



# Ausarbeitung

---

ENTWICKLUNG MOBILER ANWENDUNGEN – WS2017/18



Ali Rahimpour  
Inf2310@hs-worms.de | MAT-NR.: 669867

# INHALTSVERZEICHNIS

<b>Inhaltsverzeichnis</b>	<b>2</b>
<b>1 Einleitung</b>	<b>4</b>
<b>2 Installation</b>	<b>4</b>
<b>2.1 Android Studio</b>	<b>4</b>
<b>2.1.1 Installation</b>	<b>6</b>
<b>3 Android Studio-Projekt erstellen</b>	<b>6</b>
<b>3.1 Schritt 1</b>	<b>7</b>
<b>3.2 Schritt 2</b>	<b>7</b>
<b>3.3 Schritt 3</b>	<b>8</b>
<b>3.4 Schritt 4</b>	<b>8</b>
<b>3.5 Schritt 5</b>	<b>9</b>
<b>4 Problem beim Installation</b>	<b>11</b>
<b>4.1 Lösungsvorschlag</b>	<b>11</b>
<b>5 Android Virtual Device (AVD)</b>	<b>13</b>
<b>5.1 Emulator starten</b>	<b>16</b>
<b>6 App auf dem Handy installieren</b>	<b>18</b>
<b>6.1 USB-Debugging aktivieren</b>	<b>18</b>
<b>6.2 Android Gerät mit dem PC verbinden</b>	<b>20</b>
<b>6.3 Die App auf dem Gerät installieren und starten</b>	<b>20</b>
<b>7 1. Teil</b>	<b>22</b>
<b>7.1 Aufgabe 1 - Aufgabestellung</b>	<b>22</b>
<b>7.1.1 MyNameApp Layout</b>	<b>22</b>
<b>7.1.2 MainActivity</b>	<b>24</b>
<b>7.1.3 Codeausschnitte</b>	<b>25</b>
<b>7.1.4 Text Farbe</b>	<b>25</b>
<b>7.1.5 Schriftgröße ändern</b>	<b>26</b>
<b>7.1.6 Android Asset Studio</b>	<b>27</b>
<b>7.1.7 Die App Übersicht &amp; Video</b>	<b>28</b>
<b>8 3. Teil: Projekt</b>	<b>29</b>
<b>8.1 Idee: Fitness_app</b>	<b>29</b>

<b>8.2 Sensoren</b>	<b>29</b>
8.2.1 Einleitung	29
8.2.2 Sensoren im Android-SDK	30
8.2.3 Sensoren Testen	31
<b>8.3 Ziel &amp; Anforderungen an das Projekt</b>	<b>33</b>
<b>8.4 Das Design vorbereiten</b>	<b>33</b>
8.4.1 Planung - Mockup	33
8.4.2 Benutzeroberfläche und XML Datei	35
8.4.3 String.xml	37
<b>8.5 Implementation</b>	<b>38</b>
<b>8.6 Den Sensor ansprechen</b>	<b>38</b>
8.6.1 SensorManager & STEP_COUNTER Sensor	39
8.6.2 onResume() & onPause()	40
<b>8.7 Schritte über den Sensor erfassen</b>	<b>42</b>
<b>8.8 Start, Stop und Reset einbauen</b>	<b>43</b>
<b>8.9 Die SettingsActivity</b>	<b>45</b>
8.9.1 Menü	47
8.9.2 Einstellungsoptionen ausdünnen	48
<b>8.10 Die Kalorien &amp; Meter berechnen</b>	<b>49</b>
<b>8.11 Problem beim Screen Rotation</b>	<b>50</b>
<b>8.12 Stoppuhr</b>	<b>51</b>
<b>8.13 Icon</b>	<b>52</b>
<b>8.14 Andere Aufgetretene Probleme</b>	<b>53</b>
8.14.1 Problem 1	53
8.14.2 Problem 2	54
8.14.3 Problem 3	54
8.14.4 Problem 4	55
<b>8.15 Die App Übersicht &amp; Video</b>	<b>57</b>
<b>9 Fazit</b>	<b>59</b>
<b>10 Projekt Übersicht</b>	<b>60</b>
<b>11 Quellenverzeichnis</b>	<b>61</b>

## 1 EINLEITUNG

Eine App ist die Abkürzung für eine Applikation oder Anwendung die heute immer mehr an Bedeutung gewinnt. In unsere Zeit der Smartphone Überwelle ist es kaum vorstellbar ohne Apps auszukommen, da diese viele Funktionen auf dem eigenen Smartphone vereinfachen, beschleunigen und oft überhaupt erst möglich machen. Daher sind Apps heutzutage kaum wegdenkbar.

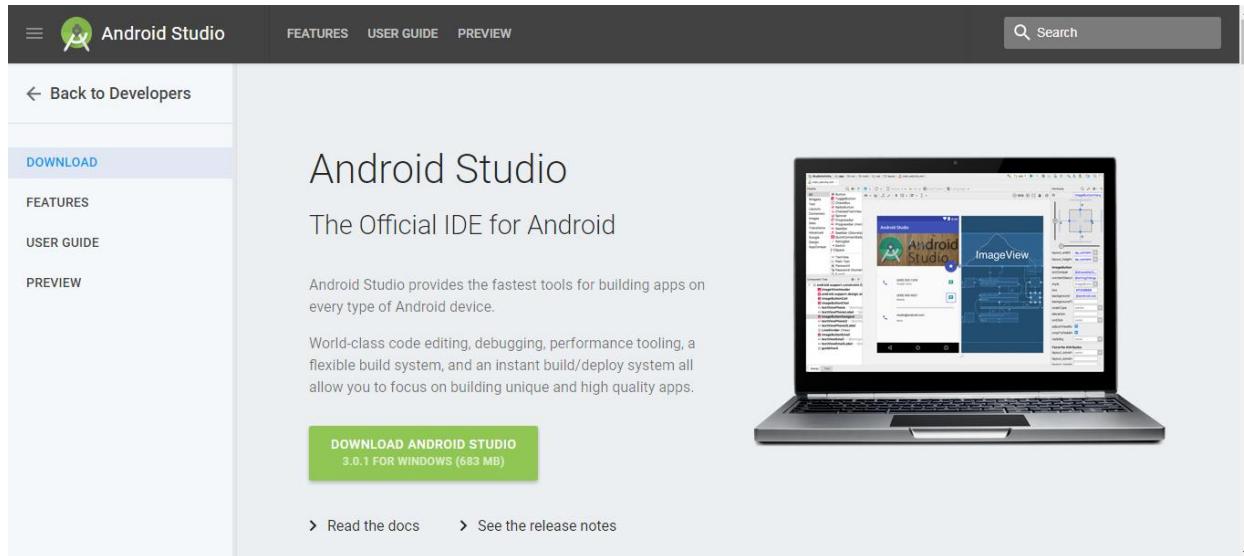
In dieser Ausarbeitung werde ich die Installation von Android Studio 3.0 ausführlich beschreiben um später eine eigene App programmieren.

## 2 INSTALLATION

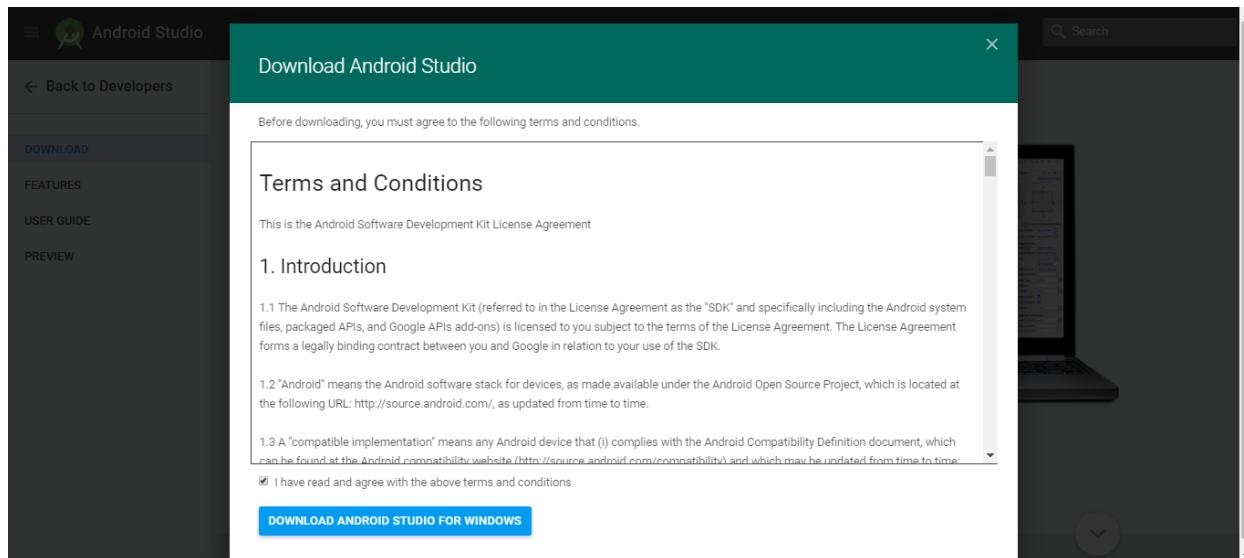
### 2.1 ANDROID STUDIO

Nun gelangt man zu der Einleitung von der Android Installation. Von hier aus kann das Android Studio mit der gewünschten Kompatibilität des eigenen Betriebssystems heruntergeladen werden.

Link zum Download: <https://developer.android.com/sdk/index.html>



Mit einem Klick auf den großen grünen Button kann der Benutzer den Download starten. Bevor der Download beginnt, muss aber noch den „*Terms and Conditions*“ von Android Studio, also den Nutzungsbedingungen, zugestimmt werden.



Nachdem der Download abgeschlossen ist, kann mit der Installation von Android Studio begonnen werden.

### 2.1.1 Installation

Für die Installation von Android Studio gibt es zahlreiche umfassende Anleitungen im Internet. Ich habe bei der Installation eine Anleitung aus dem Netz genommen.

<http://www.programmierenlernenhq.de/tutorial-android-studio-2-installieren/>

In dieser Anleitung wird die Version 2.1 von Android Studio installiert.

Die Installation von Android Studio wird schrittweise beschrieben. Zunächst wird gezeigt, wie man prüft, ob der Java SDK richtig eingerichtet ist und funktioniert (u.a. ob der Java-Path korrekt gesetzt ist). Dies ist notwendig, da das JDK 8 für die Installation und Ausführung von Android Studio benötigt wird.

Anschließend wird die Installation von Android Studio schrittweise beschrieben und wird gezeigt, welche Einstellungen vorgenommen werden soll, die für das Entwickeln einer App notwendig sind.

Als letzten Schritt wird die Installation von Intel® Hardware Accelerated Execution Manager (Intel HAXM) erklärt. Der HAXM ist eine Hardware-unterstützte Virtuelle Umgebung, die den Entwicklungsprozess von Android Anwendungen beschleunigt.

## 3 ANDROID STUDIO-PROJEKT ERSTELLEN

Der Erstellungsprozess besteht aus den folgenden fünf einfachen Arbeitsschritten und dauert einen kurzen Moment.

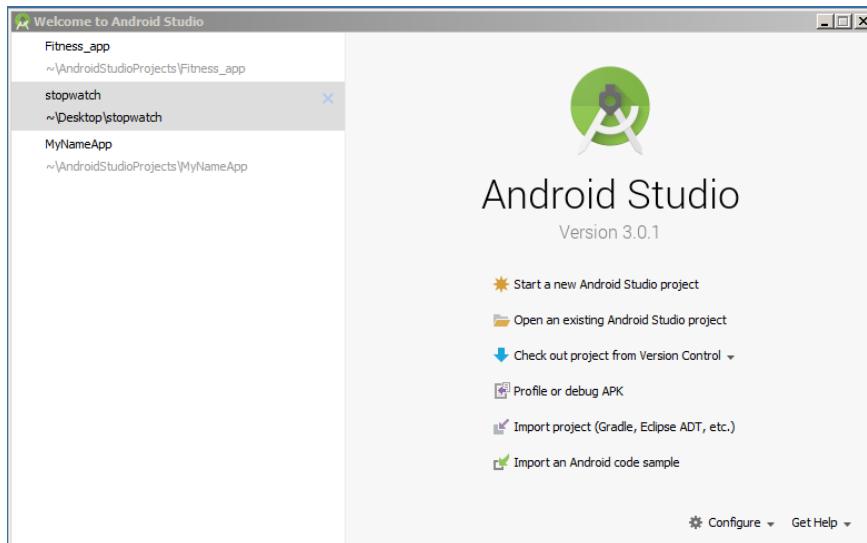
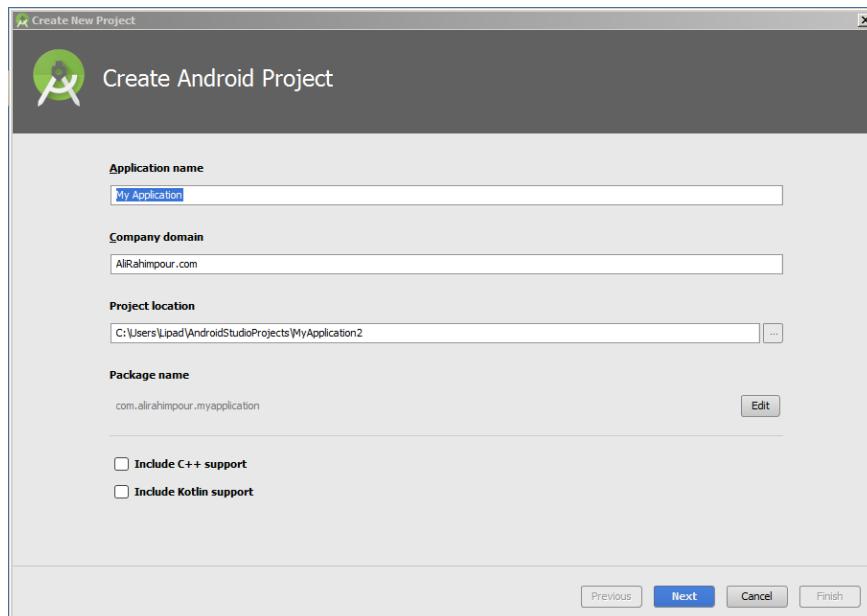


### 3.1 SCHRITT 1

Mit einem Klick auf „Start a New Android Studio Project“ starten wir das neue Projekt „New Project“, dass uns durch die Erstellung eines neuen Android Projekts führt.

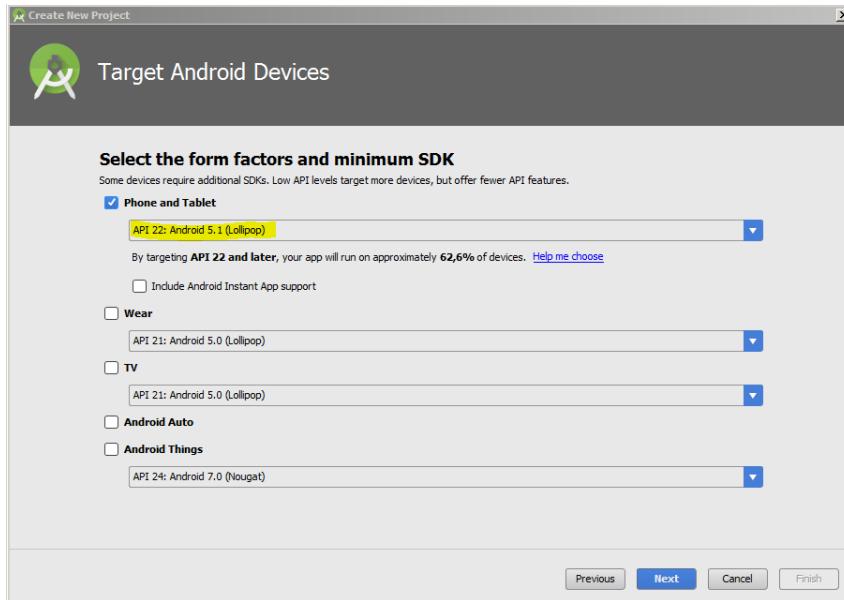
### 3.2 SCHRITT 2

Der New Project-Dialog erscheint, in dem wir Einstellungen für unser Projekt vornehmen können. Zuerst der „Application Name“ bleibt so wie es ist (oder kann durch einen anderen Namen getauscht werden) „My Application“ und als Company Domain schreibe ich „AliRahimpour.com“



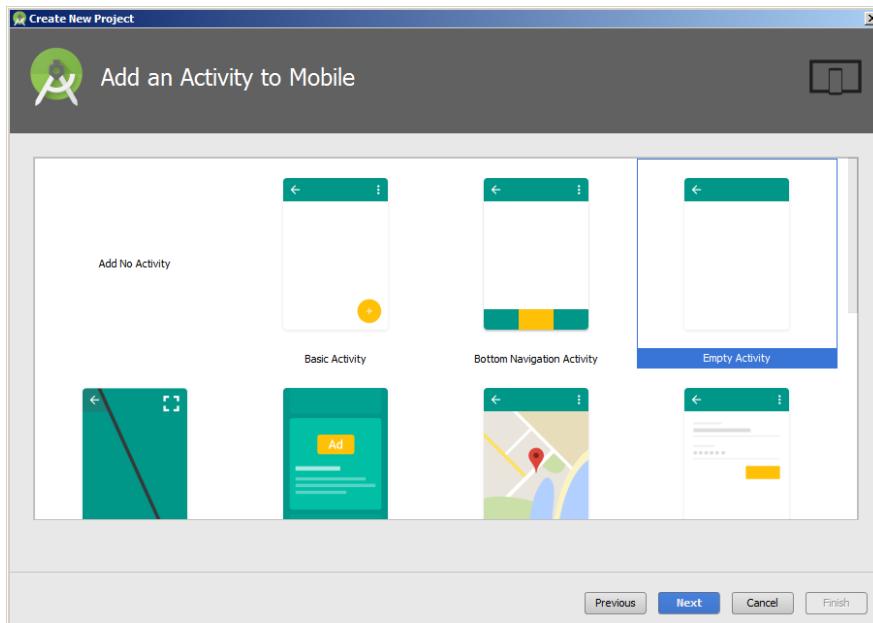
### 3.3 SCHRITT 3

Als nächsten Schritt geben wir den Minimum SDK vor. Der Minimum SDK legt fest, welche Android-Version mindestens auf dem mobilen Gerät installiert sein muss, damit unsere App darauf läuft.



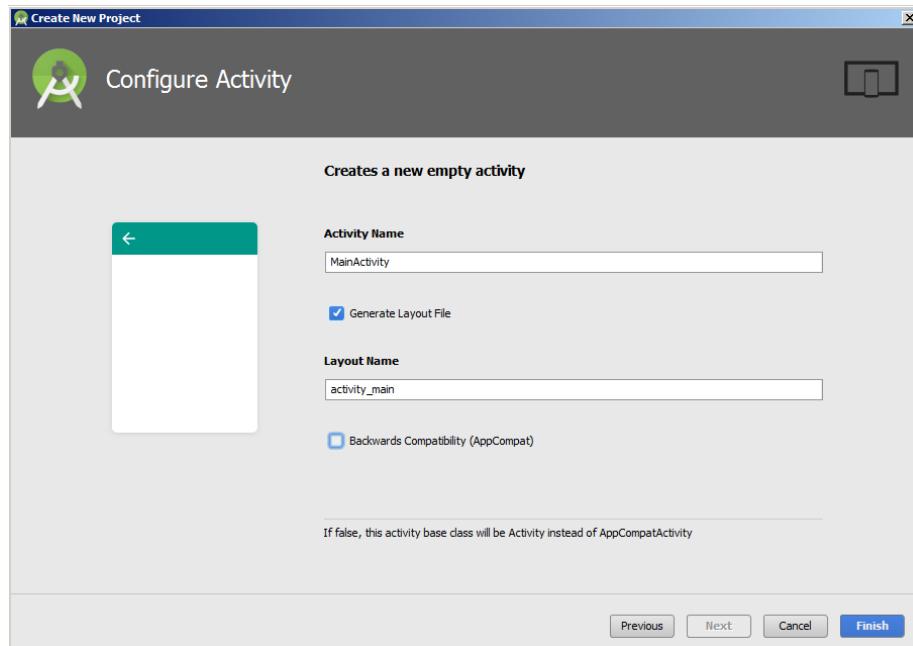
### 3.4 SCHRITT 4

Als nächsten Schritt legt man den grundsätzlichen Aufbau der App fest. Dazu wählt man die Option „Empty Activity“ aus, wodurch ein Android Projekt angelegt wird.

