31.8.–3.9.2015
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

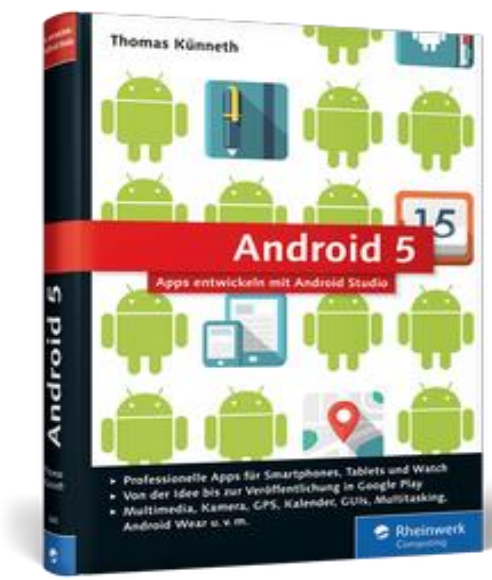
Wer hat an der Uhr gedreht?

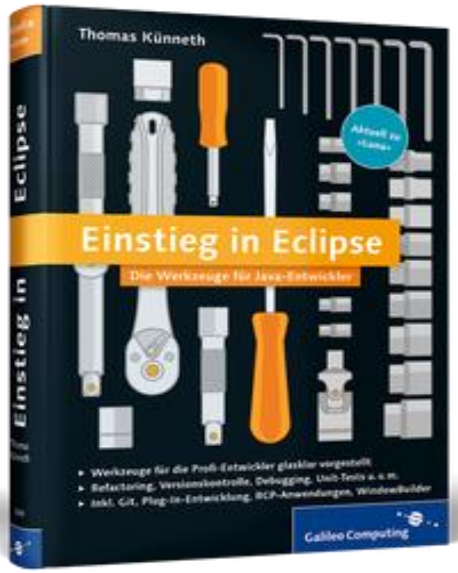
App-Entwicklung für Android Wear

Thomas Künneth, M.A.

MATHEMA Software GmbH

Über mich

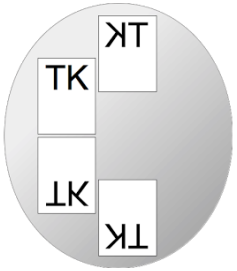

Thomas Künneth
Android 5
Apps entwickeln mit Android Studio
Android 5
Rheinwerk Computing


Thomas Künneth
Einstieg in Eclipse
Die Werkzeuge für Java-Entwickler
Eclipse
Galileo Computing


Thomas Künneth
Java für Windows
Sold out
Galileo Computing


29
Aug


11:03 AM


TK
TK
TK
TK

Was ist Android Wear?

- Android-Version speziell für Smartwatches und ggf. weitere Wearables (aktuell: 5.1.1)
- wurde am 18. März 2014 angekündigt (einschl. Entwickler-Vorschauversion)
- am 25. Juni 2014 wurden auf der Google I/O die Geräte Samsung Gear Live und LG G Watch vorgestellt und konnten im Play Store vorbestellt werden
- Mitte 2015 ca. 10 Modelle verfügbar

Foto © Thomas Künneth

Merkmale

- Steuerung über Spracherkennung, Wischgesten und Handbewegungen
- je nach Modell unterschiedlich viele Sensoren (z. B. Schrittzähler, Messung der Herzfrequenz, Kompass, Gyroskop)
- Vibrationsmotor (wird z. B. für Benachrichtigungen genutzt)
- unterstützt unterschiedliche Anzeigegrößen und Gehäuseformen

- derzeit kein Lautsprecher
- derzeit keine Telefoniefunktionen

Nutzungsvoraussetzungen unter iOS

- Support von iOS wurde am 31.08.2015 angekündigt
- für iPhone 5, 5c, 5s, 6, 6 Plus
- iOS 8.2 oder neuer

- offiziell werden nur aktuelle Uhrenmodelle unterstützt
- es scheinen aber auch ältere verwendet werden zu können

- Companion App aus App Store

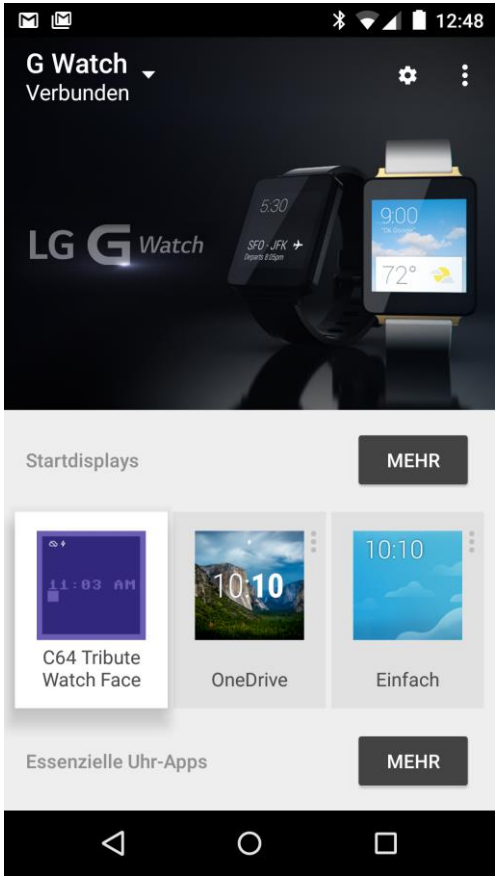
Nutzungsvoraussetzungen unter Android

- benötigt Smartphone oder Tablet mit Android 4.3 oder aktueller
- Verbindung über Bluetooth LE und WI-FI
- Companion App aus Google Play

Companion App

- Verbindungen verwalten
- Bestimmte Geräteeinstellungen
- Aktionen Apps zuweisen
- Uhr-Screenshots erstellen

- Welche Funktionen die App unter iOS bietet, ist derzeit (Stand 01.09.2015) noch nicht ganz klar



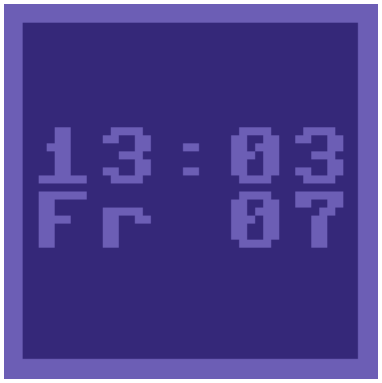
Anhängsel

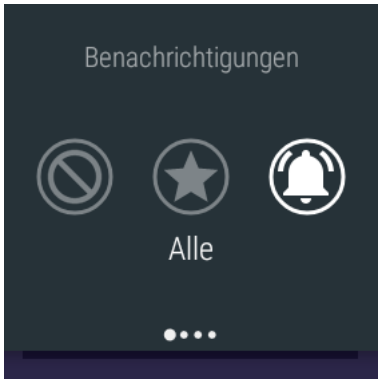
Permanenter Betrieb ohne Smartphone/Tablet und Companion App derzeit nicht praktikabel

