



# Native Android App Entwicklung

Susanne Braun, 14.08.2013, JUG Darmstadt



## Unser Angebot

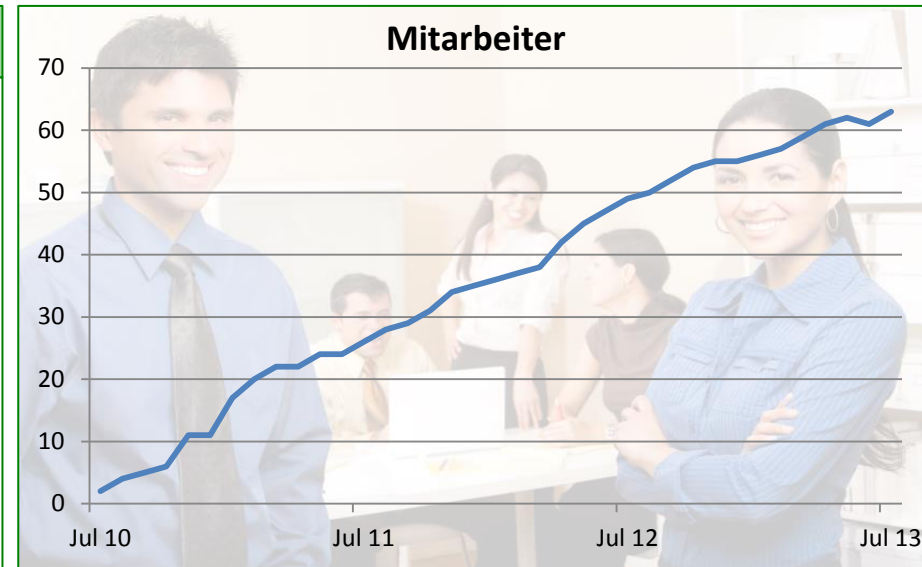
- Individuelle Softwarelösungen für IT-Kernsysteme (Schwerpunkte: Java/JEE, .NET, Open Source, RIA, u.a.)
- Technologie- und Architektur-Beratung (Schwerpunkte: IT-Architektur und -Modernisierung, EAM, SOA, BPM, Security, u.a.)
- Management von Software-Projekten
- Integration dedizierter Software-Produkte (Schwerpunkte: CMS, Portal)

## Unsere Kunden

- Wir entwickeln individuelle Softwarelösungen für die Kernkompetenzen unserer Kunden und beraten in aktuellen Fragestellungen von Technologie und Architektur.
- Accso arbeitet branchenübergreifend für namhafte Unternehmen, dazu zählen Deutsche Telekom, Deutsche Bahn, Deutsche Flugsicherung, Deutsche Wetterdienst, DekaBank, AOK, ZDF, SWR, HR, WDR, DuMont-Verlag, HRS, Maxdome, Vodafone und weitere.

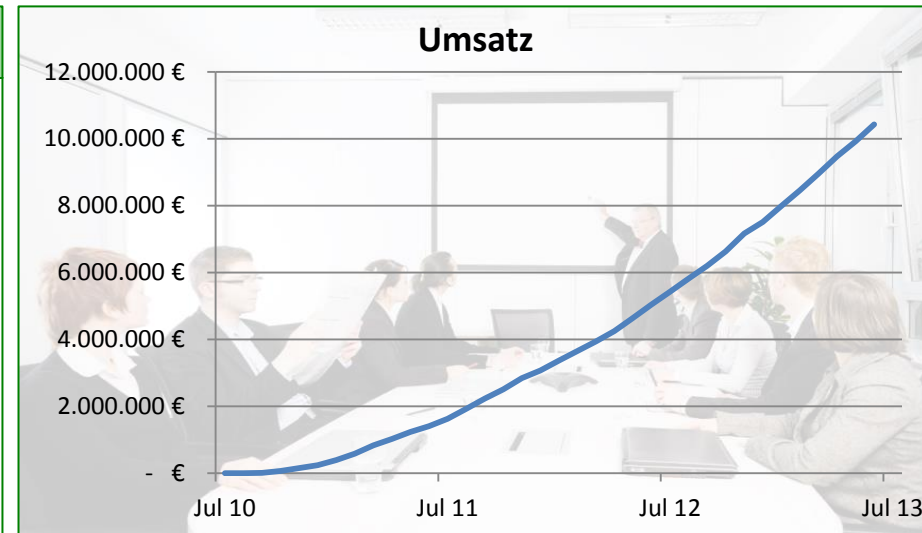
## Unsere Mitarbeiter

- 63 Mitarbeiter (Stand 1.7.2013)
- Software-Ingenieure und IT-Berater
- Exzellent ausgebildet (fast alle Mitarbeiter haben einen Hochschulabschluss, jeder sechste mit Promotion)
- sehr erfahren (> 80% mit mehr- bzw. langjähriger Berufserfahrung)
- hoch motiviert