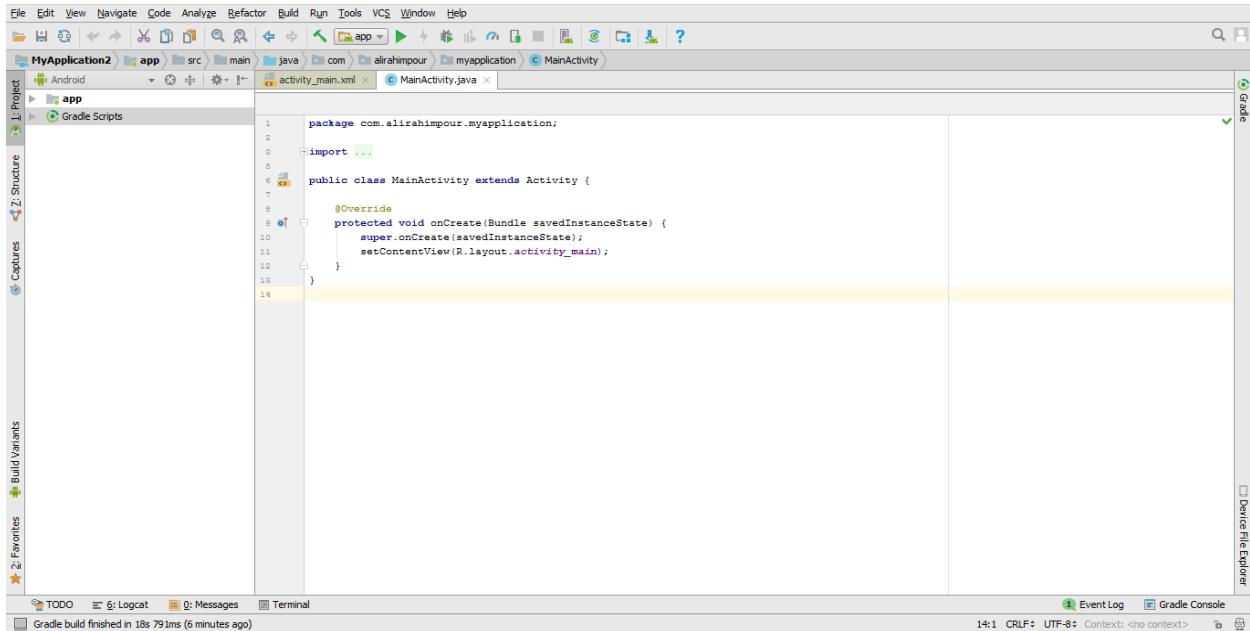


### 3.5 SCHRITT 5

Als letzten Schritt wählt man einen Namen Für Haupt-Activity und legt man fest wie das Layout dieser Activity heißen soll.



Unser Android Studio Projekt sollte nun von der Entwicklungsumgebung nach unseren Vorgaben generiert werden. Dieser Vorgang nimmt einige Zeit in Anspruch und sollte auf keinen Fall unterbrochen werden. Manchmal werden Zwischenmeldungen eingeblendet und informieren über den aktuellen Fortschritt. Nachdem das Projekt generiert wurde, sollte sich Android Studio komplett geladen haben. Man sieht als erstes die Klassendatei „`MainActivity.java`“ und auch kann man zum „`activity_main.xml`“ wechseln und das Layout zu gucken

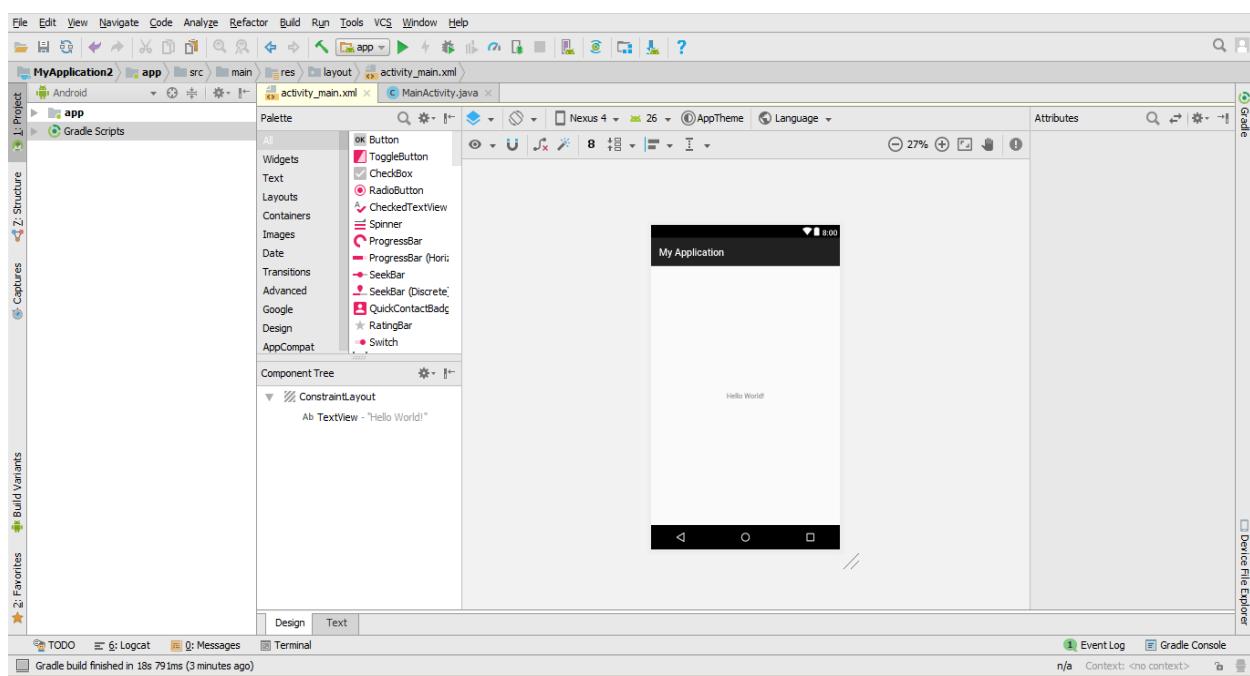


The screenshot shows the Android Studio interface with the Java code for `MainActivity.java` in the center. The code defines a `MainActivity` class that overrides the `onCreate` method to set the content view to `R.layout.activity_main`. The code editor has syntax highlighting and a yellow selection bar.

```

1 package com.alirahimpour.myapplication;
2
3 import ...
4
5 public class MainActivity extends Activity {
6     ...
7     @Override
8     protected void onCreate(Bundle savedInstanceState) {
9         super.onCreate(savedInstanceState);
10        setContentView(R.layout.activity_main);
11    }
12 }
13
14

```



## 4 PROBLEM BEIM INSTALLATION

Nach der installation, kam es zu einer Fehlermeldung: „*HAXM is not working and emulator runs in emulation mode*“

```

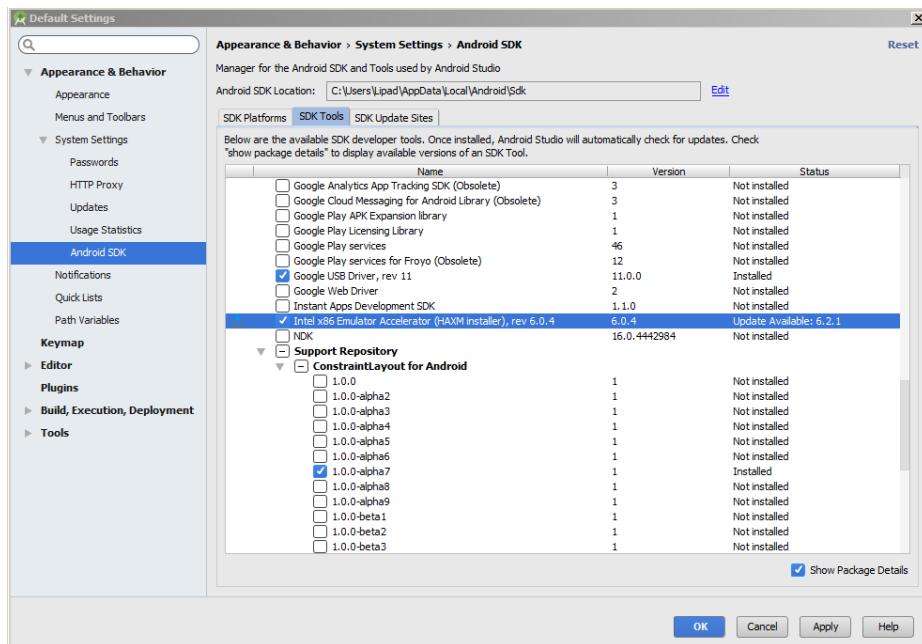
Label:
Blocks: 16896
Block groups: 1
Reserved block group size: 7
Created filesystem with 11/4224 inodes and 1302/16896 blocks
emulator: device fd:800
HAXM is not working and emulator runs in emulation mode
emulator: The memory needed by this AVD exceeds the max specified in your HAXM configuration.
emulator: AVD      RAM size = 1536 MB
emulator: HAXM max RAM size = 512 MB
emulator: You might want to adjust your AVD RAM size and/or HAXM configuration to run in fast virt mode.
creating window 59 80 449 797
emulator: emulator window was out of view and was recentered
emulator: UpdateCheck: current version '24.4.0', last version '24.4.0'

```

### 4.1 LÖSUNGSVORSCHLAG

Um diese Fehler zu beheben muss man folgende Schritte durchführen:

1. auf *SDK Manager*
2. *SDK Tools*
3. *Intel x86Emulator Accelerator(HAXM installer)*
4. wählen(hacken)
5. *Apply (um zu installieren)*



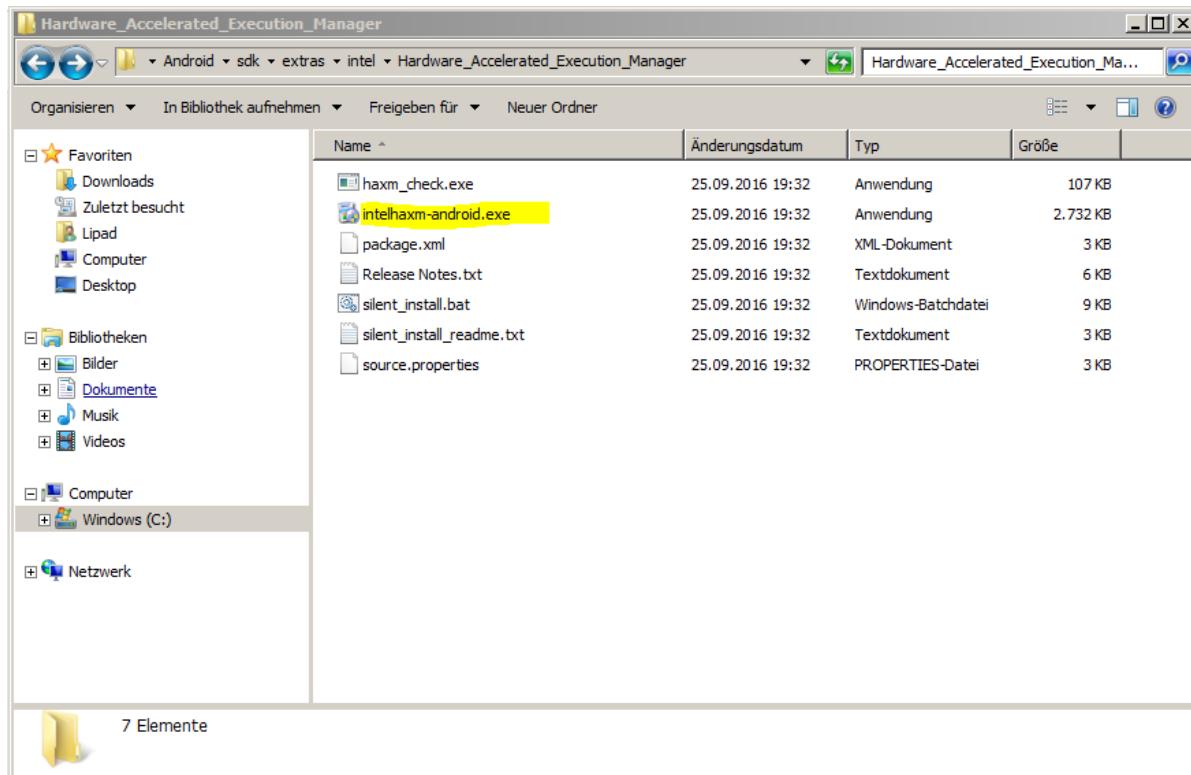
Ich habe immer wieder dieselbe Fehlermeldung erhalten! Daher habe ich die Fehlermeldung gegoogelt. Nach einer Weile Suche im Internet bin ich auf den folgenden Link gestoßen:

<https://www.youtube.com/watch?v=ZLcozYGkxI0>

Jetzt sollte man den HAXM-Installer ausführen. Das Verzeichnis befindet sich im Android SDK-Verzeichnis unter folgendem Pfad:

C:\Users\<User\_Name>\AppData\Local\Android\sdk\extras\intel\Hardware\_Accelerated\_Execution\_Manager\.

In dem HAXM-Verzeichnis befindet sich die Installationsdatei „intelhaxm-android.exe.“ Auf diese klickt man nun doppelt und startet die Installation von HAXM.



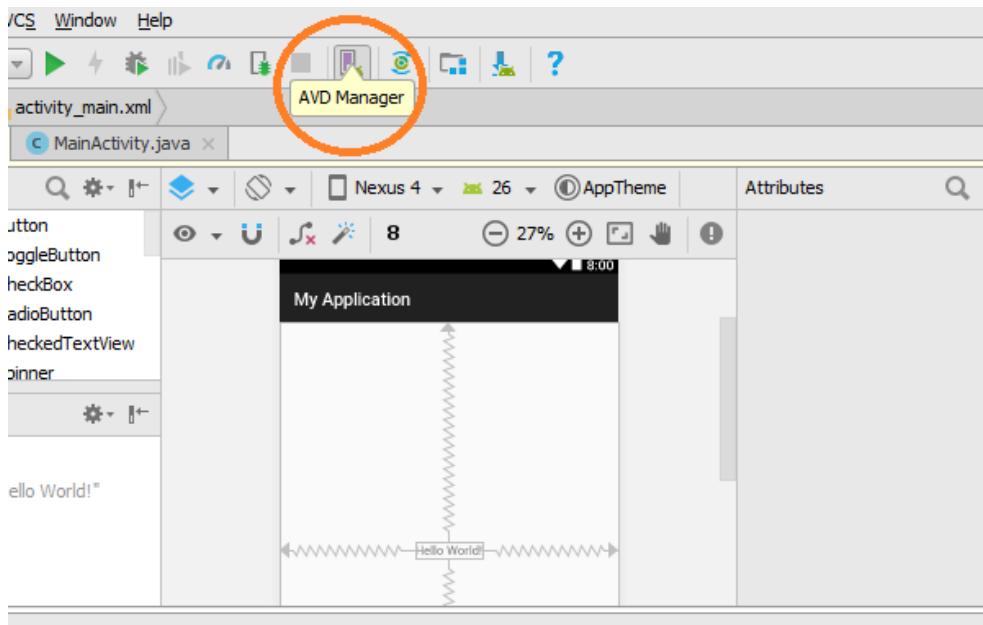
Der HAXM ist eine Hardware unterstützte virtuelle Umgebung, die den Entwicklungsprozess von Android Anwendungen beschleunigt.

Diese benötigen wir für das Emulieren eines Android Virtual Devices (AVD). Zum Schluss musste ich die Eigenschaften von meinem Emulator bearbeiten (unter AVD Manager).

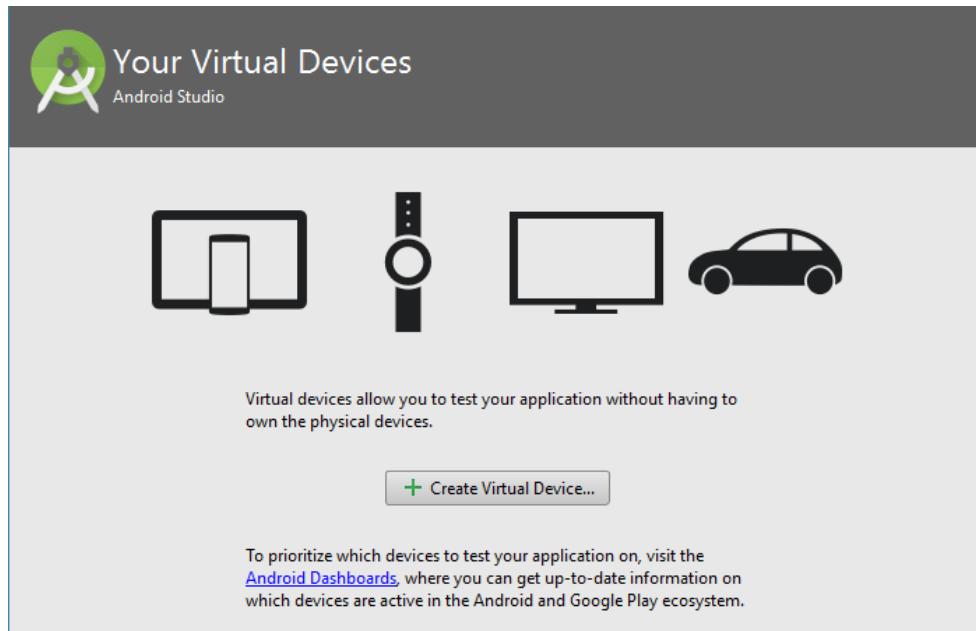
Danach habe ich ein virtuelles Gerät erstellt und Emulator gestartet. Wie folgt:

## 5 ANDROID VIRTUAL DEVICE (AVD)

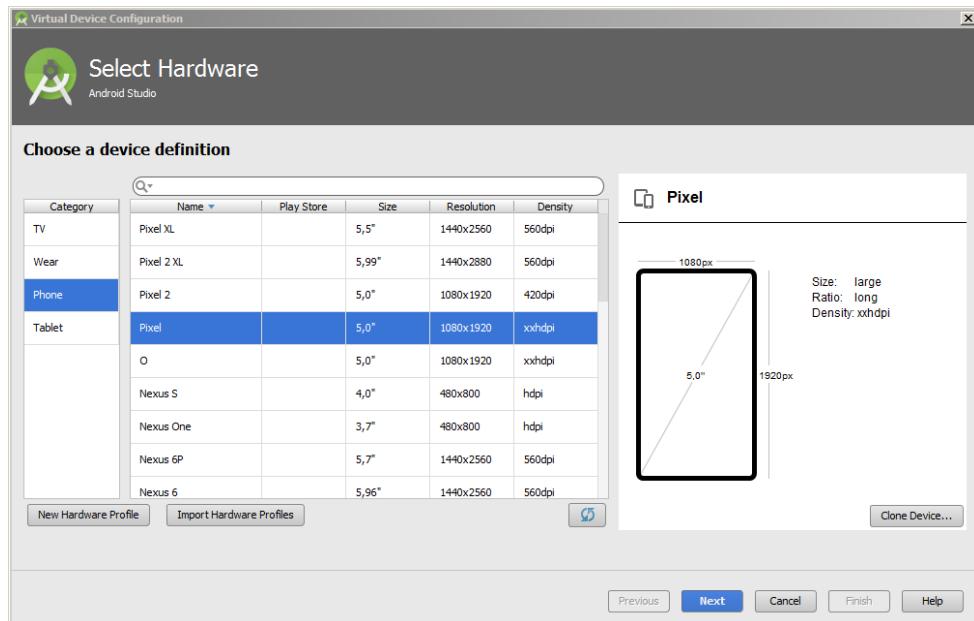
Wir können den Manager auf zwei Arten öffnen. Mit einem Klick auf das AVD Manager-Symbol in der oberen Menüleiste oder direkt über das Hauptmenü, indem wir auf Tools>Android>AVD Manager klicken.



