDeath()  
    .build());
```

Richtiger Umgang mit Nebenläufigkeit

- Unterscheidung zwischen kurzzeitigen und länger laufenden Aktionen
- Für kurze nebenläufige Aufgaben: z. B. Thread, AsyncTask
- Für kurze wiederkehrende Aufgaben: Timer und TimerTask
- Ausschließlich innerhalb von Services und Activities

Demo: StopWatch

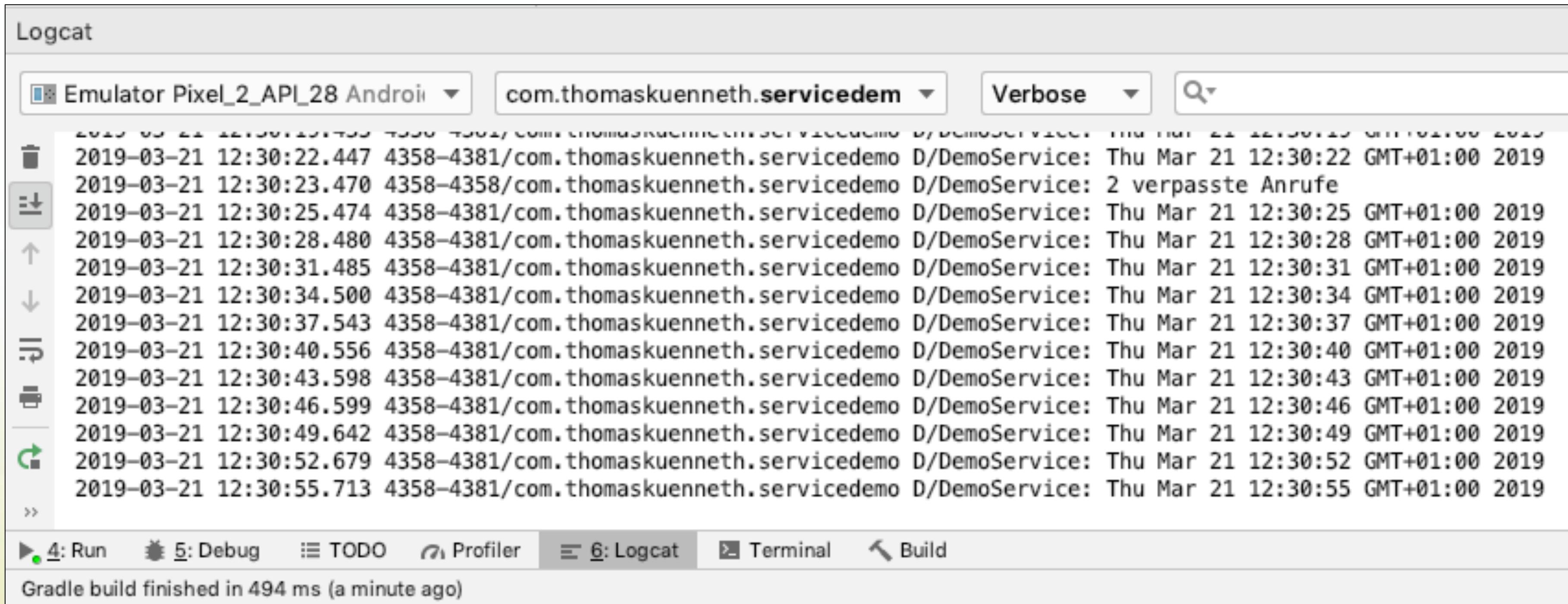
Git-Repo: https://github.com/tkenneth/android_architecture/tree/master/StopWatch



Services

- Vordergrundservices für Nutzer sichtbar (Abspielen von Musik, Videos, ...)
- Hintergrundservices für Nutzer nicht sichtbar (Datenbank bereinigen, Synchronisierung mit Backend, ...)
- Gebundene Services stellen anderen Komponenten eine Schnittstelle (API) zur Verfügung

Demo: ServiceDemo



The screenshot shows the Android Studio Logcat window. The title bar says "Logcat". The device dropdown shows "Emulator Pixel_2_API_28 Android". The package dropdown shows "com.thomaskuenneth.servicedemo". The log level dropdown is set to "Verbose". The search bar contains a magnifying glass icon. The main area displays a list of log entries from March 21, 2019, at 12:30:22 to 12:30:55. Each entry shows a timestamp, process ID, and log message. The log messages are all identical: "D/DemoService: Thu Mar 21 12:30:XX GMT+01:00 2019". Below the log list are several icons for filtering and navigating the logs. At the bottom, there are tabs for Run, Debug, TODO, Profiler, Logcat (which is selected), Terminal, and Build. A status message at the bottom says "Gradle build finished in 494 ms (a minute ago)".

