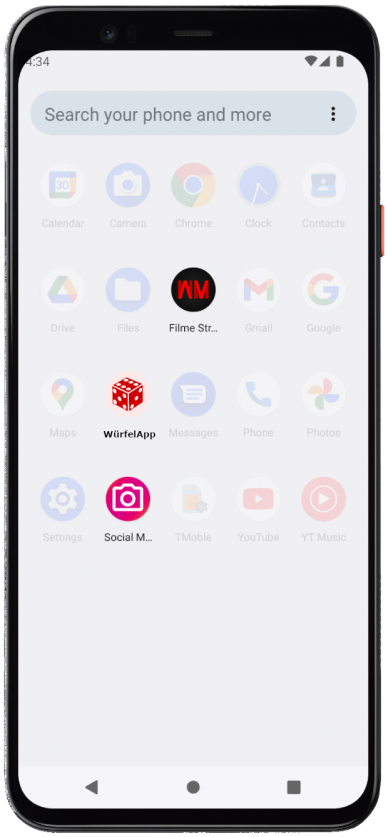
Kotlin 2.4 - App Icon

Aufgaben

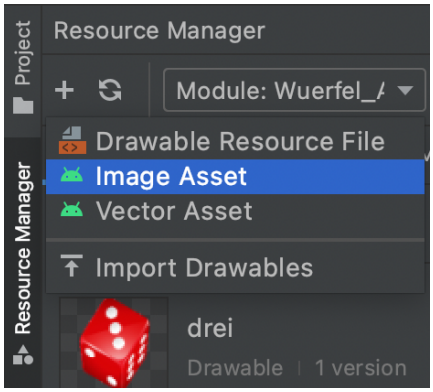
Hinweis: Zu bearbeiten sind Aufgaben 1 und 2

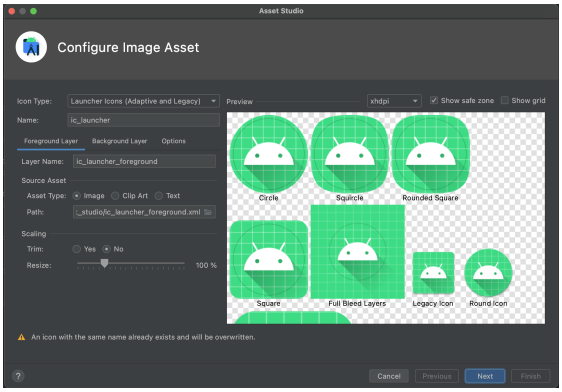
1. CUSTOM ICONS - Icons

In dieser Aufgabe verpassen wir den drei uns bekannten Apps “Filme Streaming Service”, “WürfelApp” und “Social Media Profil” jeweils ein Icon.

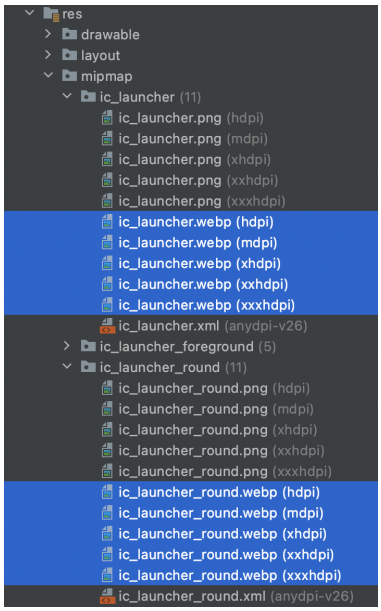


- Das Icon einer App stellt man mit dem Asset Tool ein
    - Das Tool öffnet sich, wenn du im Resource Manager ein neues Bild hinzufügst





Hinweis: Wenn du beim Ausführen der App eine Fehlermeldung mit “Duplicate resources” erhältst, musst du die im “mipmap” Ordner gedoppelten Dateien mit der “.webp” Endung löschen:

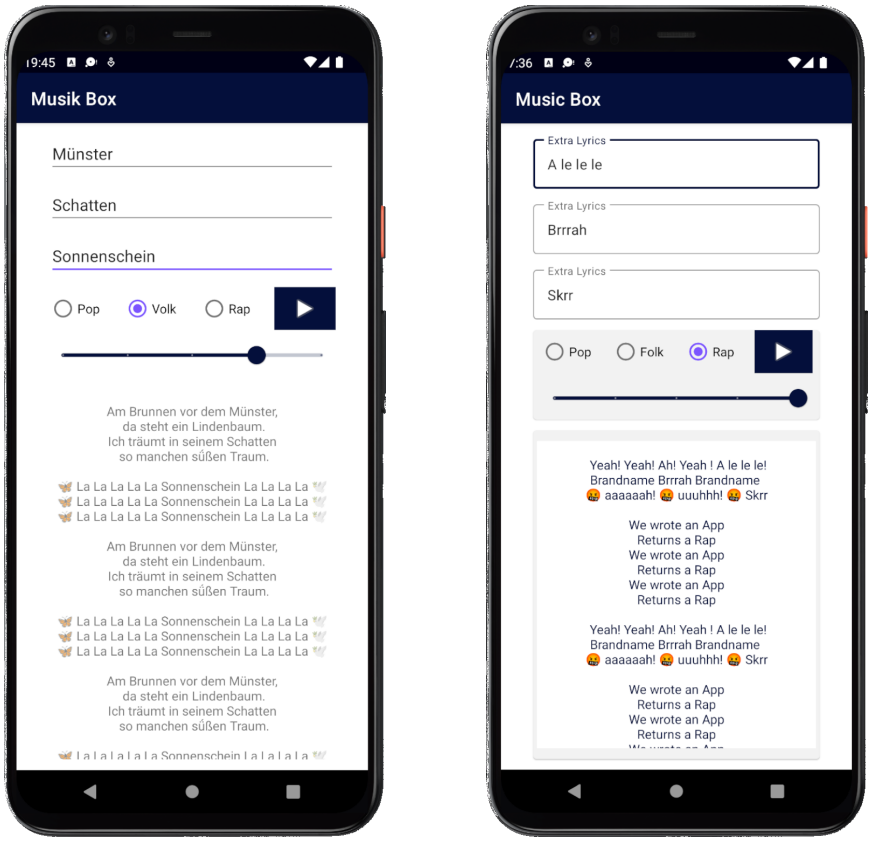


- Öffne das Projekt “Wuerfel App” und gib der App ein Icon
    - Im Ordner “ICONS” findest du hierfür die Vektorgrafik “ic_wuerfel_foreground”
    - Als Icon Hintergrund soll eine einfache Farbfläche mit einer Farbe deiner Wahl eingestellt werden
    - Starte die App im Emulator und wirf einen Blick in das App-Menü. Überprüfe, ob sich das Icon geändert hat
- Öffne das Projekt “Filme Streaming Service” und gib der App ein Icon
    - Im Ordner “ICONS” findest du hierfür die .png Dateien “ic_watchme_foreground” und “ic_watchme_background”
    - Starte die App im Emulator und wirf einen Blick in das App-Menü. Überprüfe, ob sich das Icon geändert hat
- Öffne das Projekt “Social Media Profil” und gib der App ein Icon
    - Im Ordner “ICONS” findest du hierfür die .png Datei “ic_socialmedia_background”
    - Als Icon Vordergrund soll das Kamerasymbol “camera alt” (Outlined) aus der Clipart Bibliothek ausgewählt werden und in weiß eingefügt werden
    - Starte die App im Emulator und wirf einen Blick in das App-Menü. Überprüfe, ob sich das Icon geändert hat

Viel Erfolg! 

2. Musik Box - Material Components

Wir wollen das Design der App Oberfläche mithilfe von Material Components auf das nächste Level bringen.



- Öffne das Projekt “Musik Box” und die Datei “activity_main.xml”
- Ersetze die drei `EditText` Komponenten mit `TextInputLayout` Komponenten und übernehme Eigenschaften wie `margin`, `hint` …
    - jedes `TextInputLayout` enthält jeweils ein `TextInputEditText`, welches vor allem die gleiche ID wie die vorherige `EditText` Komponente hat, hier siehst du ein Beispiel:

```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/firstTextLayout"
    style="@style/Widget.MaterialComponents.TextInputLayout.OutlinedBox"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginHorizontal="36dp"
    android:layout_marginTop="12dp"
    android:hint="@string/extra_lyrics"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <com.google.android.material.textfield.TextInputEditText
        android:id="@+id/firstText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

</com.google.android.material.textfield.TextInputLayout>
```

- Die `RadioGroup` und der `Slider` werden zusammen in ein vertikales “LinearLayout” verpackt, welches wiederum in einer `MaterialCardView` enthalten ist
    - Ändere die Hintergrundfarbe der `MaterialCardView` auf die Farbe `backgroundL1` aus `colors.xml`
    - setze vernünftige Constraints und Abstände
- Verpacke anschließend noch die TextView in eine `MaterialCardView` und ändere die Schriftfarbe der TextView auf `SI_DeepPurple` aus `colors.xml`
    - setze auch hier jeweils sinnvolle Constraints und Abstände und ändere den Hintergrund der `MaterialCardView` auf `backgroundL1` aus `colors.xml`
- Führe die App aus und überprüfe, ob sie so aussieht, wie du es dir vorgestellt hast

Viel Erfolg!