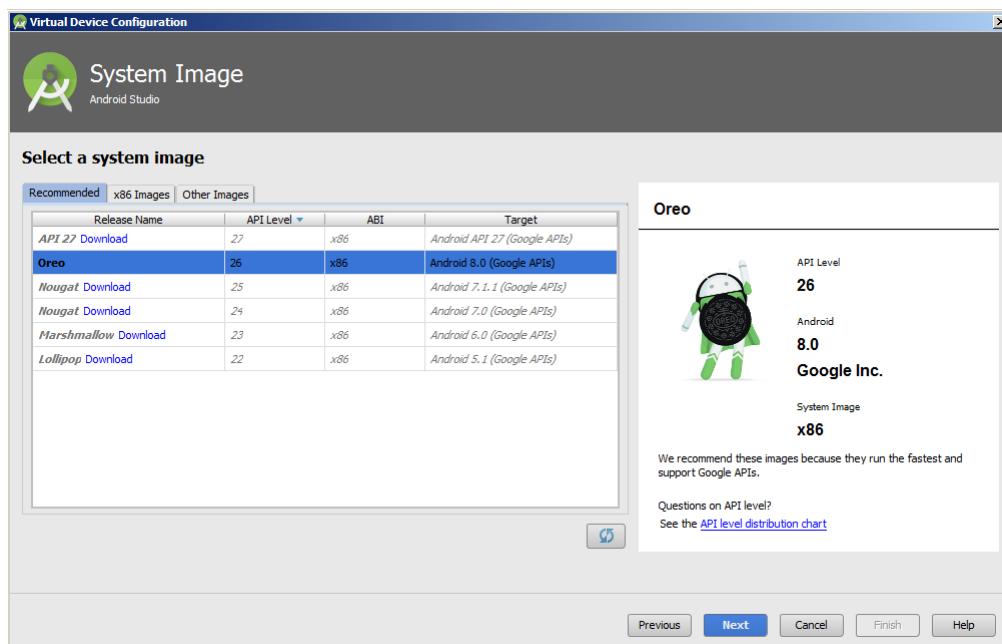
Nachdem der AVD Manager das erste Mal gestartet wurde, öffnet sich der „Your Virtual Devices“-Dialog des Managers. Wir haben an dieser Stelle die Möglichkeit ein Android Virtual Device einzurichten. Mit einem Klick auf „Create Virtual Device“ können wir ein neues AVD erstellen lassen:



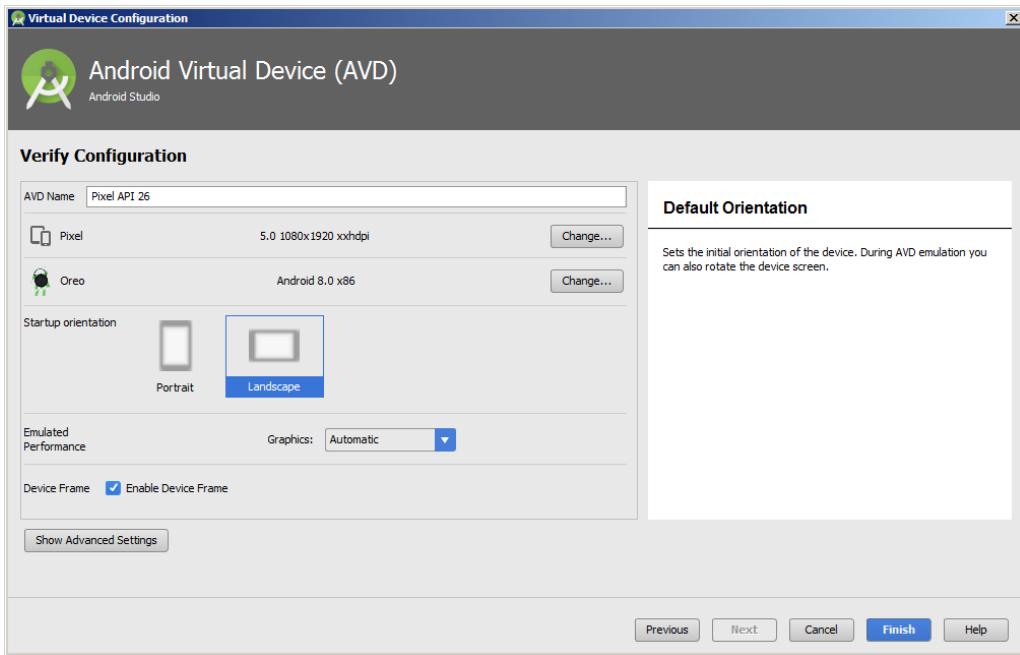
Als Erstes muss man ein Hardware-Profil für das AVD festlegen. Ich wähle ein 5 Zoll große Pixel.



Dann habe ich als System Image die neueste Android Version nämlich Oreo ausgewählt. Das ausgewählte System Image wird auf dem Emulator ausgeführt werden.

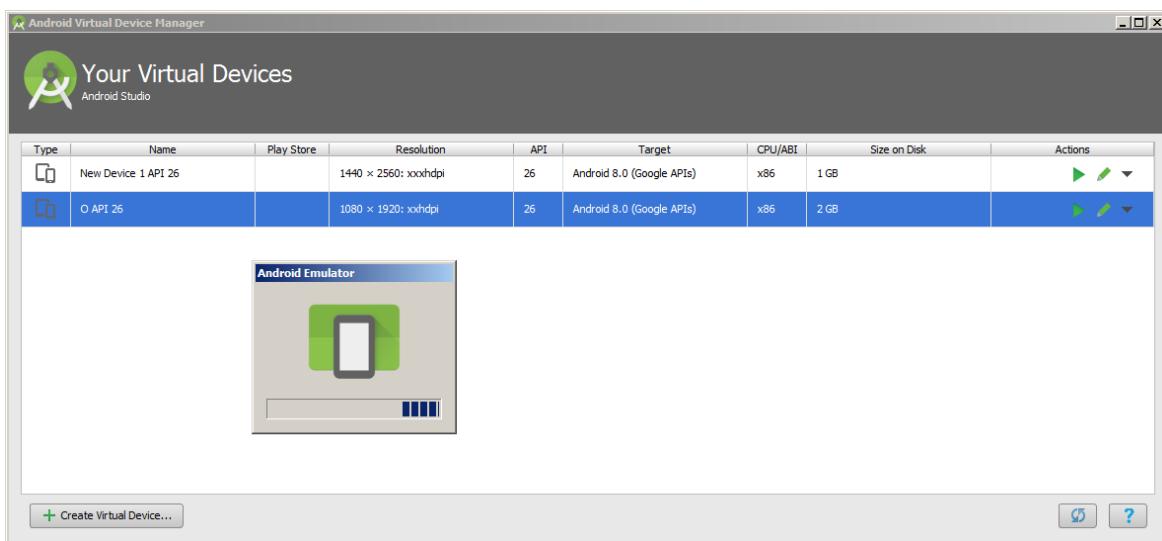


AVD wurde fast eingerichtet. In diesem letzten Dialog habe ich den Namen von Emulator festgelegt. wenn man den Knopf "show advanced settings" drückt, sieht man die erweiterten Einstellungen z.B. RAM-Größe oder Netzwerkeinstellungen.



## 5.1 EMULATOR STARTEN

Nachdem das Android Virtual Device (AVD) erstellt wurde, möchten wir es in dem Android Emulator ausführen.



Das erstellte AVD wird nun im Emulator gestartet, indem man „Android virtual Device manager“ wieder startet und beim erstellten AVD den roten Play-Knopf drückt.

Der Startvorgang kann manchmal sehr lange dauern. Da mein Laptop mit 8 GB RAM und einer SSD Festplatte ausgestattet ist, dauert den Lagevorgang (zum Starten vom Emulator) bei mir etwa 50 Sekunden.



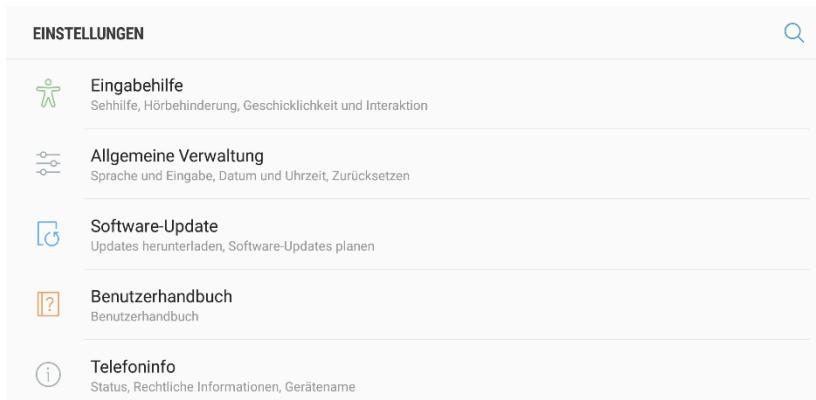
## 6 APP AUF DEM HANDY INSTALLIEREN

Da in meinem Projekt einen Sensor ansprechen werden soll, muss die App auch auf einem physikalischen Android Gerät installiert werden, damit man die App testen kann. Bevor das Android Gerät mit dem PC verbunden werden kann, muss die Verbindung zugelassen werden. Dazu müssen die Entwickleroptionen (Developer Options) aktiviert werden. Auf Smartphones (Handys) und Tablets, auf denen Android 4.2 oder neuer installiert ist, sind die Entwickleroptionen (Developer Options) standardmäßig versteckt.

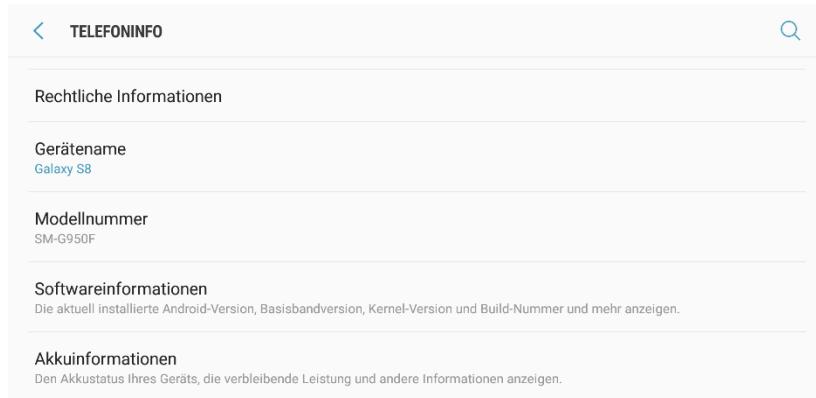
### 6.1 USB-DEBUGGING AKTIVIEREN

Mit diesen Schritten kann man die Entwickleroptionen (Developer Options) bei einem Android Gerät aktivieren, in meinem Fall „Samsung Galaxy s8“:

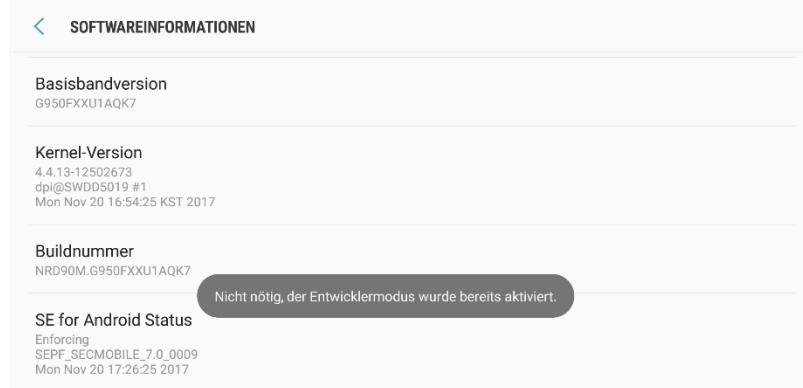
1. in *Einstellungen* vom Handy auf den Tab *Telefoninfo* klicken.



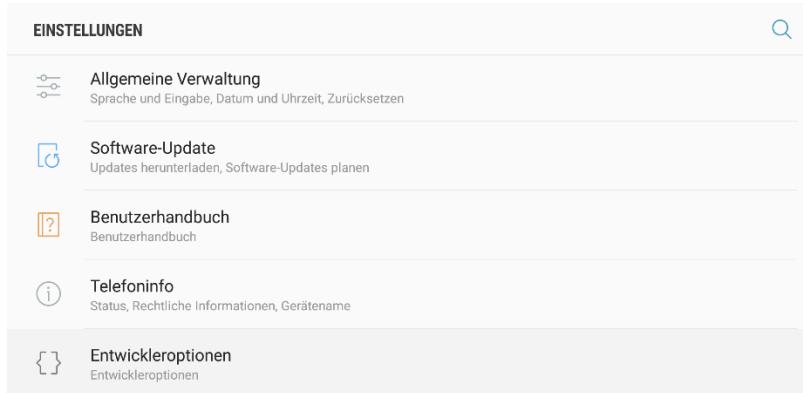
2. In Tab *Telefoninfo* ganz nach unten scrollen und auf *Softwareinformationen* klicken.



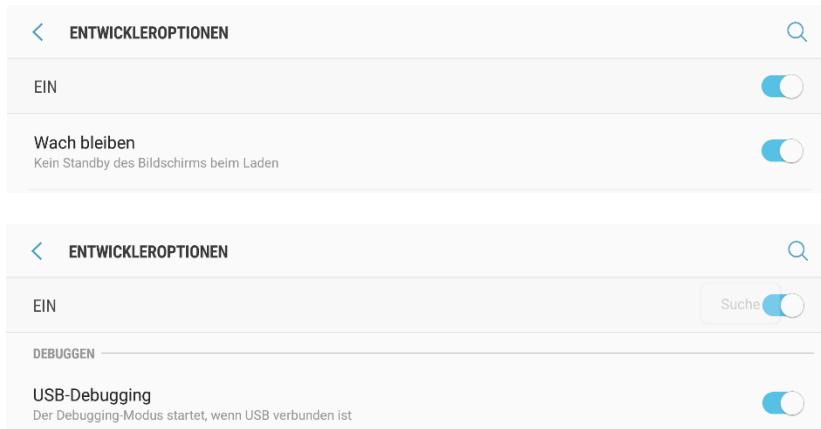
3. In *Softwareinformationen* jetzt 7-mal auf den Eintrag „Buildnummer“ klicken. Dann erscheint eine Meldung, dass jetzt die *Entwickleroptionen* aktiviert wurden.



4. In Optionen ganz nach unten scrollen und auf *Entwickleroptionen* klicken.



5. In Tab Entwickleroptionen müssen die Optionen *Entwickleroptionen, wach bleiben* und *USB-Debugging* aktiviert sein.



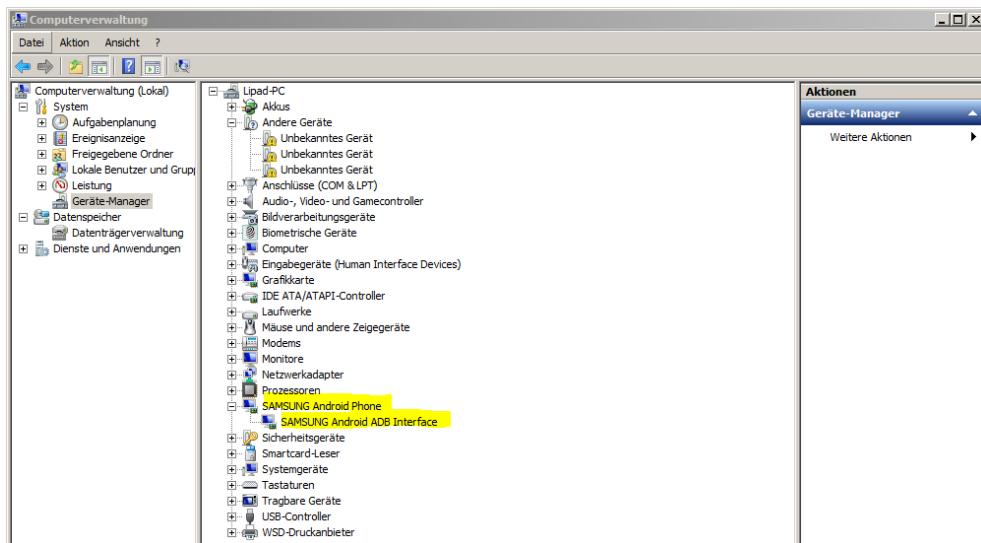
## 6.2 ANDROID GERÄT MIT DEM PC VERBINDEN

Die Treiberinstallation verläuft je nach Art des Android Geräts unterschiedlich. Eine ausführliche Installationsanleitung findet ihr hier:

<https://developer.android.com/studio/run/oem-usb.html#InstallingDriver>

Nachdem der USB Treiber für das Android Gerät nach Angaben von Google bzw. des Herstellers installiert wurde, kann man überprüfen, ob das Gerät bei der Installation korrekt eingerichtet wurde. Dazu öffnet man den Gerätemanager von Windows. Das Gerät muss für diesen Schritt am PC über USB angeschlossen sein.

Im Gerätemanager sollte das Gerät jetzt unter Android Phone als Android ADB Interface aufgeführt sein.

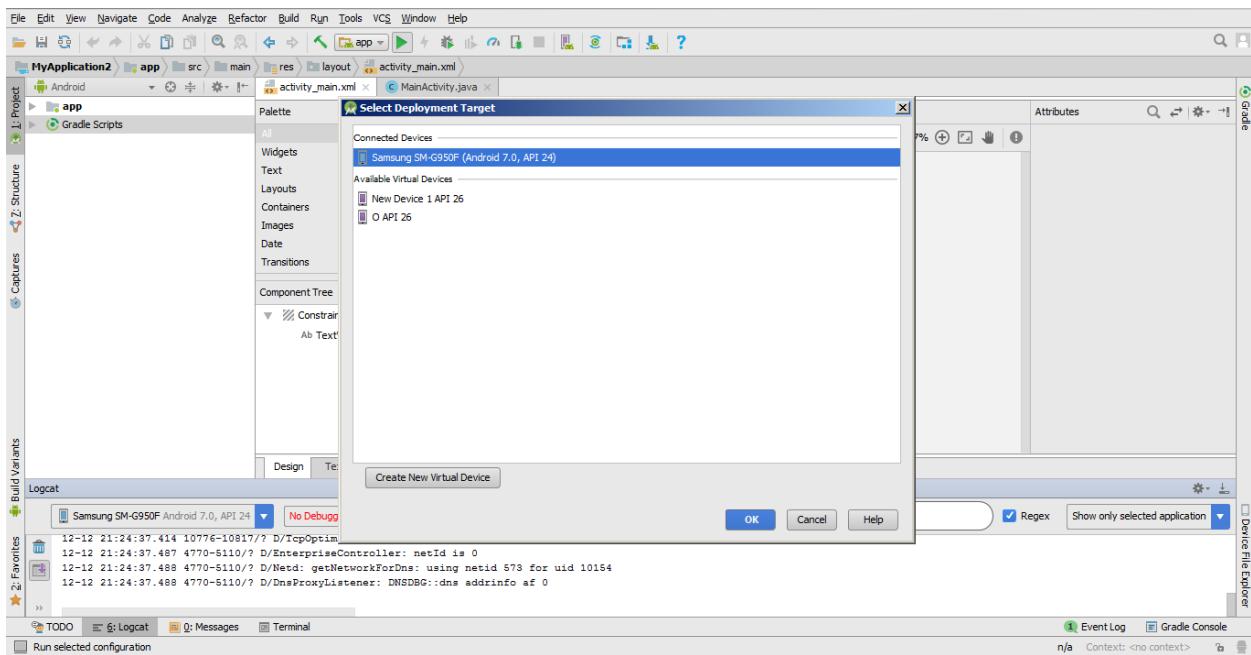


## 6.3 DIE APP AUF DEM GERÄT INSTALLIEREN UND STARTEN

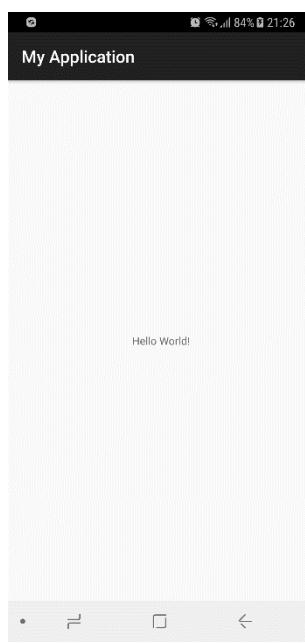
Jetzt wo mein Samsung Galaxy S8 mit dem PC über die *Android Debug Bridge* verbunden ist, sind die Voraussetzungen für das Installieren unserer App auf dem Gerät erfüllt.

Ich öffne das einfache Android Projekt, welches ich vorher erstellt habe. Die einfache App ist bereits funktionstüchtig und kann ausgeführt werden.

Nachdem das Android Studio Projekt geöffnet wurde, überprüfen ich den Status der angeschlossenen Geräte. Mein Smartphone oder sollte jetzt in dem Android *Monitor-View* unter *Devices* zu sehen sein.



Dann installiere ich die App auf meinem Handy.



## 7 1. TEIL

Bitte beachten Sie:

Diesen ersten Teil des Praktikums soll jeder selbst machen. Sie dürfen das Entwicklungssystem, nach Rücksprache, selbst wählen. Beschreiben Sie den Lösungsweg und Ihre Lösung schriftlich in einer Ausarbeitung. Ziel des ersten Teils ist es, grundlegende Kenntnisse über die Entwicklung mobiler Anwendungen zu erwerben.

### 7.1 AUFGABE 1 - AUFGABESTELLUNG

Aufgabe 1:

- Installieren Sie eine passende Entwicklungsumgebung auf Ihrem Notebook.
- Entwickeln Sie eine Anwendung/App, die „Hallo (Ihr Name)“ ausgibt
  - a. Fügen Sie einen Button hinzu, bei dessen Drücken dieser Text erscheint.
  - b. Geben Sie dem Benutzer die Möglichkeit Schriftgröße und Textfarbe des dargestellten Textes zu editieren. Schauen Sie sich hierfür verschiedene Elemente einer GUI an (NumberPicker; EditText).