```
2019-03-21 12:30:22.447 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:22 GMT+01:00 2019
2019-03-21 12:30:23.470 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: 2 verpasste Anrufe
2019-03-21 12:30:25.474 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:25 GMT+01:00 2019
2019-03-21 12:30:28.480 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:28 GMT+01:00 2019
2019-03-21 12:30:31.485 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:31 GMT+01:00 2019
2019-03-21 12:30:34.500 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:34 GMT+01:00 2019
2019-03-21 12:30:37.543 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:37 GMT+01:00 2019
2019-03-21 12:30:40.556 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:40 GMT+01:00 2019
2019-03-21 12:30:43.598 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:43 GMT+01:00 2019
2019-03-21 12:30:46.599 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:46 GMT+01:00 2019
2019-03-21 12:30:49.642 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:49 GMT+01:00 2019
2019-03-21 12:30:52.679 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:52 GMT+01:00 2019
2019-03-21 12:30:55.713 4358-4381/com.thomaskuenneth.servicedemo D/DemoService: Thu Mar 21 12:30:55 GMT+01:00 2019
```

Ruhemodus simulieren

- adb shell dumpsys deviceidle force-idle
Bringt das System in den Ruhemodus
(ggf. vorher adb shell dumpsys deviceidle enable)
- adb shell dumpsys deviceidle unforce
Erzwungenen Ruhemodus beenden
- adb shell dumpsys battery reset
Gerät reaktivieren

Job Scheduler / Job Manager

- Zur Umsetzung von periodischer Hintergrundaktivitäten
- Ausführung kann an Bedingungen geknüpft werden
- Optimal in das Energiesparkonzept von Android integriert