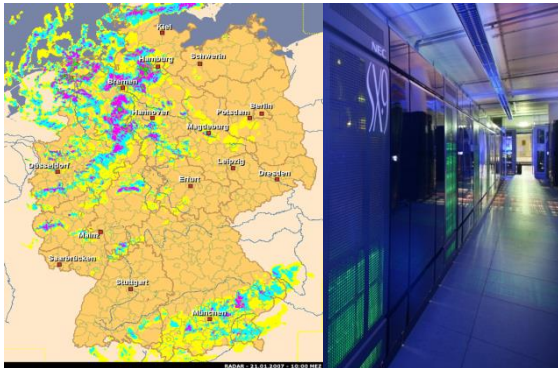
## Fakten

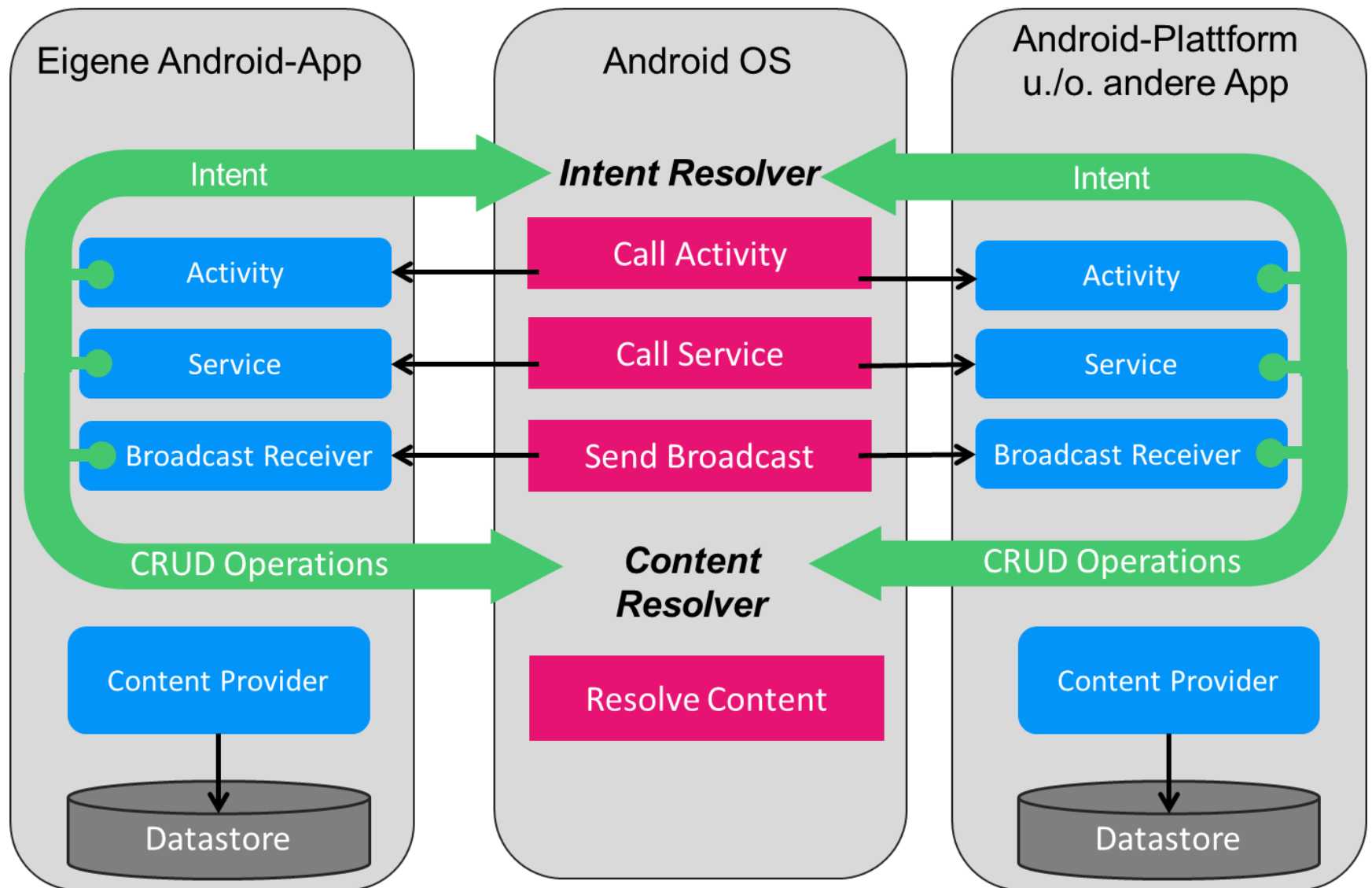
- Gegründet 2010 in Darmstadt
- GmbH mit Kapitalbeteiligung des Landes Hessen
- Geschäftsführung: Jürgen Artmann, Dr. Markus Voß
- Niederlassungen: Darmstadt, Köln





# Unsere **Kunden** schätzen die Kompetenz des Accso-Teams und arbeiten gerne mit uns zusammen





## Konzepte und API

- Umfangreiche Konzepte
- Komplexes Lifecycle-Management und API

## Rückwärts-Kompatibilität

- Rollout von Updates liegt bei den Herstellern -> Immer noch sehr viele Devices mit älterer OS-Version „in the wild“
- Support-Bibliothek von Google nicht immer aktuell und teilweise auch mit Fehlern.

## Development-Tools

- Kein guter freier GUI-Builder
- Eclipse manchmal schwerfällig, häufig Neustart von ADB notwendig, Treiber-Problematik beim Testen mit echter Hardware
- Hardware-Beschleunigung für Emulator nicht für alle Versionen und nicht für Google-API

## Testing

- Zu viel Hardware
- Zu viele Anpassungen durch Hersteller
- Erfahrene Tester erforderlich



## Activity-Klasse

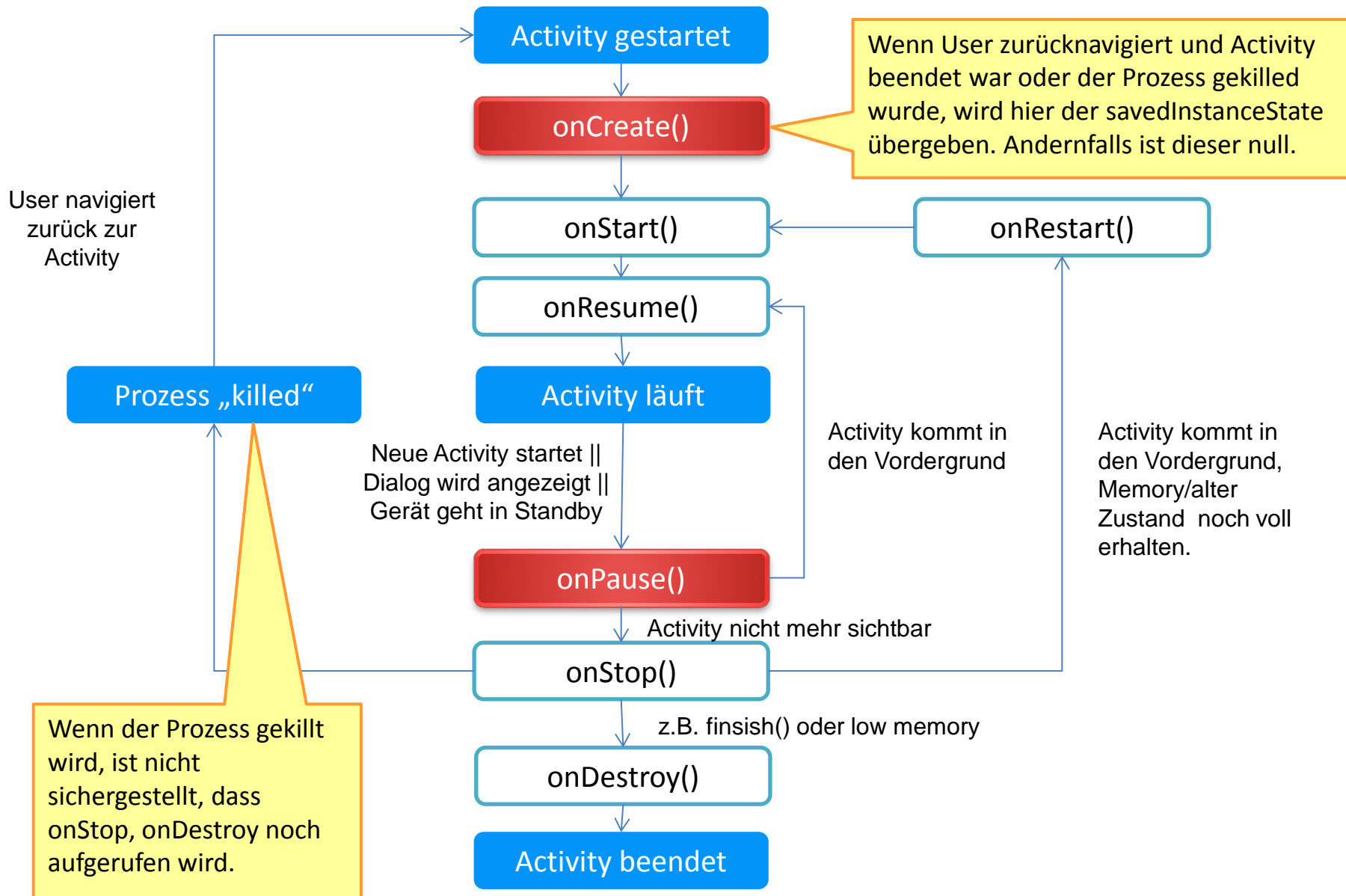
```
public class NotesActivity extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_notes);  
    }  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        getMenuInflater().inflate(R.menu.activity_notes,  
            menu);  
        return true;  
    }  
}
```