#### 7.1.1 MyNameApp Layout

Ich habe ein neues Projekt ausgewählt und habe es "MyNameApp" genannt. Danach habe ich von der „Palette“ welche sich im Android-Studio an der linken Seite der App-Layout befindet, den Punkt „TextView“ genommen und in die Mitte oben positioniert, dann einen Button in die Mitte unten positioniert und habe sie "CLICK" genannt. Der Button und das TextView welches aus den Werkzeugleiste in den Bildschirm positioniert wurden, generieren automatisch einen XML Code in activity\_main.xml.

```
<TextView
    android:id="@+id/TextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:gravity="center_vertical"
    android:textAlignment="viewStart"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.109"
    tools:layout_constraintBottom_creator="1"
    tools:layout_constraintLeft_creator="1"
    tools:layout_constraintRight_creator="1"
    tools:layout_constraintTop_creator="1" />

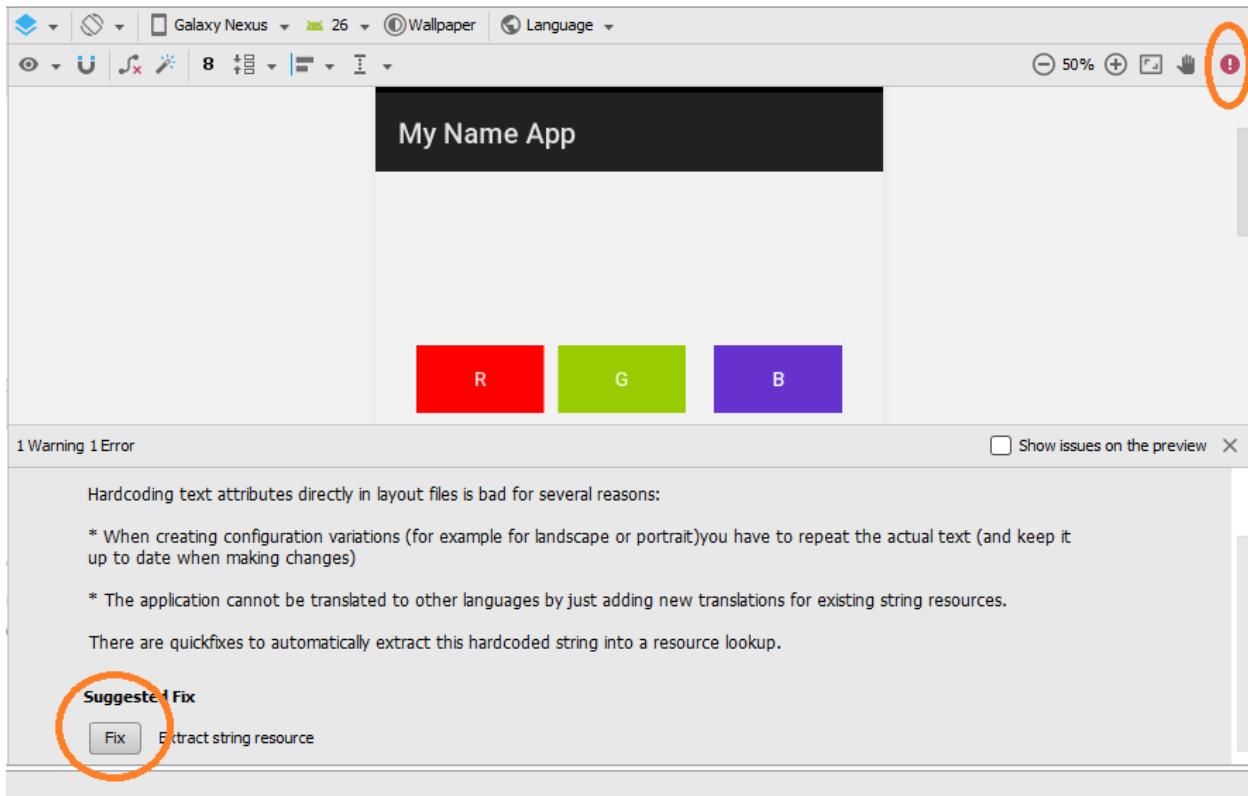
<Button
    android:id="@+id/button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginBottom="35dp"
    android:text="Click"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    tools:layout_constraintBottom_creator="1"
    tools:layout_constraintLeft_creator="1"
    tools:layout_constraintRight_creator="1" />
```

Dann habe ich von der Palette ein horizontales „Linear Layout“ genommen und in der Mitte unter dem „TextView“ positioniert. Im „Linear Layout“ habe ich drei Button in hinzugefügt. Auf der rechten Seite, bei den „Attributes“ den „Text“ den jeweiligen Buttons geändert, und sie R, G und B genannt und die Hintergrundfarbe der Buttons geändert.

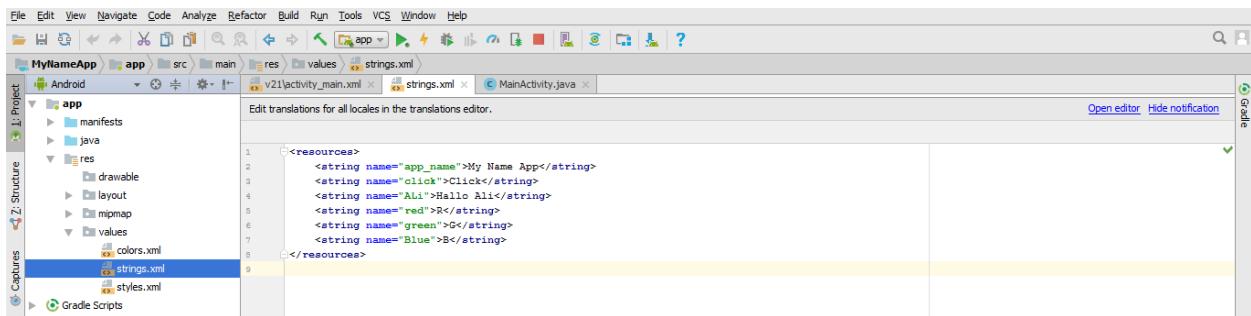
In der Datei Activity\_main.xml ist jetzt die Warnung „hardcoded string "R", should use @string resource“ zu sehen. Da ich die Namen von Buttons nicht in der Datei String.xml eingetragen habe, erscheint diese Warnung. Um es zu beheben kann man einfach oben rechts auf dem kleinen Ausrufezeichen klicken, dann öffnet sich einen Tab, in den man mit dem klicken auf den Knopf „Fix“ die Texte in der Datei einträgt. Natürlich kann man dies auch selber machen, indem man das per Hand in der Datei reinschreibt.

Ich gebe als Name für die Strings Red, Green und Blue. Die Texte werden dann über diese Namen von den Ressourcen aufgerufen. Z.B.:

```
    android:text="@string/red"
```



Die Datei String.xml ist unter „Values“ zu finden in den Tab „Project“ und wird ebenfalls mit einem neuen Projekt immer generiert. In der strings.xml sind alle mögliche Texte in der App definiert. Damit kann die App ganz einfach in andere Sprachen übersetzt werden. Dazu müssen nur die Tags dieser Datei übersetzt werden. Ich füge jede Text von der App der Datei hinzu bzw. kann man es automatisch hinzufügen lassen.



„NumberPicker“ habe ich für die Textgröße aus den „Widgets“ genommen und habe es zwischen die Farbknöpfe und CLICK-Button gesetzt.

### 7.1.2 MainActivity

In dem folgenden Codeabschnitt der „MainActivity“ -Klasse werden für jedes Element im Layout eine Variable definiert.

```
public class MainActivity extends AppCompatActivity {

    TextView txt;
    Button Ok;

    //button
    Button btnRed;
    Button btnBlue;
    Button btnGreen;

    // text size
    public NumberPicker np;
```

### 7.1.3 Codeausschnitte

Die Methode onCreate() ist eine der zustände einer App bzw. Activity Lifecycle. Es muss in jedem Activity implementiert sein. Sie erzeugt das Activity sowie dessen Ansicht/View. wird nur einmal aufgerufen, wenn die App startet, die Elemente werden hier angenommen und zugewiesen.

Die definierte Variablen werden hier mittels ID (findViewById()) mit den Layouts-Elementen verbunden. wenn der Button geklickt wird, wird onClick() aufgerufen und durch „setText“ kann meine Text in den TextView angegeben werden.

```
txt = (TextView) findViewById(R.id.TextView);
Ok = (Button) findViewById(R.id.button);
Ok.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick (View v) {
        txt.setText(R.string.Ali);
    }
});

btnRed = (Button) findViewById(R.id.red);
btnBlue= (Button) findViewById(R.id.blue);
btnGreen= (Button) findViewById(R.id.green);
```

### 7.1.4 Text Farbe

setOnClickListener meldet einen Listener an, dessen onClick() Methode bei einem Klick auf den Button aufgerufen wird. Durch den „setTextColor“ konnte ich die Farbe meiner Text „Hallo Ali“ festlegen.

```

btnRed.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        txt.setTextColor(Color.parseColor("#FF0000"));
    }
});
btnBlue.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {
        txt.setTextColor(Color.parseColor("#0000FF"));
    }
});
btnGreen.setOnClickListener(new View.OnClickListener() {

    public void onClick(View v) {
        txt.setTextColor(Color.parseColor("#006400"));
    }
});

```

### 7.1.5 Schriftgröße ändern

Die Größe der Textausgabe wird durch den „NumberPicker“ geändert. Den „NumberPicker“ habe ich für die Textgröße aus den „Widgets“ genommen und habe damit weitergearbeitet. Durch „setMinValue“ und „setMaxValue“ werden die minimale und die maximale Textgröße festgelegt.

```

np.setMinValue(10);

np.setMaxValue(30);

np.setWrapSelectorWheel(true);

np.setOnValueChangedListener(
    new NumberPicker.OnValueChangeListener() {
        @Override
        public void onValueChange(NumberPicker picker, int oldVal, int newVal) {

            txt.setTextSize(newVal);
        }
});

```

### 7.1.6 Android Asset Studio

Ich habe über das Online Programm Android Asset Studio für meine Kleine App ein Icon erstellt und es über „Image Asset“ zum Ressourcen hinzugefügt.

<https://romannurik.github.io/AndroidAssetStudio/>

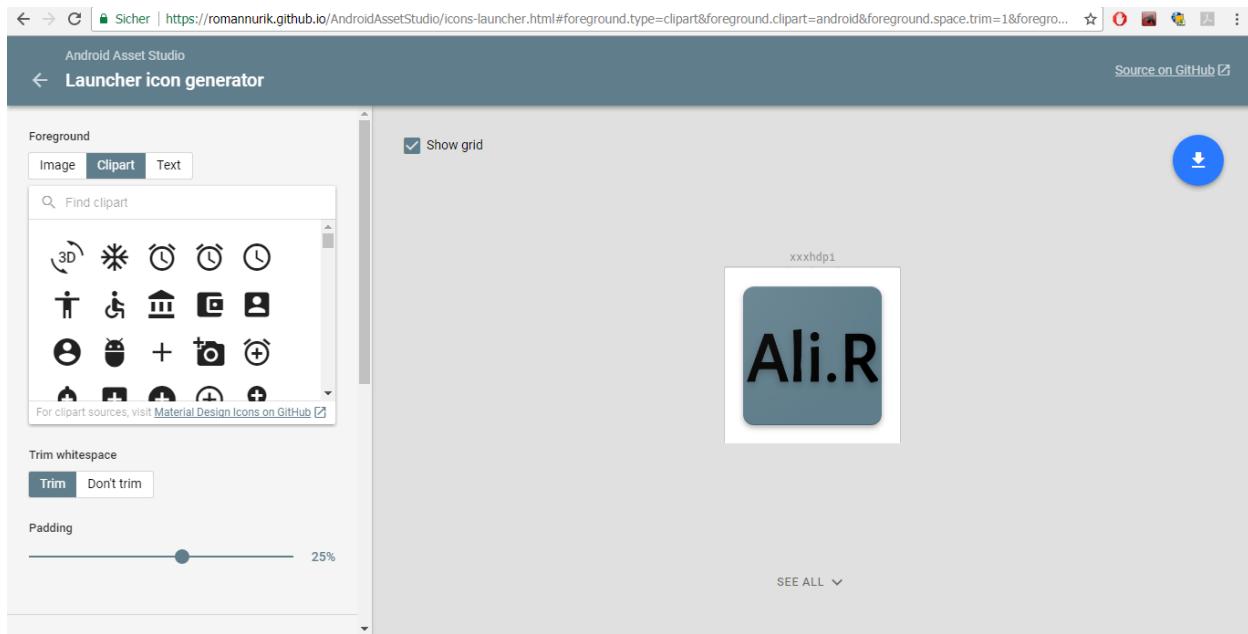
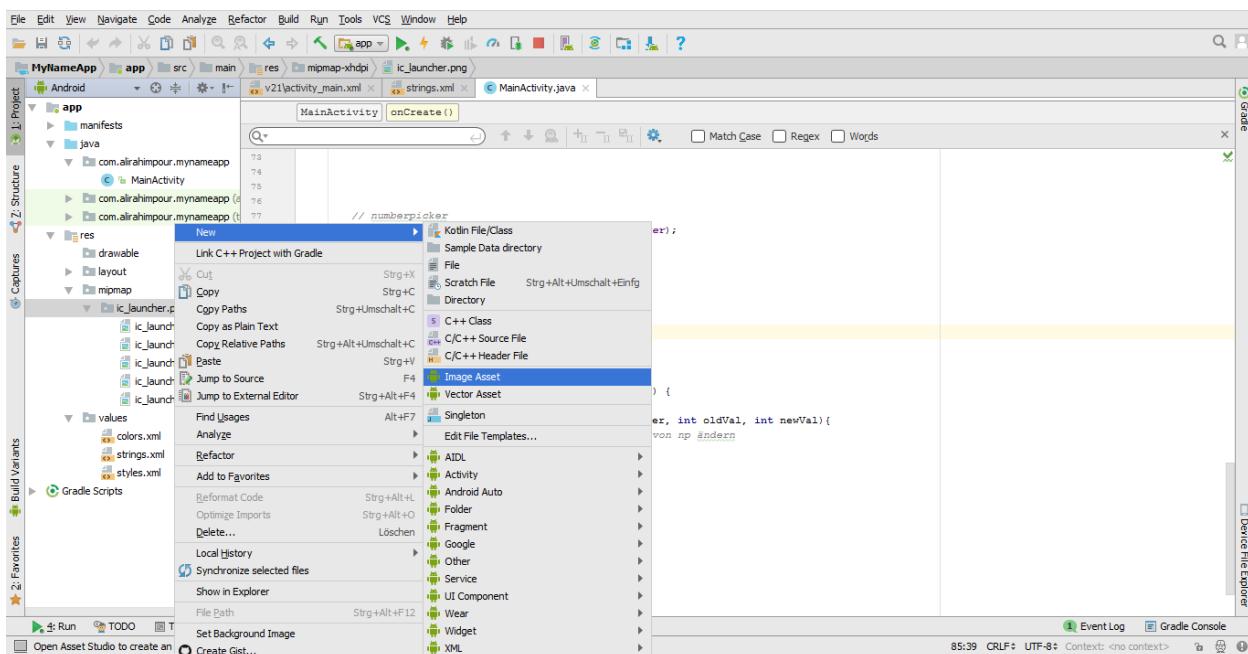
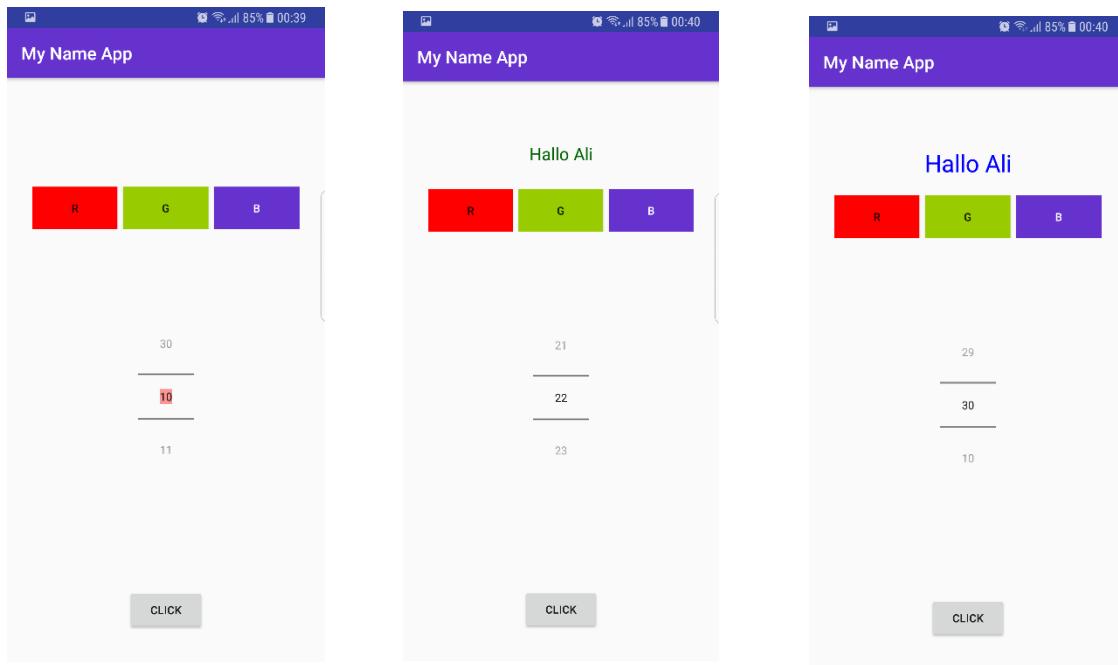


Image Asset:



### 7.1.7 Die App Übersicht & Video



Eine kurze Videoaufnahme von der App können Sie sich über den YouTube-Link oder QR-Code ansehen:



## 8 3. TEIL: PROJEKT

### 8.1 IDEE: FITNESS\_APP

„Fitness App“ verwendet den eingebauten Sensor step counter, um die Schritte zu zählen. GPS-tracking wird nicht verwendet, daher verbraucht sie sehr wenig Akkuleistung, deshalb ist „Fitness App“ für die Personen, die sehr lange laufen wollen sehr praktisch, da man möglicherweise keine zusätzliche Powerbank mitnehmen muss. Sie rechnet außerdem die verbrannten Kalorien, Laufentfernung und die Zeit.

Es ist keine Anmeldung oder Registrierung erforderlich. Die App sammelt oder speichert zu keinem Zeitpunkt personenbezogene Daten oder teilt die Daten mit den anderen. Einerseits sind die genommenen Daten sicher, andererseits die Bedingung der App ist daher leichter. Außerdem man kann die Zählfunktion jederzeit pausieren, starten oder zurücksetzen und wieder bei 0 anfangen.

Die App wurde sehr schlicht konstruiert. Das klare Design hilft bei der einfachen Verwendung. Um die Genauigkeit der Ausgaben zu verbessern, sollte man die Schrittänge und verbrachte Kalorien pro km in den Einstellungen eingeben. Diese werden zur Berechnung der Laufentfernung und die Kalorien verwendet.

### 8.2 SENSOREN

#### 8.2.1 Einleitung

Die Entwicklung von Smartphones wird immer schneller, sie verfügen über mehr Rechenleistung, neue Technologien und immer mehr Sensoren. Smartphones stecken voller Sensoren, die ständig Daten sammeln. Entsprechende Programme können damit sehen, hören und fühlen, was alles in der Umgebung des Gerätes geschieht.

Sensoren werden für unterschiedliche Zwecke eingesetzt, der Helligkeitssensor beispielsweise passt die Displaybeleuchtung abhängig von dem Umgebungslicht automatisch an, der Annäherungssensor deaktiviert beim Telefonieren das Display, wenn man das Gerät nah ans Ohr hält, um versehentliche Eingaben durch Berührung mit der Wange zu verhindern. Bewegungssensible Sensoren wie der Accelerometer, Orientierungssensor und das Gyroskop werden zur Ausrichtung des Displays (Hoch oder Querformat), zur Steuerung von Spielen, aber vor allem zur Untergrund- und Innenraum-Navigation genutzt.

## 8.2.2 Sensoren im Android-SDK

Die Android-Plattform unterstützt drei große Kategorien von Sensoren:

- **Bewegungssensoren**

Diese Sensoren messen Beschleunigungskräfte und Rotationskräfte entlang drei Achsen. Zu dieser Kategorie gehören z.B. Beschleunigungsmesser, Schwerkraftsenso-ren, Gyroskope und Rotationsvektorsensoren.

- **Umweltsensoren**

Diese Sensoren messen verschiedene Umgebungsbedingungen, wie z.B. Temperatur, Druck, Beleuchtung und Feuchtigkeit. Zu dieser Kategorie gehören z.B. Barometer, Photometer und Thermometer.

- **Positionssensoren**

Diese Sensoren messen die physikalische Position eines Geräts. Zu dieser Kategorie gehören z.B. Orientierungssensoren und Magnetometer.

Man kann diese Sensoren zugreifen und mithilfe des Android Sensor-Framework Rohdaten erwerben. Sensor-Framework ist ein Teil von „*android.hardware*“ Package und umfasst die folgenden Klassen und Schnittstellen:

- **SensorManager**

Die Funktionen, damit man auf die Sensoren im Handy zugreifen kann, sind über die SensorManager-Klasse erreichbar. Diese Klasse bietet verschiedene Methoden und konstanten für den Zugriff auf die Sensoren.

- **Sensor**

Man Kann diese Klasse verwenden, um eine Instanz eines bestimmten Sensors zu erstellen. (In meinem Fall: TYPE\_STEP\_COUNTER) Diese Klasse bietet verschiedene Methoden, mit denen man die Fähigkeiten eines Sensors bestimmen kann.

