WP-SmartHome
v2-Einführung in Android

Prof. Dr. Ing. Birgit Wendholt

Was ist Android?

- ein Software-Stack für mobile Geräte bestehend aus
  - Betriebssystem
  - Middleware
  - Kernapplikationen

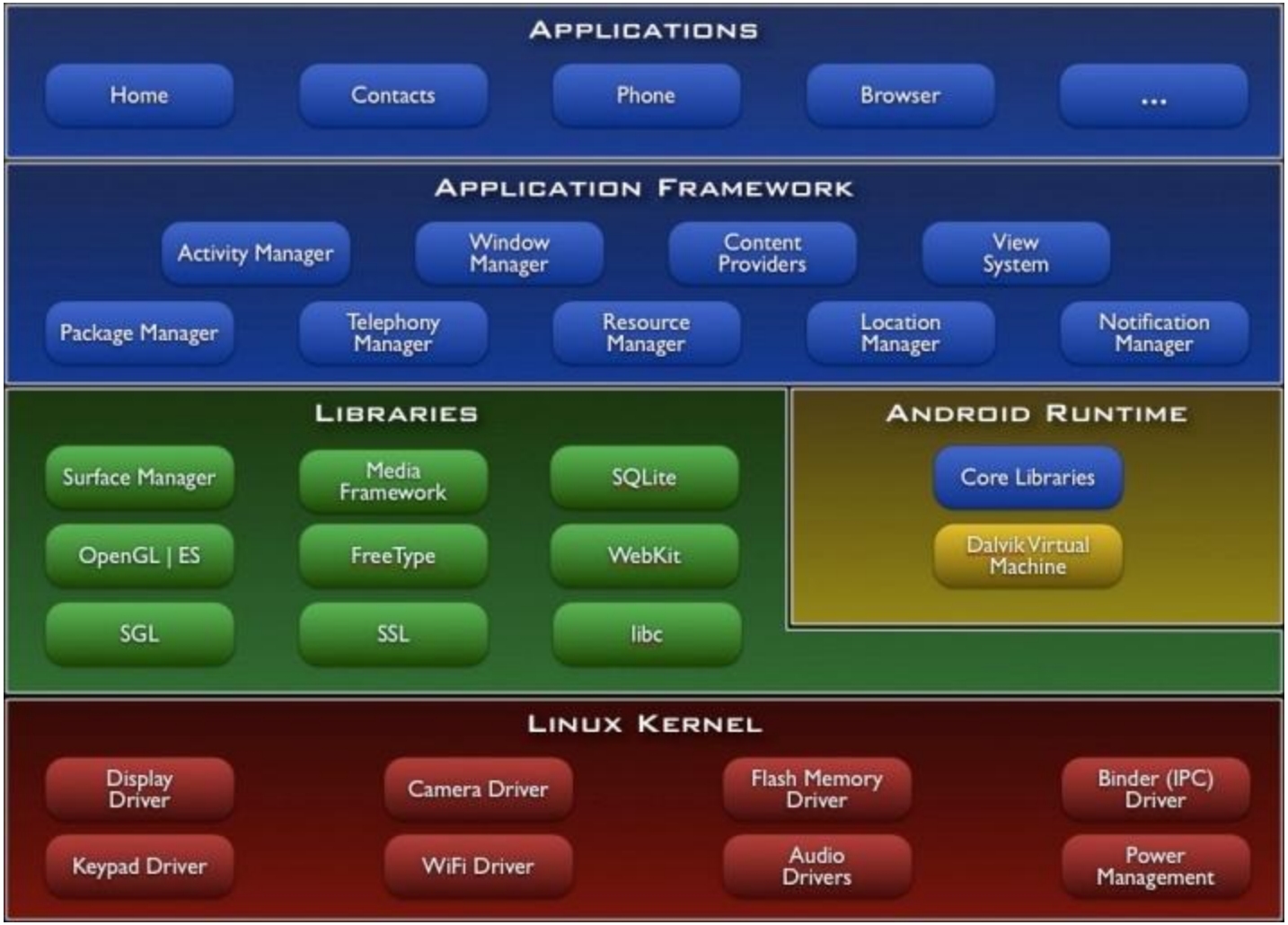
- ein SDK mit Tools und APIs für die Entwicklung von Anwendung auf der Android Platform in Java.

- eine Laufzeitumgebung für Java-Applikationen auf mobilen Geräten. (→ Dalvik Virtual Machine). Dalvik führt mehrere virtuelle Maschinen effizient auf mobilen Geräten aus und ist für Geräte mit geringem Speicher optimiert.

Was enthält Android?

- **Applikations-Framework** für die Erstellung von Applikation
- **Dalvik virtuelle Maschine** optimiert für mobile Geräte
- **Integrierter Browser** basierend auf dem Open Source WebKit
- **Optimierte Graphik für** 2D; 3D Graphik basierend auf der OpenGL ES 1.0/2.0 Spezifikation
- **SQLite** für das Speichern strukturierter Daten
- **Medien Unterstützung** für gängige Audio-, Video-, und Bildformate (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)
- **Telefonie**
- **Bluetooth, EDGE, 3G, und WiFi**
- **Kamera, GPS, Kompass und Beschleunigungssensor**
- **Entwicklungsumgebung** Geräte Emulator, Debugging Tools, Speicher und Performance Profiling, Eclipse Plugin

Android Architektur



Bestehende Android Applikationen

- Beispiele für Applikationen
  - Email client
  - SMS program
  - Calendar
  - Karten
  - Browser
  - Kontakte, etc.