- Die von Google empfohlene Vorgehensweise

Jobs testen

- adb shell dumpsys jobscheduler | grep my.pkg
gibt Informationen über Jobs aus
- adb shell cmd jobscheduler run -f my.pkg id
startet einen Job

Demo: JobSchedulerDemo



Abstürze analysieren

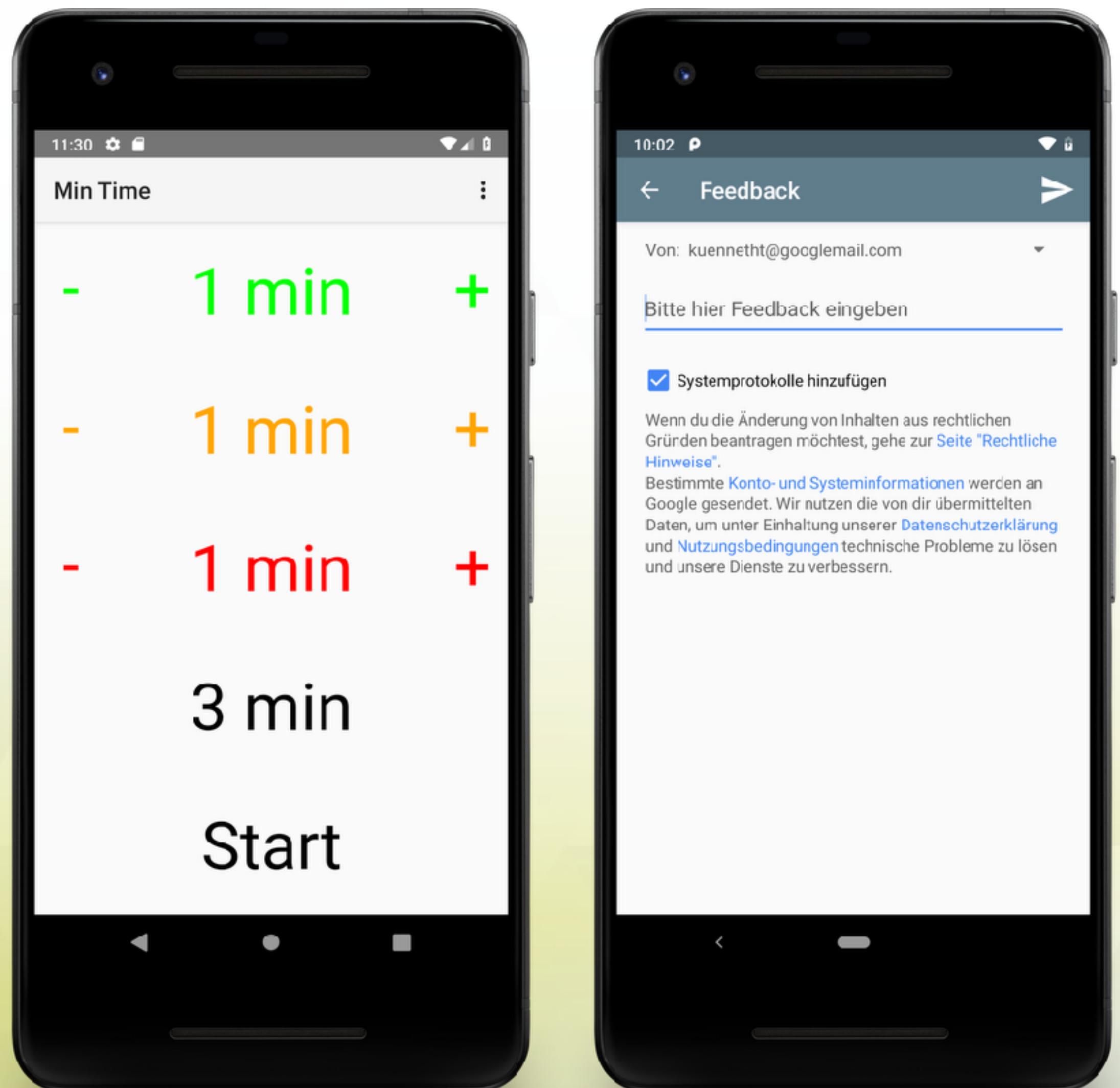
Bug Reports

- Echtes Gerät unter Entwickleroptionen
- Emulator: Extended Controls/Bug Report
- adb bugreport E:\Reports\MyBugReports

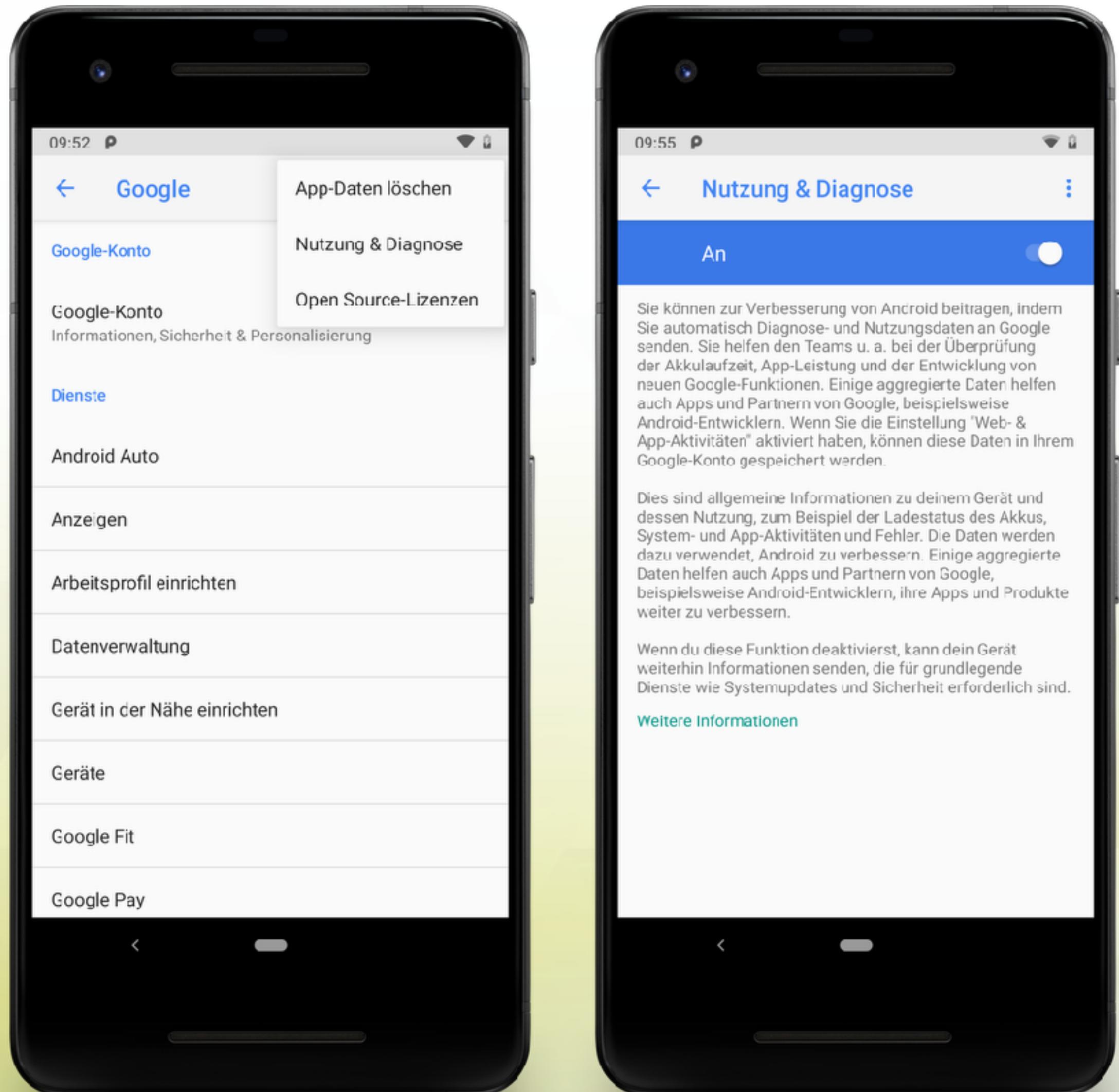


- Auf Nutzer-Geräten nicht ohne Weiteres möglich
- Android kann Crash Reports senden
- Zugriff über Play Console

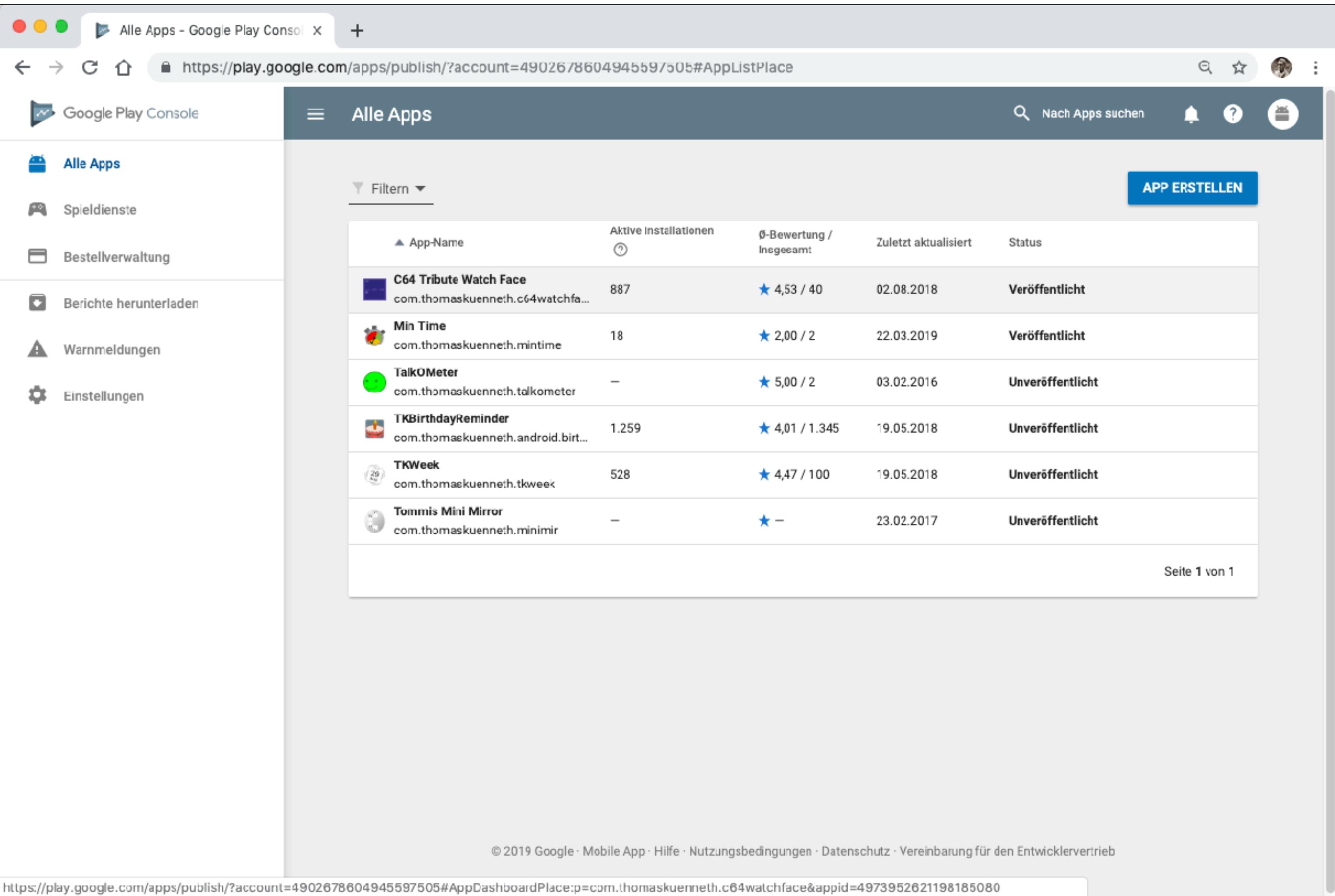
Demo: Min Time



Diagnosedaten senden



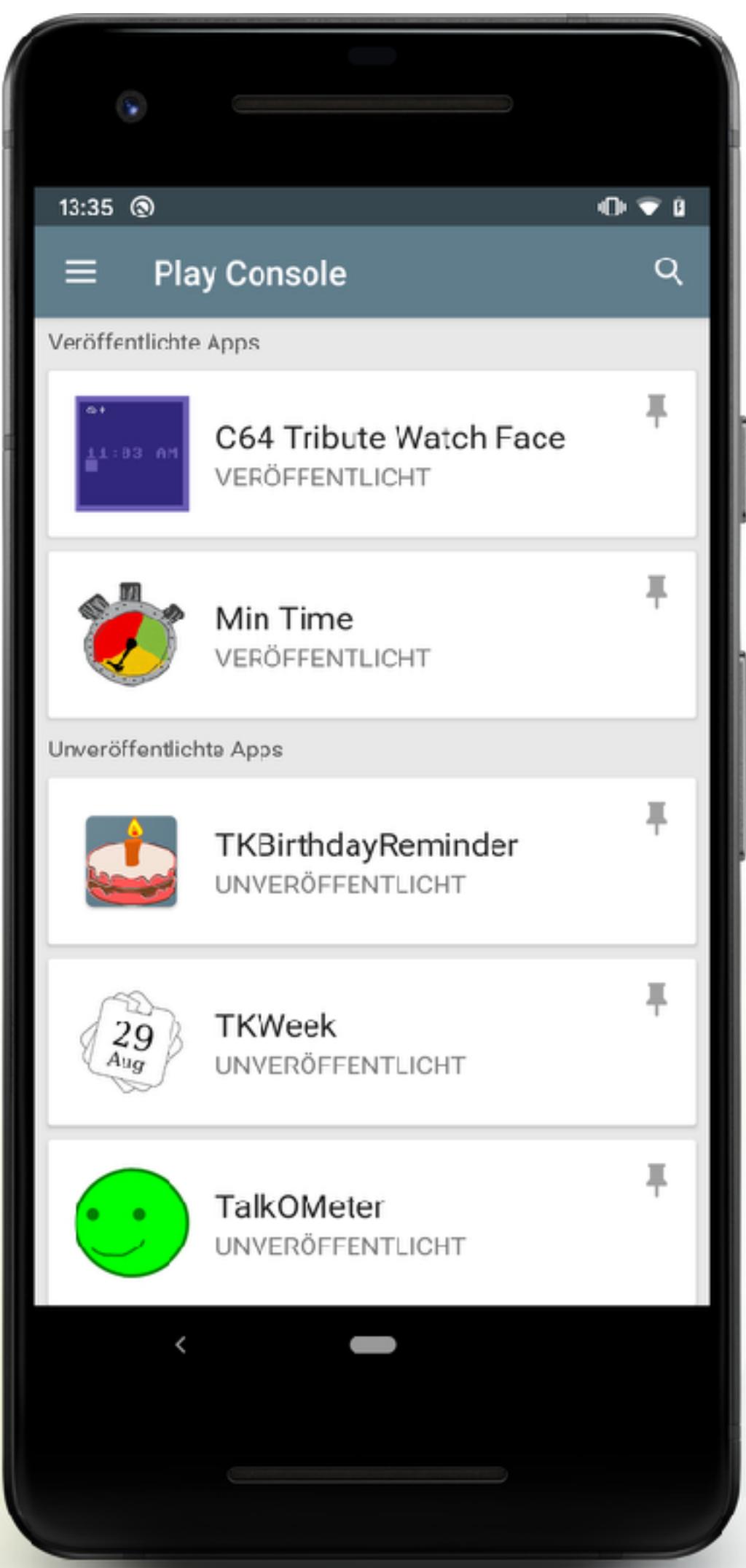
Crash Reports



The screenshot shows the Google Play Console interface. The left sidebar has a 'Alle Apps' section with links for Spieldienste, Bestellverwaltung, Berichte herunterladen, Warnmeldungen, and Einstellungen. The main area is titled 'Alle Apps' and contains a table of installed apps. The columns are: App-Name, Aktive Installationen, Ø-Bewertung / Ingeeamt, Zuletzt aktualisiert, and Status. The table lists six apps:

App-Name	Aktive Installationen	Ø-Bewertung / Ingeeamt	Zuletzt aktualisiert	Status
C64 Tribute Watch Face com.thomaskuenneth.c64watchfa...	887	★ 4,53 / 40	02.08.2018	Veröffentlicht
Min Time com.thomaskuenneth.mintime	18	★ 2,00 / 2	22.03.2019	Veröffentlicht
TalkOMeter com.thomaskuenneth.talkometer	—	★ 5,00 / 2	03.02.2016	Unveröffentlicht
TKBirthdayReminder com.thomaskuenneth.android.birt...	1.259	★ 4,01 / 1.345	19.05.2018	Unveröffentlicht
TKWeek com.thomaskuenneth.tkweek	528	★ 4,47 / 100	19.05.2018	Unveröffentlicht
Tommis Mini Mirror com.thomaskuenneth.minimir	—	★ —	23.02.2017	Unveröffentlicht