- **SensorEvent**

Das System erstellt mit dieser Klasse ein "sensor event"-Objekt, das die Informationen über ein "sensor event" liefert. Ein "sensor event"-Objekt enthält die folgenden Informationen: die Rohsensordaten, den Typ des Sensors, der das Ereignis (event) generiert hat, die Genauigkeit und so weiter.

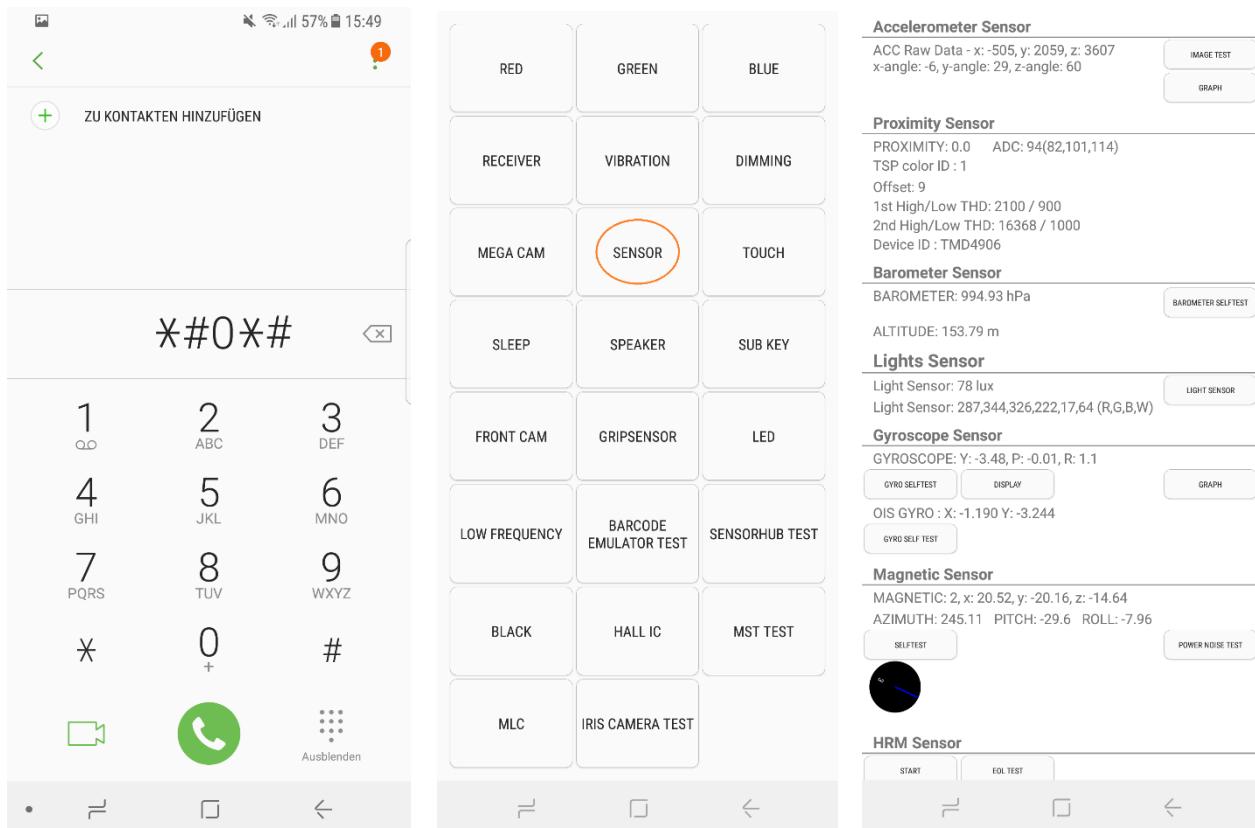
- **SensorEventListener**

Man kann diese Schnittstelle verwenden, um zwei Callback-Methoden zu erstellen, die Benachrichtigungen (sensor events) empfangen, wenn sich die Sensorgenaugkeit oder die Sensorwerte ändern

### 8.2.3 Sensoren Testen

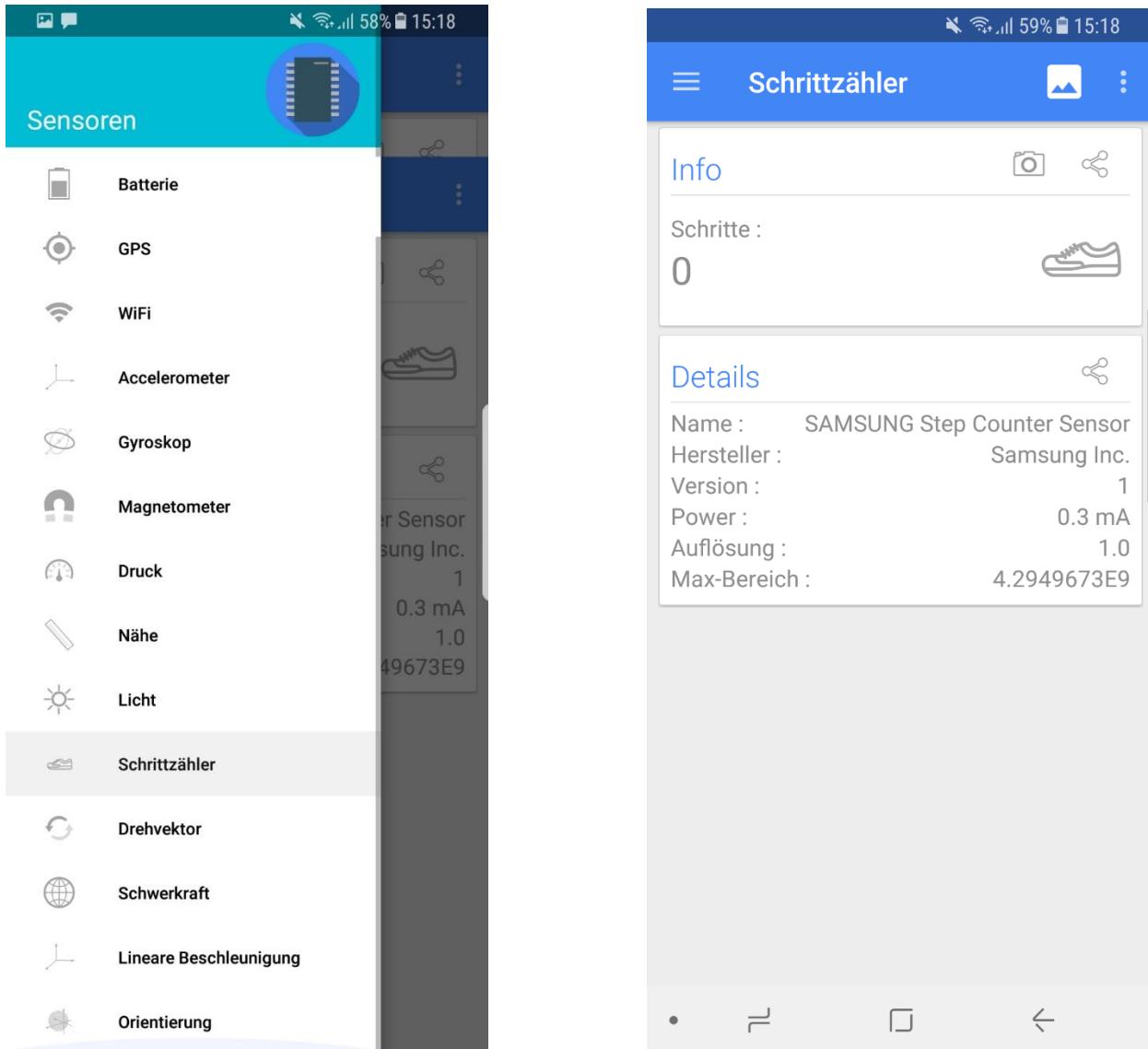
Jedes Gerät besitzt ein Programm zum Testen der Funktionen. Um in meinem Handy also, Samsung Galaxy S8 die Sensoren und deren erfasste Daten gesehen werden zu können, geht man vom Homescreen aus auf dem Telefon App und dort auf den Reiter "Tastenfeld". Man gibt nun auf dem Tastenfeld folgenden Code ein: \*#0\*#

Nachdem man den Code eingegeben hat, erscheint auf dem Bildschirm ein Menü mit sehr vielen Kacheln, wobei jede Kachel für ein Test-Menü steht. Hier kann man nun die Sensoren testen und die erfasste Daten sehen.



Alternative kann man die Sensoren-Test App von Google Play herunterladen.

Ich habe die App „Sensors Multitool“ installiert und dadurch konnte ich einen besseren Überblick über die eingebaute Sensoren meines Hany's haben. In den Abbildungen sieht man den Step Counter Sensor (Schrittzählersensor) genauer.



### 8.3 ZIEL & ANFORDERUNGEN AN DAS PROJEKT

Es soll zum einen möglich sein, Schritte zu zählen, zum anderen sollen die verbrauchten Kalorien und die Entfernung berechnet werden. Dabei soll die Zeit durch eine Stoppuhr erfasst werden.

Außerdem soll ein „Setting-Menü“ eingefügt werden, die es ermöglicht, die individuellen Informationen (Schrittlänge und Kalorie pro km) einzugeben zu können, die beim Berechnung der Entfernung und der verbrauchten Kalorien berücksichtigt werden sollen.

### 8.4 DAS DESIGN VORBEREITEN

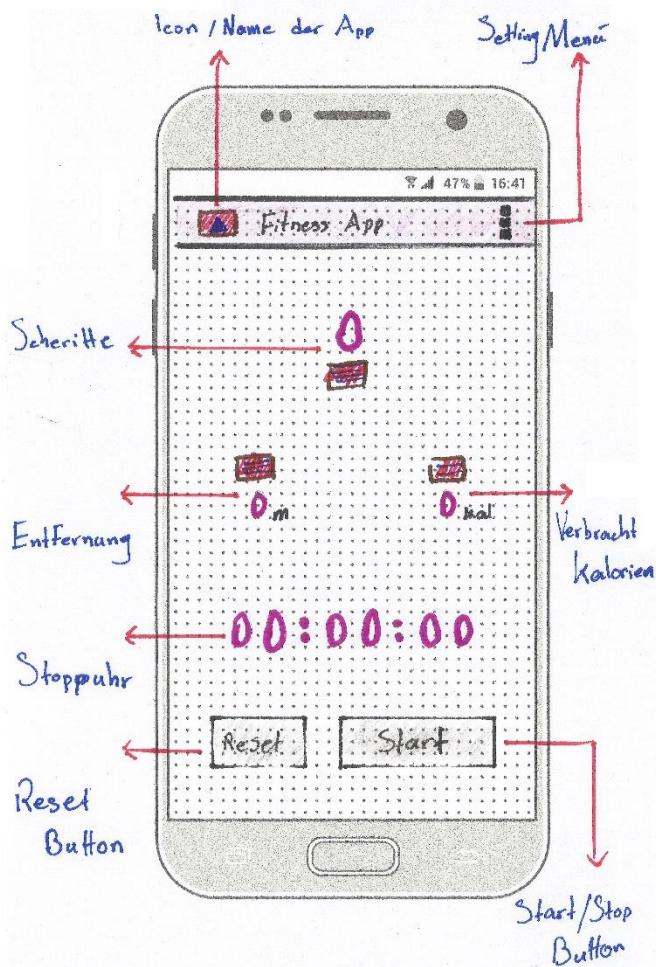
#### 8.4.1 Planung - Mockup

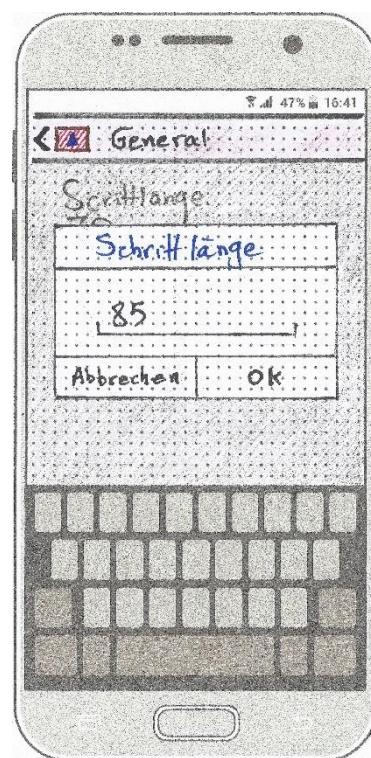
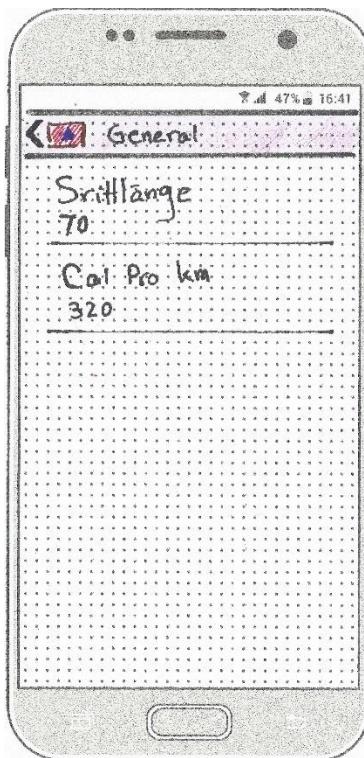
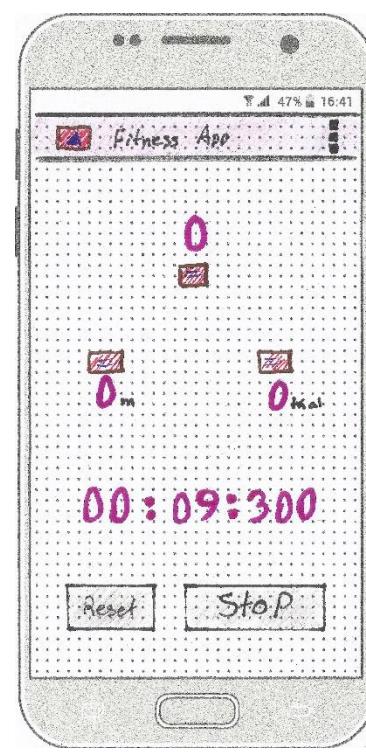
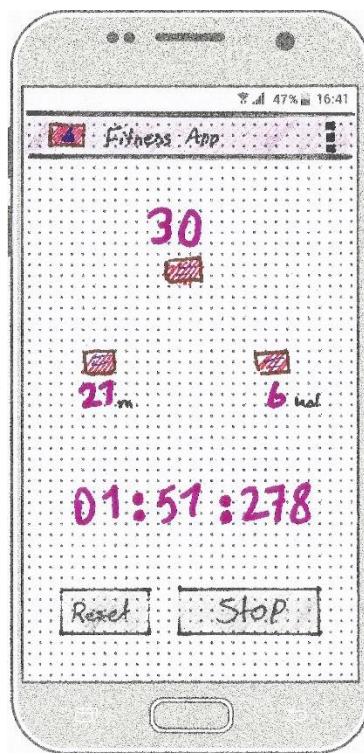
Um einen Überblick zu schaffen, werden zu Beginn Mockup erstellt, die die Ziele bildlich darstellen. Die Mockups dienen als „roter Faden“.

Die Felder und Elemente aus der App werden in die richtige Position gerückt um dem Mockup gerecht zu werden. Buttons wie z.B. der Reset Button werden gleich zu Beginn eingefügt.

Ich habe meine Ziele bzw. wie die App aussehen soll, auf dem Papier gezeichnet.

In der nächsten Seite sieht man die App in der verschiedenen Funktion Situationen. Wie z.B., wenn man auf dem Start/Stop Knopf drückt oder beim Eingeben von der Schrittlänge in den Einstellungen.



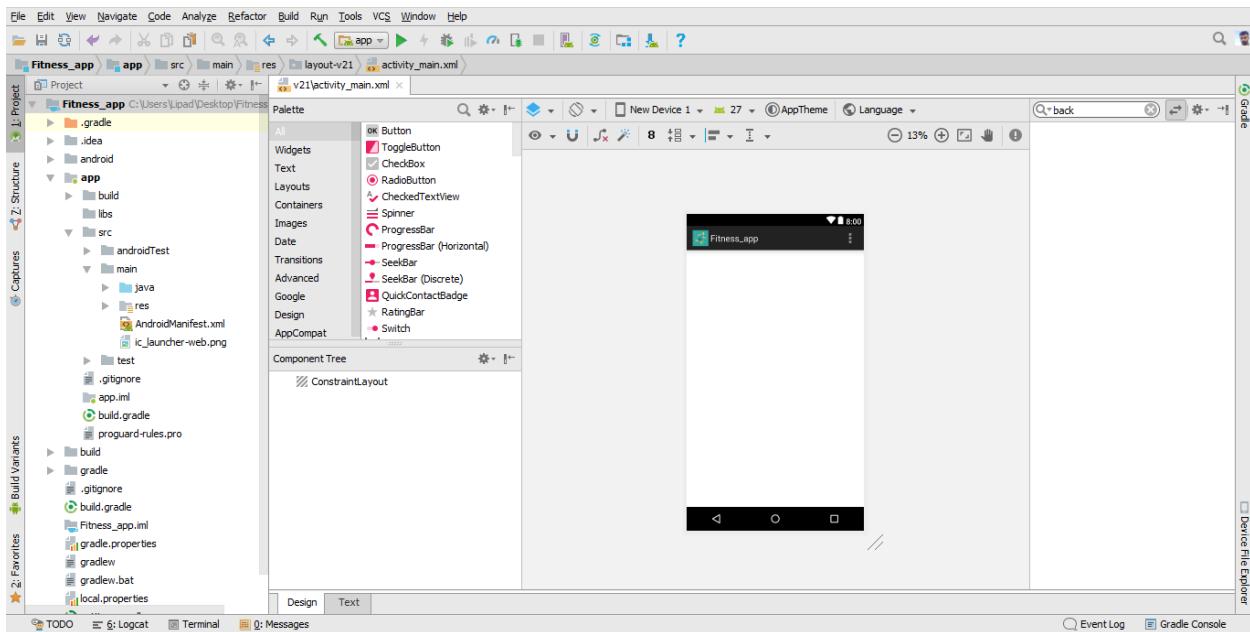


### 8.4.2 Benutzeroberfläche und XML Datei

Die Auszeichnungssprache XML wird zur Darstellung hierarchisch strukturierter Daten verwendet. Die Daten sind in Textform angegeben.

Die „*Extensible Markup Language*“ (engl. „erweiterbare Auszeichnungssprache“, XML) ist für Menschen und Maschinen gleichermaßen verständlich und wird für den plattform- und implementationsunabhängigen Austausch von Daten zwischen Computersystemen eingesetzt, insbesondere über das Internet. Eine XML-Datei besteht aus Textzeichen und ist auch von Menschen leicht lesbar. Binärdaten enthält ein XML-Dokument per Definition nicht.

Das Project-Fenster zeigt als wesentlichen App-Bestandteil im Pfad res/layout die Datei *activity\_main.xml*, welche die Bedienoberfläche der einzigen Aktivität unserer Anwendung definiert. Wird diese Datei in der Editorzone geöffnet, ist per Voreinstellung der Design-Modus aktiv:



Mit dem Klicken auf Tab „Text“ neben „Design“ kann man den XML-Quellcode ansehen. Manche Änderungen der GUI-Definition lassen sich allerdings ausschließlich oder bequemer im Text-Modus erreichen. Man editiert den XML-Code zur Definition der Bedienoberfläche und kann den Effekt in einem Preview-Fenster besichtigen.

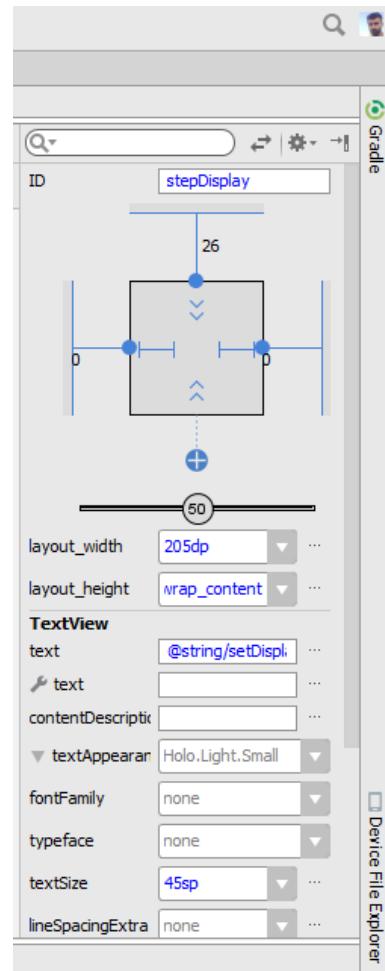
Ich habe von der Werkzeugeiste(Element), welche sich im Android-Studio unter der "Palette" befinden den Punkt „*TextView*“ 2-mal genommen (Schritte und Stoppuhr) und so wie ich im Mockup gezeichnet habe, in der App positioniert. Dann habe ich zwei *LinearLayout* genommen, um in dem den Start/Stop und Reset Knöpfe und zwei „*TextView*“ zur Darstellung von berechneten Entfernung und Kalorien zu platzieren.