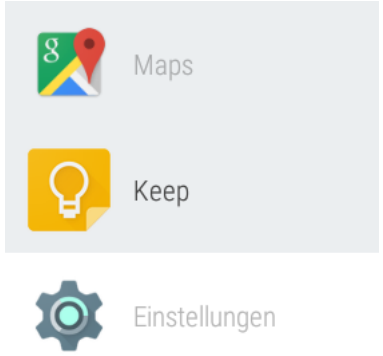
Benutzeroberfläche

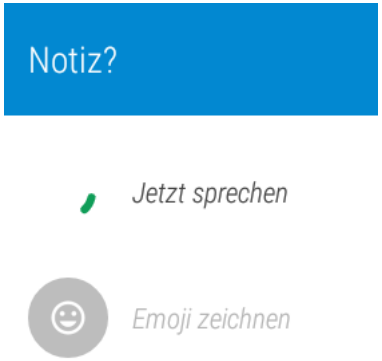
- Üblicherweise zeigt die Smartwatch das änderbare Zifferblatt (Startdisplay)
- Es kann von einem so genannten Kontextstrom ganz oder teilweise verdeckt werden
- Das Verhalten der Uhr kann durch Wischen vom oberen Bildschirmrand nach unten konfiguriert werden
  - Anzuzeigende Benachrichtigungen
  - Kinomodus
  - Sonnenlicht-Modus
- Antippen öffnet mehrseitigen Anwendungsstarter

Benutzeroberfläche (Screenshots)







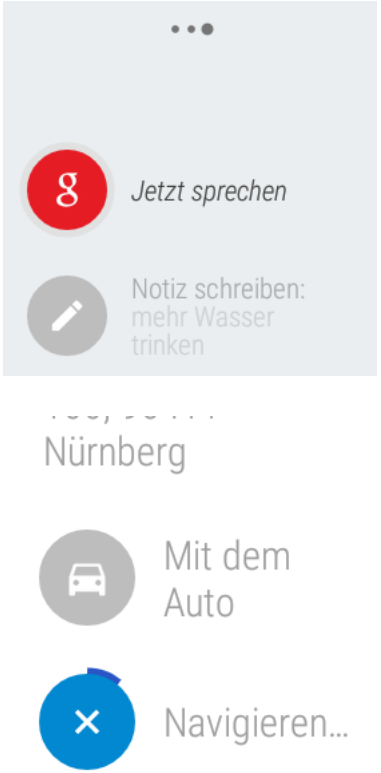


Der Kontextstrom

- vertikal verlaufende Liste von Karten im Stil von Google Now
- Benutzer blättert mit Wischgesten oder Armbewegungen
- stets ein Element sichtbar
  - eine Karte
  - Kartenstapel
- Karten können aus mehreren Seiten bestehen

Sprachsteuerung

- Start der Interaktion mit „Ok Google“
  - Navigiere zu Flughafen Nürnberg
  - Meine Schritte anzeigen
  - Notiz schreiben Folien für Vortrag fertigstellen
  - Einstellungen öffnen
- Üblicherweise erscheinen nach dem Sprachkommando Karten, mit denen sich die ausgelöste Aktion steuern oder abbrechen lässt



Für Android Wear entwickeln (1/3)

Karten in den Kontextstrom einspeisen

- Karten werden auf dem Smartphone/Tablet erzeugt
- Können spezielle Erweiterungen für Wearables enthalten

Für Android Wear entwickeln (2/3)

Wearable Apps

- werden auf der Smartwatch ausgeführt
- können auf Gerätefunktionen zugreifen
- können mit Smartphone/Tablet kommunizieren
- werden über den Play Store bezogen

Für Android Wear entwickeln (3/3)

Zifferblätter (Startdisplays)

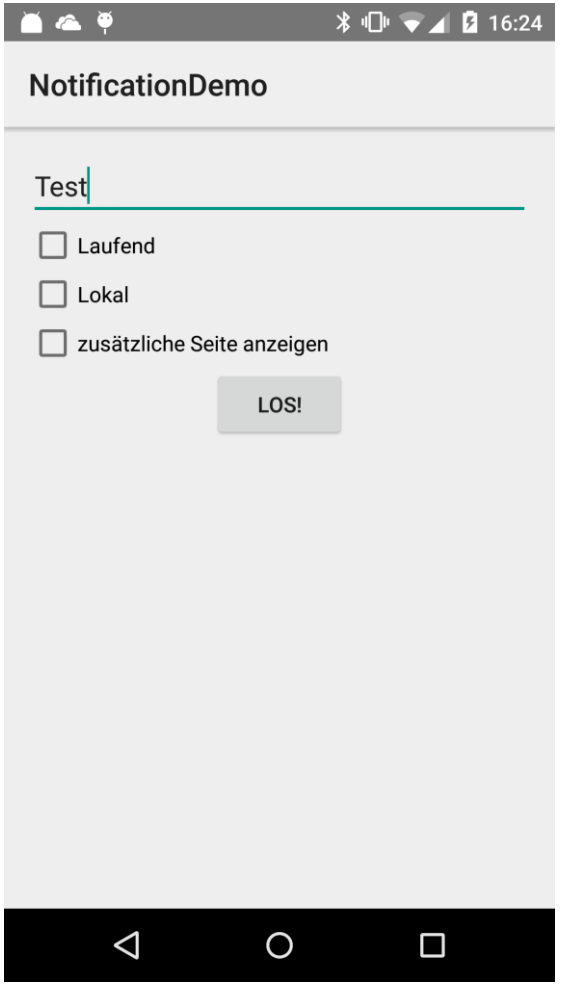
- werden auf der Smartwatch ausgeführt
- können mit Smartphone/Tablet kommunizieren, um beispielsweise Einstellungen zu speichern
- werden über den Play Store bezogen
- Spezialform einer „Wearable App“

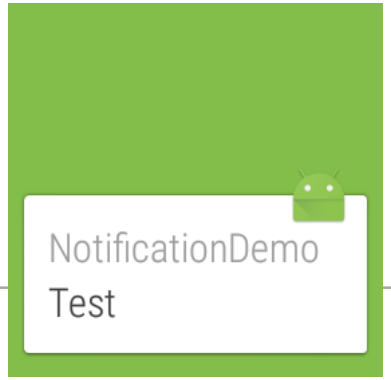
Voraussetzungen für das Entwickeln

- Java JDK, Android Studio und Android SDK
- für die Entwicklung eingerichtetes Smartphone oder Tablet (Developer-Optionen, USB-Debugging)
- mit dem Gerät verbundene Smartwatch oder konfigurierter Wear Emulator
- Alternativ: über USB angeschlossene Smartwatch
- Tipp: Java-bin-Verzeichnis und Android SDK-Verzeichnisse tools und platform-tools durch Erweitern der Umgebungsvariablen PATH verfügbar machen
- derzeit nicht offiziell unterstützt: emuliertes Smartphone/Tablet und emulierte Smartwatch