At the bottom right of the table, it says 'Seite 1 von 1'. The footer includes a copyright notice for 2019 Google and a link to the developer agreement.



MobileTech
CONFERENCE & SUMMIT

App entschleiern

The screenshot shows the Google Play Console interface for managing deobfuscation files. The left sidebar navigation includes 'Google Play Console', 'Alle Apps', 'Dashboard', 'Statistiken', 'Android vitals' (selected), 'Übersicht', 'ANRs & Abstürze', 'Deobfuscation-Dateien' (selected), 'Entwickler-Tools', and 'Release-Verwaltung'. The main content area is titled 'Deobfuscation-Dateien' and displays a message about ProGuard tooling. A table lists nine APK versions, each with a download icon, version number, and a 'HOCHLADEN' button. The table has columns for 'Hochgeladen', 'Version', 'Mapping hochgeladen um', and 'HOCHLADEN'. At the bottom right of the table, it says 'Seite 1 von 1'.

Hochgeladen	Version	Mapping hochgeladen um	
0	9.0	—	HOCHLADEN
0	8.0	—	HOCHLADEN
0	7.0	—	HOCHLADEN
0	6 (1.05)	—	HOCHLADEN
0	5 (1.04)	—	HOCHLADEN
0	4 (1.03)	—	HOCHLADEN
0	3 (1.02)	—	HOCHLADEN
0	2 (1.01)	—	HOCHLADEN
0	1 (1.0)	—	HOCHLADEN

© 2019 Google · Mobile App · Hilfe · Nutzungsbedingungen · Datenschutz · Vereinbarung für den Entwicklervertrieb

Firebase Crashlytics

- Alternative zum Crash Reporting in der Play Console
- Bietet sich an, wenn App ohnehin Firebase verwendet
- Vorteil: Ereignisse schneller sichtbar

Demo: Zufallszahl



Firebase-Test – Crashlytics

https://console.firebaseio.google.com/project/fir-test-edc3f/crashlytics/app/android:com.thomaskuenneth.firebaseio... Schließen Dokumentation ansehen

MainActivity.java line 83

com.thomaskuenneth.firebaseio.MainActivity.crash

Dieses Problem umfasst 3 Abstürze und betrifft 1 Nutzer.