## Resources:

- Layout
- Menu
- Values (Strings, Styles, ...)
- Drawables
- ...

```
<RelativeLayout  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent" >  
  
    <EditText  
        android:id="@+id/notesTextEdit"  
        android:layout_width="match_parent"  
        android:layout_height="match_parent"  
        android:ems="10"  
        android:gravity="top/center_vertical"  
        android:hint="@string/enter_notes"  
        android:inputType="textMultiLine" >  
  
        <requestFocus />  
    </EditText>  
  
</RelativeLayout>
```





Aufruf erfolgt immer, wenn Activity zerstört wird (z.B. low memory) und der User zu dieser Activity zurücknavigieren könnte.

Callback	Zweck
<code>onCreate(Bundle savedInstanceState)</code>	Initialisierung oder alten Zustand wiederherstellen mit Bundle-Werten
<code>onPause()</code>	Alle persistenten ungesicherten Daten speichern
<code>onSaveInstanceState(Bundle outState)</code>	Zustand im Bundle sichern oder ggf. persistieren in Properties oder Datenbank.
<code>onDestroy()</code>	Ressourcen releasen (z.B. Threads stoppen, Cursor schließen etc.)

- Realisierung von looser Kopplung zwischen den Laufzeit-Komponenten eines Android-Systems
- und Wiederverwendbarkeit von (Fremd-) Komponenten
- Komponenten sind z.B. **Activities**, Services, Broadcast-Receiver...
- *„Ich hätte gerne den Zucker“*
- Beschreiben eine Absichtserklärung
- ...oder das Eintreten eines bestimmten Ereignisses
- Parameter/ Daten werden in `INTENT_EXTRAS` übergeben

## Expliziter Intent:

```
Intent intent = new Intent(this, SomeActivity.class);  
intent.putExtra("Value", "Some extra value for SomeActivity");  
startActivity(intent);
```

Klasse der Activity  
wird explizit  
genannt

Daten-/Parameter-  
Übergabe

Action

Daten-URI

## Impliziter Intent - Webseite aufrufen:

```
Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.accso.de"));  
startActivity(intent);
```

## Impliziter Intent - Content-Sharing via soziale Netzwerke:

```
Intent intent = new Intent(Intent.ACTION_SEND);  
intent.setType("text/plain");  
intent.putExtra(android.content.Intent.EXTRA_TEXT, "News for you!");  
startActivity(intent);
```



Betriebssystemversionen

Anpassungen durch Hersteller

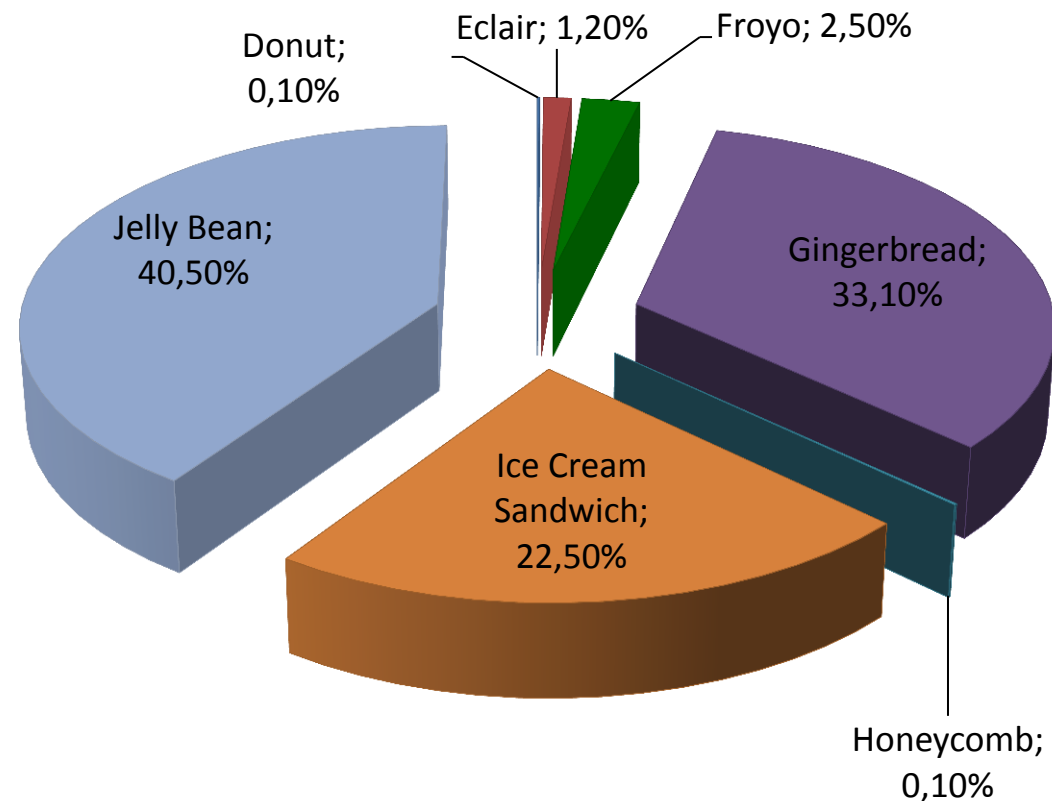
Display-Klassen

Formfaktoren: Smartphones,  
Tablets, TV, ....

