



ANDROID

KIV/MKZ - 2. přednáška

Verze 2017

L. Pešička

HISTORIE

- ◉ 2005 Google kupuje Android, Inc.
- ◉ 2007 ohlášena Open Handset Alliance
Android je open source
- ◉ 2008 vydáno Android SDK 1.0
telefon G1 od HTC
- ◉ 2009 nové verze 1.5, 1.6, 2.0, 2.1,
> 20 typů zařízení na Androidu
- ◉ 2010 verze 2.2 (Froyo), > 60 zařízení
verze 2.3 (Linux jádro 2.6.35)
- ◉ 2011 Barcelona Mobile Congress 2011
převládala zelená barva 😊

HISTORIE II. - 2011

- ◉ Android 3.0 / 3.1 / 3.2 (**Honeycomb**)
 - **únor 2011**
 - pouze pro tablety !! (konkurence první iPad 2010)
 - už nepotřebuje fyzická tlačítka
- ◉ Android 4.0 / 4.0.1 / 4.0.2 (**Ice Cream Sandwich**)
 - **říjen 2011**
 - chytré telefony (společně s tablety)
 - **Android Beam** (výměna info mezi telefony s NFC čipem, např. URL webové stránky)
 - **WiFi Direct** (přímé připojení 2 zařízení)

HISTORIE III. - 2012, 2013

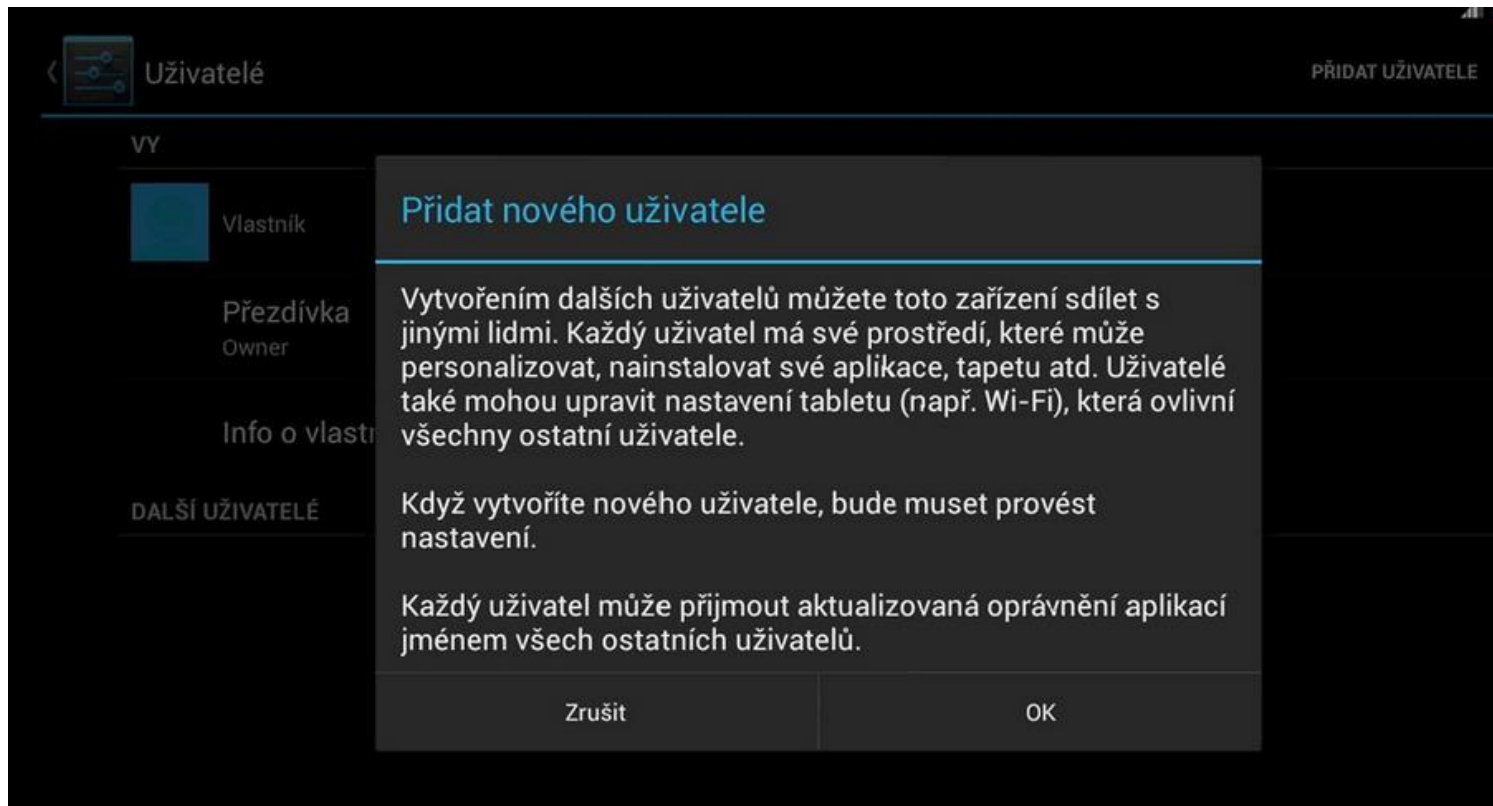
◉ Android 4.1/4.2/4.3 (Jelly Bean)

- červenec 2012
- projekt **Butter** - zrychlené a plynulejší vykreslování (60 fps) - boj proti trhanosti zobrazení
- rozpoznávání hlasu offline
- více uživatelských účtů (4.2; 4.3 omezené profily)

◉ Android 4.4 (KitKat)