Ereignisse insgesamt nach Version für die vergangenen 24 Stunden

1.0 (1) 3

Geräte Betriebssysteme Geräteteststatus
100% Google 100% Android 9 0 % Hintergrund
100% Google →

Sitzungen Alle Versio... < >

Sitzungsübersicht 1.0 (1) 9 Android SDK built for x86_64 23.03.2019, 15:07:00

Stacktrace Schlüssel Logs Daten

TXT

Fatal Exception: java.lang.NullPointerException
Attempt to invoke virtual method 'java.lang.String java.lang.String.toString()' on a null object reference

com.thomaskuenneth.firebaseio.MainActivity.crash (MainActivity.java:83)
com.thomaskuenneth.firebaseio.MainActivity.access\$100 (MainActivity.java:21)
com.thomaskuenneth.firebaseio.MainActivity\$2.onClick (MainActivity.java:46)
android.view.View.performClick (View.java:607)

Storage Hosting Functions ML Kit

Qualität Crashlytics Performance Test Lab

Analytics Dashboard Events Conversions Audiences Funnels User Properties Latest Release Retention StreamView DebugView

Spark Kostenlos Upgrade ausführen

Zusammenfassung

- Durch Profiling lassen sich unter anderem Speicherlecks und rechenzeitintensive Codeteile identifizieren
- Crash Reports versorgen Sie mit Infos zu Problemen auf fremden Geräten
- Bei Apps mit Cloud-Anbindung können Dienste außerhalb der Play Console sinnvoll sein

- Übergreifende Themen
- Effizient, effektiv und automatisiert testen
- Debuggen, profilieren und Abstürze analysieren
- Zielgerade

- Riesige Geräteauswahl großes Plus für Android
- Aber: Geräteauswahl heißt nicht nur Smartphone!
- Smartphone, Tablet, Auto, TV und Wearables erfordern individuelle Umsetzungen
- Entwickler müssen sich auf Vielfalt einstellen

- Hardware (Sensoren, Anzeigen, Speicher, ...)