## Fragmentierung



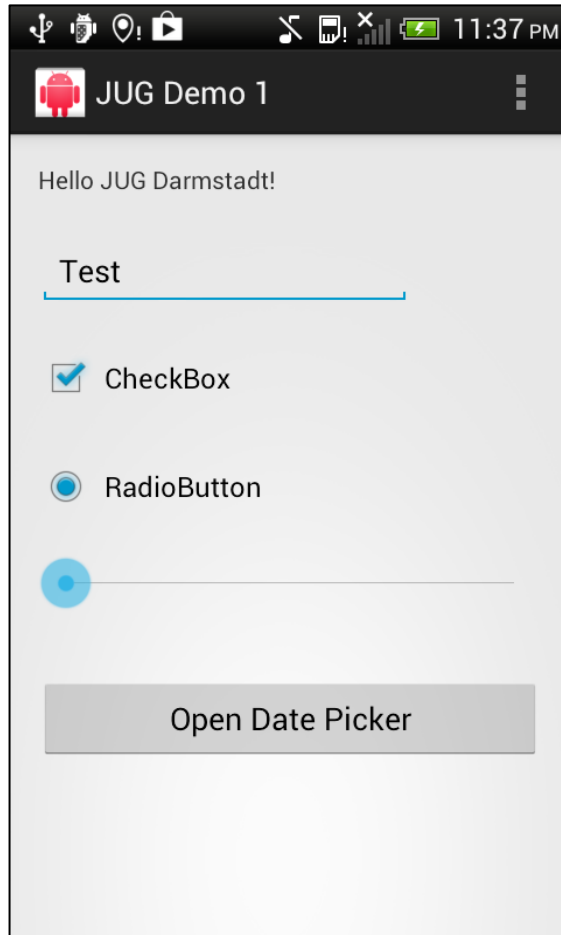
Version	Codename	API	Verteilung
1.6	Donut	4	0,10%
2.1	Eclair	7	1,20%
2.2	Froyo	8	2,50%
2.3-2.3.2	Gingerbread	9	0,10%
2.3.3-2.3.7		10	33,00%
3.2		13	0,10%
4.0.3-4.0.4	Ice Cream Sandwich	15	22,50%
4.1.x	Jelly Bean	16	34,00%
4.2.x		17	6,50%



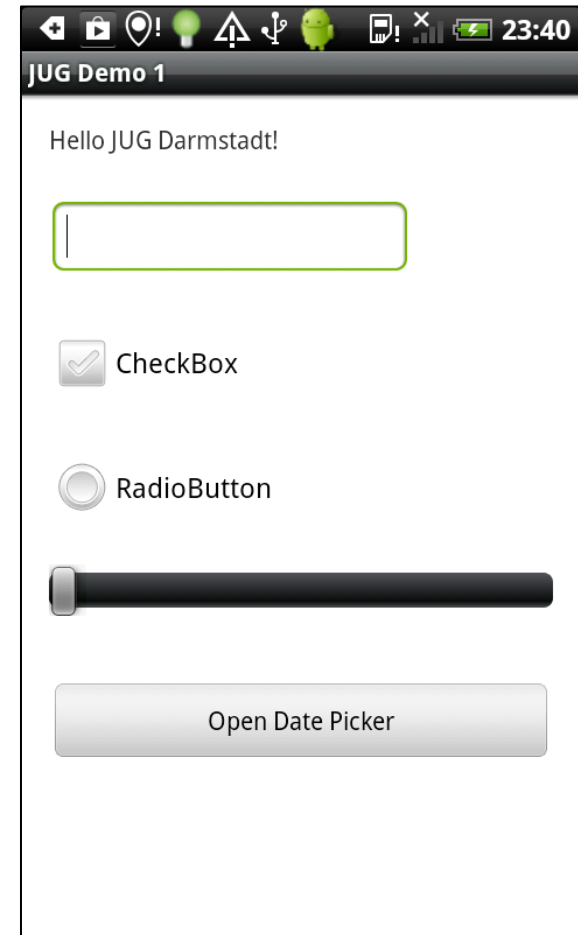
Quelle:

<http://developer.android.com/about/dashboards/index.html>

# Standard-Elemente auf unterschiedlichen OS-Versionen



Android 4.0.3, HTC

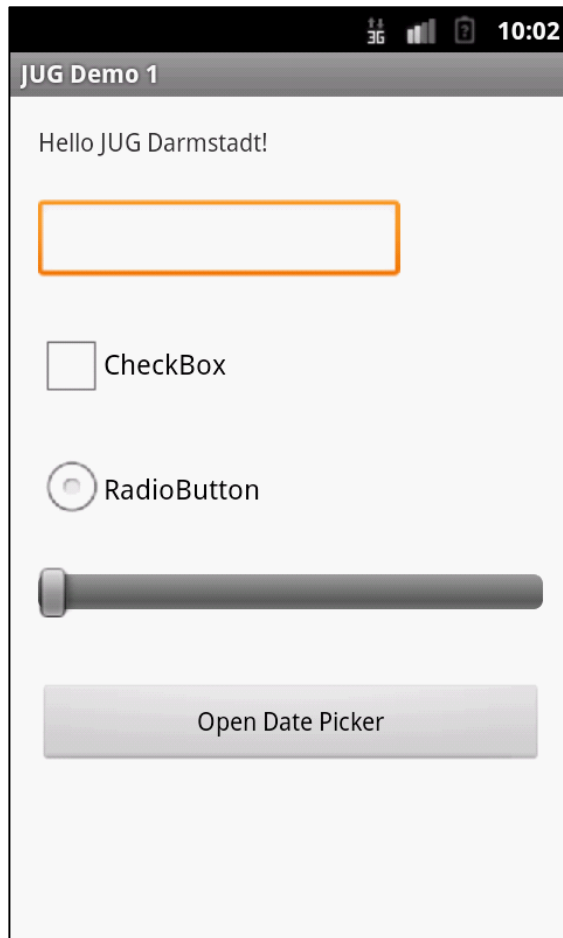


Android 2.2, HTC

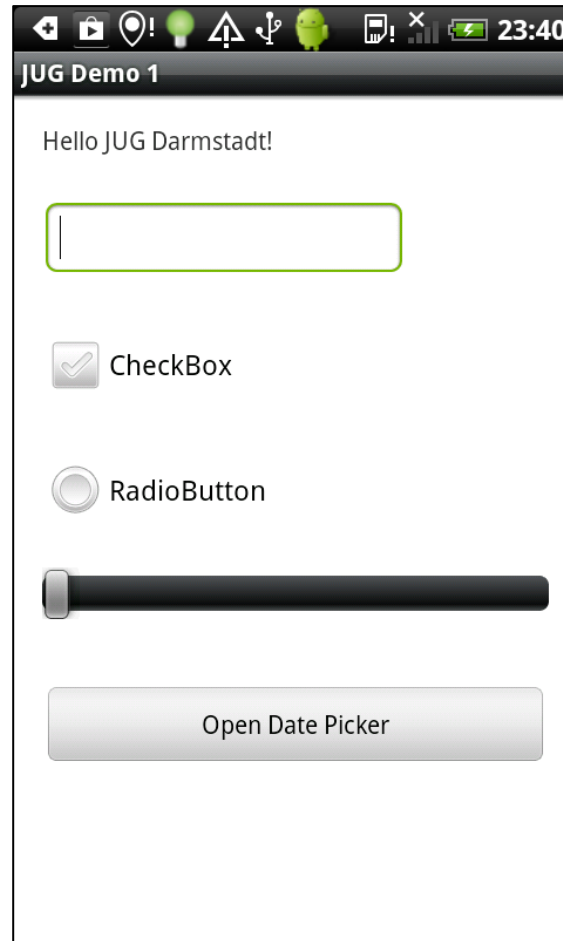
# Standard-Elemente bei unterschiedlichen Herstellern