- Alle Applikationen sind in Java geschrieben.

Das Applikations-Framework

- Unterstützung für
  - das Ansprechen von Gerätehardware,
  - Ortsinformation (GPS),
  - Starten von Hintergrundprozessen,
  - Setzen von Alarmen,
  - Hinzufügen von Notifikationen zur Statusbar, etc.
- Zugang zu den Framework API's der Kernapplikationen.
- Erweiterbare Views für Listen, Text, Buttons, Webbrowser etc.
- Content Provider für das Austauschen von Daten zwischen Applikationen, z.B. Kontakte
- Resource Manager für das Verwalten lokalisierter Strings, von Graphiken und Layout Dateien, etc.
- Notification Manager für das Anzeigen von Alerts in der Statusbar.
- Einen Activity Manager, der den Lebenszyklus von Applikationen, und einen Backstack, der die Navigation zwischen Aktivitäten verwaltet.

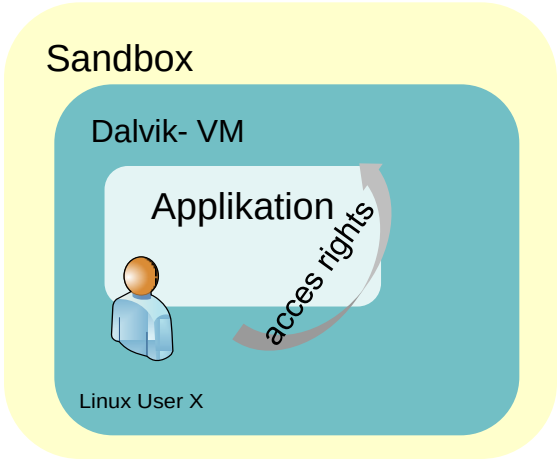
C/C++ Bibliotheken

- **System C Bibliothek** - eine von BSD abgeleitete Implementierung der Standard C Biblithek (libc)
- **Media Bibliotheken** - für das Abspielen und Aufzeichnen von gängigen Audio und Video Formaten, sowie Bilddateien wie MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- **Surface Manager** – für das Display Subsystem und die 2D / 3D Graphik
- **LibWebCore** – ein Webbrowser, als Applikation oder eingebetteter Browser
- **SGL** – die 2D Graphik-Engine
- **3D Bibliotheken**- basierend auf den OpenGL ES 1.0 APIs
- **FreeType** – für das Rendern von Bitmaps und Vector Fonts
- **SQLite** – eine “leichtgewichtige” relationale Datenbank

Android Runtime

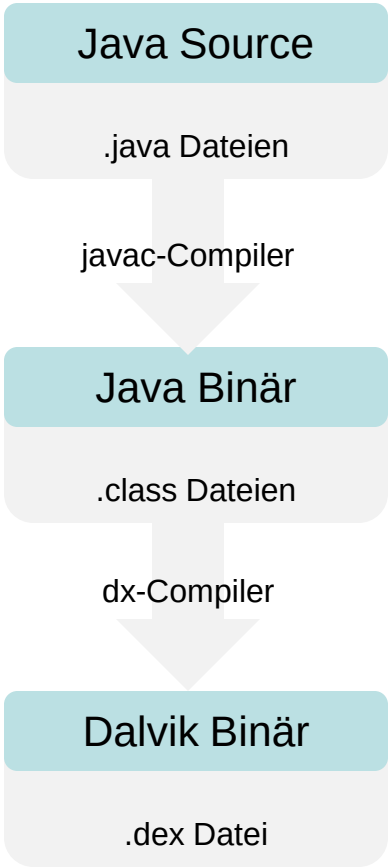
Kernbibliotheken und Laufzeit Basics

- Android enthält eine Reihe von Bibliotheken, die die Funktionalität der Kern-Bibliotheken der Java Programmiersprache nahezu vollständig anbieten.
- Jede Android Applikation läuft in einem eigenen Prozess mit einer eigenen Instanz der Dalvik Virtual Machine.
- Dalvik wurde so geschrieben, dass ein Gerät mehrere VMs effizient ausführen kann.