- Software (Android-Version, herstellerspezifische Anpassungen)
- Sprache, Einheiten, individuelle Einstellungen, Barrierefreiheit, ...

- Kleiner Gerätepark (Hardware und virtuell) unabdingbar
- Mit zusätzlichen Emulatoren arbeiten, z. B.
 - Genymotion
 - Bluestacks
- Play Store im Emulator für geschlossene und interne Tests nutzen

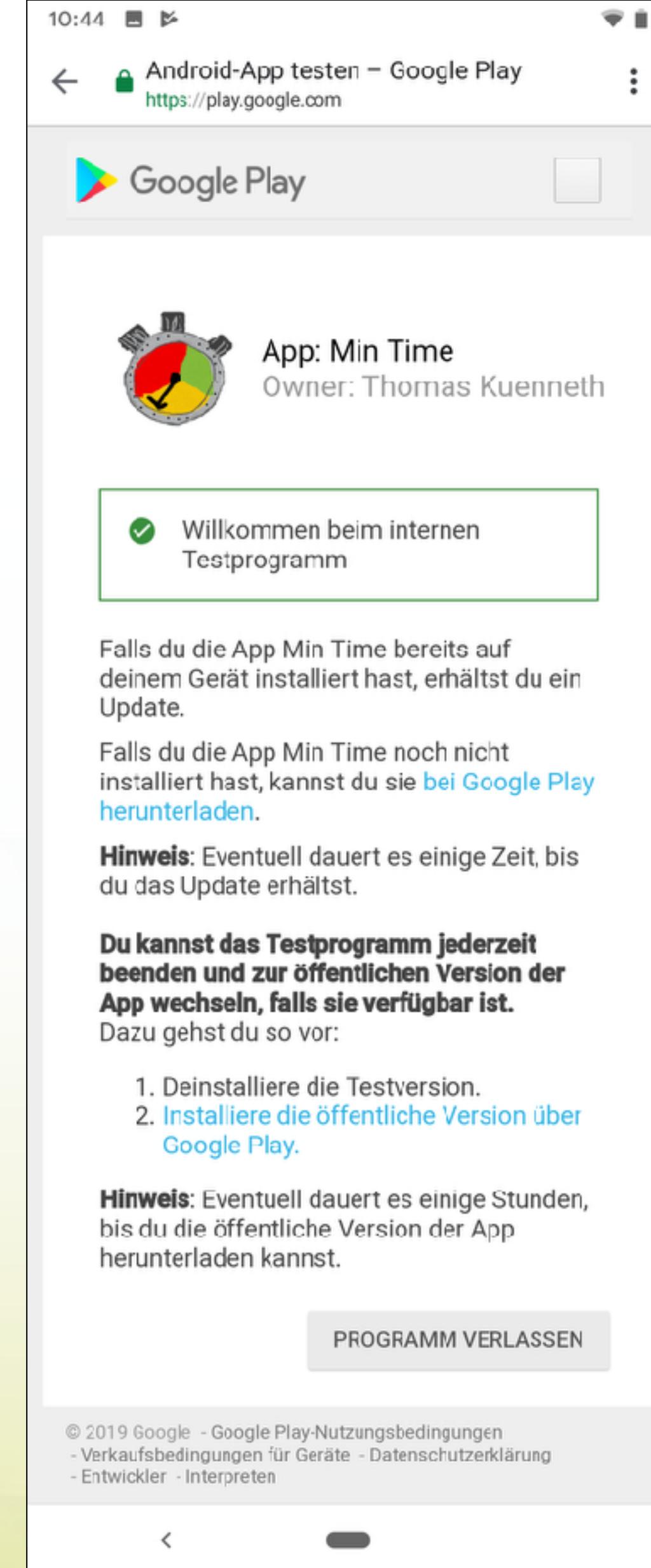
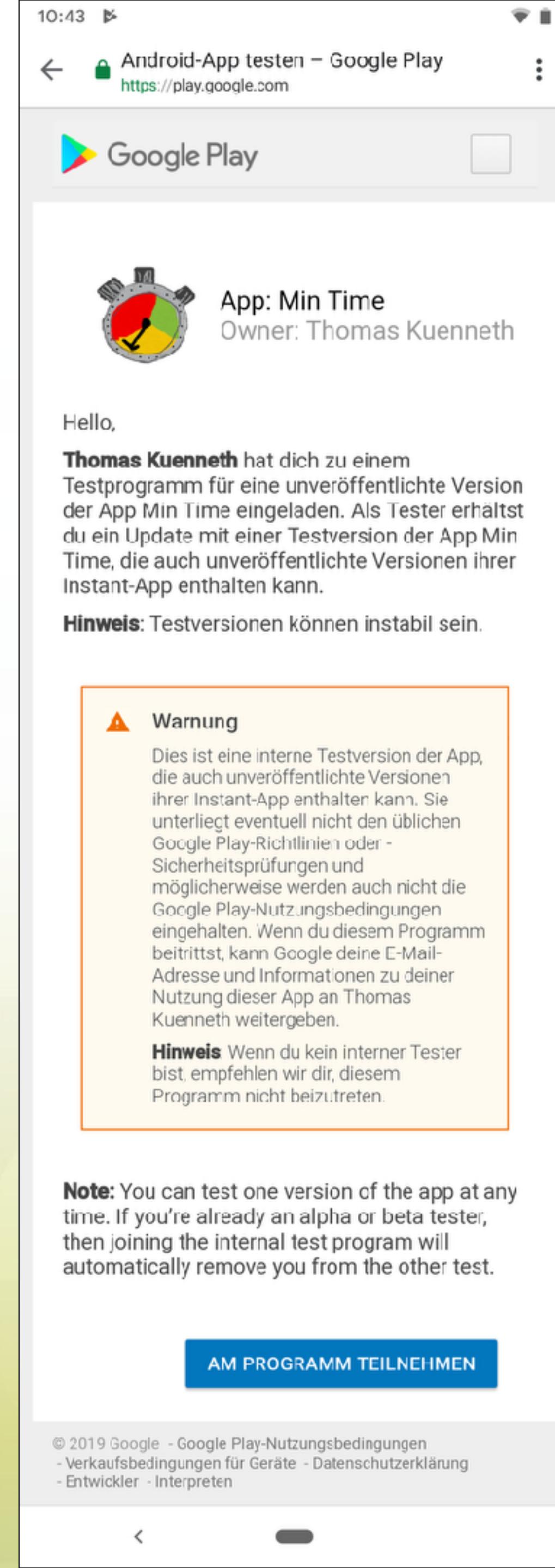
Release-Verwaltung im Play Store

- Interner Test-Track
Schnelle interne Tests und Qualitätssicherungsprüfungen
- Geschlossener Track (Alpha Test)
Vorabveröffentlichungen der App mit einer größeren Testgruppe testen
- Offener Track (Beta Test)
Test mit einer großen Gruppe
- Produktions-Track

Interner Test

- Pro App 100 interne Tester
- Unterliegen laut Google evtl. nicht den üblichen Play-Richtlinien oder - Sicherheitsüberprüfungen

The screenshot shows the Google Play Console interface for managing app releases. The left sidebar lists various sections like 'Alle Apps', 'Dashboard', 'Statistiken', 'Android vitals', 'Entwickler-Tools', and 'Release-Verwaltung'. Under 'Release-Verwaltung', 'App-Releases' is selected. The main content area is titled 'App-Releases' and shows a status bar indicating 'Min Time' is 'Veröffentlicht'. Below this, a section titled 'Interner Test' is shown, with a sub-section 'Tester verwalten'. A callout box highlights the limit of '100 interne Tester pro App'. The 'Nutzer' section lists one tester named 'Nur ich' with '1 Tester'. At the bottom right, there are buttons for 'TESTER ENTFERNEN' and 'GESPEICHERT'.



Multi-Device-Tests

- Anbieter mit Testgerät-Farmen
- Hochgeladene APKs werden auf einer Vielzahl an Geräten getestet
