Android: Time Tracker

Florian Diederichs

Gliederung

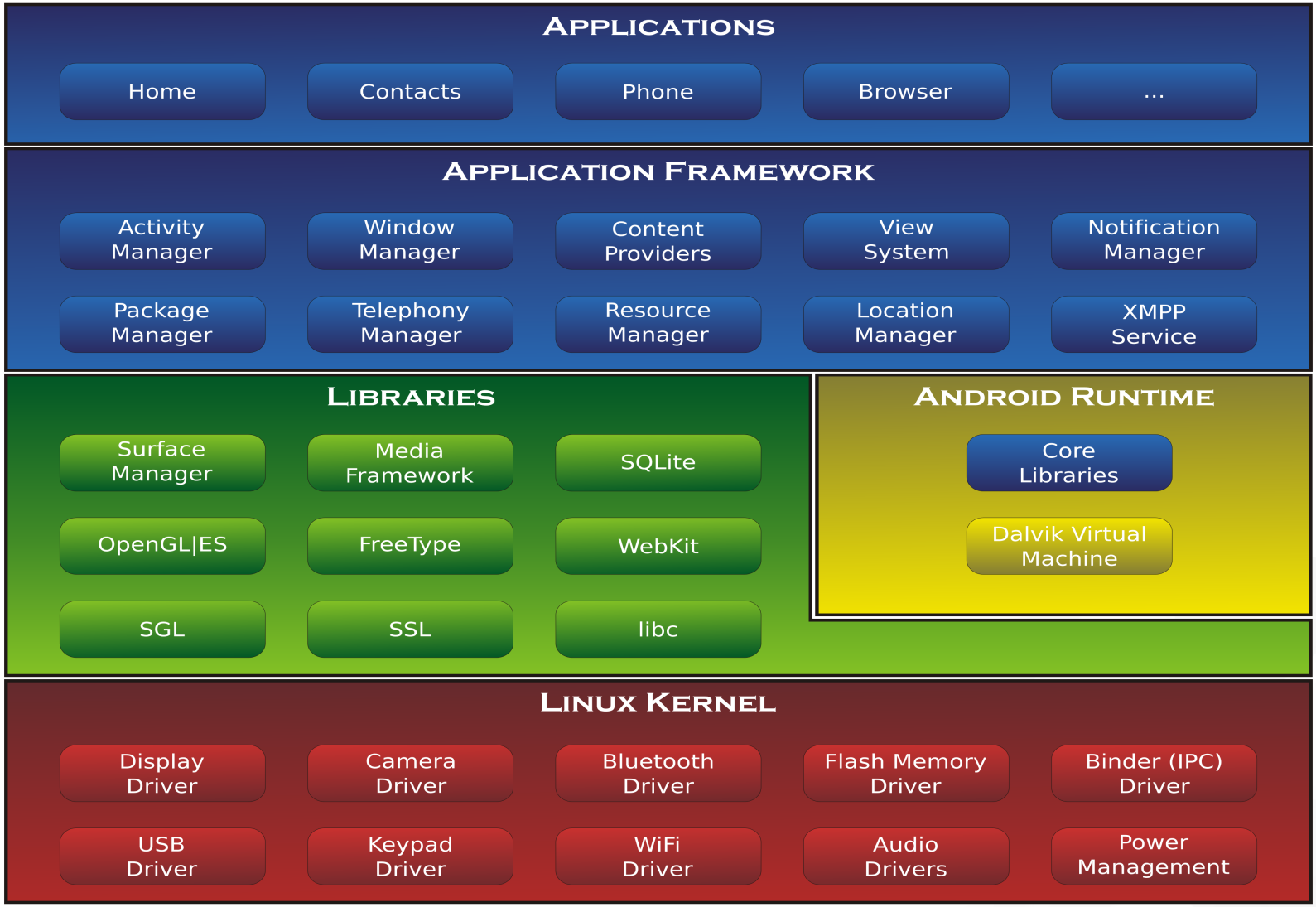
- Android Einführung
- Time Tracker App
- Aufsetzen der Entwicklungsumgebung
- Bearbeitung der Aufgaben