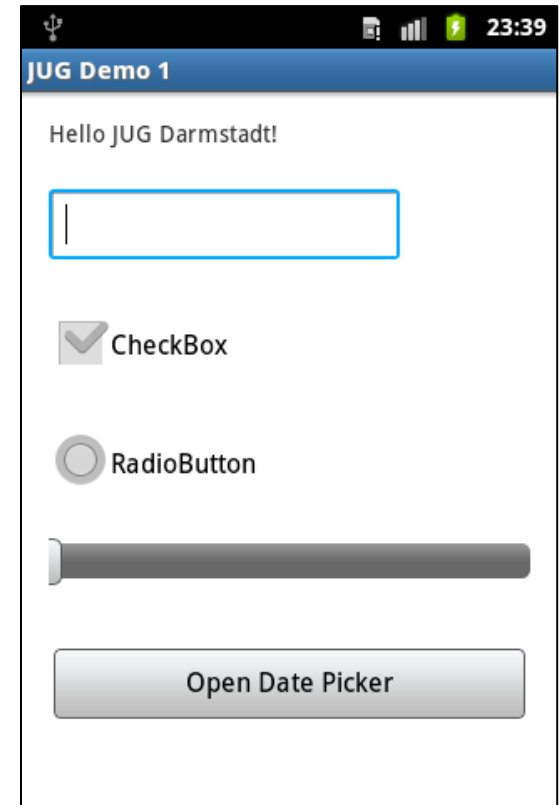
## *Light-Theme* bei unterschiedlichen Herstellern



Google 2.3



HTC, 2.2

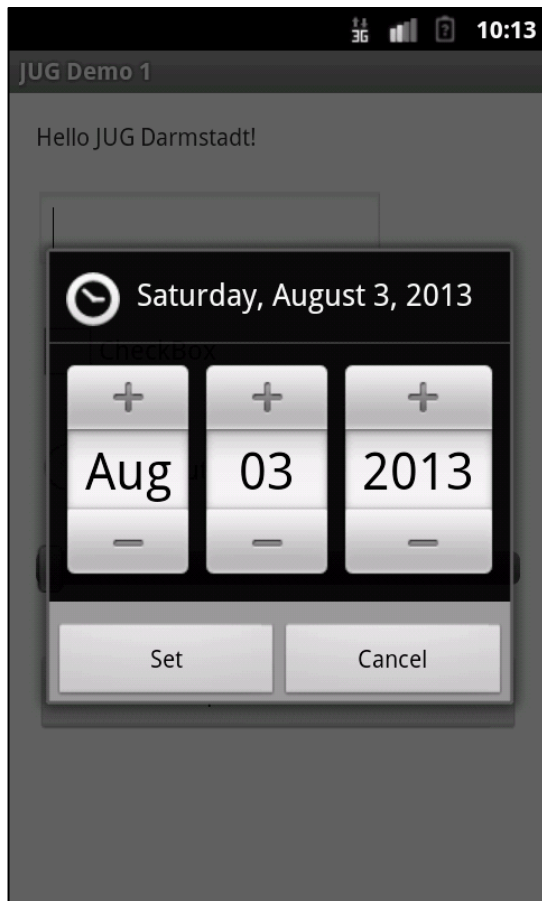


Samsung, 2.3

# Standard-Elemente bei unterschiedlichen OS-Versionen und Herstellern



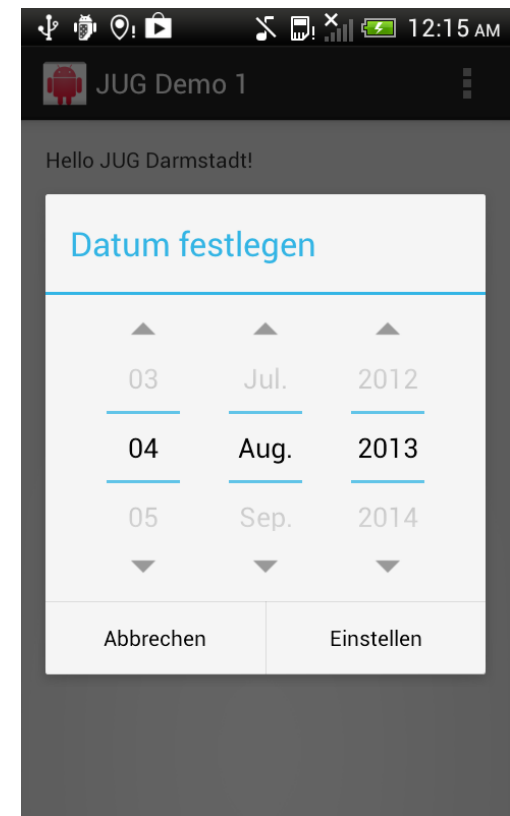
## Unterschiede bei Nutzerführung und Design



Google 2.3, Light-Theme



Samsung, 2.3, Light-Theme



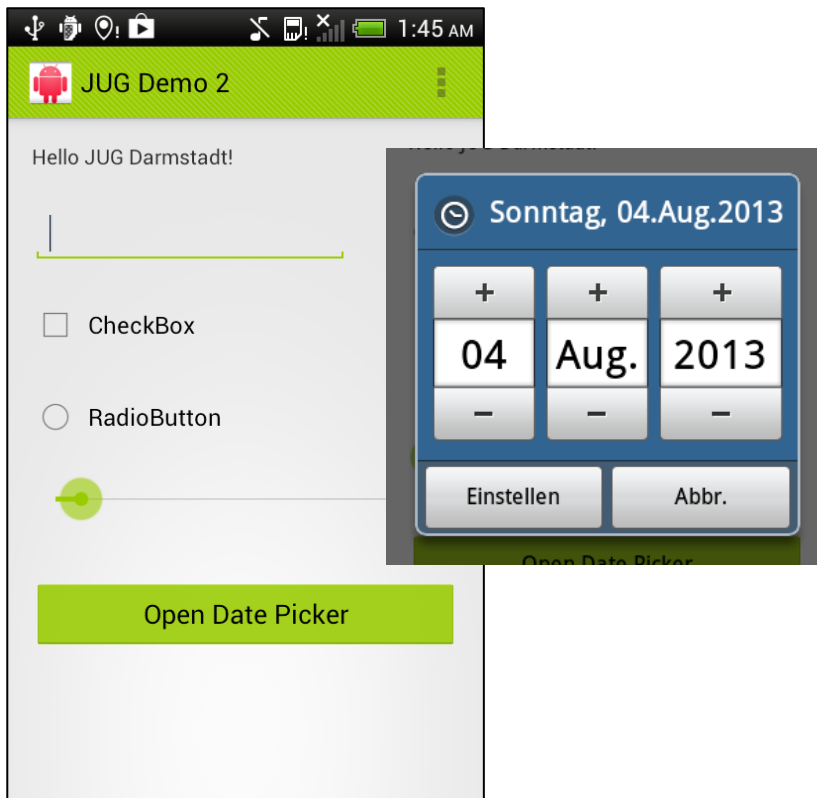
HTC, 4.0, Holo.Light-Theme

- Eigenes App Theme entwickeln
- Dennoch: Es ist (fast) unmöglich ein komplett einheitliches Design auf allen Endgeräten zu erzwingen
- Zu bedenken ist: Wenn man die Standard-Elemente unverändert übernimmt ist die Bedienung für den User oftmals intuitiver
- Auf Pre-4.0 Devices liegen die Original-Themes u. U. mit erheblichen Hersteller-Anpassungen auf dem Device vor
- Seit Android 4.0 müssen auf allen Devices mit Zugang zum Play-Store die originalen Holo-Themes unverändert vorliegen
- Tester schulen. Für unerfahrene Tester ist es schwierig zu unterscheiden, wann die Design-Vorgaben korrekt umgesetzt wurden und wann nicht

