1 Motivation und Einführung

Single- vs. Cross-Plattform:
Single: Codebasis für jede Plattform
Cross: Shared Code + Platform Code
Native-, Hybrid-, Web-Apps:
Native: Plattform(NativeApp(Binary))
Hybrid: Plattform(NativeApp(HTML))
Web: Plattform(Web Browser(HTML))
Vorteil Native Apps:

- Voller Funktionsumfang
- keine Tools/Einschränkungen von Drittanbietern

2 Grundkonzepte

Apps bestehen aus lose gekoppelten, wiederverwendbaren Komponenten (Activities, Content Providers, Services & Broadcast receivers).
Android hat die Kontrolle über ausgeführte Apps:

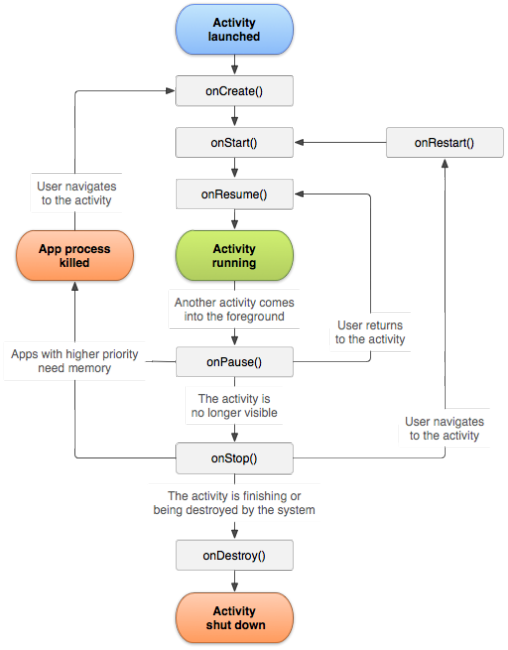
- Verwaltung des Lebenszyklus
- Kommunikation zwischen Komponenten
- Terminierung bei Bedarf (z.B. Speicherknappheit)

2.1 Activities

Beim App-Start wird die Main Activity von Android erzeugt und ausgeführt. Activities besitzen eine grafische Obefläche und verarbeiten Benutzereingaben.

```
public class MainActivity extends Activity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```

Activity Lebenszyklus & Zustände:
Android ruft beim Zustandwechsel Callback-Methoden auf der Activity auf. Diese Methoden können überschrieben werden.



2.1.1 Anwendungsfälle

- Erzeugung des GUI: onCreate()
- Datensicherun: onPause für schnelle Operationen, ansonsten onStop()
- Dienste wie Lokalisierung aktivieren/deaktivieren: onResume() und onPause()
- Zustand des GUI erhalten, z.B. bei Rotation: onSaveInstanceState() und onRestoreInstanceState()

2.2 Intents

- Die Kommunikation zwischen Komponenten erfolgt über Intents (Absicht, Vorhaben)
- Zwei Arten von Intents:
  - Explizit: Aufruf einer definierten Komponente (typischerweise für Komponenten der eigenen App)
  - Implizit: Aufruf einer passenden Komponente (typischerweise für Komponenten aus anderen Apps)
- Apps können sich im Android Manifest mit Intent Filters auf implizite Intents registrieren
- Intents werden stets von Android verarbeitet

```
// Expliziter Intent
Intent secondActivityIntent = new Intent(this,
    SecondActivity.class);
startActivity(secondActivityIntent);
// Impliziter Intent
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.setType("text/plain");
sendIntent.putExtra(Intent.EXTRA_TEXT, "Hey!");
startActivity(sendIntent);
```

2.2.1 Beispiel

```
Button button = findViewById(R.id.buttonNavigate);
button.setOnClickListener(v -> {
   //Explizit
   Intent secondActivityIntent = new Intent(this,
       SecondActivity.class);
   startActivity(secondActivityIntent);
   //Implizit
   Intent intent = new Intent(Intent.ACTION_VIEW, Uri.
       parse("http://www.ost.ch"));
   startActivity(intent);
});
```

2.3 Intents mit Parametern

Zusätzliche Parameter können als Key-Value Paar (Bundle) mit putExtra()/putExtras() übergeben werden.

```
// MainActivity.java
Intent intent = new Intent(this, SecondActivity.class);
intent.putExtra("myKey", 42);
startActivity(intent);
// SecondActivity.java
Intent intent = this.getIntent();
String parameter = intent.getStringExtra("key");
```

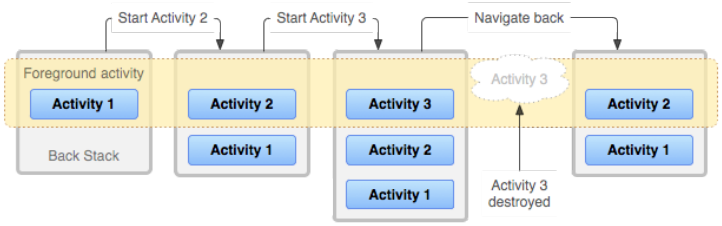
2.3.1 Hinweise

Mit Intents startet man andere Activities.
→ Ohne Rückgabewert: startActivity(Intent)
→ Mit Rückgabewert: startActivityForResult(Intent, int)
Implizite Intents müssen nicht immer einen Empfänger haben.
→ Darum immer überprüfen ob Intent einen Empfänger hat:

```
//MainActivity.java
if(intent.resolveActivity(getPackageManager()) != null){
   startActivity(intent);
}
//AndroidManifest.xml
<uses-permission android:name="android.permission.
    QUERY_ALL_PACKAGES" />
```

2.4 Back Stack (Task)

- Activities werden im Back-Stack verwaltet
- Activities eines Stacks können zu verschiedenen Apps gehören
- Dieselbe Activity kann mehrfach im selben Stack enthalten sein



Ein Back Stack wird auch Task genannt. Android verwaltet die Ausführung von Tasks. Bei Bedarf können Activities in neuen Tasks gestartet werden.

2.5 Tasks, Prozesse und Threads

- Alle Teile eines Apps werden in einer APK-Datei ausgeliefert
- Jedes APK wird mit einem eigenen Linux User installiert (Sandbox)
- Jedes APK wird in einem eigenen Linux Prozess ausgeführt
- Jeder Prozess hat mindestens einen Thread (Main Thread)

2.5.1 Main-Thread

- Automatisch erzeugt beim Start einer Applikation
- Blockierung des Main Threads führt zum ANR-Screen (Application Not Responding)
- Langlaufende Operationen immer in eigenen Threads ausführen (Runnable)
- **Achtung:** Nur der Main Thread darf das GUI aktualisieren, sonst Exception

2.6 GUI

Das GUI kann auf zwei Arten erstellt werden: Deklarativ (Beschreibung in XML) und Imperativ (Beschreibung im Quellcode).

2.7 Event Handling

Listener reagieren auf GUI-Ereignisse und werden bei GUI-Objekt registriert

```
final TextView textView = this.findViewById(R.id.
    text_example);
Button button = this.findViewById(R.id.button_example);
button.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View view) {
      textView.setText("Button pressed");
   }
});
// Lambda
button.setOnClickListener(v -> { ... });
// XML
android:onClick="onExampleButtonClicked"
public void onExampleButtonClicked(View view)
```

2.8 Resources

Alle Dateien, die keinen Code enthalten, werden als Resources bezeichnet. colors.xml für Farbwerte, dimens.xml für Dimensionen, strings.xml für Texte, styles.xml für Styles. Veränderliche Werte immer in passenden Files definieren und referenzieren. Der Zugriff erfolgt jeweils über die Resource ID. → Zugriff via **R-Klasse**

2.9 Dimensionen

Android erlaubt die Verwendung folgender Dimensionen:

- dp: Density-independent Pixels
- sp: Scale-independent Pixels
- px: Pixel
- pt: Punkte (1/72 eines physikalischen)
- in: Inch
- mm: Millimeter

Empfehlung: Für Schriften immer in sp, Alles andere in dp

2.10 Qualifier

Resources können in unterschiedlichen Varianten hinterlegt werden:

- Texte für verschiedenen Sprachen
- Bilder für verschiedenen Auflösungen
- Layouts für unterschiedliche Gerätetypen

2.10.1 Mehrsprachigkeit

Kein Hardcoded Text sondern über String resource file. Mehrere values Ordner (values_en, etc.) mit strings.xml Dateien anlegen.

2.11 App Manifest

Das AndroidManifest.xml enthält essenzielle Informationen zur App.

- ID, Name, Version und Logo
- Enthaltene Komponenten
- Hard- und Softwareanforderungen
- Benötigte Berechtigungen

2.11.1 Application ID und Version

package: Eindeutige Identifikation der App, Definiert Namespace, Reversed Internet Domain Format (ch.ost.rj.helloworld)
versionName: Ein menschenlesbarer String, Typischerweise Semantic Versioning
versionCode: Ein positiver Integer für interne Verwendung, Je höher die Zahl, desto "neuer"die App, Unterschiedliche Ansätze zur Inkrementierung

2.11.2 Application-Element

- Parent der Komponenten ist der Application-Knoten
- Application ist auch eine Klasse, die den globalen Zustand der App hält
- Eigene Ableitung von Application kann registriert werden
- Application enthält LifeCycle-Methoden, die überschrieben werden können

2.11.3 API Level

- minSdkVersion gibt an, welche Version das Gerät mindestens haben muss
- maxSdkVersion gibt an, welche Version das Gerät maximal haben darf
- targetSdkVersion ist die Version, welche die App bei der Ausführung verwendet
- compileSdkVersion gibt an, mit welcher API das App kompiliert wird

3 GUI Programmierung

4 Strukturierung, Material Design und Styling

5 Berechtigungen, Persistenz und Hardwarezugriff

6 Architektur und fortgeschrittene Themen

7 Android Jetpack