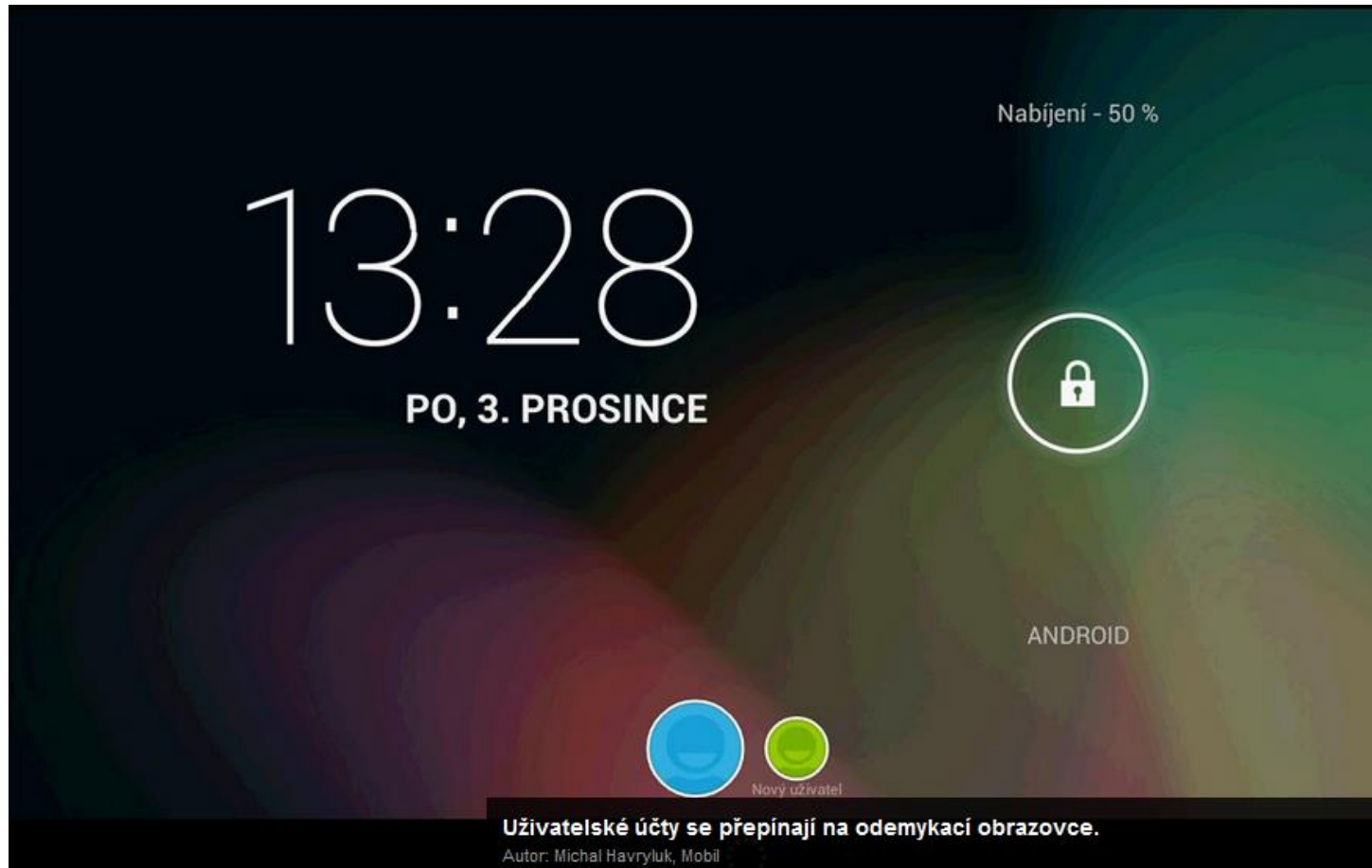
- září 2013
- wireless printing
- NFC host card emulation (nahradit kreditku)
- Webview na Chromium engine (od 5.0 do apk, odděleně od systému)

POZNÁMKA - VÍCE UŽIVATELŮ



Vlastník telefonu smí spravovat ostatní uživatele

POZNÁMKA - VÍCE UŽIVATELŮ



HISTORIE IV.

◉ Android 5.0 (Lollipop)

- Material design
- **ART runtime**
- 32 i 64bitový systém
- Nižší energetická náročnost
 - Udává se až 90 minut vydrží déle
 - Zobrazení času do nabití, v nastavení vybití baterie
- Úsporný režim - pohotovostní doba telefonu
- Šifrování v základním nastavení

HISTORIE V. - ANDROID 6.0

◉ Android 6.0 (Marshmallow)

- Nativní podpora rozpoznávání otisků prstů
- USB Type-C
- Přidělování práv (!!)
 - Kategorie oprávnění
normální x nebezpečné
 - Nedostane automaticky vše během instalace
 - Uživatel grant/deny individuální práva (přístup k foťáku, mikrofonu) při prvním požadavku
 - Zapamatuje si oprávnění, lze později odebrat
 - Starší aplikace stále vše nebo nic model

DANGEROUS PERMISSIONS

skupina	práva
CALENDAR	READ_CALENDAR, WRITE_CALENDAR
CAMERA	CAMERA
CONTACT	Read, Write, GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE, CALL_PHONE READ_CALL_LOG, WRITE_CALL_LOG, ...
SENSORS	BODY_SENSORS
SMS	SEND_SMS, RECEIVE_SMS, READ_SMS, RECEIVE_MMS, RECEIVE_WAP_PUSH
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

POKRAČOVÁNÍ

◉ Android 6.0

■ Doze

- na baterii, neaktivní, nikdo na něj fyzicky nesáhá
- -> omezení konektivity, jen notifikace s vysokou prioritou

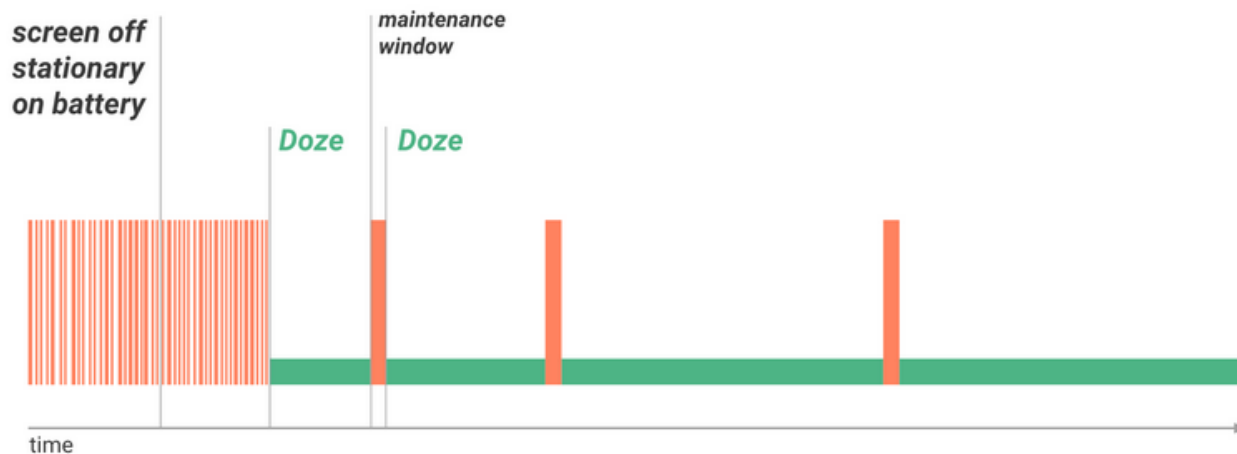


Figure 1. Doze provides a recurring maintenance window for apps to use the network and handle pending activities.

POZNÁMKA

◉ App Standby

- Další technologie pro snížení spotřeby (jako je Doze)
- Doze se týká všech aplikací na zařízení
- App Standby - ovlivňuje aplikace, které se nevyužívají příliš často
- Ztráta síťového připojení, sync joby jsou suspendovány
- Když připojen do nabíječky umožní chvíli obsloužit