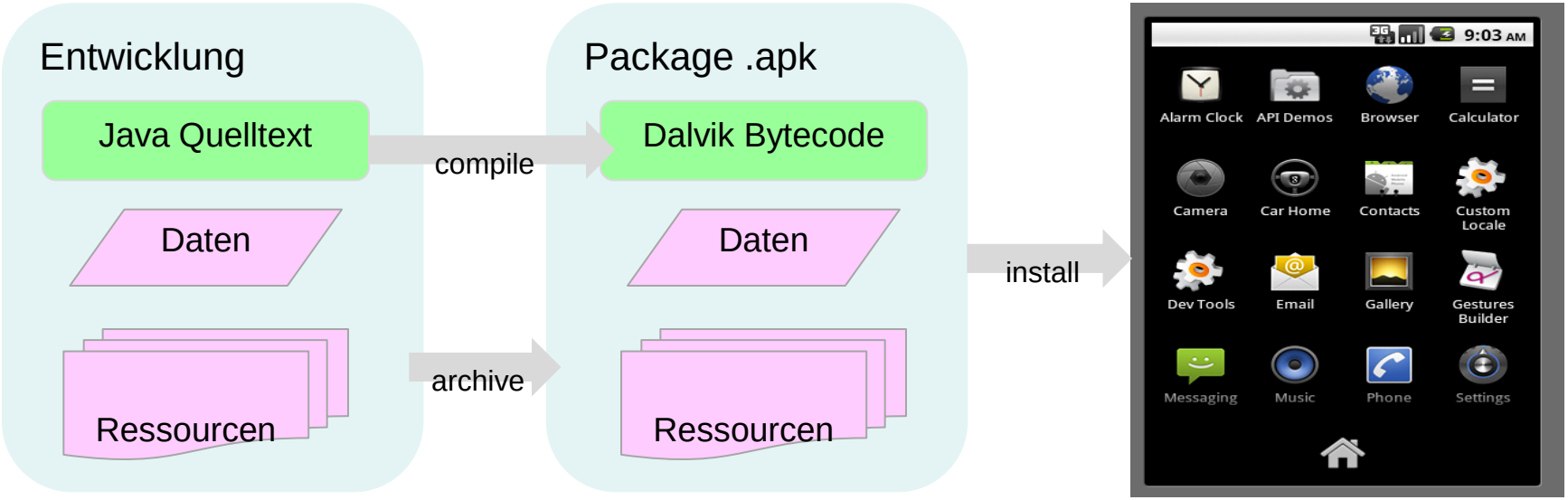
Dalvik Virtual Machine (*Dalvik VM* oder *DVM*)

- eine für mobile Geräte entwickelte virtuelle Maschine auf modernen Prozessoren (z.B. ARM-Mikroprozessoren) → Entwickler: Dan Bornstein (Google)
- Ressourcen- und Laufzeitoptimiert
- mit eigenem Compiler *dx,* der Java-Binärdateien (.class) in das Dalvik Format (.dex) übersetzt und dabei optimiert
- verwendet Linux Systemdienste für Threading, Low Level Speichermanagement, Security, Prozessverwaltung, Netzwerkdienste, Gerätetreiber



Applikationen

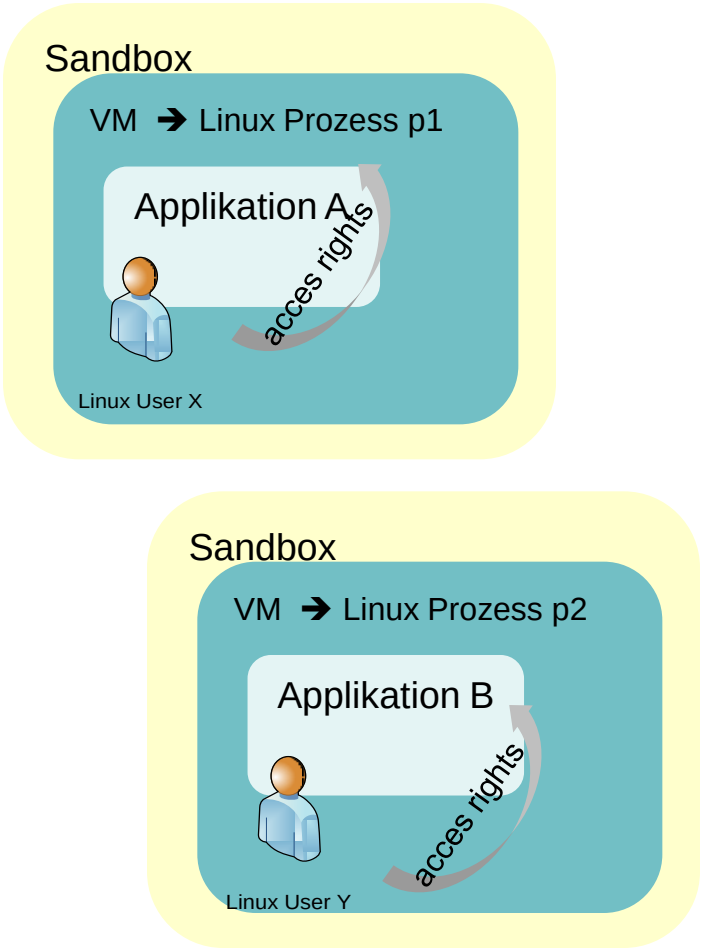
- in Java geschrieben
- durch das Android SDK kompiliert und mit allen Daten und Ressourcen in einem *Android package* (Suffix .apk) archiviert.
- eine .apk Datei wird von Android-fähigen Geräten als Applikation installiert.