1. Android Einführung

Einleitung

- Betriebssystem und Software-Plattform
- Entwickelt von Google
- Ursprünglich für mobile Endgeräte (Smartphones)
- Anfangs ARM-Prozessoren, mittlerweile auch andere
- Einsatz heute auch auf:
  - Tablets
  - Netbooks
  - Auto-Infotainment
  - Spielekonsolen
  - Set-Top-Boxen

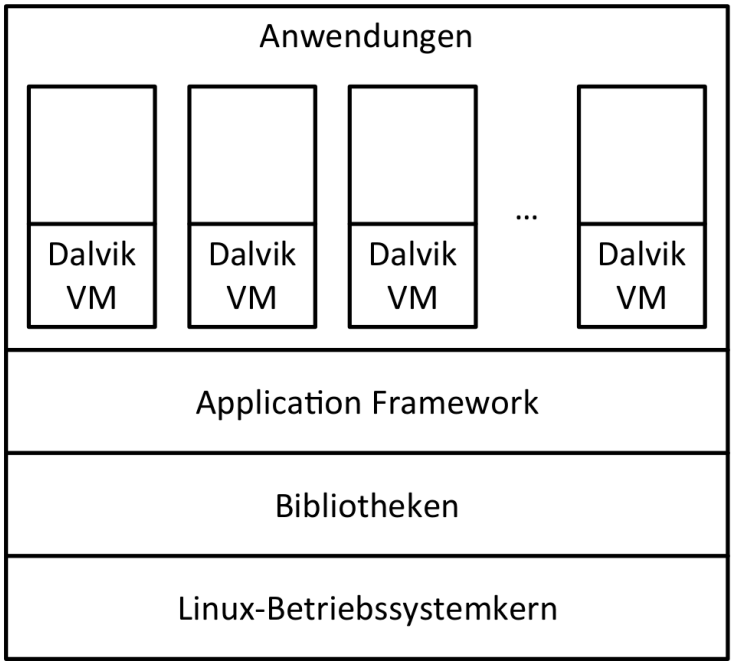
Android Architektur



Quelle: http://de.wikipedia.org/wiki/Datei:Android-System-Architecture.svg

Dalvik Virtual Machine

- Ersetzt die JVM
- Hat einen eigenen Bytecode, Cross-Compiler
- Optimiert für mobile Prozessoren
- Ressourcenschonend, eigene Instanz für jede App



Quelle: Rainer Oechsle, Java-Komponenten, Seite 270

Android Anwendungen

- Terminologie
  - KGSE Komponenten != Android Komponenten
  - KGSE Komponenten == Android Anwendungen
  - Android Komponenten == Android Bausteine

- Android Bausteine
  - Activities
  - Services
  - Content Providers
  - Broadcast Receivers

- Weitere wichtige Bestandteile
  - Datenbankschnittstelle
  - Intents
  - Android-Manifest
  - Layouts
  - Strings
  - Grafiken

Activities

- Sichtbare Teile einer Android Anwendung
- Eine Bildschirmseite entspricht einer Activity
- Erstellung durch Ableitung der Klasse *Activity*
- Implementieren bestimmter Event-Handler (Up-Calls)
- Einführung eigener Event-Handler für Reaktion auf Benutzereingaben
- Layout-Konfiguration in XML-Dateien

Hello World App

Aufruf mehrerer Activities

- Die meisten Apps bestehen aus mehreren Activities
- Activities können andere Activities starten
  - Derselben Anwendung
  - Anderer Anwendungen

Multi Activity App

Intents

- Mechanismus zum Austausch zwischen Android Bausteinen
- Drei Hauptanwendungsfälle
  - Starten einer Activity
  - Starten eines Service
  - Absenden eines systemweiten Broadcasts
- Intents sind der ,,Klebstoff” zwischen den Bausteinen einer oder mehrerer Android Anwendungen
- Intents können unterschiedliche Zusatzinformationen erhalten und an den Zielbaustein übergeben
- Explizite und Implizite Intents