- Oft sind Tests konfigurierbar
- Beispiele: Play Store Prelaunch-Bericht, Firebase Test Lab

Pre-Launch-Bericht - Min Time x Firebase-Test – Übersicht – Ein x +

https://play.google.com/apps/publish/?account=4902678604945597505#PreLaunchReportPlace:p=com.thomaskuenneth.mintime&appId=4975439517771967851&plrvc=10&...

Google Play Console Pre-Launch-Bericht Min Time Veröffentlicht

Alle Apps Dashboard Statistiken Android vitals Entwickler-Tools Release-Verwaltung Release-Dashboard App-Releases Android Instant-Apps Artefaktebibliothek Gerätetkatalog App-Signatur Pre-Launch-Bericht App-Präsenz im Play Store Nutzergewinnung Finanzberichte Nutzerfeedback

Pre-Launch-Bericht für APK 10 APK auswählen: 10 - 23.03.2019

ÜBERSICHT ABSTÜRZE LEISTUNG BEDIENUNGSHILFEN SCREENSHOTS SICHERHEIT EINSTELLUNGEN

In Pre-Launch-Berichten werden Probleme zusammengefasst, die beim Testen der App bei einer Vielzahl von Geräten aufgetreten sind. Google kann jedoch nicht garantieren, dass bei den Tests alle Probleme gefunden werden. [Weitere Informationen](#)

Zusammenfassung

Empfehlung: Fehler beheben und Warnungen prüfen

Kategorie	Fehler ⓘ	Warnungen ⓘ	Kleinere Probleme ⓘ
Alle Probleme	! 9	⚠ 5	! 1
Abstürze	9	0	0
Leistung	0	0	0
Sicherheit und Datenschutz	0	0	0
Barrierefreiheit	0	5	1

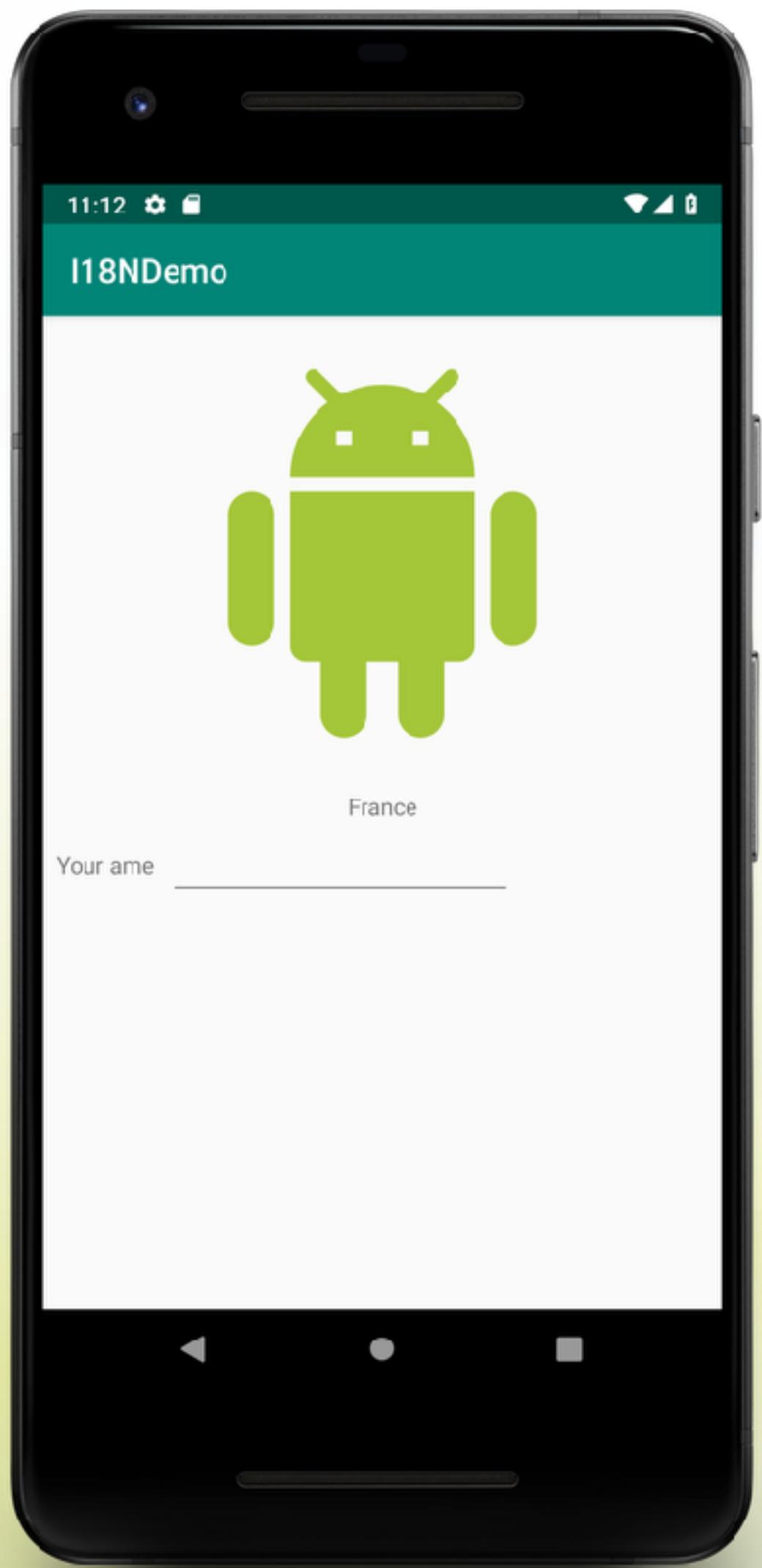
Testergebnisse

Dieses APK wurde auf 10 verschiedenen Geräten getestet. Testgeräte werden auf Grundlage zahlreicher Kriterien wie Beliebtheit, Absturzhäufigkeit oder Hersteller ausgewählt. [Weitere Informationen](#)

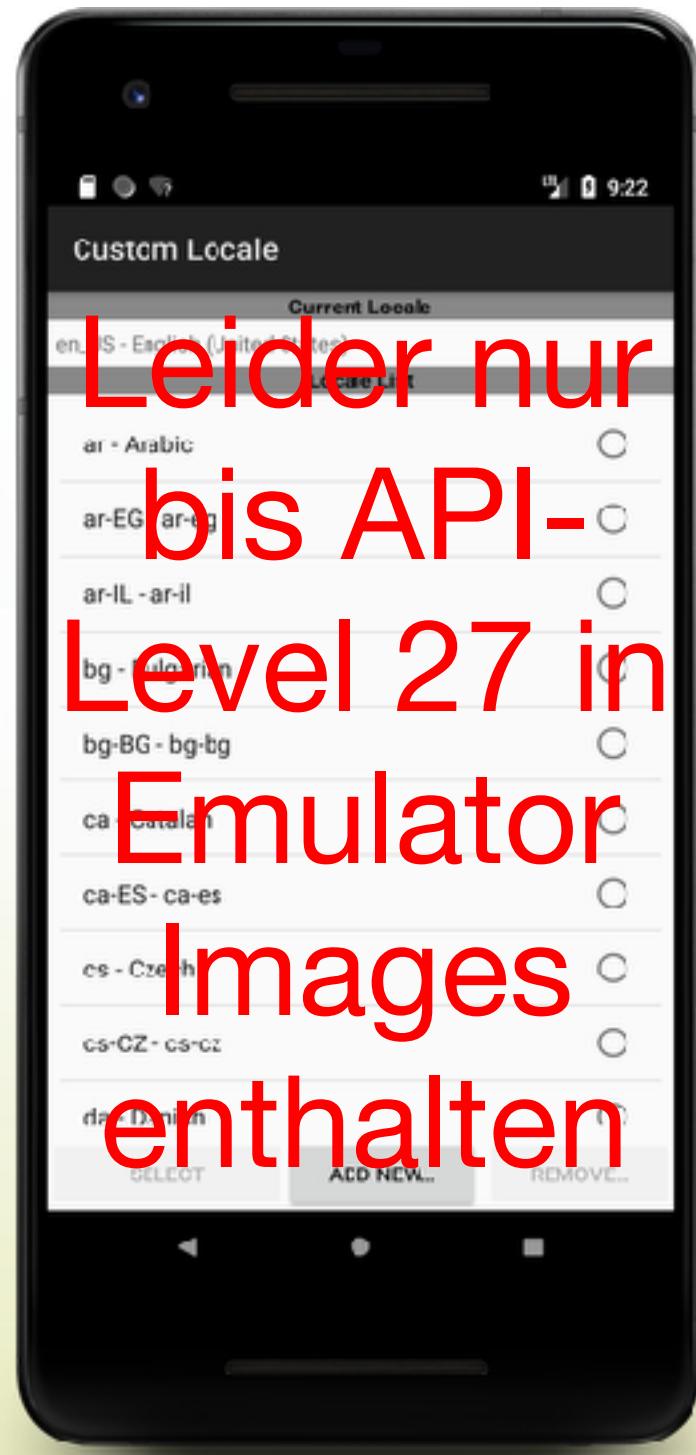
Abstürze
! 9 Fehler erkannt
Durch die Behebung der Probleme kannst du die Nutzererfahrung verbessern, höhere Bewertungen erzielen.