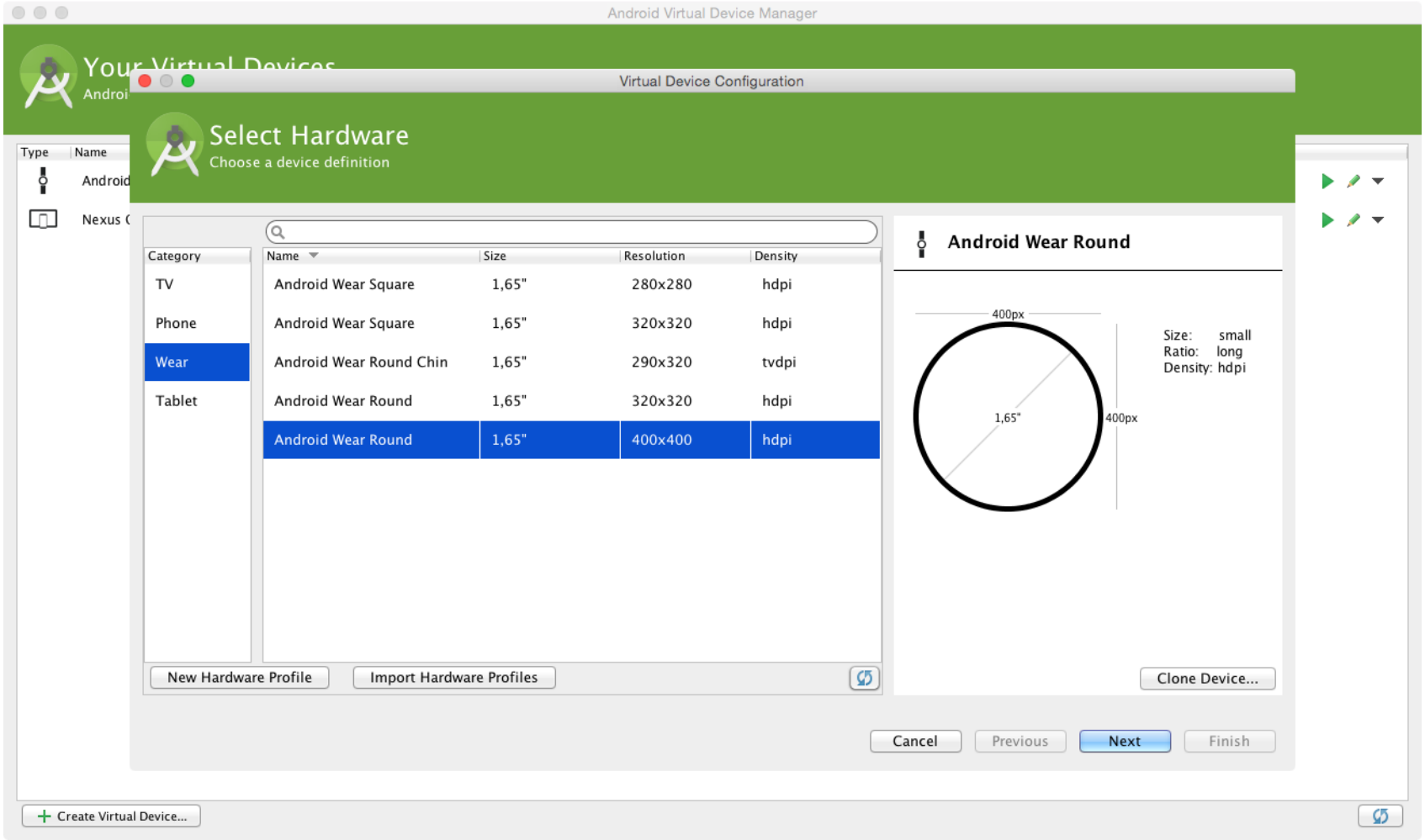
Beispiel: NotificationDemo

- läuft auf Tablet/Smartphone
- zeigt Benachrichtigungen auch auf der Smartwatch
- hat spezielle Erweiterungen für Wearables eingebaut:
  - kann zusätzliche Seite anzeigen
  - kann Spracheingaben entgegen nehmen
  - alternativ: Liste mit Optionen
  - Auswahl wird an App gesendet



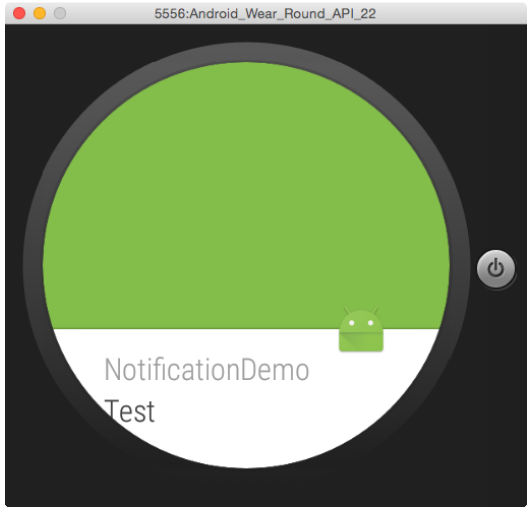


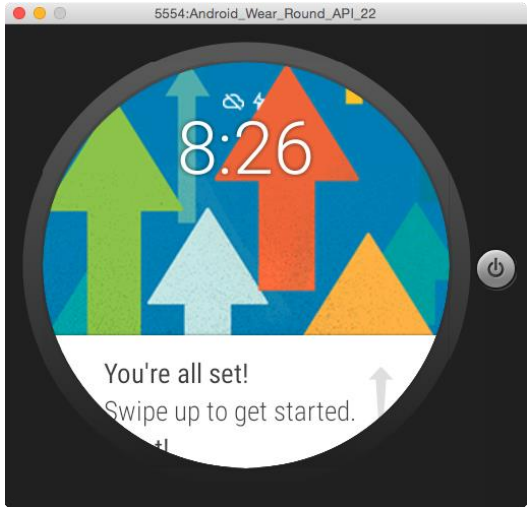
Simulierte Smartwatch konfigurieren...

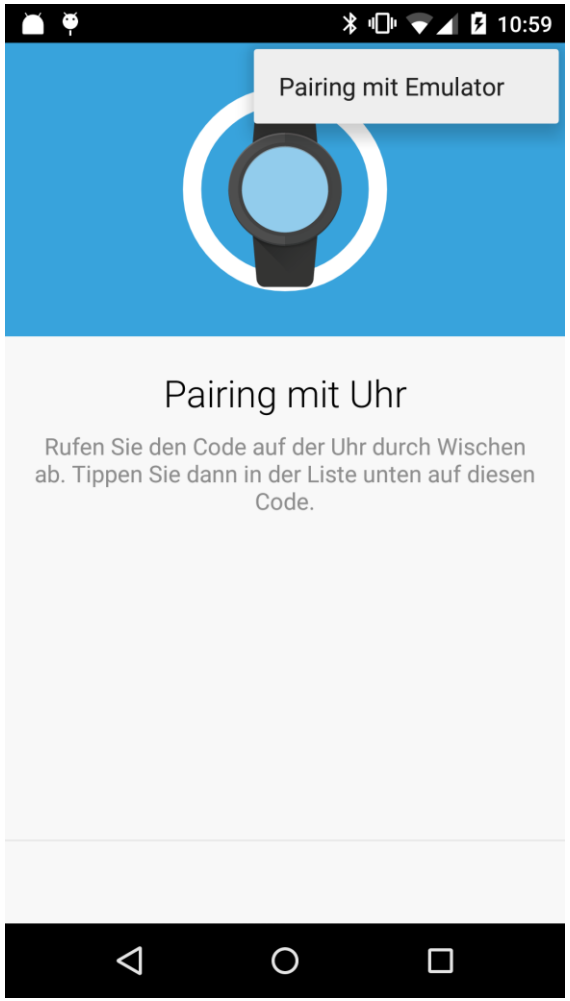


...und starten

- Wearable-Emulator starten
- in Companion auf Smartphone/Tablet App Verbindung mit neuer Uhr, Pairing mit Emulator
- in Android Studio adb -d forward tcp:5601 tcp:5601