Security Sandbox Modell

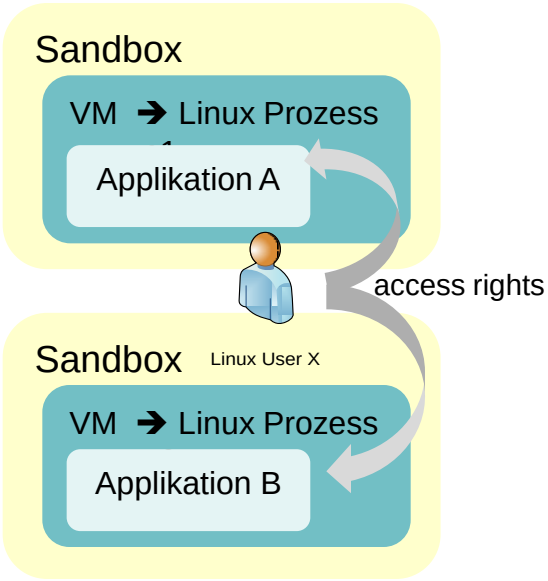
„principle of least privilege“

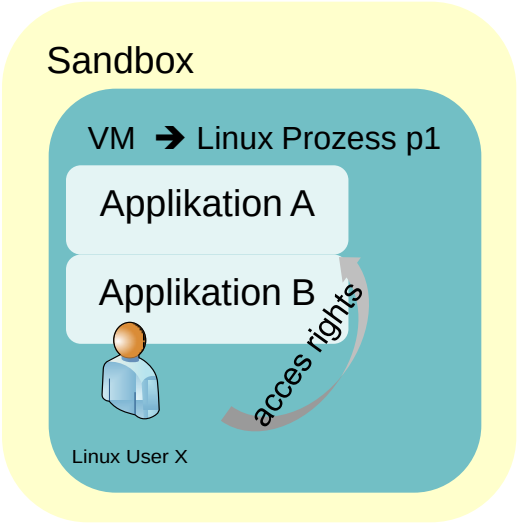
- Jede Applikation ist ein eigener Linux-Benutzer.
- Jede Applikation hat eine eindeutige Linux User ID.
- Nur diese User ID hat Zugriff auf die Dateien der Applikation.
- Jeder Prozess hat eine eigene virtuelle Maschine. Applikationen sind voneinander isoliert.
- Jede Applikation läuft in ihrem eigenen Linux Prozess. Das Android System ist verantwortlich für das Starten und Stoppen der Prozesse.



Security Sandbox -Ausnahmen

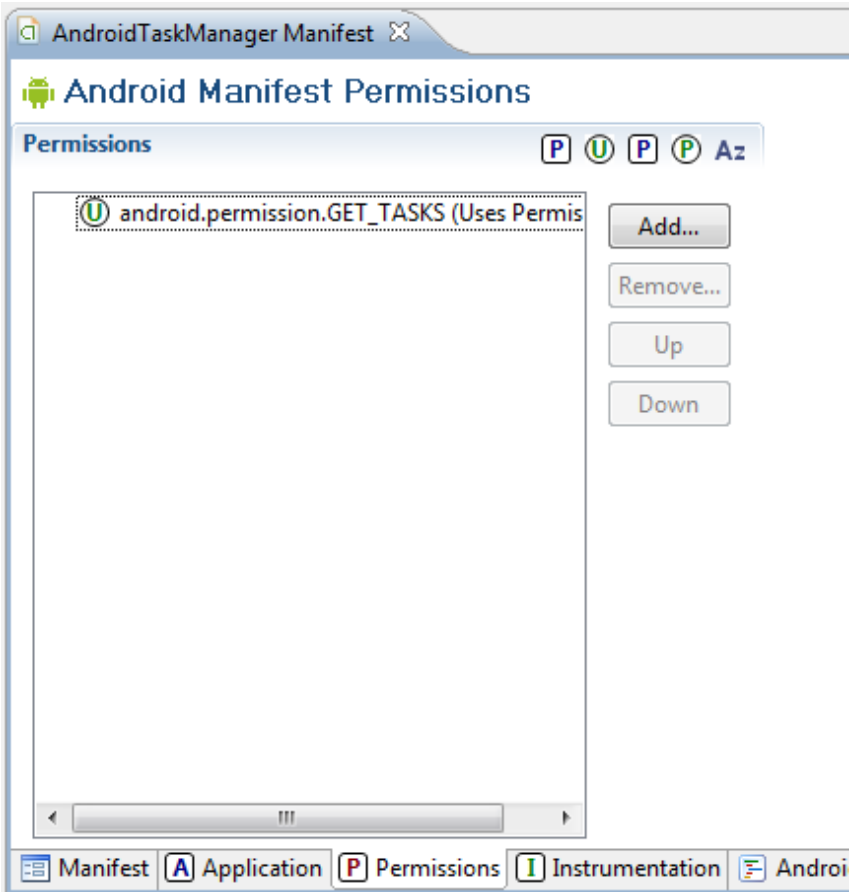
- Verschiedene Applikationen können die gleiche Linux User ID haben ➔ wechselseitiger Zugriff auf Dateien
- Applikationen mit gleicher User ID können in derselben VM ausgeführt werden.