PŘÍKLAD

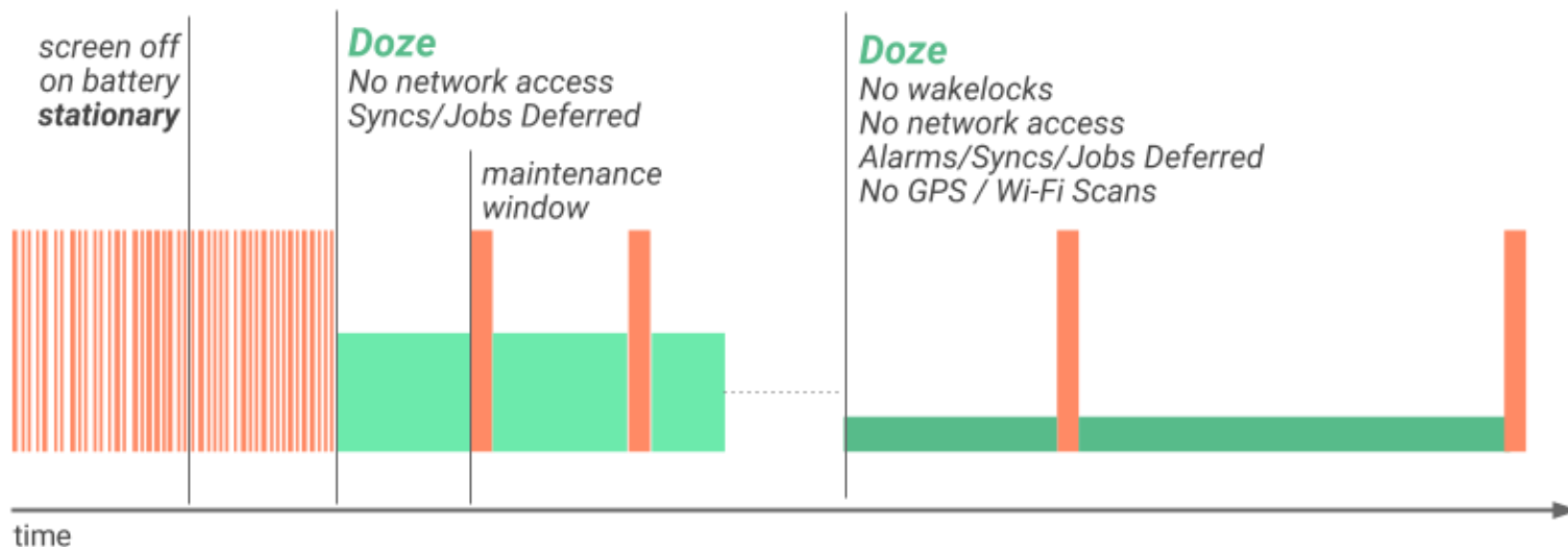
Aby se šetřila baterie, aplikace, které nebyly použity více než 3 dnů, byly nastaveny do úsporného režimu. Aplikace v úsporném režimu pravděpodobně nebudou moci poskytovat oznámení.

- ◉ Automaticky šetřit energií (po 3 dnech)
- ◉ Vždy šetřit energií
- ◉ Vypnuto

ANDROID 7 - NUGAT

◉ Vylepšený režim Doze

- Šetří baterii i když se nese zařízení v kapse
- First level of restriction
- Second level of restriction po určitém čase



VIRTUÁLNÍ STROJ (DALVIK X ART)

◉ Dalvik

- Do verze Androidu 4.3

◉ ART (Android RunTime)

- Od verze Androidu 4.4 a 5.0
- Dopředná kompilace (AOT - Ahead of Time)
- `.java` -> `bytecode` -> `bytekód Dalviku (.dex)`
- `dex2oat`: z dex souboru aplikaci pro dané zařízení
- Při instalaci aplikace se zkompiluje do nativního kódu procesoru zařízení
- Větší výkon, větší výdrž zařízení, pomalejší instalace
- Původní kompilátor JIT stále dostupný

POZNÁMKY

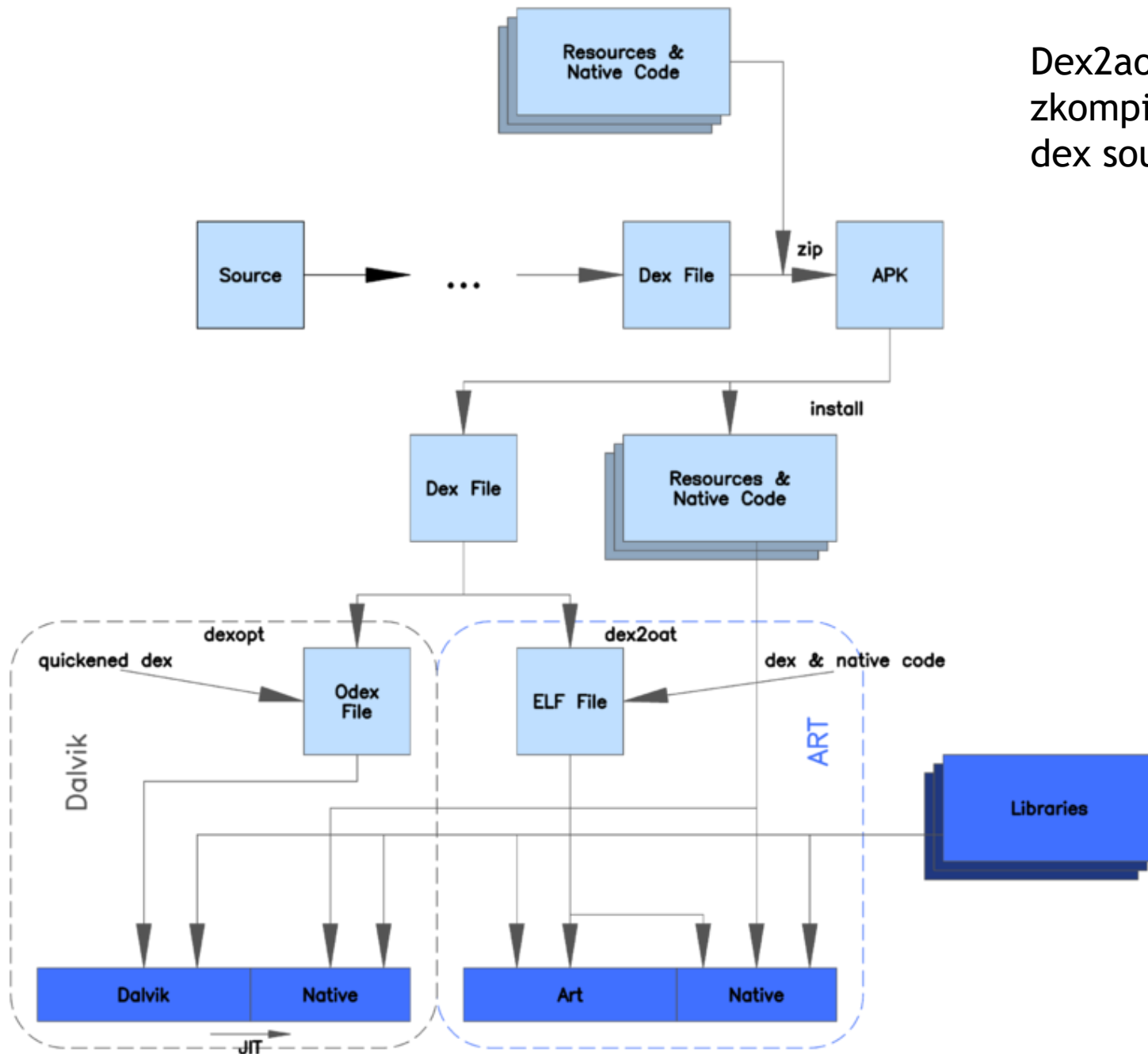
⊙ JIT kompilace (Just in time)

- Dalvik, od 2.2 Froyo
- Bytekód se kompiloval pokaždé, když byla aplikace spuštěna