Live-Demo

NotificationDemo

Emulator-Wehwehchen

- Touch-Gesten wirken „fummelig“
- Spracheingabe funktioniert nicht; Tastatur als Ersatz nicht überall wirksam
- gelegentlich Hänger in der Kommunikation (Benachrichtigungen werden nicht mehr zugestellt): hier hilft adb kill-server, adb start-server

Wie wird‘s gemacht?

Wichtig: Support-Bibliothek nutzen

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile "com.android.support:support-v4:22.0.0+"
}
```

Modul-spezifisches build.gradle

```
import android.support.v4.app.NotificationCompat;
import android.support.v4.app.NotificationManagerCompat;
import android.support.v4.app.RemoteInput;
...
```

NotificationDemoActivity.java

Einfache Benachrichtigungen bauen...

```
PendingIntent reply =
   PendingIntent.getActivity(this, 0, new
Intent(this,
      NotificationDemoActivity.class),
      PendingIntent.FLAG_UPDATE_CURRENT);
NotificationCompat.Builder notificationBuilder = new
NotificationCompat.Builder(this)

.setPriority(NotificationCompat.PRIORITY_HIGH)

.setContentTitle(getString(R.string.app_name))
        .setContentText(txt).setOngoing(ongoing)
        .setLocalOnly(local)
        .setWhen(System.currentTimeMillis())
        .setSmallIcon(R.mipmap.ic_launcher)
```

wird beim Antippen der Benachrichtigung auf Smartphone bzw. Tablet verwendet

wird nicht an Wearables weitergeleitet

wird nicht an Wearables weitergeleitet

... und anzeigen

```
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(
                NotificationDemoActivity.this);
notificationManager.notify(NOTIFICATION_ID,
notificationBuilder.build());
```

Für Wearables: Seite hinzufügen...

```
NotificationCompat.WearableExtender extender = new
NotificationCompat.WearableExtender();
...
extender.addPage(createSecondPage(txt));
```

liefert Notification-Instanz

... und zusammensetzen

```
private Notification createSecondPage(String txt) {
  StringBuilder sb = new StringBuilder();
  for (int i = 0; i < 20; i++) {
    if (sb.length() > 0) { sb.append(' '); }
      sb.append(txt);
  }
  NotificationCompat.BigTextStyle secondPageStyle =
    new NotificationCompat.BigTextStyle();
  secondPageStyle.setBigContentTitle(getString(R.string.page2))
                 .bigText(sb.toString());
    return new NotificationCompat.Builder(this)
                 .setStyle(secondPageStyle).build();
}
```

Sprachsteuerung konfigurieren

```
...
extender.addAction(createVoiceReplyAction(reply));
...
private NotificationCompat.Action
            createVoiceReplyAction(PendingIntent pendingIntent) {
  String replyLabel = getString(R.string.reply);
  String[] choices = getResources().getStringArray(R.array.choices);
  RemoteInput remoteInput = new RemoteInput.Builder(EXTRA_VOICE_REPLY)
            .setLabel(replyLabel).setChoices(choices).build();
  return new NotificationCompat.Action.Builder(R.mipmap.ic_launcher,
              getString(R.string.app_name), pendingIntent)
              .addRemoteInput(remoteInput).build();
}
```

Auswahltexte

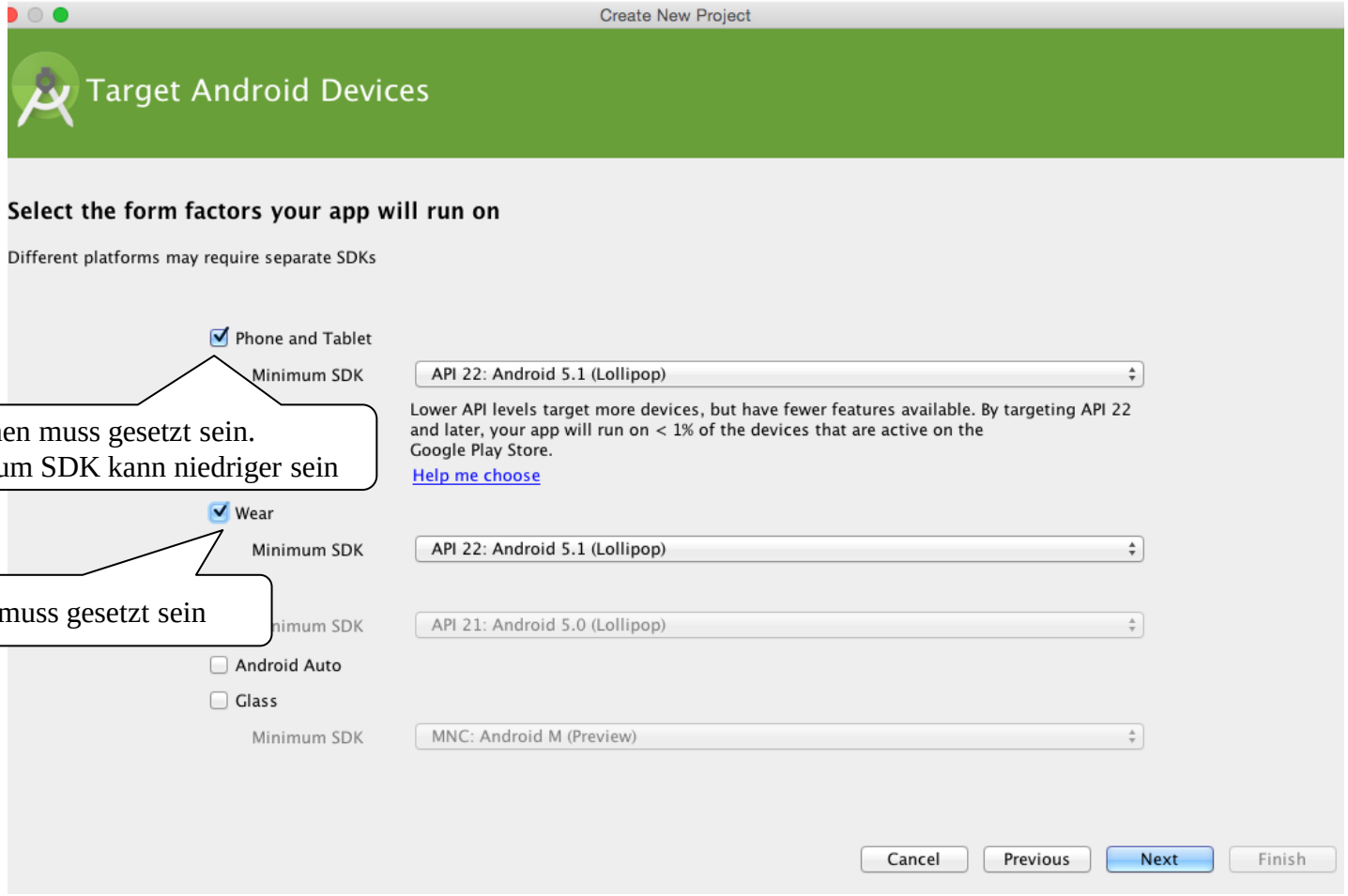
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="choices">
        <item>Antwort #1</item>
        <item>Antwort #2</item>
        <item>Antwort #3</item>
    </string-array>
</resources>
```