Intents App

Activity Start durch Explizite Intents

- Zum Aufruf einer Activity innerhalb derselben Anwendung
- Direkte Angabe der Klasse

```
Intent intent = new Intent(this, SecondActivity.class);
```

Activity Start durch Implizite Intents

- Aufgerufene Activity kann sich innerhalb oder ausserhalb der eigenen Anwendung befinden
- Angabe eines *ACTION* Parameters
- Für den individuellen *ACTION* Parameter können mehrere Activities in Frage kommen
- Bereitstellung passender Activities durch Android anhand der Activity-Filter in Android-Manifest Dateien
- Auswahl der gewünschten Activity durch den User

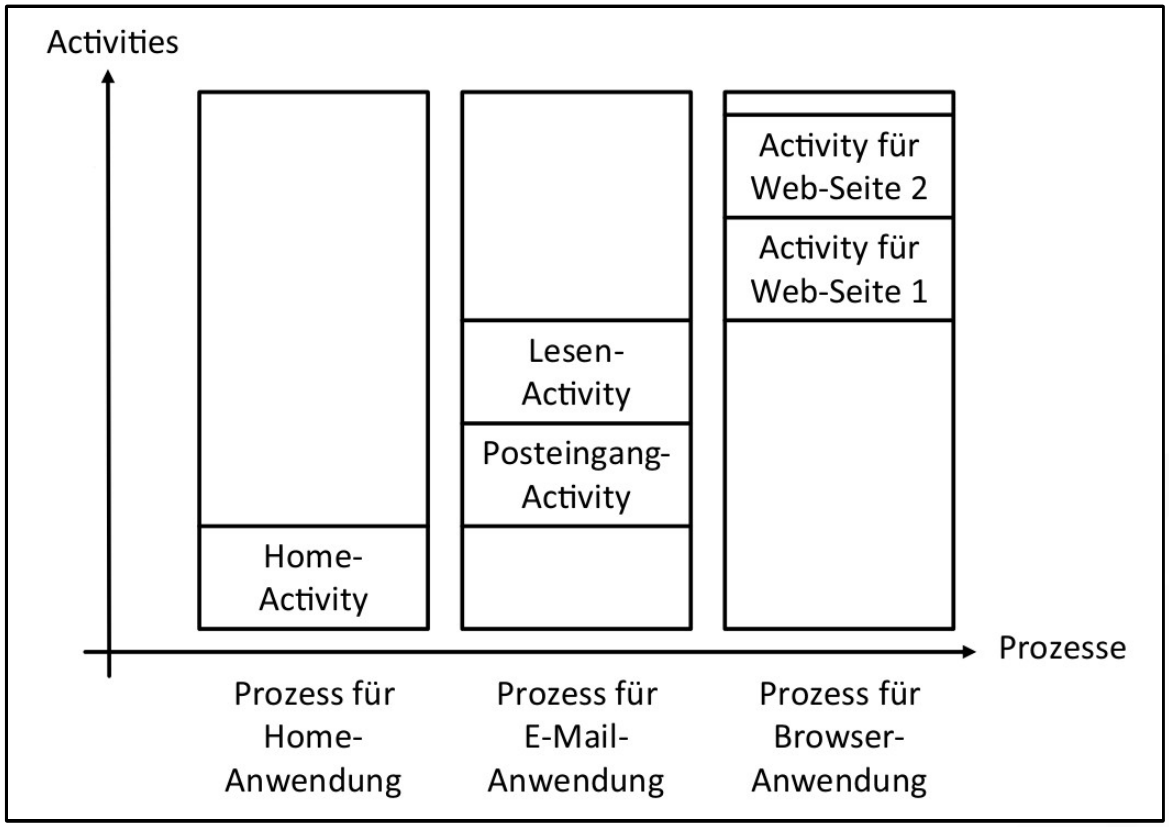
```
Intent intent = new Intent(ACTION);
```

```
<activity android:name=".SecondActivity"/>
    <intent-filter>
        <action android:name="android.acticities.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

Implizite Intents App

Activities und Prozesse

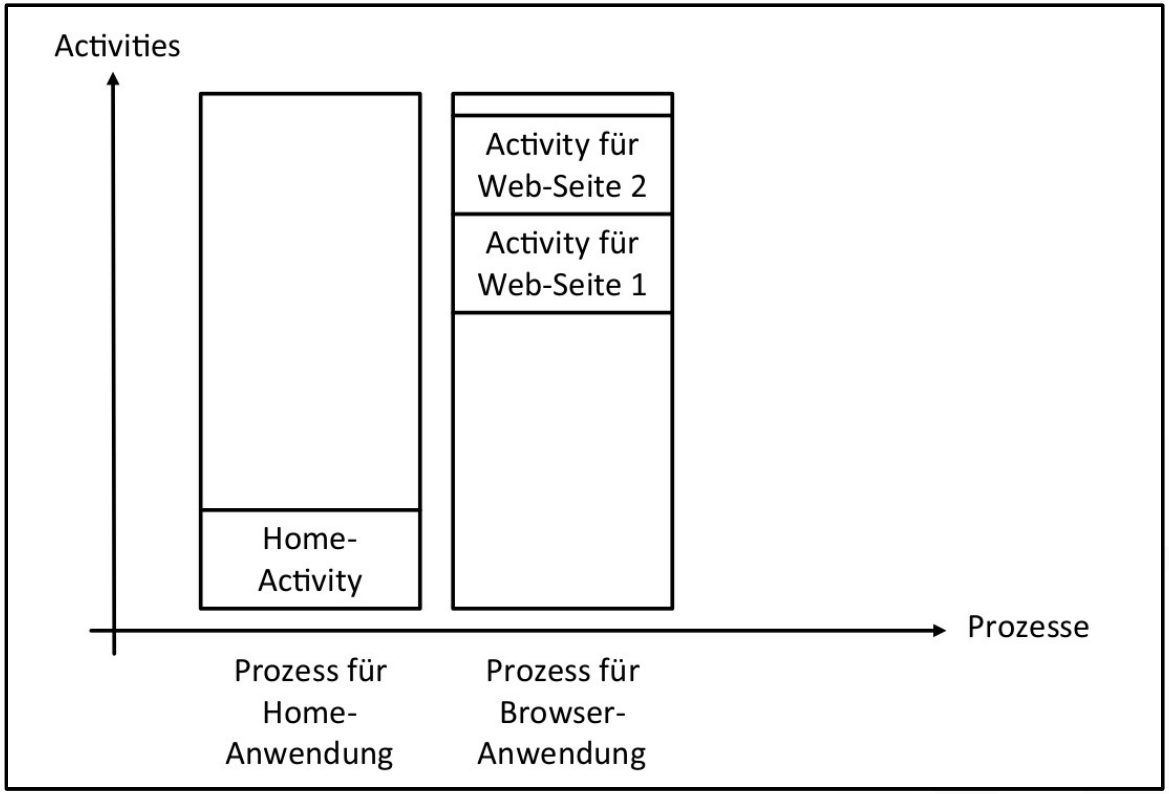
- Jede App läuft in einem eigenen Prozess (mit eigener DVM)
- Start von Activities derselben App im selben Prozess
- Geöffnete Activities bilden einen Keller, ,,Zurück”-Button



Quelle: Rainer Oechsle, Java-Komponenten, Seite 272

Activities und Prozesse (2)

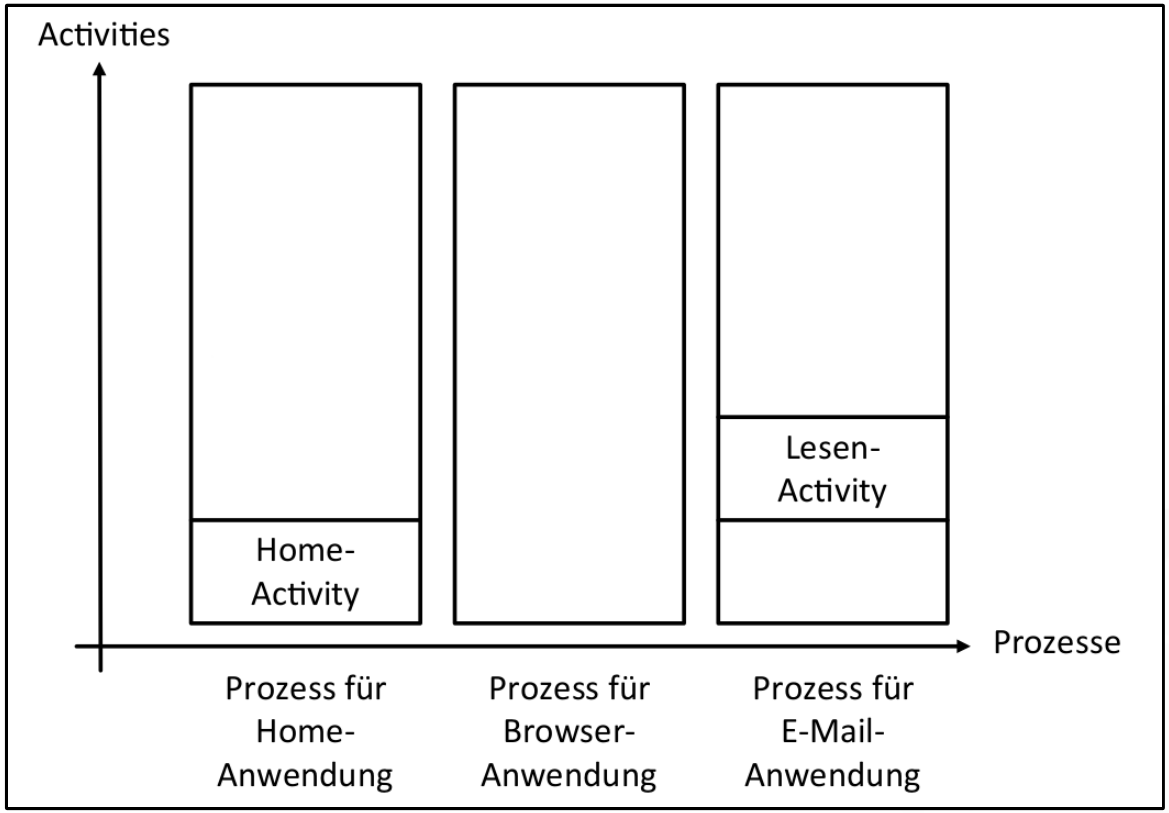
- Bei Ressourcenknappheit schießt Android Prozesse ab



Quelle: Rainer Oechsle, Java-Komponenten, Seite 273