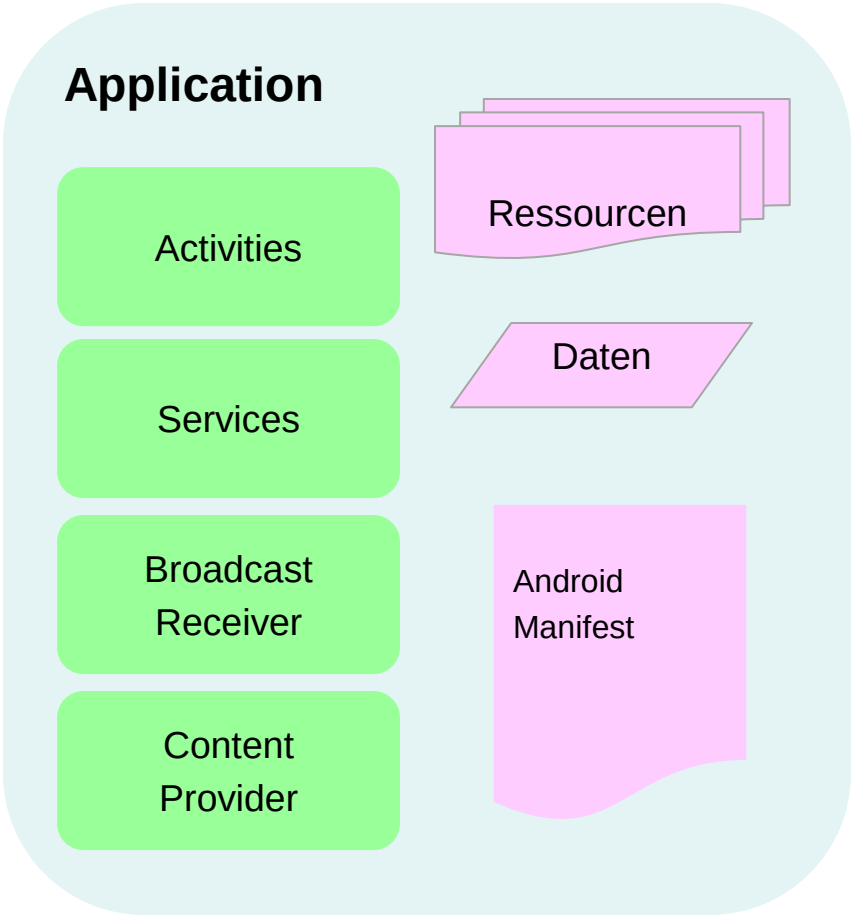
Security Sandbox - Ausnahmen

- Applikationen können Zugriffs-Berechtigungen für Daten und Dienste eines Android Gerätes anfordern:
  - für Kontakte
  - SMS Nachrichten
  - SD Karte
  - Kamera,
  - Bluetooth, etc.
- Berechtigungen müssen dem Benutzer bei der Installation vergeben werden.



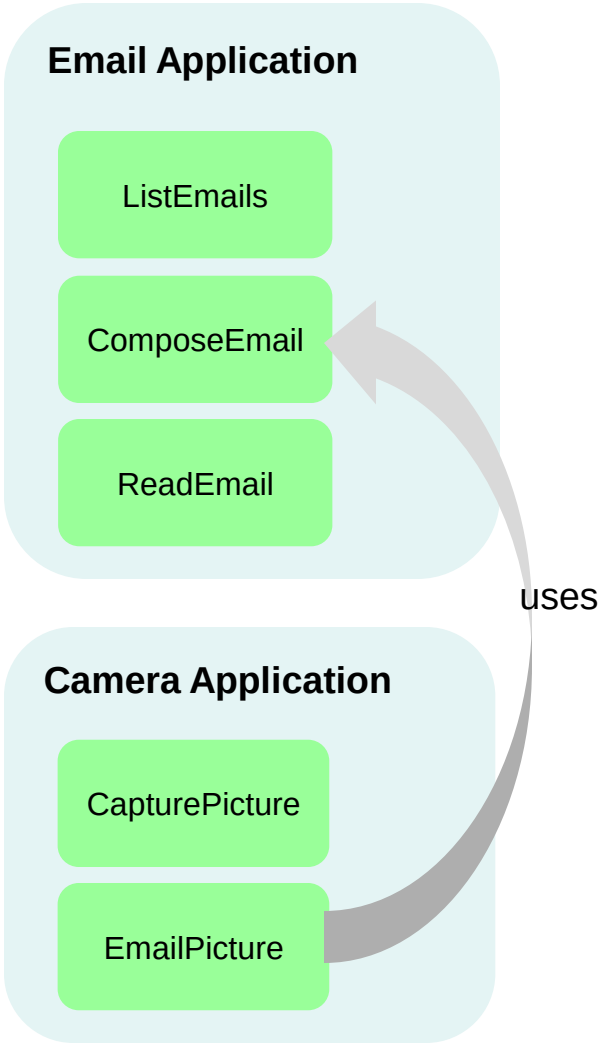
Komponenten einer Android Applikation

- = Bausteine einer Android Applikation.
- Jeder Komponententyp definiert einen spezifischen Zugang, über den das Android System mit einer Applikation interagieren kann.
- nur einige Komponenten unterstützen Benutzerinteraktion.
- **Komponenten:** Activities, Services, Broadcast Receivers, Content Providers
- **Ressourcen einer Applikation**: Dateien, die keinen Programmcode enthalten
- **Android Manifest**: Konfiguration einer Applikation



Activities

- Eine Activity repräsentiert (in der Regel) einen einzelnen Bildschirm.
- **Beispiel**: Email Applikation mit den Activities ListEmails, ComposeEmail, ReadEmail
- Alle Activities sind unabhängig voneinander.
- Applikationen können einzelne Activities einer anderen Applikation nutzen.
- **Beispiel:** Eine Kamera Applikation kann die Activity zum Schreiben einer Email nutzen und Fotos versenden.



→ Activity und Activities

Services

- Ein *Service* ist eine Komponente, die im Hintergrund läuft, um langlaufende Operationen oder Operationen für entfernte (remote) Prozesse durchzuführen.
- Ein Service hat **keine** Benutzer-Schnittstelle.
- Beispiele:
  - ein Service, der im Hintergund Musik abspielt
  - ein Service, der Videos über das Netzwerk lädt, während der Benutzer im Browser die Börsendaten liest.
- Andere Komponenten, wie z.B. Aktivitäten können einen Service starten, oder ihn binden, um mit ihm zu interagieren.
- Ein Service wird als Subklasse von Service implementiert → Services

Broadcast Receiver

→ eine Komponente, die auf system-weite Broadcasts (Meldungen) reagiert.