choices.xml

Übertragenen Text auslesen

```
Intent intent = getIntent();
if (intent != null) {
    CharSequence text = getMessageText(intent);
    if (text != null) { edittext.setText(text); }
}
...
private CharSequence getMessageText(Intent intent) {
    Bundle remoteInput = RemoteInput.getResultsFromIntent(intent);
    if (remoteInput != null) {
        return remoteInput.getCharSequence(EXTRA_VOICE_REPLY);
    }
    return null;
}
```

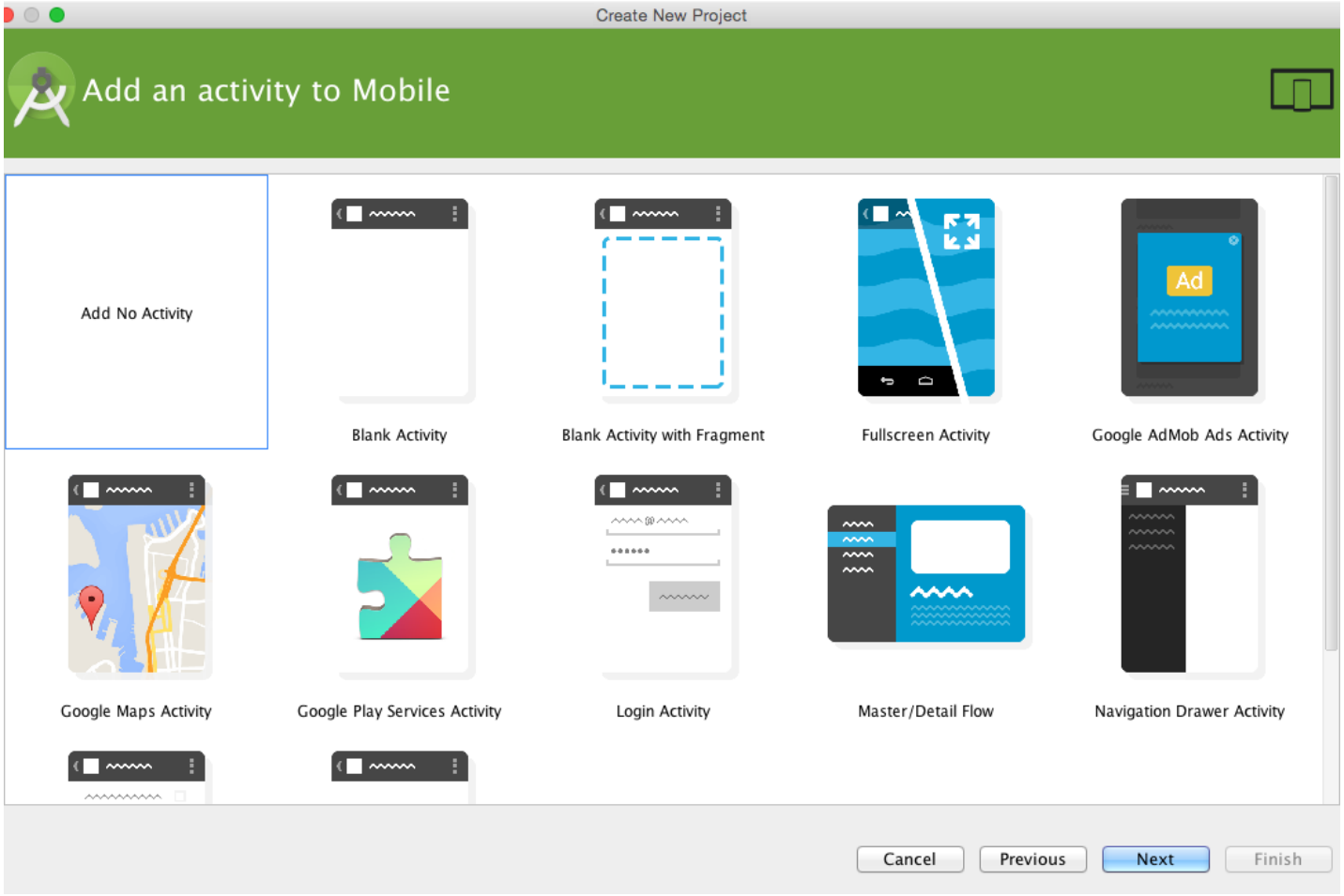
Echte Android Wear-Apps



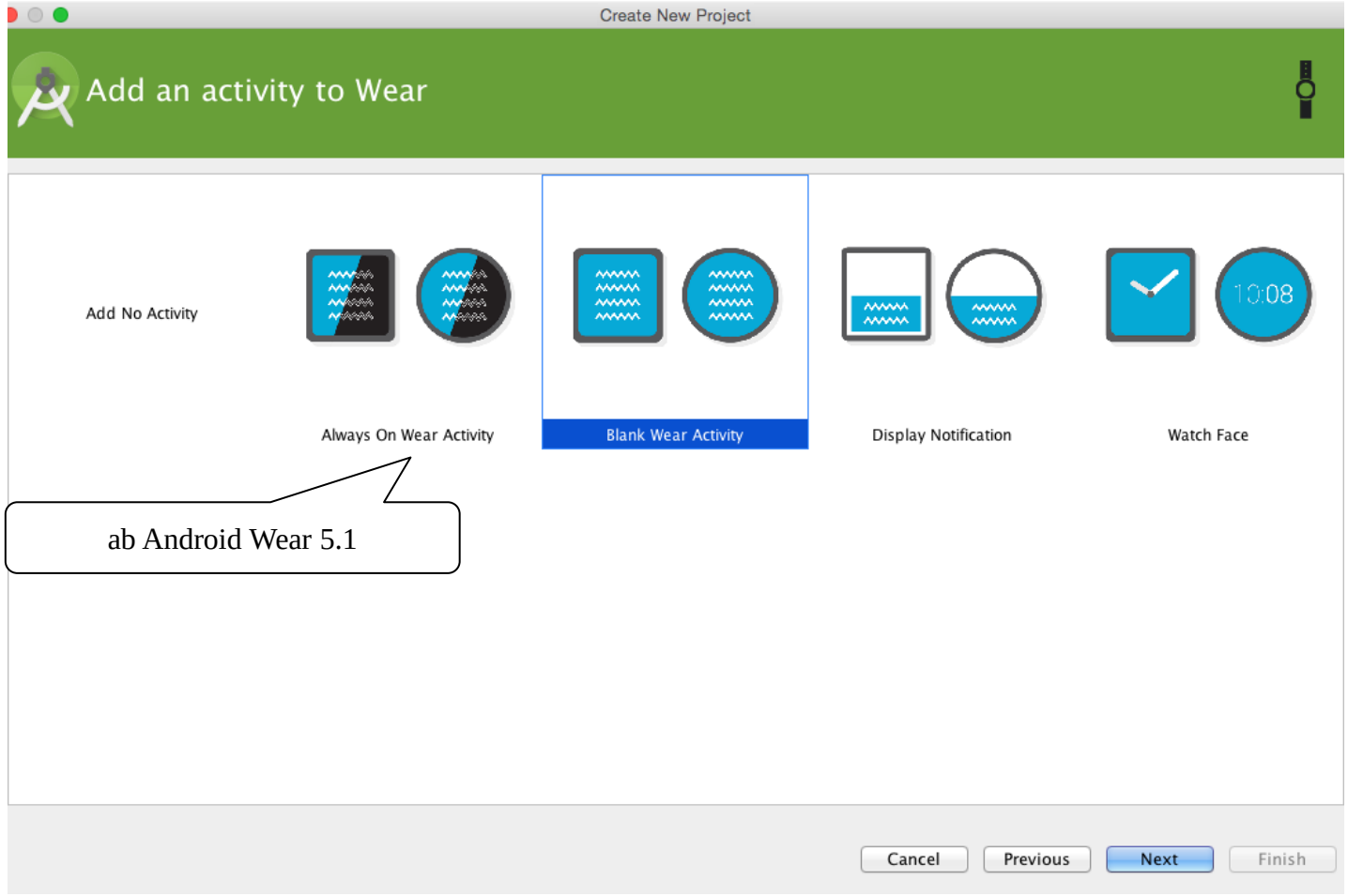
Häkchen muss gesetzt sein.
Tipp: Minimum SDK kann niedriger sein

Häkchen muss gesetzt sein

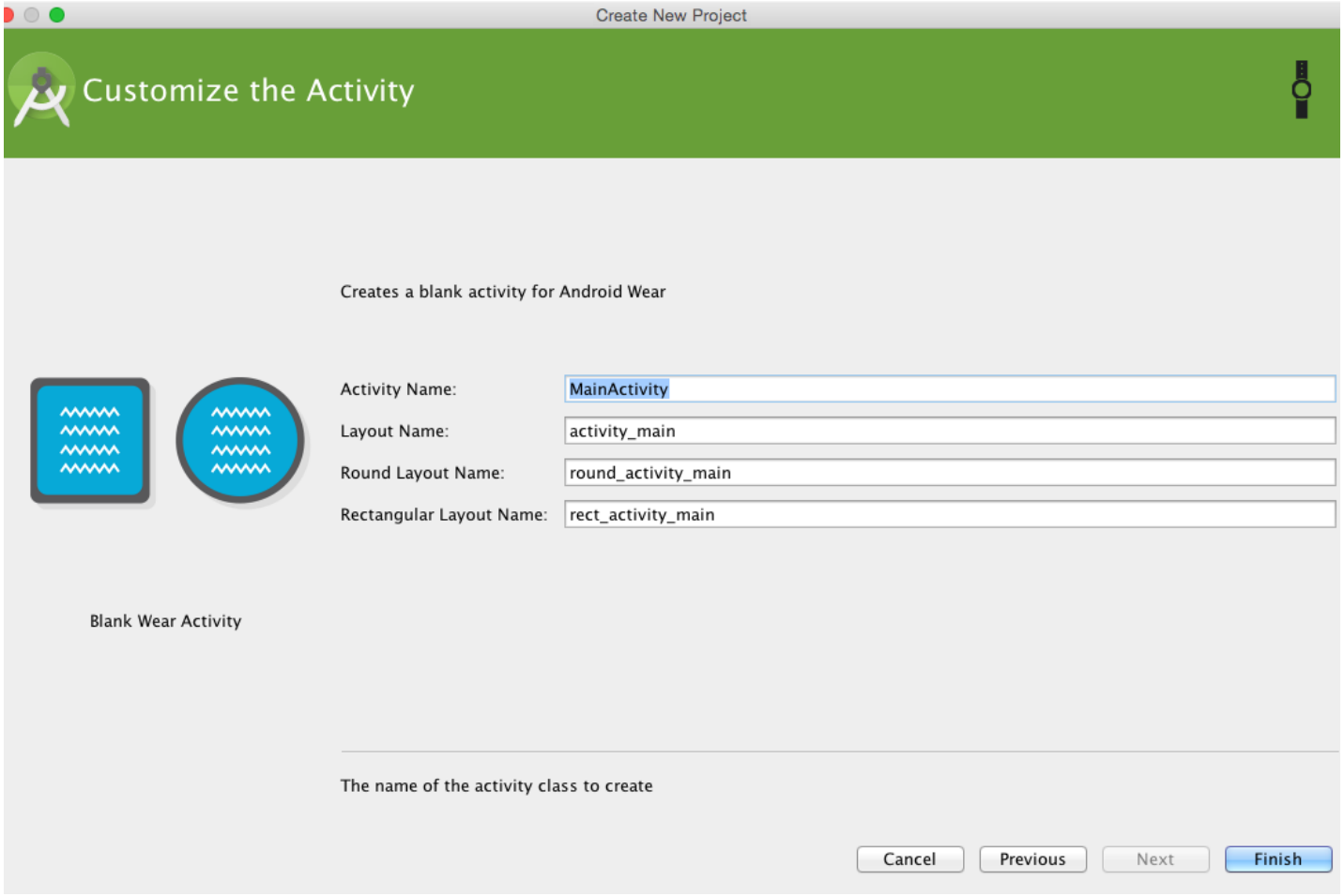
Teil für Smartphone/Tablet konfigurieren



Teil für Wearable konfigurieren (1/2)



Teil für Wearable konfigurieren (2/2)



Module in Android Studio

- In einem Android Studio-Hauptfenster wird stets ein Projekt bearbeitet
- Module bilden eine Art „Unterprojekt“
- können helfen, Android Apps zu strukturieren
- jedes Modul hat eine eigene build.gradle-Datei

- „klassische“ Android Apps bestehen aus dem Modul „app“
- wird üblicherweise durch den Projektassistenten oder im Rahmen einer Migration erzeugt

Android Wear-Projekte sind anders

- bestehen aus zwei Modulen, „mobile“ und „wear“
- „mobile“ repräsentiert eine auf Smartphone/Tablet ausgeführte App
- es kann sich um eine Rumpf-App handeln: es muss keine Activity vorhanden sein