Activities und Prozesse (3)

- Bei Rückkehr zu abgeschossenen Activities sorgt Android für die Wiederherstellung



Quelle: Rainer Oechsle, Java-Komponenten, Seite 274

Threads und Hintergrundoperationen

- Jeder Prozess hat einen MainThread
- Methoden aller Androidbausteine laufen in der Regel im MainThread
- GUI Blockade durch längere Methodenaufrufe
- ANR nach ca. 5 – 10 Sekunden
- Auslagerung längerer Operationen in andere Threads
- Verschiedene Mechanismen für Threadauslagerung
- Mechanismen für GUI-Zugriff aus anderen Threads
  - Selbst programmierte Callback-Mechanismen
  - Hilfsklassen und -methoden
  - Async-Task
- Aufräumen von Threads ist Verantwortung des Entwicklers

Threads und Hintergrundoperationen

- Jeder Prozess hat einen MainThread
- Methoden aller Androidbausteine laufen in der Regel im MainThread
- GUI Blockade durch längere Methodenaufrufe
- ANR nach ca. 5 – 10 Sekunden
- Auslagerung längerer Operationen in andere Threads
- Verschiedene Mechanismen für Threadauslagerung
- Mechanismen für GUI-Zugriff aus anderen Threads
  - Selbst programmierte Callback-Mechanismen
  - Hilfsklassen und -methoden
  - Async-Task
- Aufräumen von Threads ist Verantwortung des Entwicklers

Services

- Keine Benutzeroberfläche
- Ableitung aus Klasse *Service*
- Überschreiben bestimmter Event-Handler (Up-Call)
- Local/Remote Services