- Für möglichst große Nutzergruppe Texte internationalisieren
- Achtung: Lange Texte können Auswirkungen auf Layout haben (testen)
- Tipp: Auch Grafiken können internationalisiert werden

Demo: I18NDemo



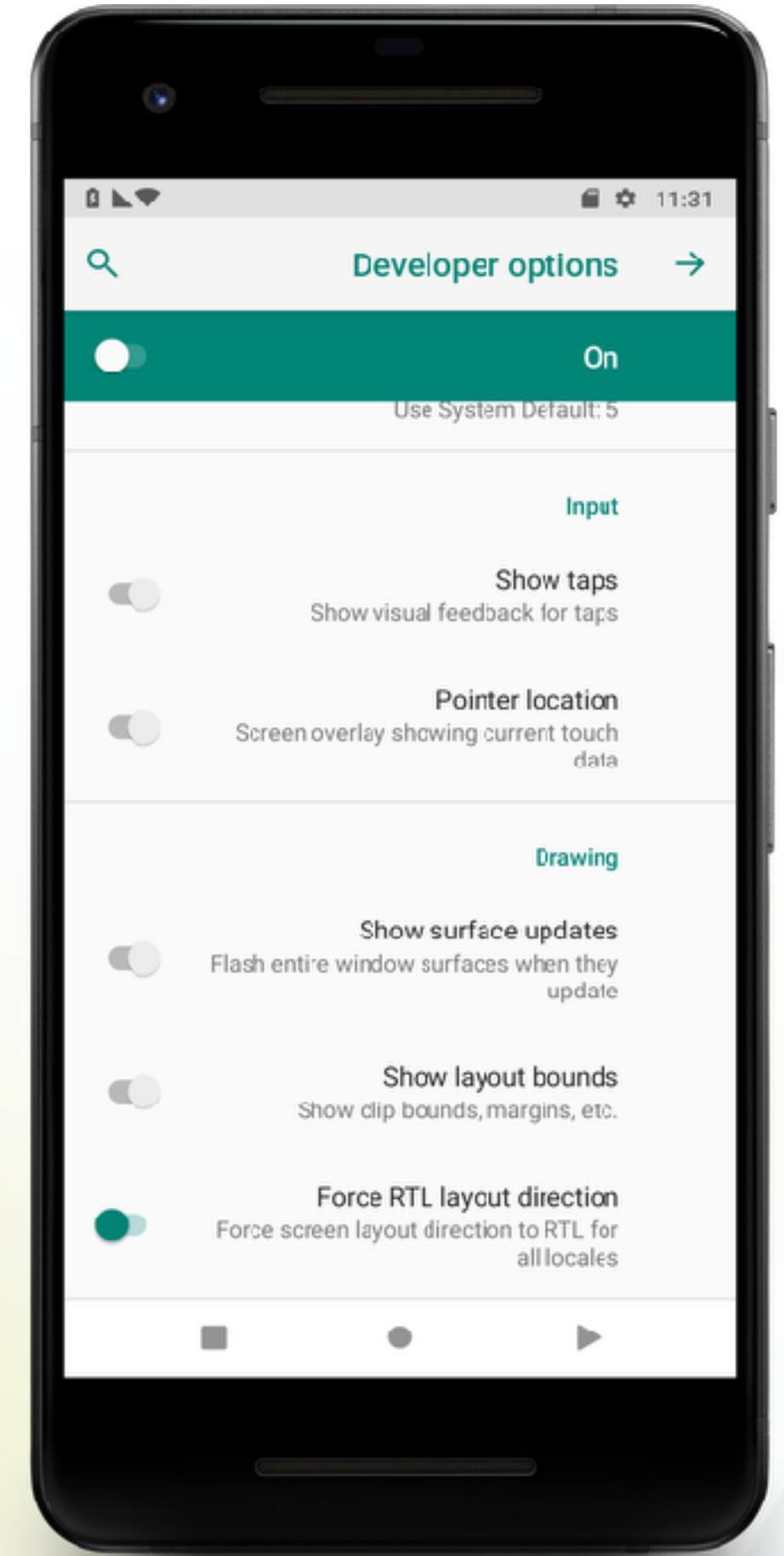
App-Verhalten testen



- Locale setzen
 - adb root
 - adb shell 'setprop persist.sys.locale fr-FR; stop;
sleep 5; start'
 - (en-GB, de-DE, ...)
- Ggf. für Emulator *Wipe Data*, falls Änderung keine Wirkung zeigt

RTL-Sprachen

- Leserichtung sollte sich auf Nutzergewohnheiten einstellen (RTL-Sprachen)
- Prüfen, ob Grafiken in RTL-Sprachen angepasst werden müssen
- Im Emulator über Entwicklereinstellungen Layoutorientierung von rechts nach links erzwingen



Endspurt

Privatsphäre achten

- Sorgsamer Umgang mit Berechtigungen
- Verantwortungsvoll und transparent mit persönlichen Daten umgehen
 - Nur was nötig ist
 - Transparenz für den Nutzer schaffen

Werbung

- In App-Werbung kann zusätzliche Einnahmequelle sein
- Setzen Sie sie sorgsam und sparsam ein
- Meiden Sie fragwürdige Angebote

Hören Sie zu!

- Regelmäßig Bewertungen lesen
- Nicht von unproduktiven Kommentaren verunsichern oder provozieren lassen
- Auf Benutzer eingehen
- Social Media nutzen

Apps durch Tests absichern

- Unit-Tests für die Fachlichkeit
- Instrumentierte Tests für Oberfläche und Android-spezifisches
- Nutzen Sie Alpha- und Beta-Tests

Nicht nur die App braucht Pflege

- Testen und Profilen spielt beim Backend eine genauso wichtige Rolle
- Begleitende Websites, FAQs und How Tos müssen auf dem aktuellen Stand gehalten werden

Updates

- Aktualisieren Sie die App regelmäßig, aber sinnvoll
- Halten Sie die Codebasis auf einem aktuellen Stand
- Halten Sie sich selbst auf dem Laufenden



Vielen Dank

www.mathema.de

thomas.kuenneth@mathema.de

 @tkuenneth