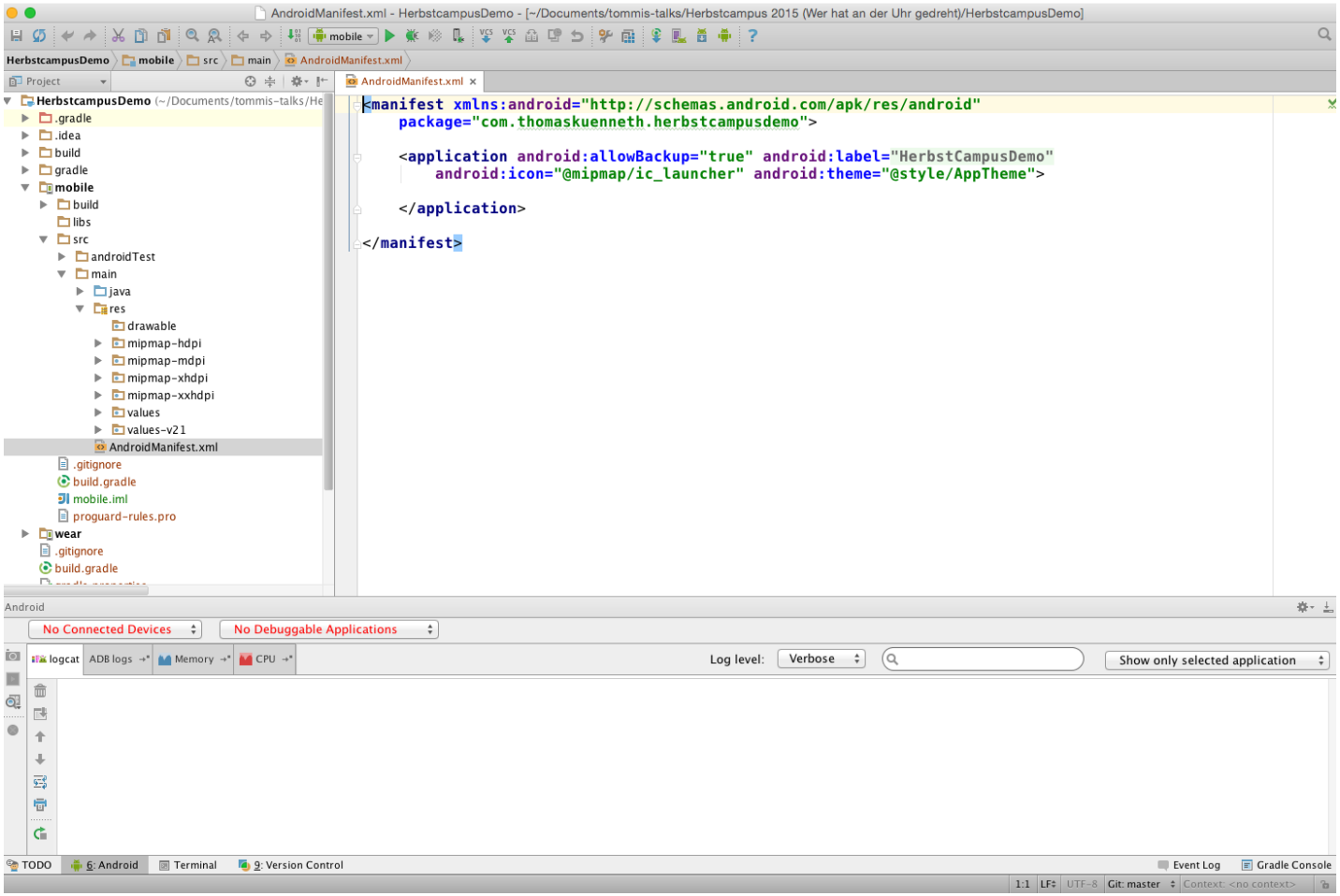
- „wear“ repräsentiert Wearable Apps

Wozu braucht man dann den „mobile“-Zweig?

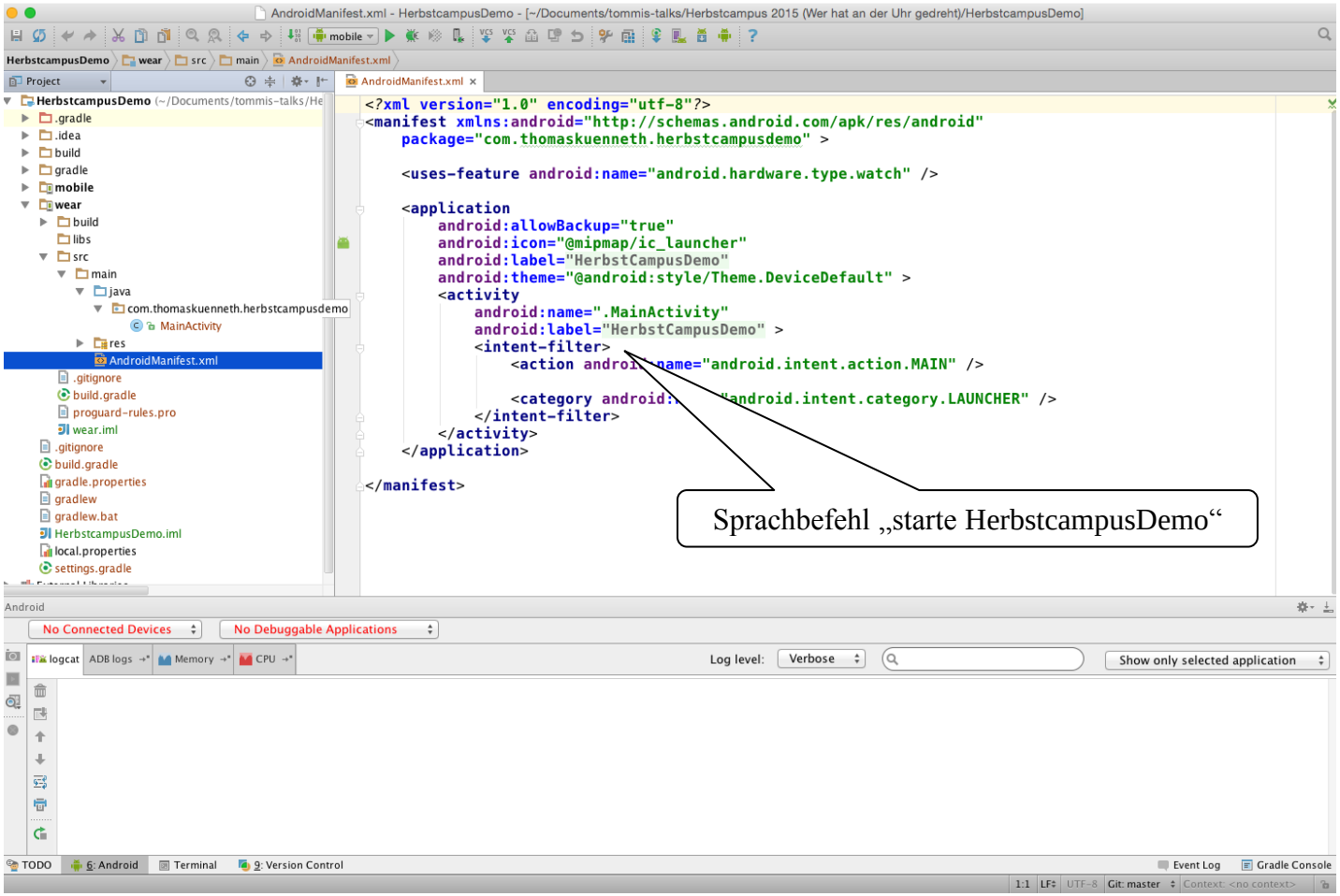
Wearable Apps fahren Trittbrett

- keine Store-App auf der Smartwatch (Formfaktor, Speicher, Rechenpower, Internet-Anbindung, ...)
- Wearable Apps werden mit dem Smartphone/Tablet in Google Play heruntergeladen und automatisch auf die Uhr kopiert
- Deinstallation ebenfalls über Smartphone/Tablet
- viele Wearable Apps *brauchen* eine Begleit-App auf dem Smartphone