- Start über explizite oder implizite Intents
- Auslagerung von Quellcode in eine eigene Komponente
- Service-Methoden laufen per Standard im MainThread
- Gut geeignet für Hintergrundoperationen
- Andere Prozess-Priorität als einfache Thread-Auslagerung
- Gebundene/Ungebundene Services

Broadcast Receivers

- Reagieren auf Broadcast Intents
- Broadcast Intents sind Systemnachrichten:
  - Hochfahren (ACTION_BOOT_COMPLETED)
  - Niedriger Batteriestand (ACTION_BATTERY_LOW)
  - viele weitere
- Können über die Laufzeit der App hinweg existieren
- Konfiguration über Intent-Filter

Content Provider

- Austausch strukturierter Daten über Anwendungs- und Prozessgrenzen hinweg
- Zugriff auf Dateninhalt, nicht die Daten selbst → Abstraktion über internes Datenmodell
- Hauptanwendungen: Datenbank und Dateizugriffe
- Berechtigungssystem für Lese- und Schreibzugriffe (normale Android-Berechtigungen)
- Konfiguration über Android-Manifest
- Klassen welche auf einen CP zugreifen möchten nutzen ein Content Resolver (CR) Objekt aus ihrem eigenen Adressraum
- CR stellt die API-Methoden des zugehörigen CP zur Verfügung
- Interprozesskommunikation zwischen CR und CP

Content Provider

- Beispiele für CP:
  - Calendar Provider
  - Contacts Provider
  - Media Store Provider
- Verschiedene Operationen möglich:
  - insert / delete / update / query
  - openFile
  - ...
- Zugriff über Content-URIs