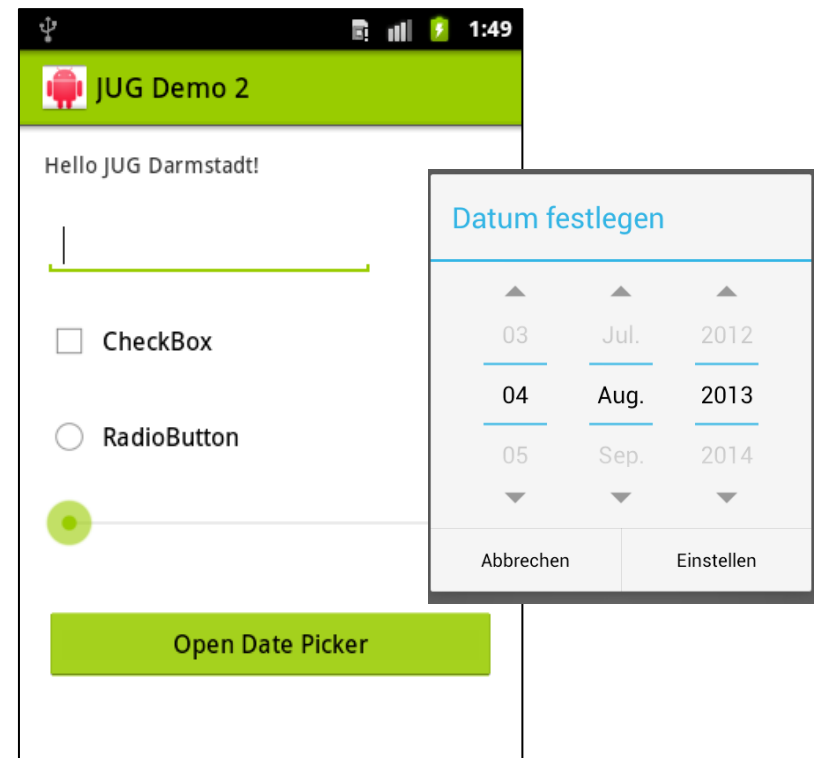


Android Asset Studio:

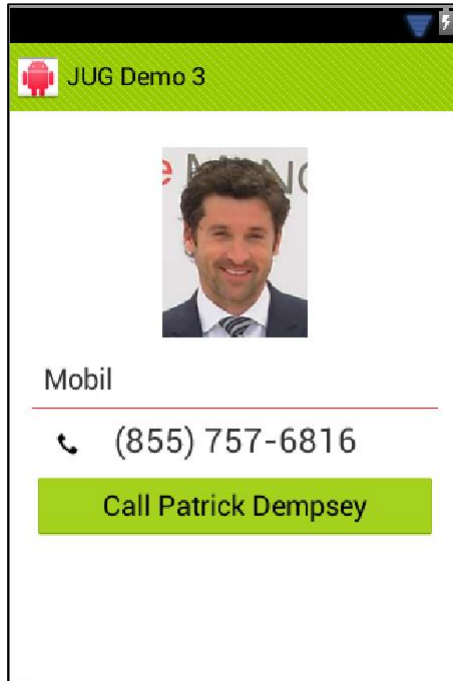
<http://android-ui-utils.googlecode.com/hg/asset-studio/dist/index.html>