Projektstruktur (mobile)



Projektstruktur (wear)

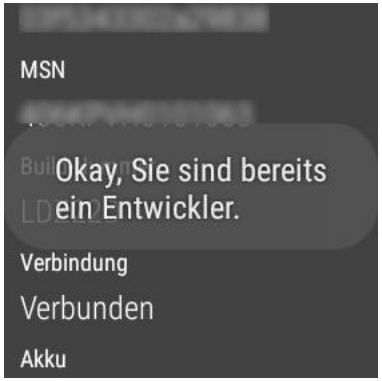


Beispiel Akkustandsanzeige

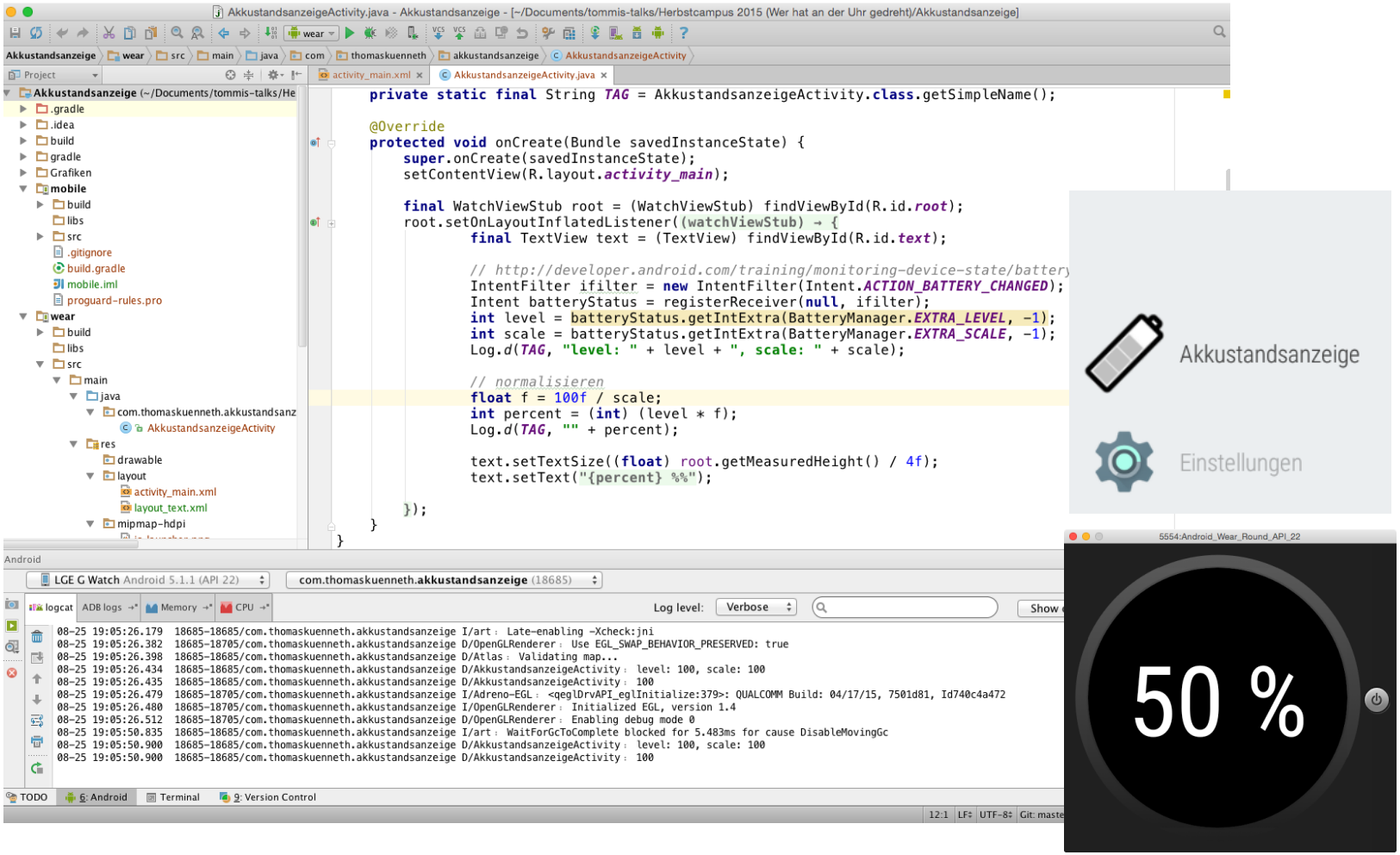
- Start im Emulator, auf der Uhr via Docking-Station oder über Smartphone/Tablet
- Wearable App funktioniert nicht innerhalb eines APKs, das mit Entwicklerzertifikat signiert wurde
- Deshalb...
  - ... mit Produktionszertifikat signieren oder...
  - ... über Bluetooth debuggen

Debuggen über Bluetooth einrichten

- in den Einstellungen Entwickler-Optionen freischalten (7 mal Build-Nummer antippen)
- Debug over Bluetooth aktivieren
- In der Companion App Debugging over Bluetooth aktivieren
- In Android Studio:
  - adb forward tcp:4444 localabstract:/adb-hub
  - adb connect localhost:4444
- Beliebige adb-Kommandos mit adb -s localhost:4444 <command> absetzen
- Verbindung kommt nicht immer zustande
- wirkt gelegentlich zäh



Walkthrough: Akkustandsanzeige



Fazit (1/2)

- Aus Anwendersicht macht Android Wear großen Spaß
  - Bedienkonzept in sich schlüssig
  - Zusammenspiel zwischen Wearable und Smartphone/Tablet ist gelungen
  - wichtige Apps vorinstalliert
- Aus Entwicklersicht ist positiv:
  - bekannte Programmiersprache
  - bekannte Tools
  - bekannte APIs

Fazit (2/2)

- Unschön ist...
  - Tools unnötig hakelig
  - Doku nicht immer ausreichend
  - Undurchsichtige Verzahnung von Plattform und Google-Diensten
- Setzt sich Android Wear als Massenphänomen durch?
  - Analysten wissen zu berichten, dass sich die Apple Watch besser verkauft als alle Android Wear-Geräte zusammen
  - Möglich ist aber auch, dass sich wie bei Smartphones das Blatt wendet. Stichworte: Herstelleranzahl, Preis

Fragen, Anregungen, Diskussion

Vielen Dank!

Thomas Künneth, M.A.

MATHEMA Software GmbH

thomas.kuenneth@mathema.de

@tkuenneth

http://kuennetht.blogspot.de/