- Viele der Meldungen werden vom System generiert, z.B.:
  - Bildschirm wurde ausgestellt
  - Batterie ist fast leer,
  - Foto wurde geschossen

- Applikationen können ihrerseits Meldungen initiieren, z.B. um anderen Applikationen mitzuteilen, dass Daten über das Netz auf das Gerät geladen wurden.

- Broadcast Receiver haben keine Benutzer-Schnittstelle, können aber einen Hinweis für die Statuszeile generieren (→ StatusBar Notifikation), um den Benutzer auf ein Ereignis hinzuweisen.

- Broadcast Receiver sind Kommunikations-Fenster für andere Komponenten und sollten selber nicht zuviel Applikationslogik beinhalten.

- Jede Meldung wird als → Intent ausgeliefert (→ BroadcastReceiver)

ContentProvider

→ verwaltet Daten für den gemeinsamen Zugriff von Aktivitäten unterschiedlicher Applikationen

- Daten einer Applikation können im Dateisystem, einer SQLite Datenbank, im Web, etc. persistent abgelegt werden. Mit Hilfe des Content Provider können andere Applikationen diese Daten lesen oder modifizieren.

- Beispiel:
  - Android Content Provider, der die Kontaktdaten von Benutzern verwaltet.
  - Jede Applikation, mit den entsprechenden Berechtigungen kann Konaktdaten lesen und schreiben.

    (→ ContactsContract.Data)

- Content Provider werden auch eingesetzt, wenn private Daten einer Applikation anderen Applikationen zur Verfügung gestellt werden sollen.

  (→ Note Pad Beispiel)

- Subklassen von ContentProvider müssen eine Reihe von API Methoden implementieren, um anderen Applikationen Transaktionen mit den Daten zu erlauben. (→ Content Providers)

Aktivieren von Komponenten

Aktivitäten, Services und Broadcast Receiver

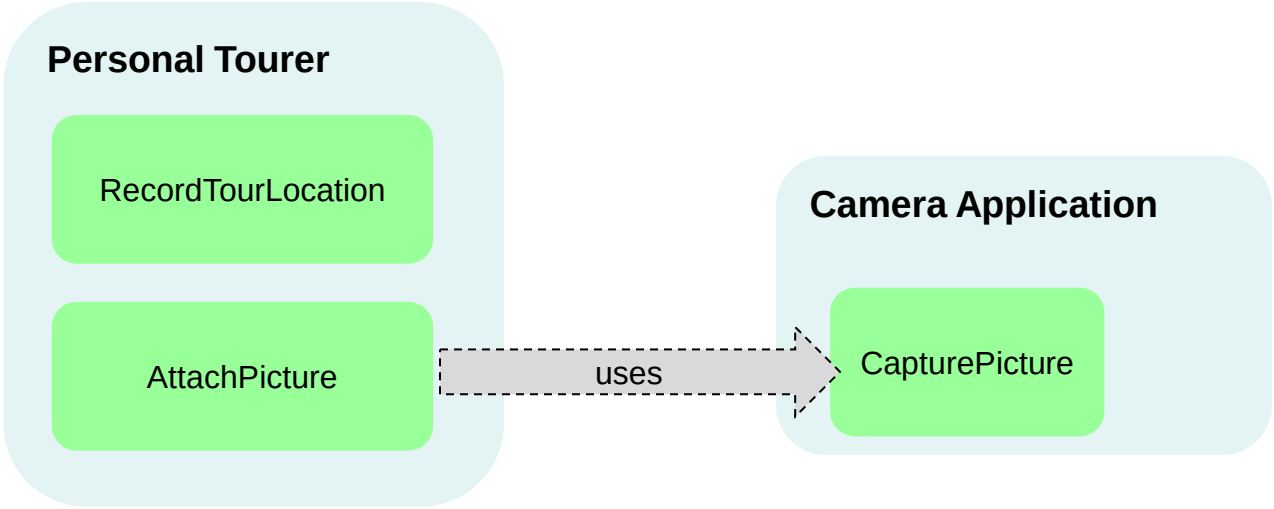
→ aktiviert durch das Versenden asynchroner Nachrichten, genannt *Intent*

Content Provider

→ aktiviert über eine Anfrage an den ContentResolver, der alle direkten Transaktionen mit dem Content Provider für Applikationen behandelt.