*LinearLayout* ordnet die eingebetteten Steuerelemente entweder horizontal (nebeneinander) oder vertikal (untereinander) an. Die Ausrichtung wird über den XML Parameter `android:orientation` festgelegt. Mögliche Werte sind horizontal und vertikal. Fehlt `android:orientation`, wird horizontal verwendet.

den „*Properties*“ in der rechten Seite muss habe ich die Eigenschaft „*Gravity*“ auf „*Center*“ gesetzt damit die Elemente mittig sind. In den „*Properties*“ kann man viele Einstellungen und Eigenschaften der Elemente beeinflussen und ändern.

Beispielweise habe ich die ID der Elemente, mit denen man später die Elemente in den Quellcode ansprechen kann, die Texte, die in den „*TextView*“ gezeigt werden sollen, das Hintergrundfoto der App und andere Änderungen vorgenommen.

man kann allerdings unzählige andere Änderungen vornehmen. Die Änderungen kann man dann direkt in den Text-Editor ansehen.



```

<android.support.constraint.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:layout_editor_absoluteY="174dp" >

    <Button
        android:id="@+id/startStopBtn"
        style="@android:style/Widget.Button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:elevation="0dp"
        android:onClick="startStop"
        android:text="@string/start" />

    </LinearLayout>
    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="213dp"
        android:layout_height="0dp"
        android:layout_marginBottom="272dp"
        android:background="@android:color/transparent"
        android:orientation="horizontal"
        android:weightSum="1"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/meterDisplay"
            android:layout_width="107dp"
            android:layout_height="46dp" />
    </LinearLayout>

```

#### 8.4.3 String.xml

Im Verzeichnis „values“ liegt die Datei strings.xml. Hier werden alle Texte abgelegt. Wenn man das Android-Studio verwendet, kann man das Verzeichnis nicht etwa anlegen, sondern muss das Verzeichnis values mit der rechten Maustaste anklicken.

```

<resources>
    <string name="app_name">Fitness_app</string>
    <string name="schritte">Schritte</string>
    <string name="start">Start</string>
    <string name="reset">Reset</string>
    <string name="_0000_kal">0 Kal</string>
    <string name="_0000_m">0 m</string>
    <string name="switchStart">Start</string>
    <string name="switchStop">Stop</string>
    <string name="title_activity_settings">Settings</string>
    <!-- Strings related to Settings -->
    <!-- Example General settings -->
    <string name="pref_header_general">General</string>
    <string name="pref_title_social_recommendations">Enable social recommendations</string>
    <string name="pref_description_social_recommendations">Recommendations for people to contact
        based on your message history
    </string>
    <string name="pref_title_display_name">example_name</string>
    <string name="pref_default_display_name">John Smith</string>
    <string name="pref_title_add_friends_to_messages">text</string>
    <string-array name="pref_example_list_titles">
        <item>Always</item>
        <item>When possible</item>
        <item>Never</item>
    </string-array>
    <string-array name="pref_example_list_values">
        <item>1</item>
    </string-array>

```

## 8.5 IMPLEMENTATION

Die Methode „`onCreate()`“ muss in jedem Activity implementiert sein. Sie erzeugt das Activity sowie dessen Ansicht/View. wird nur einmal aufgerufen, wenn die App startet, die Elemente werden hier angenommen und zugewiesen.

Die Elemente die ich im Layout hinterlegt habe, werde ich im Code erstmal definieren. Dann werden sie in „`onCreate()`“ implementiert.

```
private Button startStopBtn;
private Button resetBtn;
private TextView meterDisplay, kalDisplay, stepDisplay;
private int currentSteps = 0;
private int initialSteps = 0;

private boolean isCounting = false;
```

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    startStopBtn = findViewById(R.id.startStopBtn);
    resetBtn = findViewById(R.id.resetBtn);

    kalDisplay = findViewById(R.id.kalDisplay);
    meterDisplay = findViewById(R.id.meterDisplay);
    stepDisplay = findViewById(R.id.stepDisplay);

}
```

## 8.6 DEN SENSOR ANSPRECHEN

Erstmal muss man die Schnittstelle „`SensorEventListener`“ implementieren. Sie Wird verwendet, um Meldungen vom SensorManager zu empfangen, wenn sich die Sensorwerte geändert haben.

```
public class MainActivity extends Activity implements SensorEventListener {
```

Sobald man die Schnittstelle implementiert, bekommt man eine Fehlermeldung, die man lösen kann, indem man auf der roten Lampe klickt, um die fehlende Methode dieser Schnittstelle zu generieren.

SensorEventListener hat zwei Public Methoden:

```
public void onSensorChanged(SensorEvent sensorEvent) {  
}  
  
public void onAccuracyChanged(Sensor sensor, int i) {  
}
```

- **onAccuracyChanged** wird aufgerufen, wenn sich die Genauigkeit eines Sensors geändert hat.
- **onSensorChanged** wird aufgerufen, wenn sich die Sensorwerte geändert haben. Die Länge und der Inhalt des Werte-Arrays variieren abhängig davon, welcher Sensor überwacht wird. Später werden wir diese Methode brauchen.

### 8.6.1 SensorManager & STEP\_COUNTER Sensor

Um einen Sensor verwenden zu können, braucht man SensorManager und die Sensor Bezeichnung und sie müssen wie alle anderen Elemente erstmal definiert und danach in „*onCreate()*“ implementiert werden

```
private SensorManager sm;  
private Sensor stepCounter;
```

```
sm = (SensorManager) getSystemService(SENSOR_SERVICE);  
stepCounter = sm.getDefaultSensor(Sensor.TYPE_STEP_COUNTER);
```

Mit dem SensorManager können Sie auf die Sensoren des Geräts zugreifen.

Um die Schritte erfassen zu können, brauchen wir den Schrittzählersensor anzusprechen. Ein Sensor dieses Typs gibt die Anzahl der Schritte zurück, die der Benutzer ausgeführt hat. Der Wert wird als Gleitkommazahl zurückgegeben (der Bruchteil wird auf null gesetzt). Dieser Sensor ist in Hardware implementiert. Ich verwende dafür den „*TYPE\_STEP\_COUNTER*“ Sensor. Es ist eine Konstante, die einen Schrittzählersensor beschreibt. Es gibt allerdings „*TYPE\_STEP\_DETECTOR*“ auch, um Schritte zu erfassen. *STEP\_COUNTER* ist jedoch genauer und filtert die falschen Meldungen, die z.B. durch schütteln statt Schritte verursacht werden.

#### 8.6.2 onResume() & onPause()

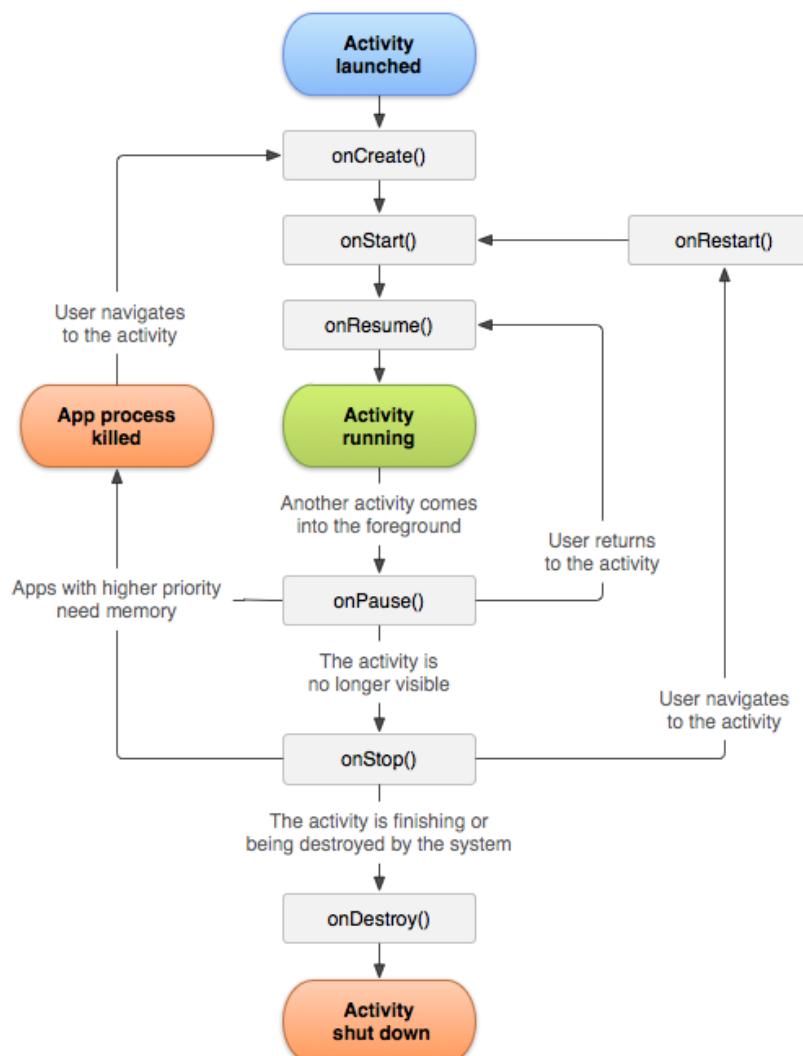
Die Vordergrund-Lifetime einer Aktivität erfolgt zwischen einem Aufruf von „*onResume()*“ bis zu einem entsprechenden Aufruf von „*onPause()*“. Während dieser Zeit steht *activity* vor allen anderen *activities*.

Eine *activity* kann häufig zwischen den Status *resumed* also "Wiederaufnahme" und *Pause* wechseln - beispielsweise, wenn das Gerät in den Ruhezustand versetzt wird, wenn ein *activity result* geliefert wird. Daher sollte der Code in diesen Methoden relativ leicht sein.

„*onPause()*“ ist ein Teil des activity lifecycle und wird aufgerufen, wenn eine activity in den Hintergrund tritt, aber (noch) nicht gekillt wurde. Das Gegenteil zu dieser Methode ist „*onResume()*“.

„*onResume()*“ ist ebenfalls ein Teil des activity lifecycle und wird aufgerufen, wenn Ihre activity mit dem Benutzer interagieren sollte. Dies ist ein guter Ort, um Animationen zu starten, exklusive Geräte wie die z.B. Kamera zu öffnen oder die Sensoren zu benutzen usw.

Android activity lifecycle:



Bildquelle: [https://developer.android.com/guide/components/images/activity\\_lifecycle.png](https://developer.android.com/guide/components/images/activity_lifecycle.png)

Man muss jetzt in „`onResume()`“ (wenn die App verwendet wird) auf dem SensorManager eine Listener für gegebene Sensoren registrieren.

Immer wenn man die App in Vordergrund bringt bzw. startet, Register hinzugefügt, ansonsten (wenn alles im Hintergrund gerückt wird), wird die Zustand „`onPause()`“ aufgerufen und in diesem Zustand möchte ich, dass den Sensor nicht mehr registriert ist.

```

protected void onResume() {
    super.onResume();
    sm.registerListener(this, stepCounter,
SensorManager.SENSOR_DELAY_UI);
}

protected void onPause() {
    super.onPause();
    sm.unregisterListener(this);
}

```

## 8.7 SCHRITTE ÜBER DEN SENSOR ERFASSEN

Die eigentliche Arbeit der Berechnung und Erfassung der Schritte muss man in der Methode „*onSensorChanged()*“ hinterlegen. Man berechnet da also wie viele Schritte da gemacht worden sind. Man muss allerdings beachten, dass es immer die Anzahl der Schritte seitdem letzten Restart von System in dem SensorEvent gespeichert wird. Das heißt beim Allerersten mal Start der App, kann tausende von Schritte, die schon erfasst worden sind, gezeigt werden. Deshalb erstelle ich zwei Hilfsvariablen um diese Problem zu lösen.

```

private int currentSteps = 0;
private int initialSteps = 0;

```

- **currentSteps**: die Anzahl der Schritte, die seitdem Start der App erfasst worden sind.
- **initialSteps**: Die Anzahl der Schritte, die schon in dem SensorEvent seit dem letzten System-reboot gespeichert sind.

Wenn die App zum allerersten Mal gestartet wird, wird erste Value von sensorEvent nämlich Value[0] dem Variable initialSteps zugewiesen. Ansonsten sollte den Unterschied zwischen Value[0] und initialSteps dem currentSteps zugewiesen werden, damit man die Anzahl der Schritte ab dem Zeitpunkt vom Start der App haben kann. Danach wird der Wert von currentSteps angezeigt.

```

if (initialSteps == 0) {
    initialSteps = (int)sensorEvent.values[0];
} else{
    currentSteps = (int)sensorEvent.values[0] - initialSteps;
}

stepDisplay.setText(String.format("%d", currentSteps));

```

## 8.8 START, STOP UND RESET EINBAUEN

Wenn der Benutzer auf eine Schaltfläche klickt, erhält das Button-Objekt ein On-Click-Event. Um den Click-Eventhandler für eine Schaltfläche zu definieren, fügt man das „*android:onClick*“-Attribut zum <Button> -Element in dem XML-Layout hinzu. Der Wert für dieses Attribut muss der Name von der Methode sein, die Sie als Reaktion auf ein Klickereignis aufrufen möchten.

```

<Button
    android:id="@+id/startStopBtn"
    style="@android:style/Widget.Button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_weight="1"
    android:elevation="0dp"
    android:onClick="startStop"
    android:text="@string/start"
    android:visibility="visible" />

```

Ich erstelle zunächst eine Hilfsvariable vom Typ Boolean und setze ich sie auf fasle. Dann erstelle ich die Handler für meine Zwei Buttons, Start/Stop und Reset-Button.

Beim Start/Stop Button, muss man dafür sorgen, dass jedes Mal, wenn man auf dem Start/Stop Button drückt, ändert sich den Text, also wenn auf dem Button „Start“ steht sollte mit dem Drucken vom Button sich zum „Stop“ ändern und umgekehrt. Dafür verwende ich die Hilfsvariable „*isCounting*“. Über „*SetText()*“ und eine if-statement sorgt man dafür, dass den richtigen Text auf dem Button steht.

```
isCounting = !isCounting;
```

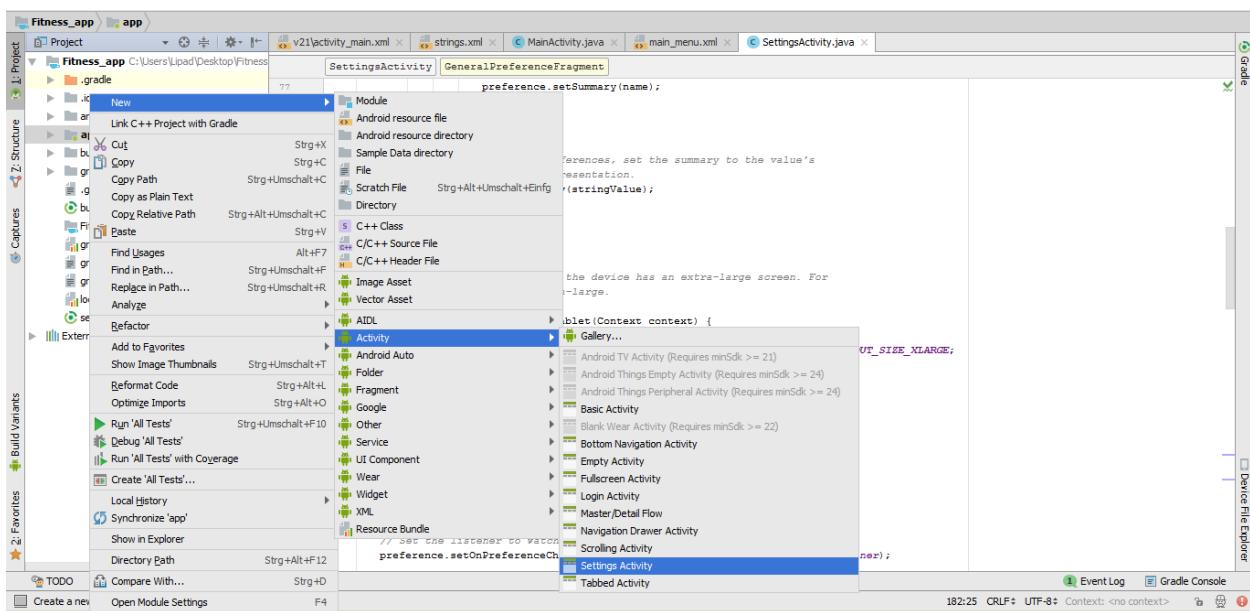
Beim Drucken von Reset-Button werden alle Variablen zurückgesetzt, zum Beispiel wird „*isCounting*“ auf False oder *initialSteps* auf „0“ gesetzt usw. Um den Code übersichtlicher zu machen, implementiere ich den Start/Stop umdreh-funktion als eine extra Methode.

```
public void reset(View view) {  
  
    stepDisplay.setText("0");  
    isCounting = false;  
  
    switchBtn();  
  
    initialSteps = 0;  
    currentSteps = 0;  
}  
  
  
public void startStop(View view) {  
  
    isCounting = !isCounting;  
    switchBtn();  
  
}  
  
  
public void switchBtn() {  
    if (isCounting){  
        startStopBtn.setText(R.string.switchStop);  
  
        StartTime = SystemClock.uptimeMillis();  
  