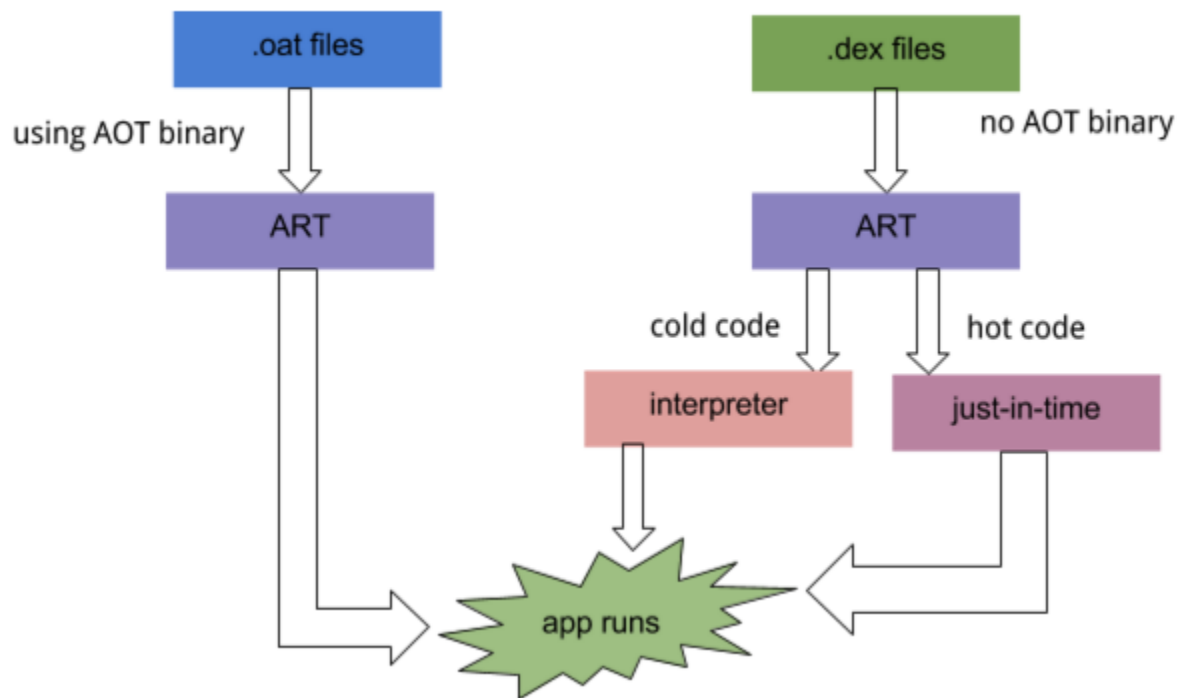
⊙ AOT kompilace (Ahead of Time)

- ART
- Kompilace **při instalaci** aplikace
- Pak běží nativní kód
- Používá na vstupu stejný bytekód jako Dalvik

Dex2aot
zkompiluje
dex soubor



SPUŠTĚNÍ KÓDU



<http://source.android.com/devices/tech/dalvik/jit-compiler.html>

WIFI DIRECT

- ◉ přenos dat bez nutnosti vytváření přístupového bodu (access pointu)
- ◉ analogie Bluetooth
- ◉ větší: rychlost, dosah, zabezpečení WPA2
- ◉ neplést s ad-hoc režimem
 - větší rychlost
 - jednodušší nastavení
 - zároveň i připojení do Internetu

WIFI DIRECT - RYCHLOST

technologie	max. rychlost Mbps	Dosah metrů
Bluetooth 2.0	3	10
Bluetooth 3.0HS	24	100
Bluetooth 4.0	24	100
WiFi Direct	54	200

zdroj: <http://mobilizujeme.cz/clanky/wi-fi-direct-objevte-moznosti-nove-technologie-vedecke-okenko/>

Ukázka:

<http://nearfield.cz/clanky/podivejte-se-jak-funguje-s-beam-na-galaxy-s-iii-video-44>

VÝVOJ

- ◉ Android SDK zdarma
 - včetně emulátoru zařízení
- ◉ dříve Eclipse + ADT plugin
- ◉ dnes **Android Studio**

- ◉ Různé další zajímavé nástroje např. Adobe AIR, App Inventor
<http://appinventor.mit.edu/>

ANDROID STUDIO

- ◉ zdarma ke stažení
- ◉ postaveno nad prostředím IntelliJ IDEA
- ◉ aktuální verze 2.2.3 (únor 2017)

ANDROID STUDIO



obrázek: <http://www.redmondpie.com/download-android-studio-ide-for-windows-os-x-and-linux/>

HELLO WORLD 😊

```
package com.example.helloandroid;
```

```
import android.app.Activity;  
import android.os.Bundle;  
import android.widget.TextView;
```

```
public class MainActivity extends Activity {
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        TextView tv = new TextView(this);  
        tv.setText("Hello, Android");  
        setContentView(tv);
```

```
    }  
}
```



propojení UI s aktivitou

HALLO WORLD PO ANDROIDSKU 😊

```
package com.example.helloandroid;
```

```
import android.app.Activity;  
import android.os.Bundle;
```

```
public class MainActivity extends Activity {
```

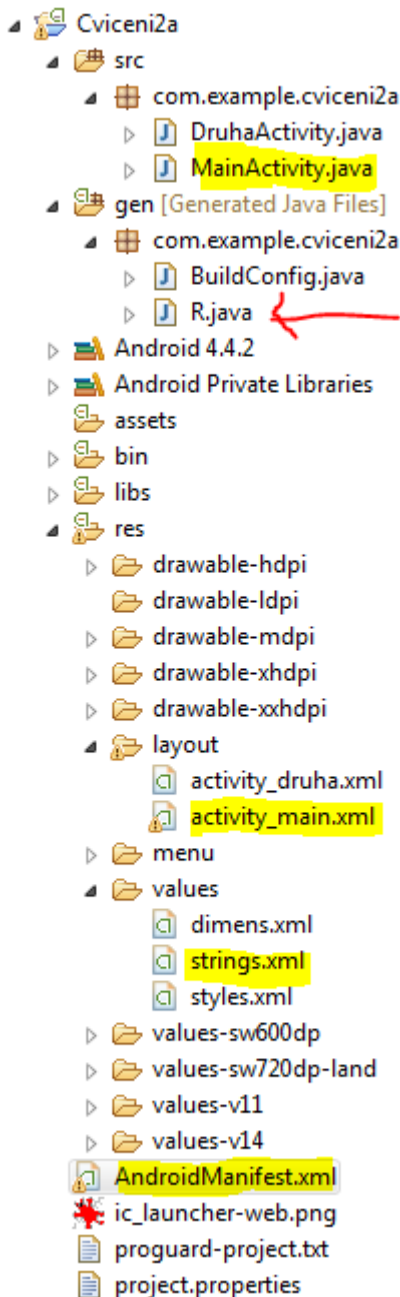
```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);
```

```
    }
```

```
}
```

propojení UI definovaného v
res/layout/activity_main.xml s aktivitou



Android projekt v Eclipse

dvě aktivity:

MainActivity.java

DruhaActivity.java

generovaná třída:

R.java

layout - UI v XML:

activity_main.xml

activity_druha.xml

values:

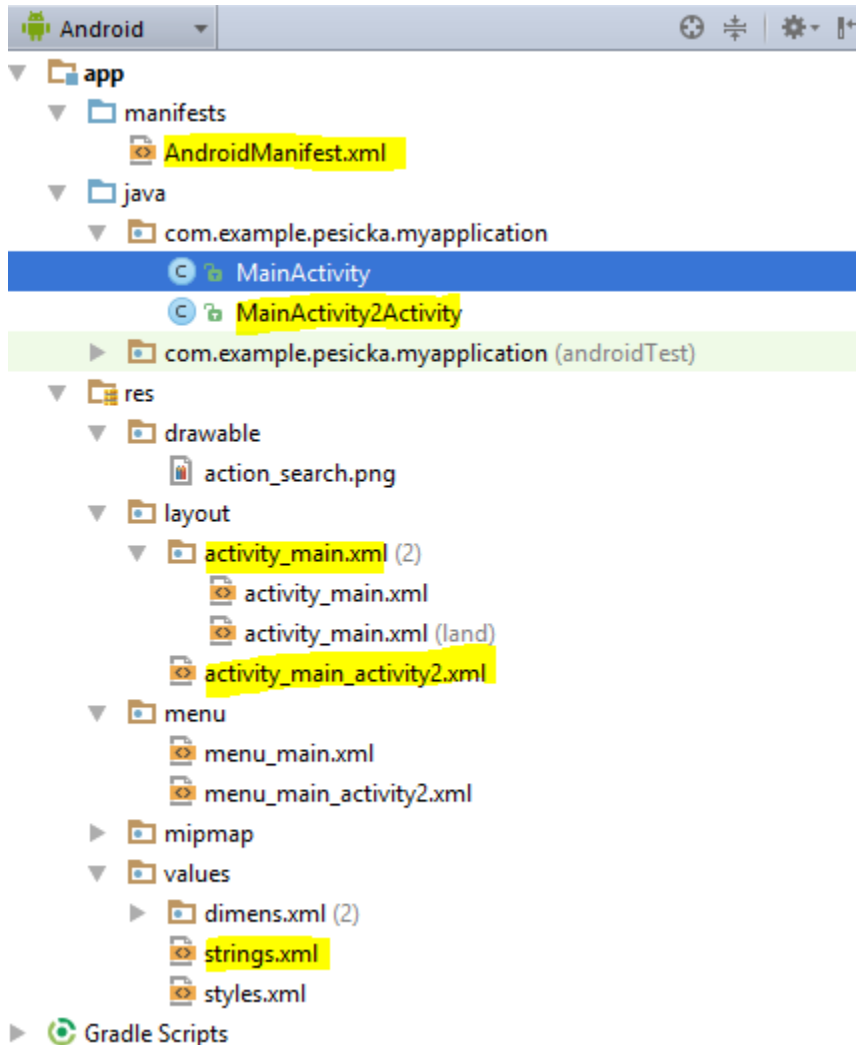
strings.xml

(values-es .. španělsky)

soubor s manifestem:

AndroidManifest.xml

Android projekt v Android Studiu



Manifest

2 aktivita

3 layouty

první aktivita

první aktivita landscape

druhá aktivita

Nyní ještě navíc

content_main.xml

includovaný do

activity_main.xml

Menu

první aktivita

druhá aktivita

AUTOMATICKY GENEROVANÁ TŘÍDA R.JAVA

```
package com.example.cviceni2a;
```

```
public final class R {  
    public static final class attr {  
    }  
    public static final class drawable {  
        public static final int icon=0x7f020000;  
    }  
    public static final class id {  
        public static final int textview=0x7f050000;  
    }  
    public static final class layout {  
        public static final int activity_main=0x7f030000;  
    }  
    public static final class string {  
        public static final int app_name=0x7f040001;  
        public static final int hello=0x7f040000;  
    }  
}
```

```
setContentView(R.layout.activity_main);
```

XML SOUBOR S DEFINICÍ UI

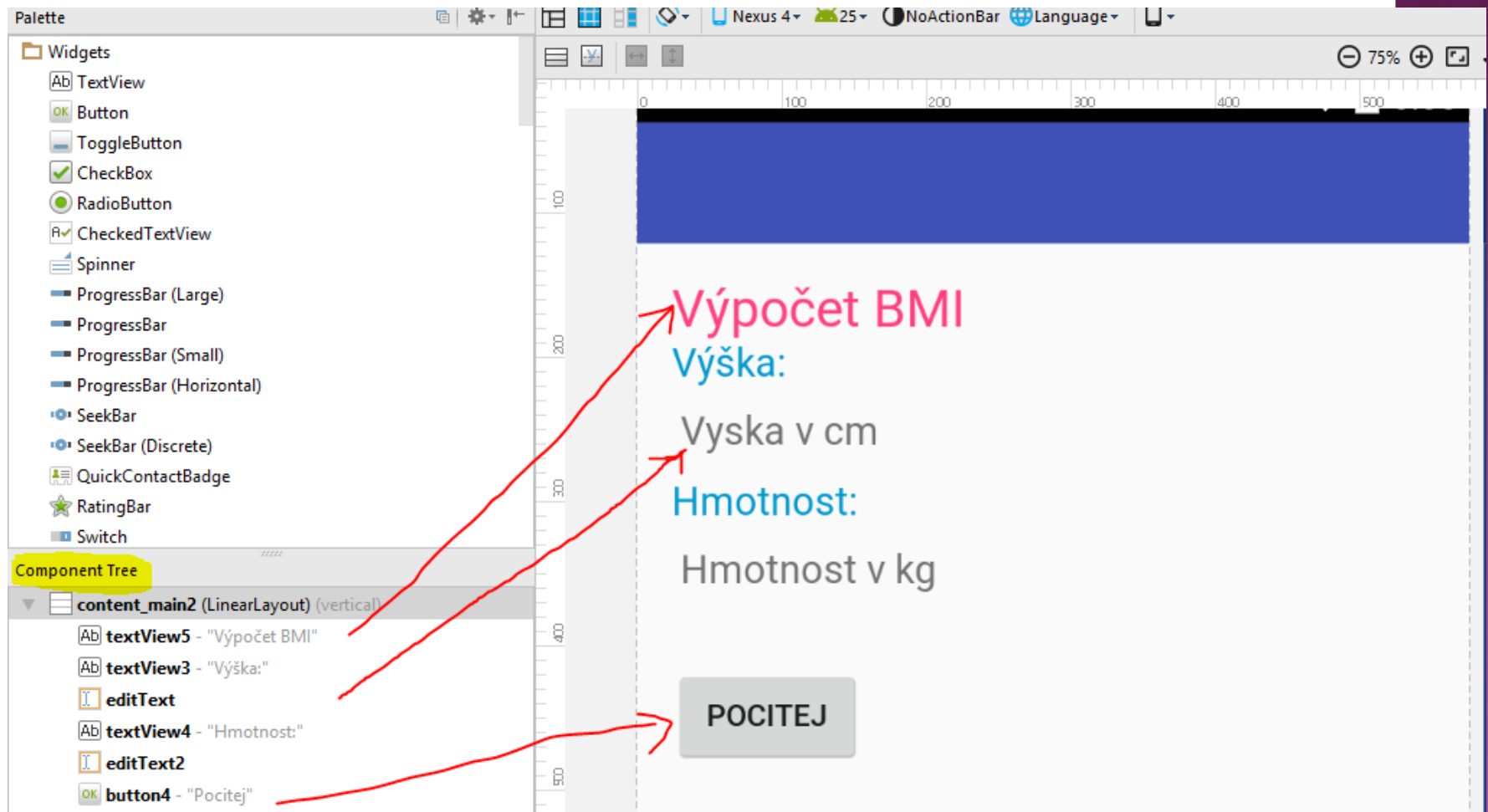
takto např. může vypadat activity_main.xml:

```
<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"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:text="@string/hello_world" />

</RelativeLayout>
```

PŘÍKLAD UI





ANDROID V KOSTCE

- ◉ Linuxové jádro řady 2.6, 3.x
 - Abstrakce - odděluje HW a zbytek SW
 - Správa procesů, paměti, síťování, bezpečnost
- ◉ Dalvik virtul machine (a nyní ART)
 - Syntaxe Javy, ale pozor není kompatibilní s J2ME
 - Může běžet více virtuálních strojů současně
 - Formát **.dex** (Dalvik executable)
- ◉ Každá Android aplikace
 - samostatný proces
 - vlastní Dalvik virtual machine
 - unikátní Linux user ID => bezpečnost