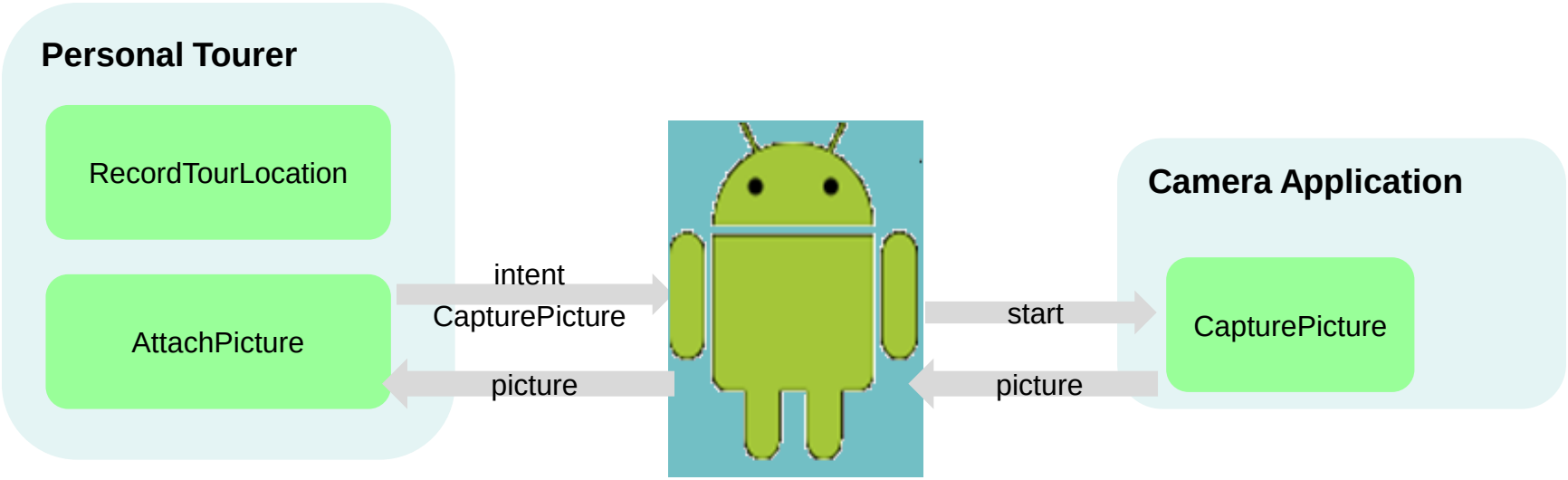
Aktivieren von Komponenten über Intents -1

- Jede Applikation kann Komponenten einer anderen Applikation starten.
- **Beispiel:** Personal Tourer möchte Fotos an POI's anhängen und dazu die Kamera Applikation nutzen.
- Es ist möglich eine Aktivität einer anderen Applikation zu nutzen, ohne den Source Code der anderen einzubinden. Wenn diese endet, kann deren Ergebnis an die eigene Applikation übergeben werden (im Beispiel Fotos).



Aktivieren von Komponenten über Intents - 2

- Android führt jede Applikation in einem separaten Prozess mit speziellen Berechtigungen aus ➔ Zugriff auf Komponenten anderer Applikationen eingeschränkt.
- Nur Android hat die Berechtigung Komponenten zu starten.
- Komponenten senden an Android eine Nachricht mit der Absicht (*intent*) eine Komponente zu starten.
- Android übernimmt die Aktivierung der Komponente und den Transport der Ergebnisse.



Intents

- Intents verbinden zur Laufzeit einzelne Komponenten innerhalb und zwischen Applikationen.
- Ein Intent Objekt definiert eine Nachricht, um eine Komponente **explizit** oder über einen Komponententyp **implizit** zu aktivieren.
- **Broadcast Receiver:**
  - Der Intent enthält einen Meldungstext über das aufgetretene Ereignis, z.B. “battery low”.
- **Aktivitäten und Services:**
  - Der Intent definiert die Aktion und ggf. eine URI für Daten, die in der Aktion bearbeitet werden sollen. (z.B. eine Aufforderung ein Video über das Web zu laden).
  - Wenn Aktivitäten gestartet werden, die ein Ergebnis berechnen, wird dieses Ergebnis als Intent zurückgeliefert. Z.B. ließe sich ein Intent erzeugen, der den Benutzer zur Auswahl eines Kontaktes auffordert, der Ergebnis-Intent würde dann eine URI enthalten, die den selektierten Kontakt adressiert.
  - Kommunikation über Intents ist asynchron

Intent Aktionen und Filter - 1

Intent Aktionen

- Platzhalter für Aktionen und ggf der assoziierten Daten.
- Android sucht nach Komponenten auf dem Gerät, die die Aktion ausführen können und startet eine Komponente. Gibt es mehrere Komponenten, dann muss der Benutzer eine auswählen.
- Android vergleicht dazu die *Aktion*, die mit einem Intent verschickt wird, mit den *Intent Filtern* in den Manifesten der installierten Applikationen.

→Intents and Intent Filters.

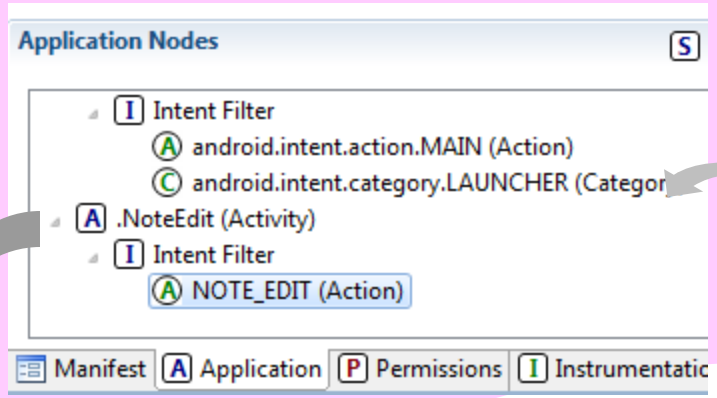
Intent Filter

- Applikationen beschreiben mit *Intent Filtern* die Fähigkeiten einer Komponente in ihrem Manifest.
- **Beispiel**:
  - Eine ***NotePad*** Applikation mit der Aktivität (***NoteEdit***) deklariert einen Intent Filter im Manifest für ***NOTE_EDIT*** Intents.
  - Die Aktivität ***Appointments*** der ***Calendar*** Applikation erzeugt einen Intent mit der ***NOTE_EDIT*** Aktion.
  - Android machted ***NOTE_EDIT*** gegen die ***NoteEdit*** Aktivität und startet diese.