```
$scheme://$authority/$dataDescriptor[/$id]
content://com.example.app.provider/contacts
content://com.example.app.provider/contacts/1
```

Android und Datenbanken

- Warum Datenbanken für Android?
  - Persistenz
  - Zwischenspeicherung
  - Kombination mit Web-Synchronisation
- SQLite im Lieferumfang enthalten

SQLite

- Schlankes DBMS, optimiert für mobile Plattformen
- Kein Server, standalone
- Viele Features trotz geringer Größe
- Datenbank in einer einzelnen Datei gespeichert
- Nur für eine einzelne App erreichbar
- Konvention: Datenbank-Manager-Klasse (DMK), erweitert SQLiteOpenHelper
- DMK dient der Schemaverwaltung und Versionierung
- Direkter Zugriff auf die Datenbank über die Klasse SQLiteDatabase
- Software-Schicht abstrahiert den Zugriff:
  - Data Access Objects (DAOs)
  - Content Providers

2. Time Tracker

Anforderungen

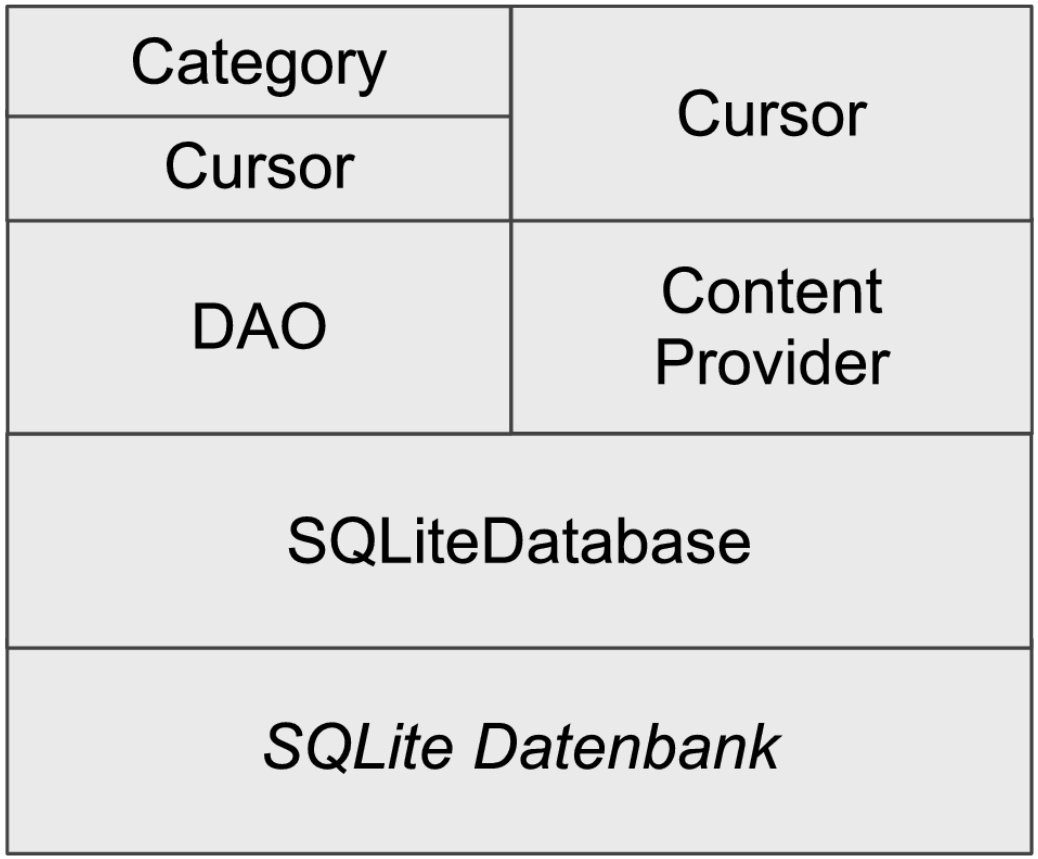
- Time Tracking
- Tätigkeit == Track
- Benutzereingaben ,,in Echtzeit”
- Einfaches Starten und Stoppen
- Was/Wann/Wo
- Nur ein Track gleichzeitig
- Nachbearbeitung und Verwaltung von Tracks
- Statistiken
- Verwaltung von Kategorien