ANDROID V KOSTCE

- ◉ Linuxové jádro

- Jazyk C
- libc odvozená od BSD

- ◉ Uživatelské aplikace

- V Javě
(lze i nativní kód)
- Odlišnosti oproti J2SE i J2ME
- Distribuce aplikací - Android package **.apk**

ANDROID - BEZPEČNOST

- ◉ Android více možností publikace aplikací (x 1 AppStore pro iPhone)
- ◉ uživatel při instalaci aplikace musí schválit práva (co aplikace vyžaduje) - otázkou je, zda tak nečiní formálně
- ◉ Uživatel schválí práva ve chvíli, kdy je aplikace vyžaduje - Android 6 a výše

BEZPEČNOST (PŘÍKLADY)

⦿ <http://www.svetandroida.cz/meli-jste-v-mobilu-skodlivy-program-vime-co-vas-ceka-201103>

1. škodlivé aplikace **odstraněny z Marketu, zastaveny vývojářské účty**
2. dotčené programy **vzdáleně odstraněny** z postižených zařízení
3. technologií **push** (**bez interakce uživatele**) instalována aktualizace, která zabrání útočníkovi v přístupu k dalším informacím

LINUXOVÉ JÁDRO

- ◉ verze 2.6.*, 3.x
- ◉ základní systémové služby:
 - bezpečnost
 - správa paměti
 - správa procesů
 - síťování (network stack)
 - ovladače (driver model)
- ◉ oddělení HW a SW

APLIKACE - OCHRANA

- ◉ psány v Javě
- ◉ kompilovány do Android Package (.apk)
- ◉ 1 .apk - 1 aplikace
- ◉ každá aplikace vlastní security box
 - každá - odlišný uživatel (Linux user ID)
 - práva na všechny soubory aplikace jen tomuto uživateli
 - každý proces má **vlastní virtuální stroj** (izolace)
 - každá aplikace běží ve **vlastním procesu**
 - při startu aplikace je proces „zygota“ forkován, sdílená knihovny jsou read-only, stačí jen 1x

ZYGOTA A DALVIK VM

- ◉ biologie:
"The first cell formed when an organism is produced. "
- ◉ studený boot VM při startu systému
- ◉ **zygota** poslouchá na soketu příchozí příkazy
- ◉ **ActivityManagerService** do soketu zapisuje, když potřebuje nový proces pro aplikaci
- ◉ **Zygota** si přečte příkaz a zavolá **fork()**
- ◉ child proces dostane připravenou VM v které poběží

proč neběží víc aplikací ve stejném VM? Design decision 😊

ZYGOTA - ODKAZY

Podrobný popis pro zájemce:

<http://allenlsy.com/android-kernel-3/>

...

<http://allenlsy.com/android-kernel-1/>

AKTIVITA, APLIKACE, PROCES

defaultně - aktivity aplikace běží
ve stejném procesu

pokud je v manifestu u aktivit:

`android:process="cz.zcu.kiv.p1"`

`android:process="cz.zcu.kiv.p2"`

aktivita běží v samostatném procesu se
stejným userid

podrobněji např.

<http://stackoverflow.com/questions/6468126/every-activity-in-android-is-a-process-or-one-application-is-one-process>

APLIKACE - PŘÍSTUP K DATŮM, SENZORŮM

aplikace může vyžádat přístup:
ke kontaktům uživatele, SMS zprávám,
SD kartě, kameře, Bluetooth aj.

všechna práva musí udělit uživatel v **době instalace** aplikace (změny od Android 6)

Programátor uvede práva do manifestu:

```
<manifest>  
<uses-permission android:name="android.permission.INTERNET"/>  
<application> ... </application>  
</manifest>
```

PODROBNÝ POHLED DO ÚTROB ANDROIDU

Chcete-li se podrobně seznámit, jak dochází ke spouštění Android aplikací nad Linuxovým jádrem, doporučuji:

<http://coltf.blogspot.cz/p/android-os-processes-and-zygote.html>

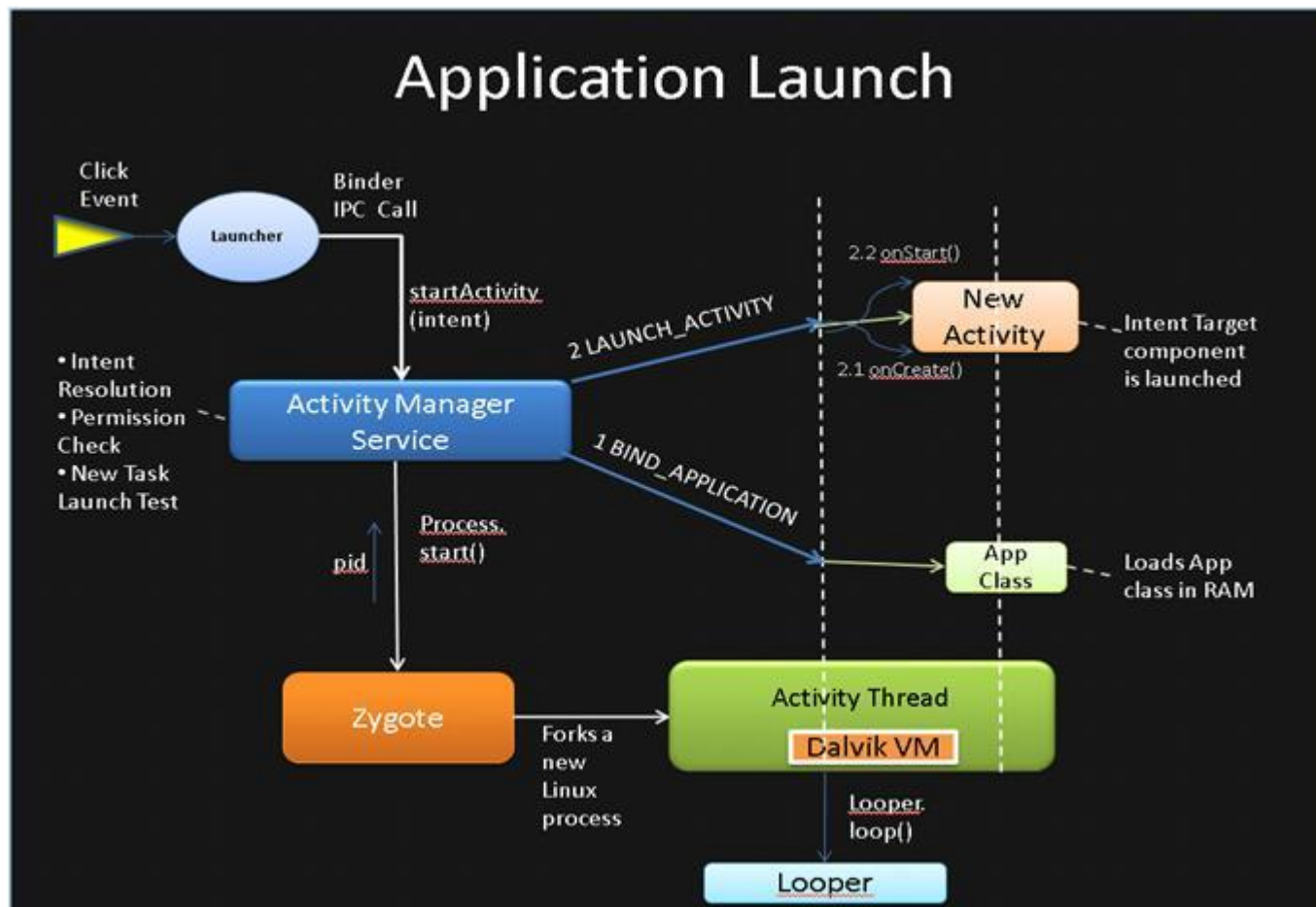
boot Linuxu => init proces -> start daemonů