Intent Filter und Aktionen - 2

NotePad Application

Manifest



Calendar Application

YearTable

MonthTable

Appointments

1: entry at installation time

Intent Filter Pool

NOTE_EDIT → NoteEdit (Activity)

…

…

2: send intent: NOTE_EDIT

3: lookup activity

4: launch activity

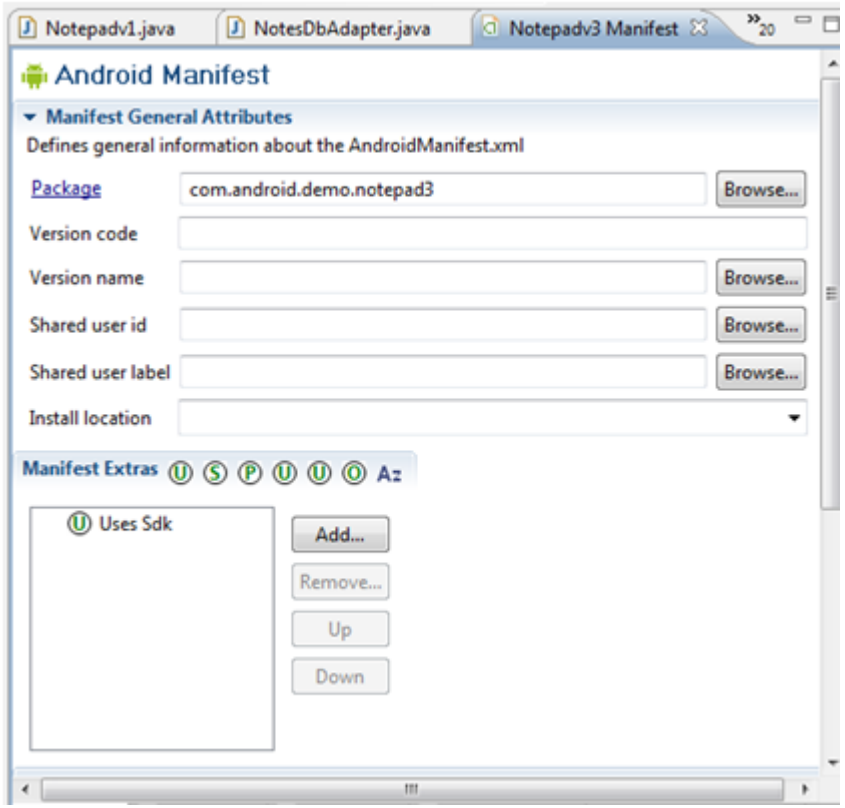
Beispiel: Kommunikation zwischen Aktivitäten

- → Vorgriff auf das Notepab Beispiel
- → Demo der Applikation und Code Walkthrough

Notepad Activity ***NoteEdit Activity*** ***Notepad Activity***

Das Android Manifest

- Komponenten der Applikation müssen in der Datei ***AndroidManifest.xml***, dem “Manifest”, deklariert werden.
- Das Android System liest die Manifest-Datei vor der Ausführung der Komponenten.
- Einstellungen für Applikationen:
  - benötigte Rechte, z.B. Zugriff auf Internet-, Systemdienste oder die Kontakte
  - minimaler API Level
  - Hardware- und Software- Eigen schaften, wie Kamera, Multitouch Screen etc.
  - API Bibliotheken, z.B. die Google Maps Library
  - etc.



Konfiguration von Applikationen im Manifest

Hintergrund

- Android unterstützt verschiedene Gerätetypen mit unterschiedlichen Hardware- und Softwaremerkmalen.
- Die Anforderungen einer Applikation müssen vor der Installation gegen die Ausstattung eines Gerätes geprüft werden.
- Applikationen müssen daher ihre Hardware- und Softwareanforderungen im Manifest deklarieren.

Merkmale

- **Bildschirmgröße und Auflösung**
- **Eingabe Konfiguration**: z.B. Tastatur, Trackball
- **Geräteeigenschaften**: z.B. Kamera, Lichtsensor, Bluetooth, eine spezielle Version von OpenGL, Touchscreen, etc.
- **Android Version mit API Level**

Ressourcen einer Applikation

- Eine Applikation besteht aus Java Quelltext und separaten Ressourcen, wie z.B. Bilder, Audiodateien, etc.
- **Adaptierbarkeit**: Alles was sich auf die visuelle Darstellung der Applikation bezieht, wie z.B. Animationen, Menus, Styles, Farben, Layout sollte in XML Dateien deklariert sein.
- Jede Ressource in einem Android Projekt erhält eine eindeutige ID (Integer), über die die Ressource angesprochen werden kann.
- Ressourcen werden in einem Unterverzeichnis von ***res/*** abgelegt.

→ Application Resources

- **Beispiel**:
  - –Eine Applikation speichert das Bild ***logo.png*** im Verzeichnis ***res/drawable***/
  - –Das Android SDK generiert daraus eine resource ID mit Namen ***R.drawable.logo***.
- Android hat vordefinierte *Qualifier* um alternative Ressourcen zu verwalten. Ein Qualifier wird an den Verzeichnisnamen angehängt.
- **Beispiel**:
  - –Buttons werden im Portrait Modus vertikal, im Landscape horizontal angeordnet
  - →zwei Verzeichnisse für die zwei Layouts mit den entsprechenden Qualifiern
  - → Anroid selektiert das Layout abhängig von der Orientierung des Gerätes