- Verwendung von Standard UI Klassen und Layouts

Time Tracker App

Datenbank: Definition und Verwaltung

- SQLite
- *DatabaseHelper*
- *TrackTable* und *CategoryTable*
- Versionierung für Schemaänderungen

Datenbank Zugriff



Quelle: Eigene Darstellung

Main Activity

- Layout
- Button Handler
- State Refresh
- findViewById()

Edit Track Activity

- Layout
- Styles
- IDs
- Attribute
- Zeiten
- Intent Extras → Create/Update
- Kategorie-Spinner
- Interaktion mit ListActivity und MainActivity

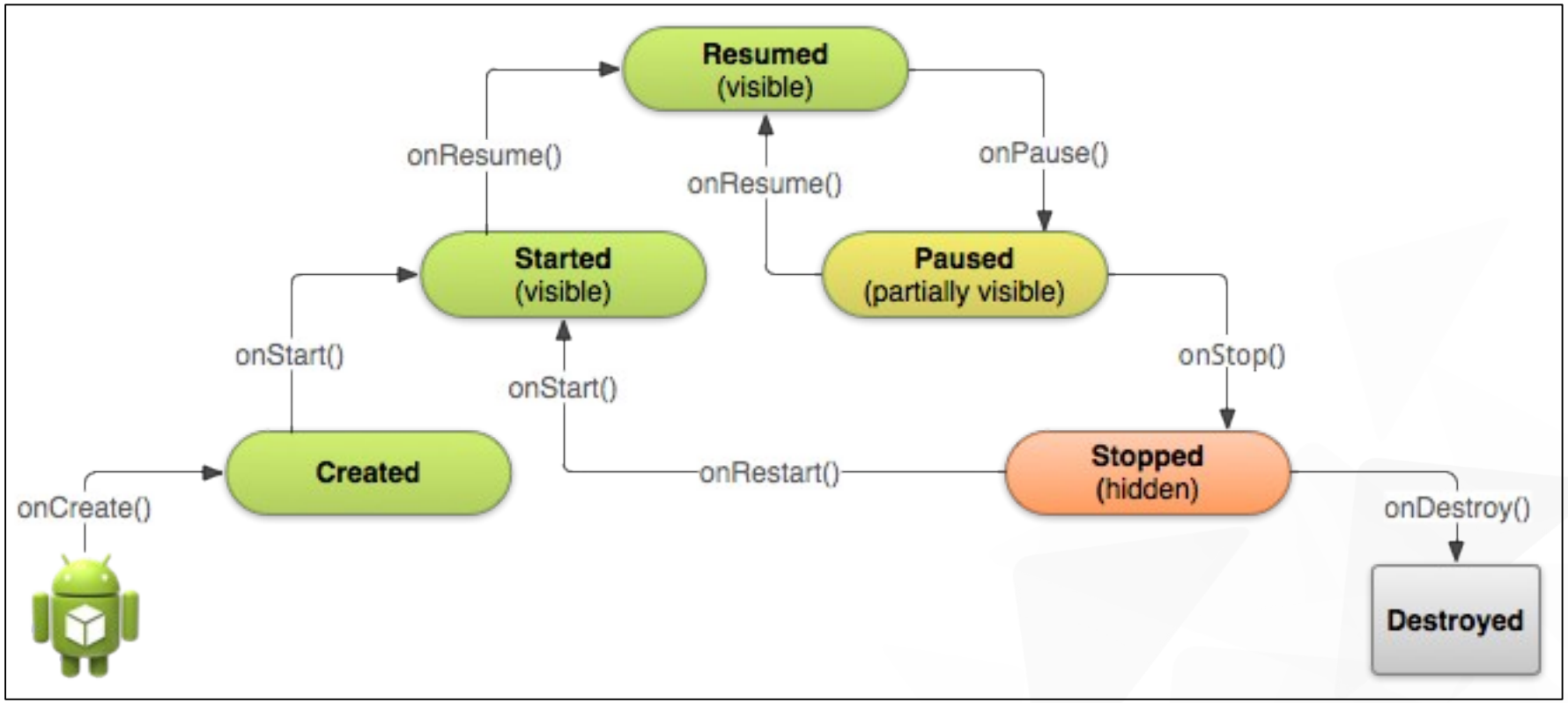
Geodaten und Adressen Lookup

- Koordinaten
  - Bereitgestellt durch Android Framework
  - Listener
  - Rechte
- Adressen-Lookup
  - Geocoding-API
  - AsyncTask
  - Rechtliche Probleme
  - Alternative: *Geocode*

Track Listenansicht

- Basisklasse ListActivity
- AdapterView und Adapter
- Cursor Adapter
- Layout
- Kontext Menü

Kategorie Listenansicht



Quelle: Offizielle Android Dokumentation, http://developer.android.com/training/basics/activity-lifecycle/starting.html (Stand: 27.02.2015)

Android als Komponentenframework

- E1: Komponente als klar ident. Einheit, konform zum KFW
  - APK Dateien als Container mit klarer Struktur