    }else{  
        startStopBtn.setText(R.string.switchStart);  
    }  
}
```

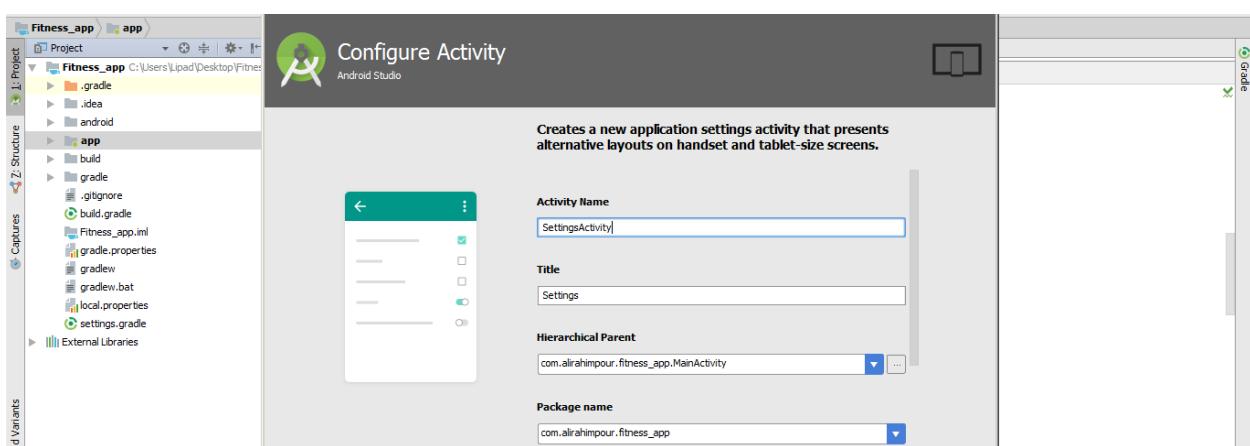
## 8.9 DIE SETTINGSACTIVITY

Apps enthalten häufig Einstellungen, mit denen Benutzer App-Funktionen und Verhaltensweisen ändern können. In einigen Apps können Benutzer beispielsweise angeben, ob Benachrichtigungen aktiviert sind, oder angeben, wie oft die Anwendung Daten mit der Cloud synchronisiert. In meinem Fall, der Benutzer soll in der Lage sein, eigene Schrittänge und Verbrauchte Kalorien pro Kilometer selbst eingeben, damit die Entfernung und die verbrauchte Kalorie genauer berechnet werden können.

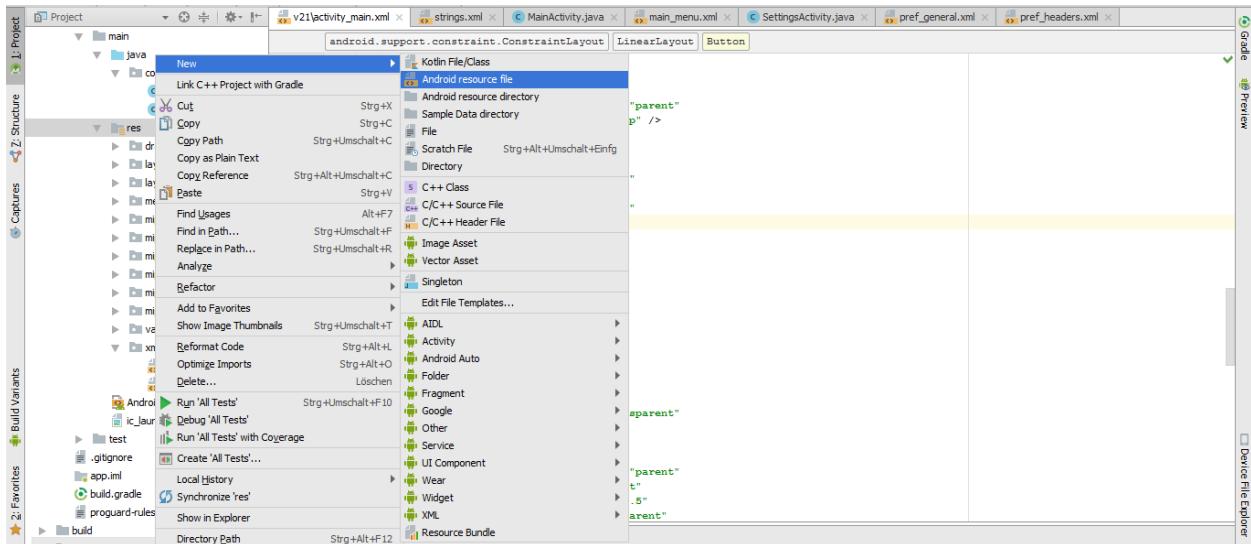
Dafür erstelle ich zunächst eine leere Setting Activity, wie folgt:



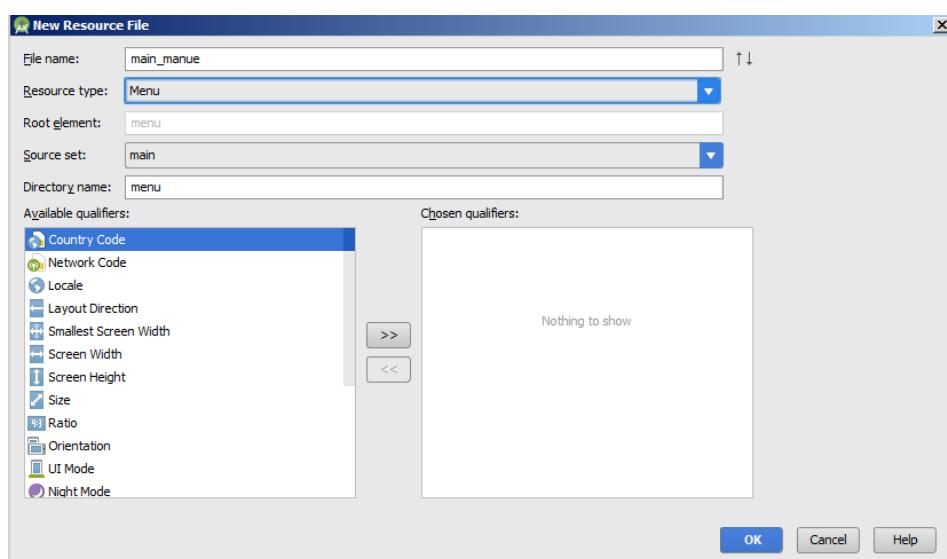
Dann wählen wir aus der „Hierarchical Parent“ „MainActivity“.



Es werden SettingActivity und einige Einstellungs Beispiele automatisch im Ordner „res -> xml“ generiert. Bevor ich die automatisch generierte SettingActivity ausdüne, erstelle ich eine menü, indem ich mit dem rechten maustaste auf dem Verzeichnis „res“ klicke und aus der menü in „new“, „Android resource File“ wähle.



Dann gebe ich eine „File name“ und „resource type“.



### 8.9.1 Menü

Danach muss man nur Menu- und Item-element als im Code definieren. Item-element erstellt ein MenuItem, das ein einzelnes Element in einem Menü darstellt. Dieses Element kann ein verschachteltes <menu> -Element enthalten, um ein Untermenü zu erstellen.

Menu-element Definiert ein Menü, das ein Container für Menüelemente ist. Ein <menu> -Element muss der „root node“ (Wurzelknoten) für die Datei sein und ein oder mehrere <item>- und <group>- Elemente enthalten.

Mein Menü hat nur ein Item „Einstellungen“. Der Name ist in der Datei string.xml gespeichert.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      tools:context=".MainActivity">

    <item
        android:id="@+id/settingsItem"
        android:title="@string/settings"
        android:icon="@android:drawable/ic_menu_edit"

    />

</menu>
```

In MainActivity muss man das ganze einbinden und dafür sorgen, dass das Menü initialisiert wird. Dafür braucht man zwei Methode.

- **onCreateOptionsMenu:**  
Initialisiert die Menüelemente anhand „*MenuInflater*“. Diese Klasse (MenuInflater) wird verwendet, um Menu-XML-Dateien in Menu-Objekte zu instanzieren.
- **onOptionsItemSelected:**  
Es wird aufgerufen, wenn ein Element im Optionsmenü ausgewählt ist.

```

public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.main_menu, menu);

    return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
    if (item.getItemId() == R.id.settingsItem) {
        Intent intent = new Intent(this, SettingsActivity.class);
        startActivity(intent);
    }

    return true;
}

```

### 8.9.2 Einstellungsoptionen ausdünnen

Da ich die automatisch generierten Strukturen so umständlich und umfangreich gar nicht brauche, dünne ich es ein bisschen aus. Ich nehme alles außer Header-Element „General“ aus der Datei raus.

Da die Dateien in der Android Struktur miteinander verbunden sind, sollte man beim Löschen das Häkchen „safe delete“ machen, damit möglichst unerwartete Fehlermeldungen verhindert werden.

Die gelöschte Header-Elemente wurden im „SettingsActivity“ verwendet, deshalb muss ich die Datei auch bearbeiten und die entsprechende Code-teile löschen.

In der Datei „General.xml“ muss auch alles außer zwei „EditTextPreference“-Tag gelöscht werden. man braucht diese, um die Werte von Schrittlänge und Kalorie pro km erfassen zu können. Danach werden die Eigenschaften von diese Tags angepasst. Zum Beispiel „key“, „Title“ oder „DefaultValue“.

„DefaultValue“ Legt man fest um sicher zu stellen, dass wenn vom Benutzer keine Informationen über Schrittlänge und/oder Kalorie pro km eingegeben wird, es zu verhindern, dass die App abstürzt, deshalb gibt man die Durchschnittswerte als „Default-Values“ ein.

```
<EditTextPreference  
    android:capitalize="words"  
    android:defaultValue="70"  
    android:inputType="number"  
    android:key="Schrittlänge"  
    android:maxLines="1"  
    android:selectAllOnFocus="true"  
    android:singleLine="true"  
    android:title="Schrittlänge" />  
  
<EditTextPreference  
    android:capitalize="words"  
    android:defaultValue="320"  
    android:inputType="number"  
    android:key="Kalorien"  
    android:maxLines="1"  
    android:selectAllOnFocus="true"  
    android:singleLine="true"  
    android:title="Cal pro km" />
```

## 8.10 DIE KALORIEN & METER BERECHNEN

Ich baue eine Methode, in der die Werte von Entfernung und verbrachte Kalorien berechnet werden und füge sie zu „`onSensorChanged()`“ (sobald die Sensor Werte sich ändern wird diese Methode aufgerufen und die Werte werden neu berechnet).

Die Entfernung ergibt sich von der Schrittlänge, indem man sie mit den zurückgelegten Schritten (CurrentSteps) multipliziert. Die Kalorie wird berechnet, indem man die berechnete Entfernung durch 1000 dividiert und das Ergebnis mit dem Wert von verbrauchten Kalorien multipliziert.

Um auf den gespeicherte Einstellungswerte zugreifen zu können, müssen wir die „`SharedPreferences`“ auslesen. Die eingegebenen Einstellungen werden hier gespeichert. Und über „`Keys`“ können wir bestimmte Einstellungen ansprechen und eingegebene Werte auslesen.

```

public void updateKalMeter() {

    SharedPreferences sharedpreferences =
    PreferenceManager.getDefaultSharedPreferences(this);
    int schritt = new Integer(sharedpreferences.getString("Schrittlänge",
    "70"));
    double schrittlänge = schritt / 100.0;
    int meter = (int)(currentSteps * schrittlänge);

    meterDisplay.setText(String.format(US, "%d m", meter));

    // 320cal auf 1000m
    int kalPerKm = new Integer(sharedpreferences.getString("Kalorien", "320"));
    int kalorien = (int)((meter / 1000.0) * kalPerKm);
    kalDisplay.setText( String.format(US, "%d Kal", kalorien));

}

```

## 8.11 PROBLEM BEIM SCREEN ROTATION

Bei der Rotation des Bildschirms wird „Activity“ zerstört und wiederaufgebaut, deshalb werden die ganzen Referenzen und Variablen bei der Rotation zurückgesetzt. Die Variablen, die in diesem Fall wichtig sind und deren Werte gemerkt werden sollte, sind „CurrentSteps“, „initialSteps“ und „isCounting“. Android Studio bietet die Möglichkeit, dieses Problem beinahe automatisch zu lösen.

In der Methode onSaveInstanceState() werden die Werte der drei Variablen persistent gespeichert und danach werden sie in der Methode onRestoreInstanceState() wieder rausgeholt. So werden die Daten bei der Rotation nicht verloren gehen.

```

protected void onSaveInstanceState(Bundle outState) {
    outState.putInt("currentSteps", currentSteps);
    outState.putInt("initialSteps", initialSteps);
    outState.putBoolean("isCounting", isCounting);
    super.onSaveInstanceState(outState);
}

protected void onRestoreInstanceState(Bundle savedInstanceState) {
    currentSteps = savedInstanceState.getInt("currentSteps");
    initialSteps = savedInstanceState.getInt("initialSteps");
    isCounting = savedInstanceState.getBoolean("isCounting");
    switchBtn();
    super.onRestoreInstanceState(savedInstanceState);
}

```

## 8.12 STOPPUHR

Stoppuhr ist eine der wichtigsten und nützlichsten Tools für Personen. Eine Stoppuhr in meiner App kann sehr hilfreich sein. Die Daten werden nicht analysierbar sein, wenn man nicht weiß, in welchem Zeitabschnitt die Daten erfasst wurden. Das macht einen großen Unterschied, ob man 10000 Schritte innerhalb 3 Tage oder innerhalb einer Stunde geschafft hat.

Im Internet gibt es zahlreiche Codebeispiele für die Stoppuhr. Ich habe die Stoppuhr aus folgendem Tutorial geändert und in meinem Code integriert.

<https://www.android-examples.com/android-create-stopwatch-example-tutorial-in-android-studio/>

```
TextView textView ;
long MillisecondTime, StartTime, TimeBuff, UpdateTime = 0L ;
Handler handler;
int Seconds, Minutes, MilliSeconds ;
```

```
public Runnable runnable = new Runnable() {

    @SuppressLint("SetTextI18n")
    public void run() {

        MillisecondTime = SystemClock.uptimeMillis() - StartTime;

        UpdateTime = TimeBuff + MillisecondTime;

        Seconds = (int) (UpdateTime / 1000);

        Minutes = Seconds / 60;

        Seconds = Seconds % 60;

        MilliSeconds = (int) (UpdateTime % 1000);

        textView.setText(String.format(Locale.GERMANY, "%d:%s:%s.%s",
            Minutes, getString(R.string.Colon), String.format(Locale.GERMANY,
            "%02d", Seconds), getString(R.string.Colon),
            String.format(Locale.GERMANY, "%03d", MilliSeconds)));

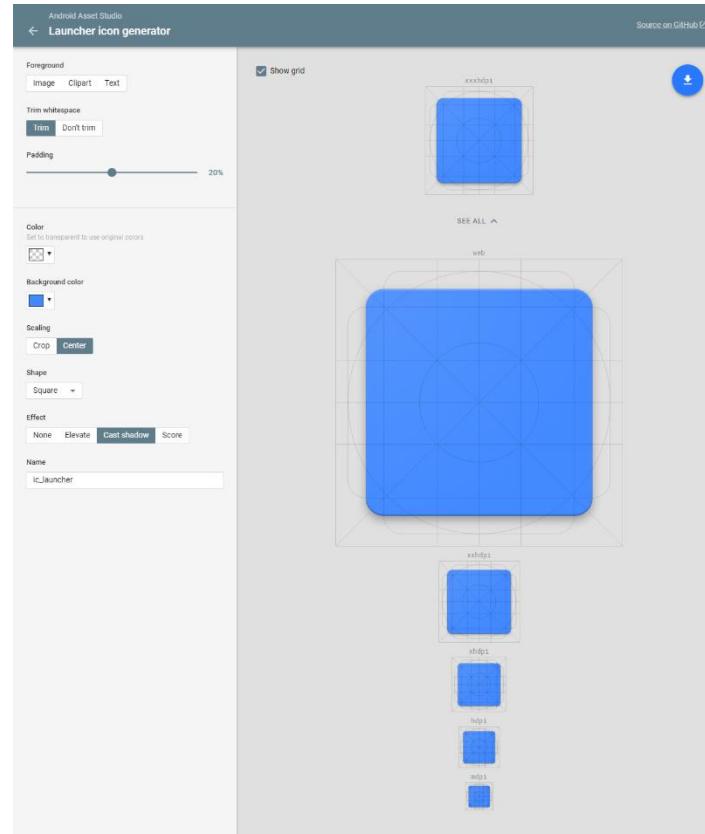
        handler.postDelayed(this, 0);
    }
};
```

## 8.13 ICON

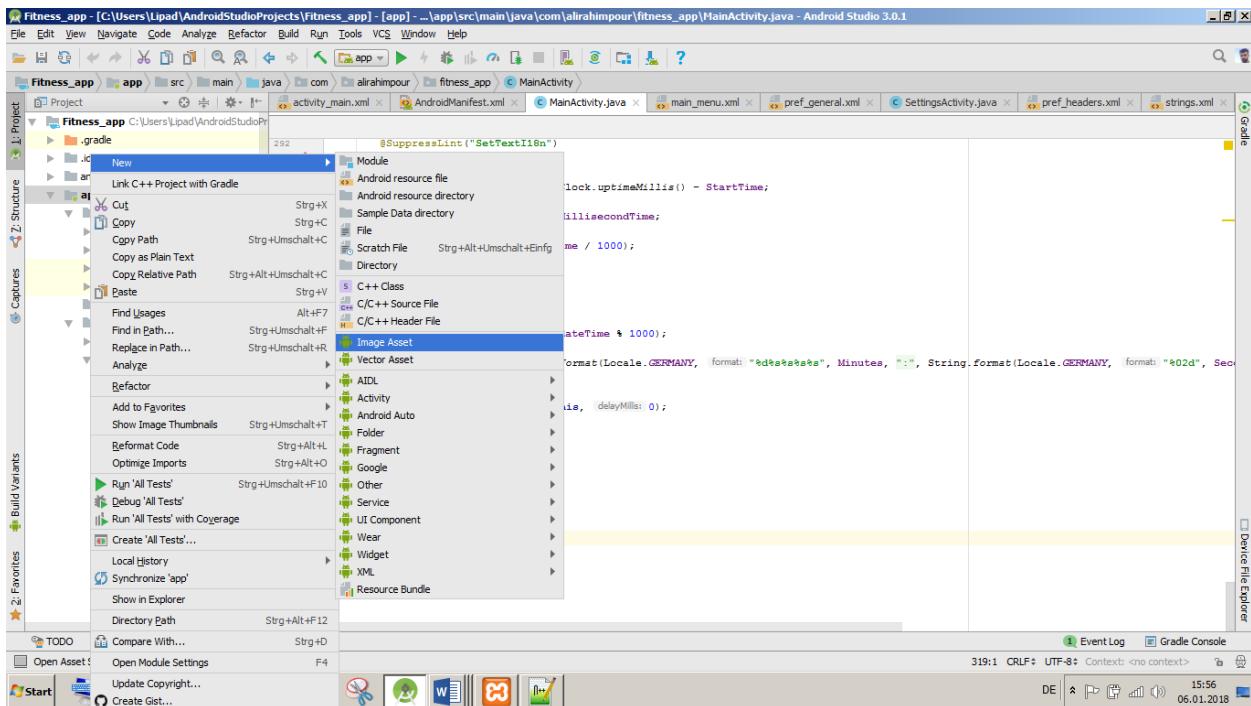
Das Icon ist zu Beginn ein einfaches Android Logo. Das Icon soll nicht verspielt sondern seriös wirken, daher wird dieses zum Schluss der App – Entwicklung durch ein anderes Icon ersetzt. Das Icon habe ich wie bei der ersten Aufgabe (7.1.6), mithilfe von „Android Asset Studio“ konstruiert.

<https://romannurik.github.io/AndroidAssetStudio/>

„Android Asset Studio“ ist eine Sammlung von Online-Tools zum einfachen Generieren von Assets wie Launcher-Symbole für die Android-App.



Um das Icon anzupassen wird mit einem Rechtsklick auf „App“, links in dem Menü „Project“ geklickt. Anschließend wird unter dem Menüpunkt „new“ der Menüpunkt „Image Assets“ ausgewählt. Jetzt kann das gewünschte Bild aus dem jeweiligen Ordner ausgewählt und als Icon hinzugefügt werden.



## 8.14 ANDERE AUFGETRETENE PROBLEME

### 8.14.1 Problem 1

Absturz von der App wegen „textView.setText()“.

**Lösung:** Formatierte String verwenden (String.format(Locale, String, Object...))

```
textView.setText(String.format(Locale.GERMANY, "%d%s%s%s", Minutes,
getString(R.string.Colon), String.format(Locale.GERMANY, "%02d",
Seconds), getString(R.string.Colon), String.format(Locale.GERMANY,
"%03d", Milliseconds)));
```

Quellen:

<https://stackoverflow.com/questions/36604835/string-format-and-locale-problems-android>

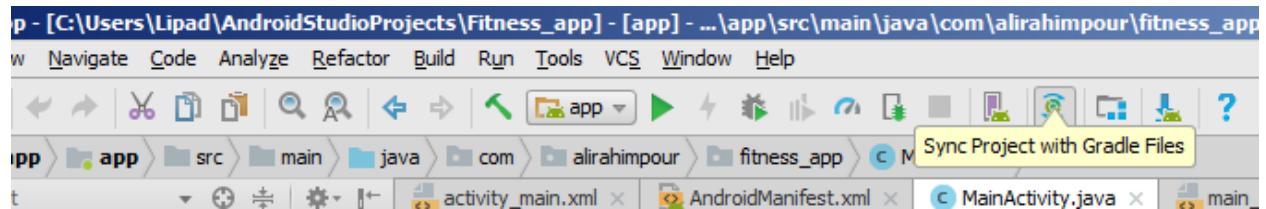
<https://github.com/couchbase/couchbase-lite-java-core/issues/1260>

<https://developer.android.com/reference/java/util/Locale.html>

### 8.14.2 Problem 2

In allen meinen Klassen, wo ich auf den Ressourcen referenziert habe, war das „R“ in Rot und es stand "cannot resolve symbol R".

**Lösung:** Ich habe das Projekt aufgeräumt und mit Gradle synchronisiert, indem ich auf dem Gradle-Symbol im gedrückt habe.



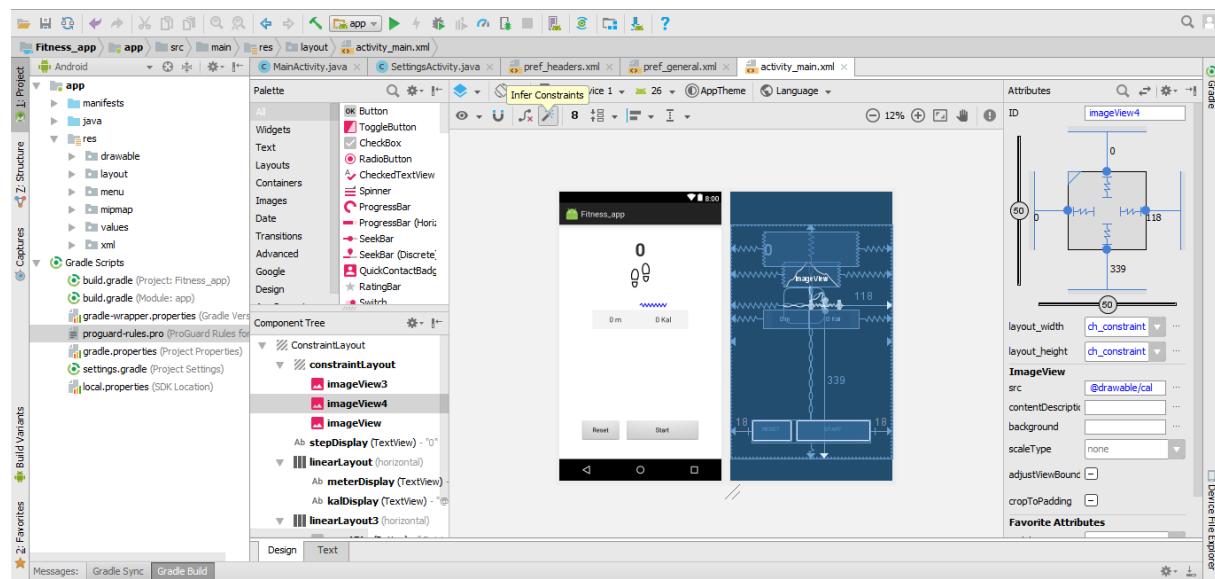
Quelle :

<https://stackoverflow.com/questions/17054000/cannot-resolve-symbol-r-in-android-studio>

### 8.14.3 Problem 3



**Lösung:** man muss das dem Zauberstab-symbol „infer constraints“ drücken. Es wechselt das „RelativeLayout“ zu einem „ConstraintLayout“.