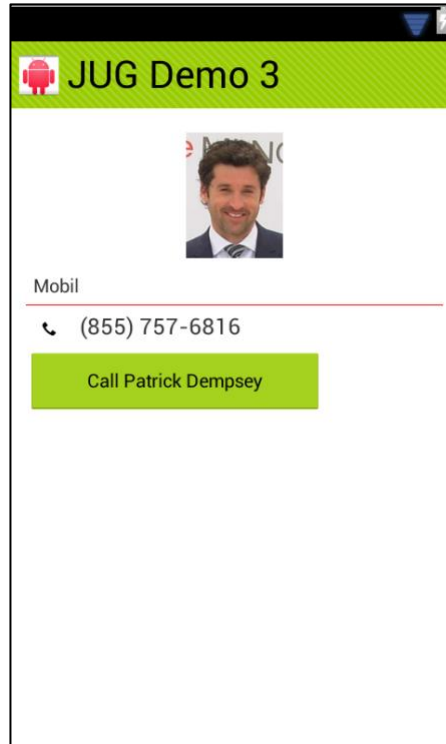
HTC, 4.0,  
Holo.Light-Theme



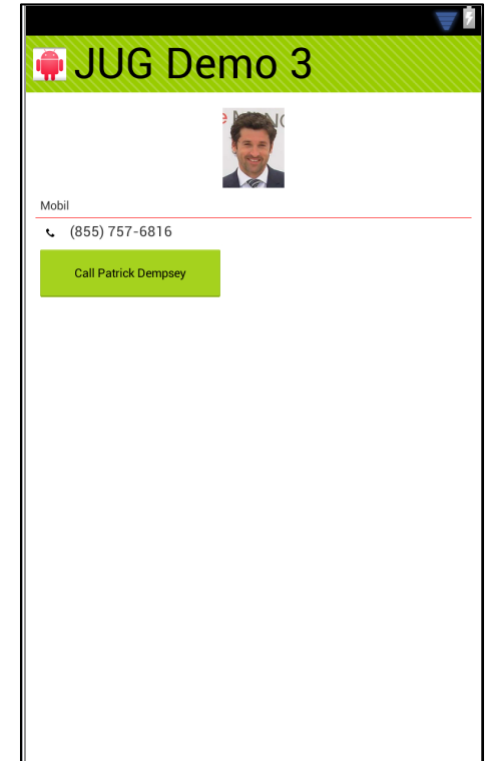
Samsung, 2.3, Light-  
Theme



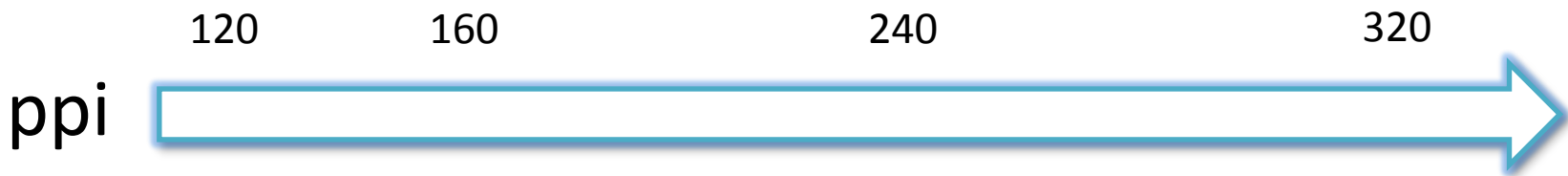
HTC Hero,  
mdpi-Display



HTC Desire,  
hdpi-Display



Nexus 4,  
xhdpi-Display



ldpi

mdpi

hdpi

xhdpi



**HTC Hero,**  
3,2",  
320x480,  
**180 ppi**



**HTC Desire,**  
3,7",  
480x800  
**252 ppi**







**Nexus 4,**  
4,7",  
1280x768,  
**316 ppi**

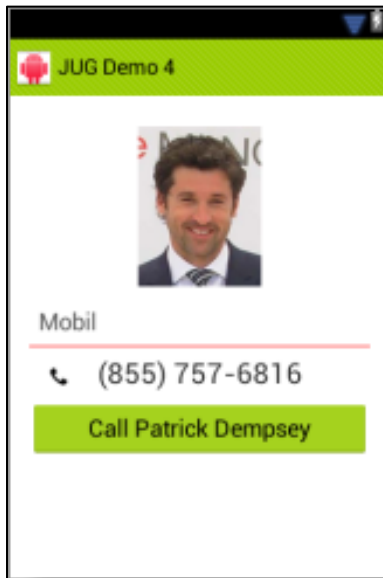
- Density-independent Pixel
- Einheit **dip** bzw. **dp**
- „Virtuelles Pixel“, dass unabhängig von der Pixeldichte ist
- 1 dp == 1 px auf mdpi-Devices
- Pendant für Text: **sp**

$$\text{px} = \text{dp} * (\text{ppi} / 160)$$

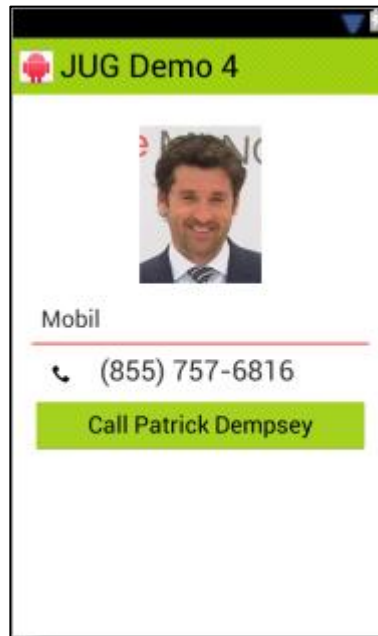
$$px = dp * (ppi / 160)$$

DP	Display-Density	Faktor	PX	Grafik
48x48	xhdpi (320 ppi)	2.0	96x96	
48x48	hdpi (240 ppi)	1.5	72x72	
48x48	mdpi (160 ppi)	1.0	48x48	
48x48	ldpi (120 ppi)	0.75	32x32	

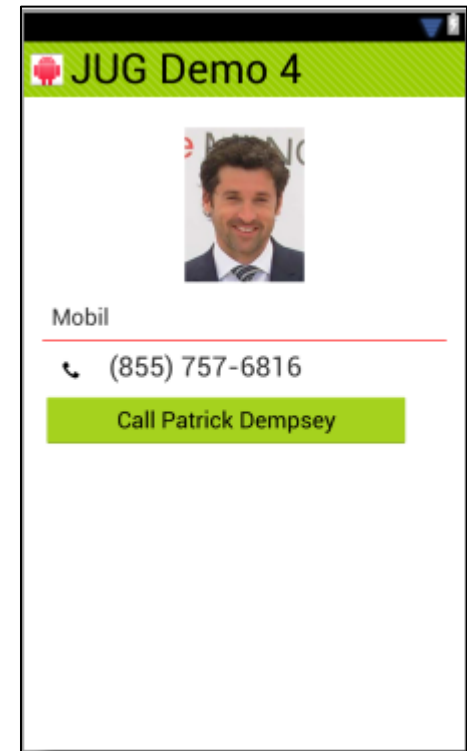




HTC Hero,  
mdpi-Display



Nexus S /  
HTC Desire,  
hdpi-Display

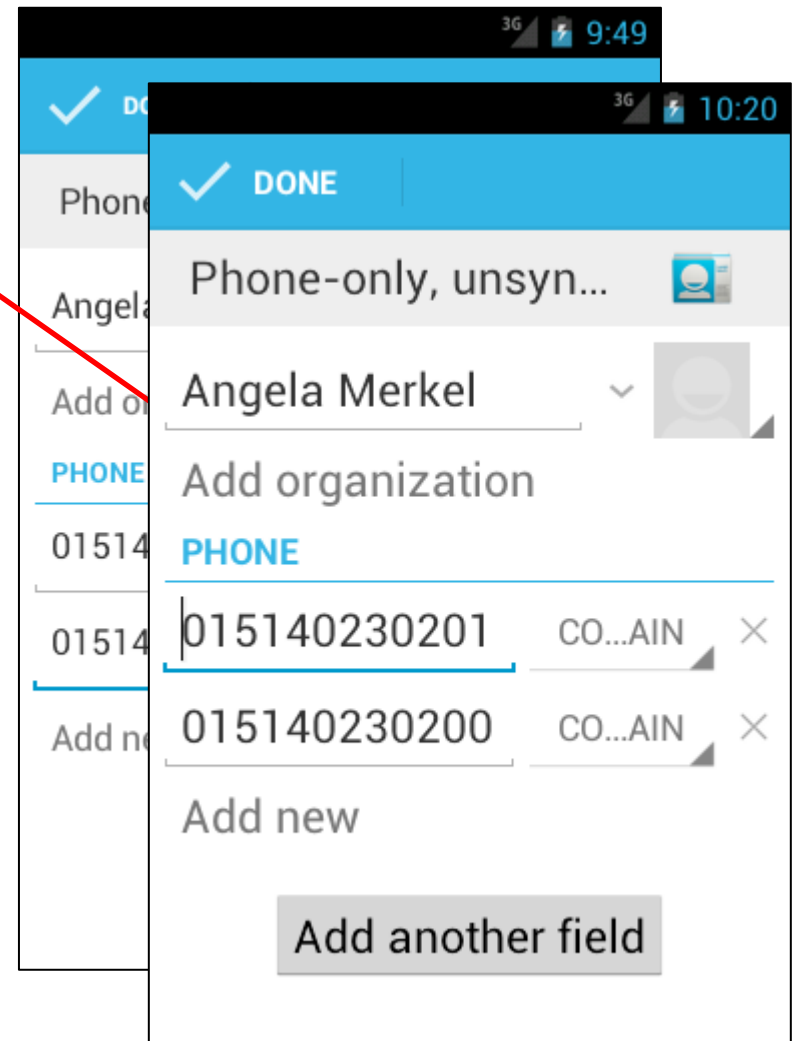


Nexus 4,  
xhdpi-Display

Horizontale Anordnung von Spinner  
und Texteingabefeld  
→ Text wird nicht vollständig  
dargestellt

Beim Entwurf hinterfragen:

- Horizontale Anordnung sinnvoll?
- Verwendung von langen Texten notwendig?
- **Achtung:** Bei Android kann die Schriftgröße in den Einstellungen verändert werden → Trägt das Design hierfür?