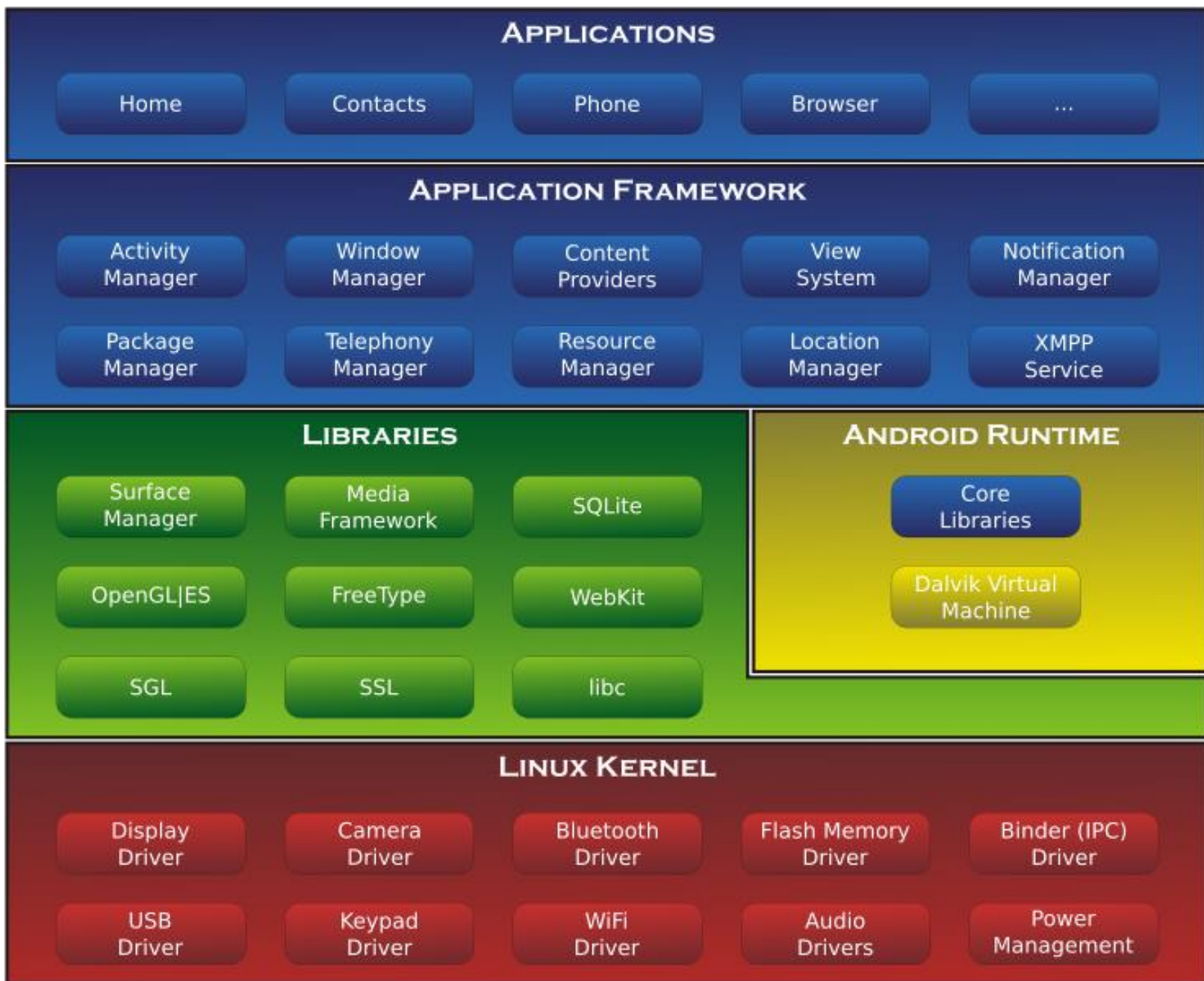
...