Mit „*ConstraintLayout*“ kann man große und komplexe Layouts mit einer flachen hierarchie-Ansicht erstellen (keine verschachtelten Ansichtsgruppen).

Quelle :

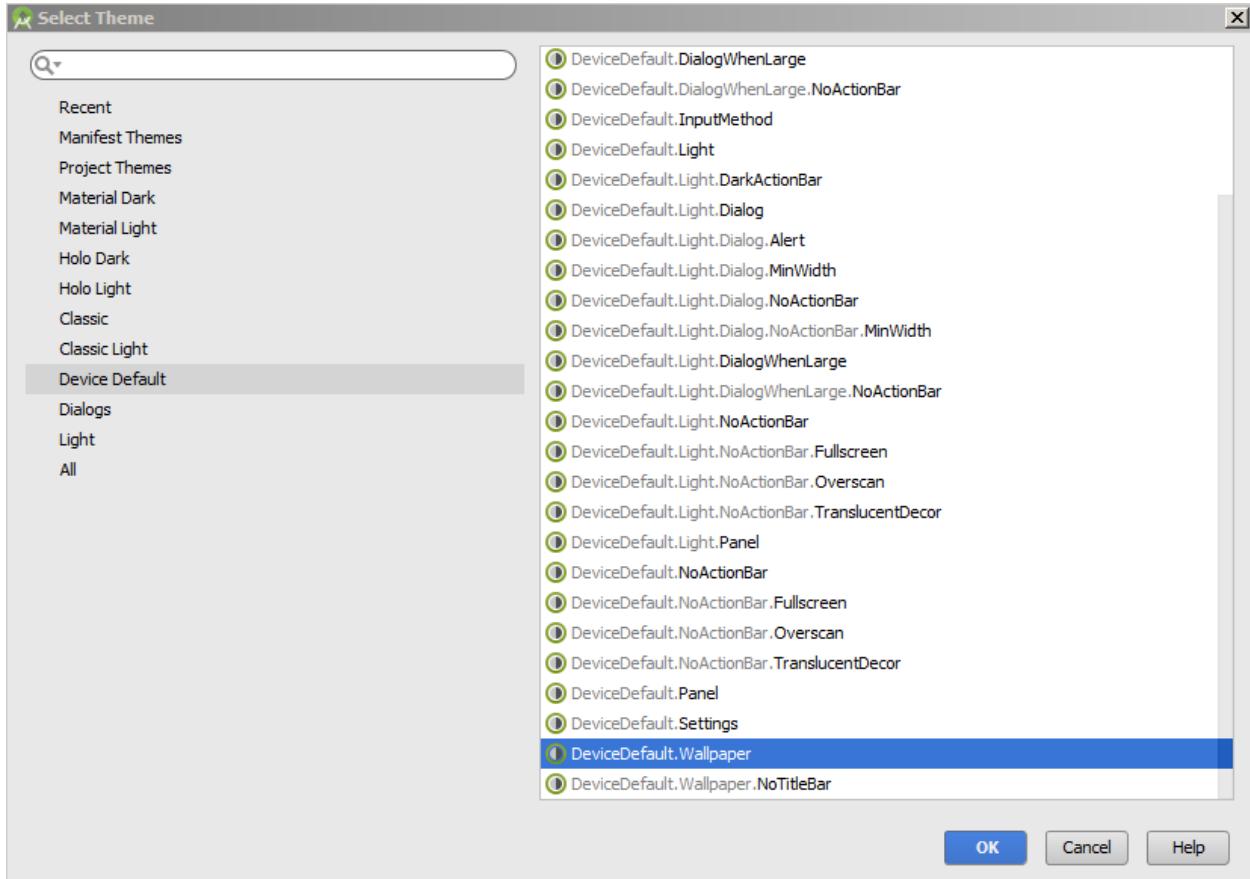
<https://www.youtube.com/watch?v=uOur51u5Nk0>

#### 8.14.4 Problem 4

Ich benutze Android Studio 3.0.1 auf Windows 7 als ich Das Programm mit Aktualisiert habe, funktionierte der ConstraintLayout-Editor nicht mehr. Es funktionierte jedoch, wenn ich den Code auf Emulator / Gerät ausgeführt habe, aber für die Design- und Vorschau-Tabs wurde es nicht angezeigt.



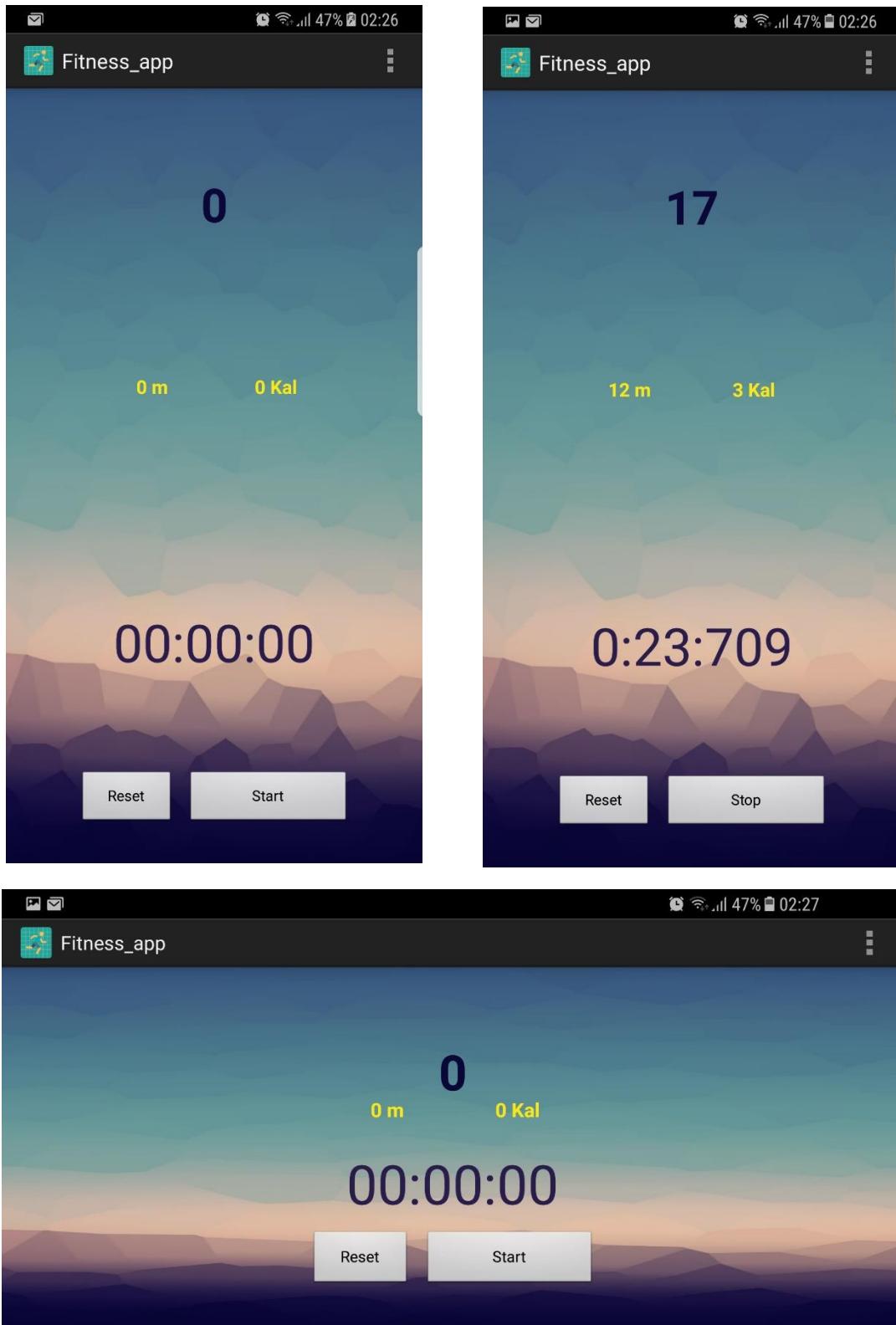
**Lösung:** Rendering Problems: Man muss einfach das App-Thema Ändern.

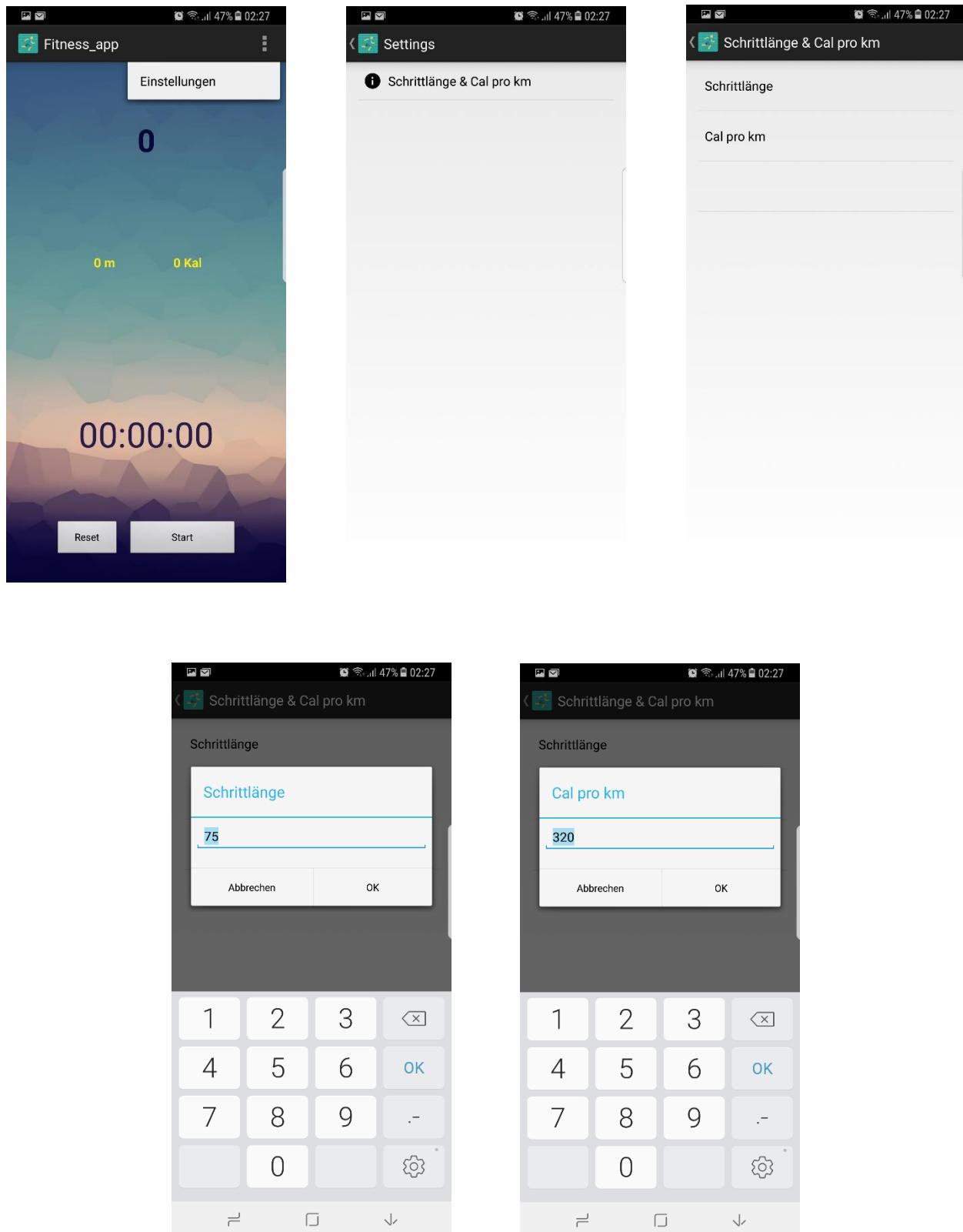


Quelle :

<https://www.youtube.com/watch?v=4MxBnwpcUjA>

## 8.15 DIE APP ÜBERSICHT & VIDEO





Eine kurze Videoaufnahme von der App können Sie sich über den YouTube-Link oder QR-Code ansehen:

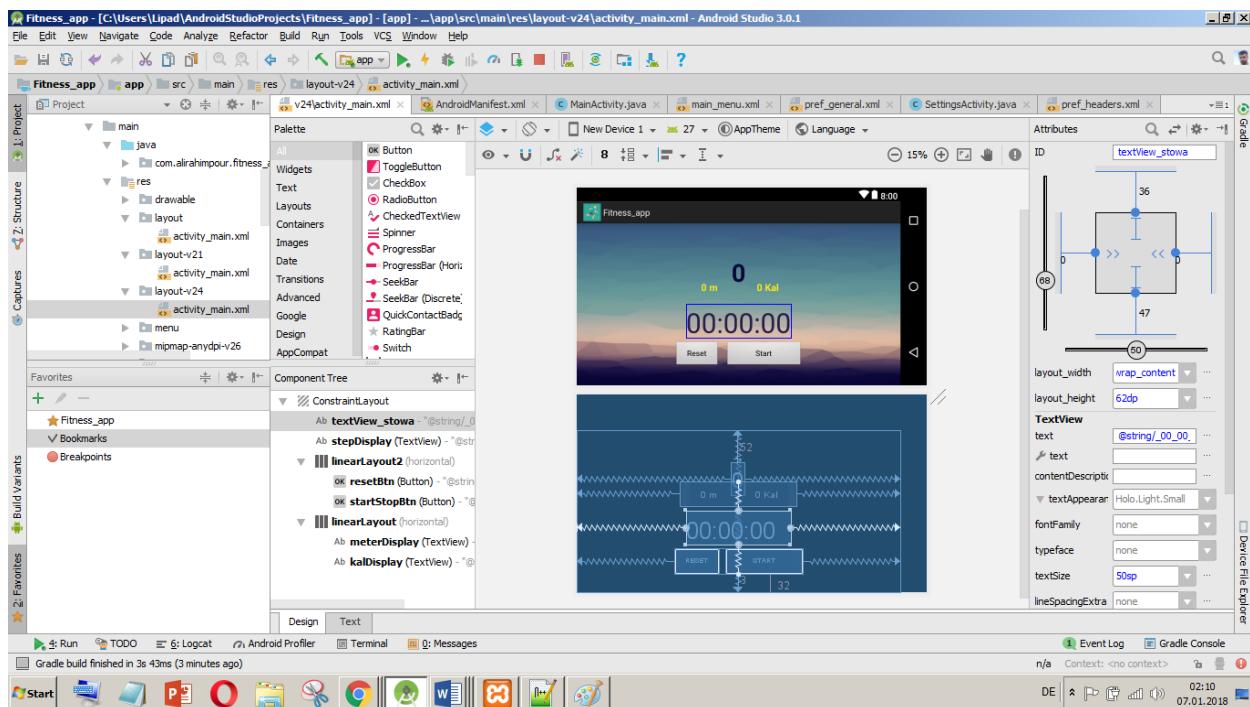
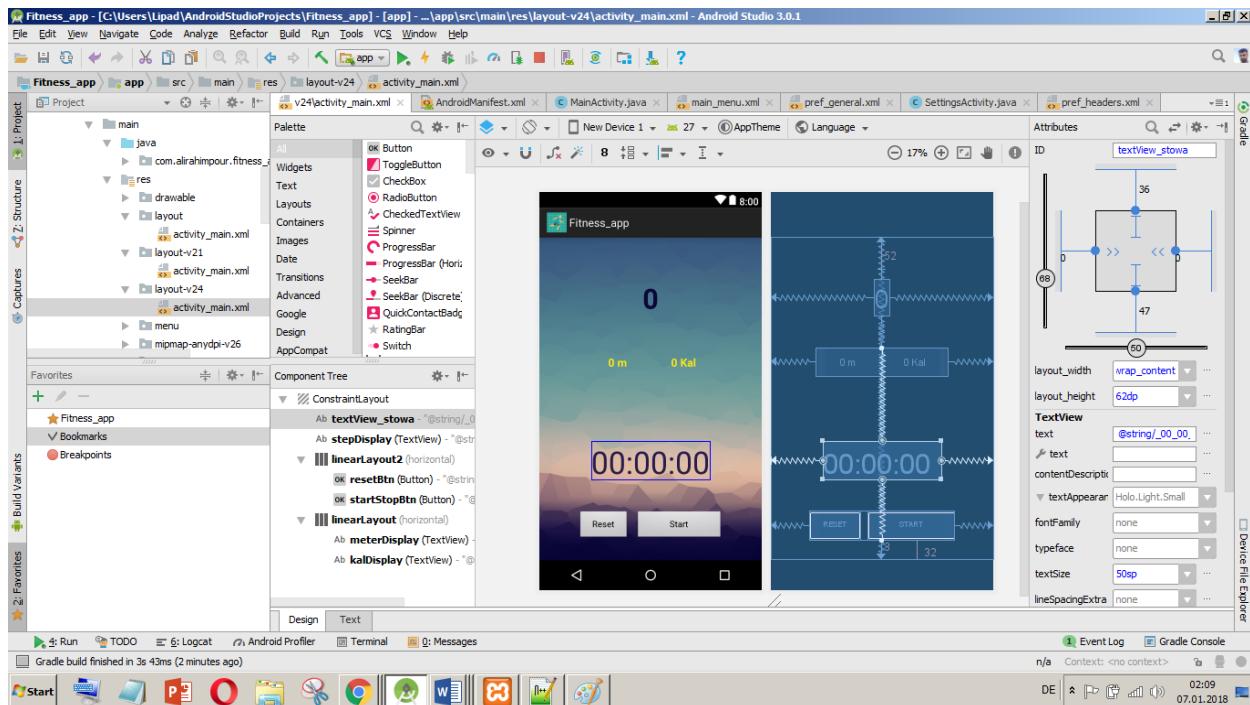


## 9 FAZIT

Das wesentliche Ergebnis dieses Projekts für mich ist die Tatsache, dass ich mich noch lange mit dem Thema App-Entwicklung beschäftigen muss, damit ich Professionelle Apps Entwickeln kann und dass, die App-Entwicklung-Prozess nicht nur aus Code schreiben besteht, sondern gibt es Viele andere Bereiche, wie Design oder Planung, die man genauso gut kann, wie das Codieren.

Ich habe jedoch Vieles Gelernt, indem ich ständig mit den Problemen und Herausforderungen der Entwicklung einer App konfrontiert war.

## 10 PROJEKT ÜBERSICHT



## 11 QUELLENVERZEICHNIS

1. <https://developer.android.com/sdk/index.html>
2. <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
3. <https://www.youtube.com/watch?v=ZLcozYGkxI0>
4. <http://stackoverflow.com/questions/22573299/how-can-i-change-the-text-with-a-button-click-android-studio-xml>.
5. <http://android4beginners.com/2013/07/lesson-1-3-how-to-change-a-color-of-text-and-background-in-textview/>
6. <https://shanelly.com/2011/12/android-3-x-and-4-x-numberpicker-example/>
7. <https://www.youtube.com/watch?v=NHXa96-r8TY>
8. <https://romannurik.github.io/AndroidAssetStudio/>
9. <https://www.youtube.com/watch?v=SDKwNh0TioE>
10. <https://developer.android.com/reference/android/app/Activity.html>
11. <https://shanelly.com/2011/12/android-3-x-and-4-x-numberpicker-example>
12. <https://romannurik.github.io/AndroidAssetStudio/index.html>
13. <https://www.youtube.com/watch?v=Q03nQVEs44Y>
14. <https://developer.android.com/reference/android/hardware/Sensor.html>
15. <http://www.zeit.de/digital/mobil/2014-05/smartphone-sensoren-iphone-samsung>
16. [https://www.syssec.rub.de/media/emma/arbeiten/2013/02/18/ba\\_Gabel\\_Andreas.pdf](https://www.syssec.rub.de/media/emma/arbeiten/2013/02/18/ba_Gabel_Andreas.pdf)

17. [https://developer.android.com/guide/topics/sensors/sensors\\_overview.html](https://developer.android.com/guide/topics/sensors/sensors_overview.html)
18. <https://www.youtube.com/watch?v=MwH0z1Hlxog>
19. <https://www.youtube.com/watch?v=CNGMWnmldaU>
20. <https://individuapp.com/softwareentwicklung/android-kurs/android-layout-klassen/>
21. <https://developer.android.com/reference/android/hardware/SensorListener.html>
22. <https://developer.android.com/reference/android/app/Activity.html>
23. [https://developer.android.com/reference/android/app/Activity.html#onPause\(\)](https://developer.android.com/reference/android/app/Activity.html#onPause())
24. <https://developer.android.com/guide/topics/ui/settings.html>
25. <http://www.programmierenlernenhq.de/tutorial-android-settings-preferences-und-einstellungen/>
26. <https://developer.android.com/guide/topics/ui/menus.html>
27. <https://developer.android.com/reference/android/preference/EditTextPreference.html>
28. <https://www.uni-trier.de/fileadmin/urt/doku/android/android.pdf>
29. <https://www.android-examples.com/android-create-stopwatch-example-tutorial-in-android-studio/>