  - Android-Manifest
- E2: Komponentenkopplung und Modularität
  - Kopplung durch Intents herausragendes Merkmal
- E3: Lebenzyklusverwaltung durch KFW
  - Ausgeklügelte Lebenszyklusverwaltung für Bausteine
- E4: Komponente definiert was sie kann bzw. benötigt
  - Rechtesystem
  - Intentfilter
  - Content Provider

Vielen Dank für die Aufmerksamkeit

3. Aufbau der Entwicklungsumgebung

Einrichtung der virtuellen Maschine

- Image-Ordner von USB-Stick kopieren
- VMware-Player starten
- ,,Open a Virtual Maschine”
- Datei in Image-Ordner auswählen
- Virtuelle Maschine starten
- Keine Updates installieren und Warnungen ignorieren
- Warten bis Login Screen erscheint und alle anderen auch soweit sind

Starten der Entwicklungsumgebung

- Einloggen (User: android, Passwort: android)
- Desktop Verknüpfung ,,Eclipse für Android” öffnen
- ,,Run” auswählen
- In Eclipse Window → Android Virtual Device Manager öffnen
- Den vorkonfigurierten Android Emulator ,,Nexus” starten
- Das Starten kann etwas dauern...
- Das Projekt Ttracker kann jetzt auf dem Emulator ausgeführt werden (einfach den grünen Run-Button oben benutzen)

4. Aufgaben