- Geräteklassen bilden
- Design gegen Mindestbreite der Klasse in dp entwickeln
- Grundsätzlich Skalierbarkeit vorsehen (match\_parent)

Smartphone

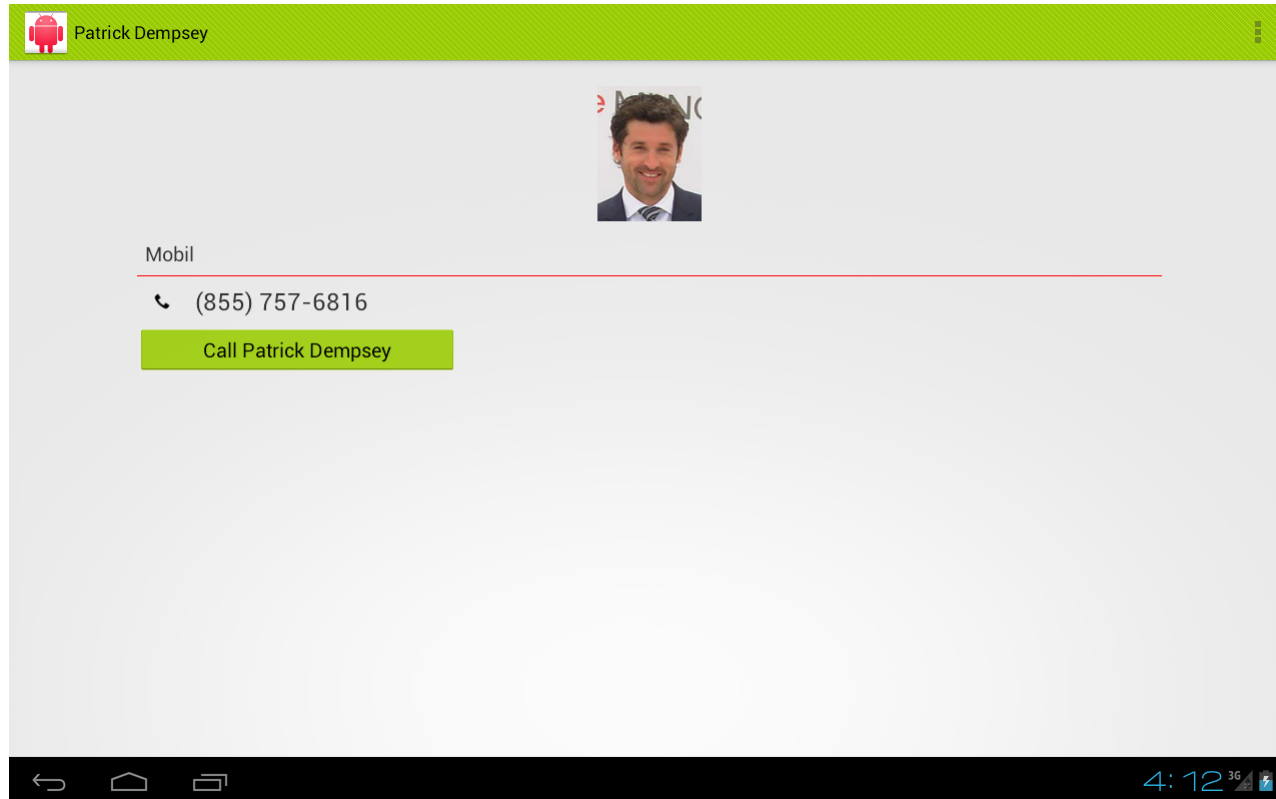


Tablet



Bucket	Display-Größe	Ordner-Name
xlarge	min. 960 dp x 720 dp	res/layout-xlarge
large	min. 640 dp x 480 dp	res/layout-large
normal	min. 470 dp x 320 dp	res/layout
small	min. 426 dp x 320 dp	res/layout-small
sw600dp	<u>s</u> mallest <u>w</u> idth at least 600dp → Tablet	res/layout-sw600dp
sw720dp	<u>s</u> mallest <u>w</u> idth at least 600dp → Tablet (10")	res/layout-sw720dp

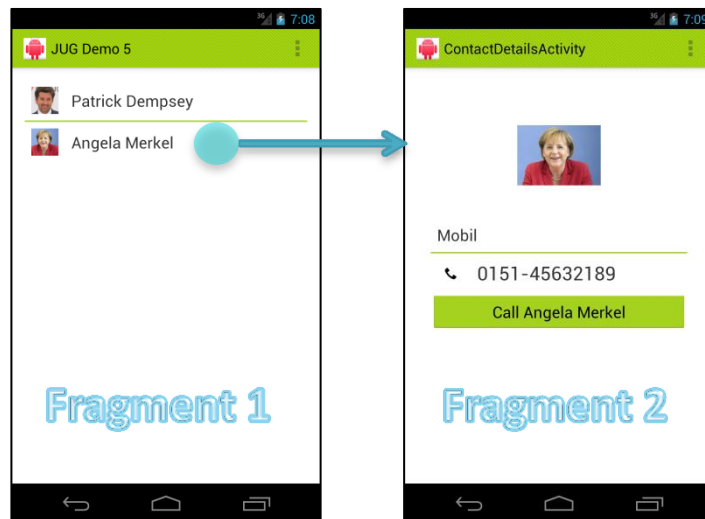
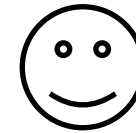
Seit Android 3.2



Tablet, mdpi-Display



## Wiederverwendbare Komponenten



- Google Support-Library v4
- Fragments-Technologie auch auf Geräten mit älterer OS-Version zum Einsatz bringen
- Allerdings: Kompilierung gegen API-Version > 3.2 nötig
- Entwickler muss sicherstellen, dass Aufrufe konsequent in die Support-Library gemacht werden
- ActionBar-Support gibt es über die v7-Support-Library
- <http://developer.android.com/tools/support-library/index.html>

- // [braun@accso.de](mailto:braun@accso.de)
- // [twitter.com/susannebraun](https://twitter.com/susannebraun)
- // [github.com/susannebraun](https://github.com/susannebraun)



*Individuelle  
Kernsysteme*

*Beschleunigte  
Softwaretechnik*

*Team≡*

Begeisterung für die  
anspruchsvollen Aufgaben unserer Kunden

**Accso≡**  
Accelerated Solutions