Activity Manager spustí HomeActivity

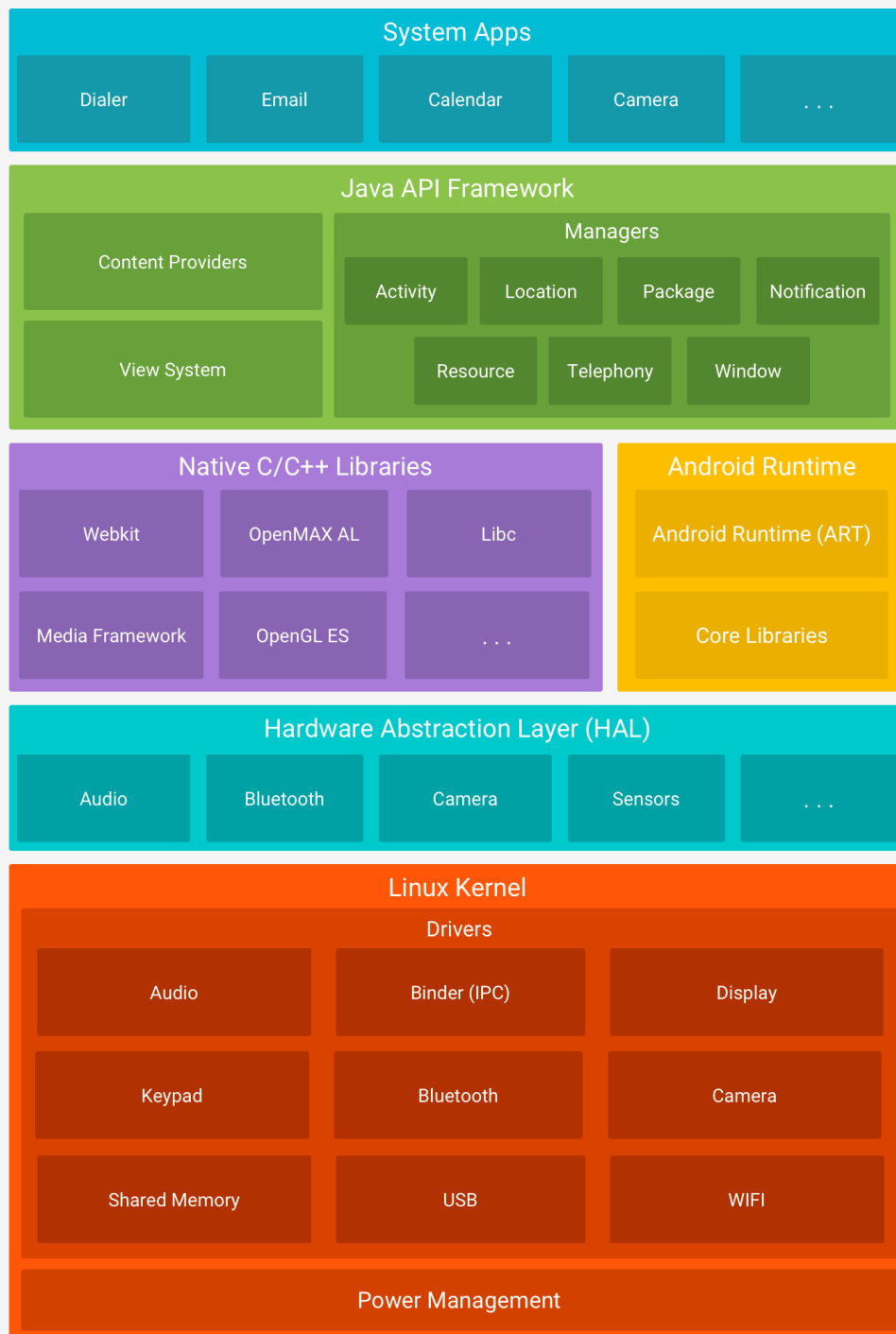
Window Manager - organizace obrazovky

SPUŠTĚNÍ AKTIVITY





zdroj: <http://en.wikipedia.org/wiki/File:Android-System-Architecture.svg>



NATIVNÍ KNIHOVNY

C/C++ open source knihovny, mimo jiné:

- ◉ **Webkit**
 - web rendering engine (používá Safari, Chrome)
- ◉ **SQLite** - SQL databáze
- ◉ **OpenGL** - 3D grafika
- ◉ **OpenSSL**
- ◉ **libc** - přepsaná verze standardní C knihovny, oproti klasické GNU glibc optimalizována pro MZ

DALVIK

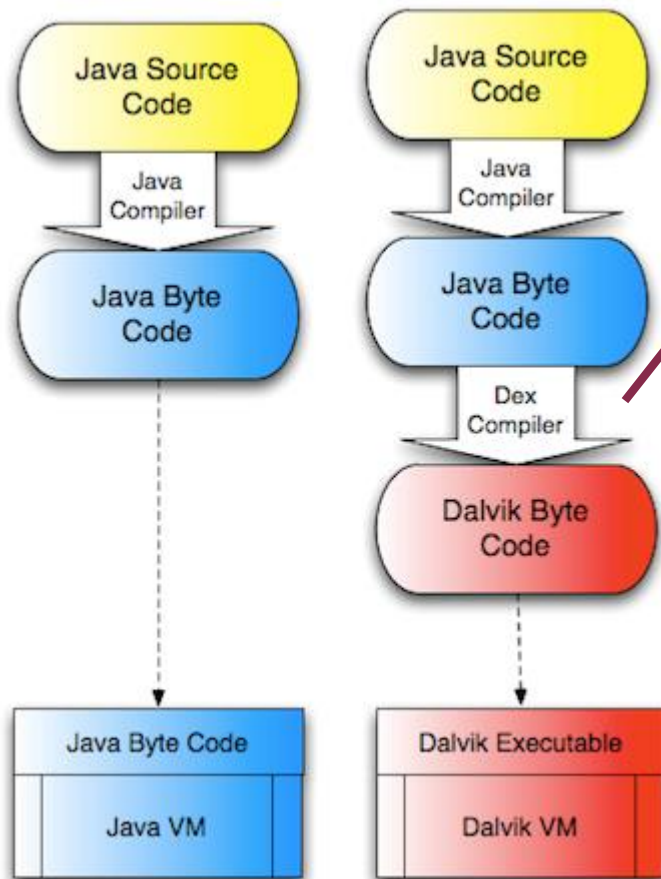
- ◉ VM pro Android
- ◉ bere v úvahu omezení
(baterie, paměť, výkon CPU)
- ◉ jsme zvyklí:
Java Source Code => Java Byte Code
- ◉ Android:
Java Source Code => Java Byte Code
=> Dalvik Byte Code

nástroj dex: *.class -> *.dex

dex size : jar size (0.44 : 1)

více tříd v jednom .dex

PŘEKLAD KÓDU



dex compiler
využívá byte code,
nikoliv java source

zdroj:
[http://ofps.oreilly.com/
titles/9781449390501/
ch02.html](http://ofps.oreilly.com/titles/9781449390501/ch02.html)

APLIKACE

1. předinstalované
 2. stažené z Android marketů či odjinud
 3. vámi vytvořené 😊
- ⦿ **.apk** (application package file)
 - dalvik executable + resources + native code

náš kód v Javě
(převedený)

obrázky,
hudba, video, XML
popis layoutů,
jazykové verze

VELIKOST APLIKACE

- ◉ omezení velikosti aplikace **pro market**
- ◉ aplikace .apk do 100MB (původně 50MB)
- ◉ možnost 2 rozšiřující soubory
main + patch (soubor 2GB max), tj. 4GB
- ◉ mimo market toto omezení není
- ◉ <http://developer.android.com/google/play/expansion-files.html>

PODPISOVÁNÍ APLIKACÍ



- ◉ aplikace **musí** být podepsány před instalací na zařízení
- ◉ pro **vývoj** - podepsané **debug key** (pozor, může vypršet ☺, viz dále)
- ◉ pro **distribuci** - podepsané vlastním privátním klíčem, **nejde debug klíčem**
- ◉ vypršení certifikátu se testuje jen v **době instalace**
- ◉ update aplikace - podepsat **stejným** certifikátem (pokud nesouhlasí, chce jiný *package name* - jako úplně novou aplikaci)

podepsat certifikátem platným např. na 25 let ☺

EXPIRACE DEBUG CERTIFIKÁTU

- ◉ vyprší 365 dní po vytvoření

debug:

[echo] Packaging bin/samples-debug.apk, and signing it with a debug key...

[exec] Debug Certificate expired on 8/4/08 3:43 PM

- ◉ řešení:
smazat **debug.keystore**, znovu se vytvoří
na Win v **c:\users\jmeno\.android**
- ◉ *více info ohledně certifikace aplikací viz*

<http://developer.android.com/guide/publishing/app-signing.html>

VERZOVÁNÍ APLIKACE

- ◉ aplikace musí být verzovaná
- ◉ součástí **manifestu aplikace**, 2 atributy (uvedeno v **build.gradle** Module)
- ◉ **android:versionCode**
 - integer číslo, nová verze aplikace vyšší číslo
 - lze ho snadno programově porovnat
- ◉ **android:versionName**
 - řetězec, vhodné pro uživatele
 - <major>.<minor>.<point>

PŘ. VERZE APLIKACE

```
<?xml version="1.0" encoding="utf-8"?>
  <manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.package.name"
    android:versionCode="2"
    android:versionName="1.1">
    <application android:icon="@drawable/icon"
      android:label="@string/app_name">
      ...
    </application>
  </manifest>
```

V Android Studio je informace o verzi aplikace a další informace vkládána Gradlem z konfiguračního souboru

ANDROID STUDIO - INFO O VERZI

◉ build.gradle (module: app)

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "23.0.3"
    defaultConfig {
        applicationId "com.example.pesic.mkz2017_cv2_p3"
        minSdkVersion 18
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
}
```

POZNÁMKA

Pozor, rozdílné pojmy:

- ◉ Verze aplikace
- ◉ Verze platformy Android

VERZE PLATFORMY - API LEVEL

- velmi rychlý vývoj verzí (Froyo, JellyBean..)
- **jednoduchá identifikace (celé číslo int)**

Android platforma poskytuje framework API:

- ◉ Základní množina balíků a tříd
- ◉ Množina XML elementů a atributů
 - Pro deklarování manifestu
 - Pro deklarování a přístup ke zdrojům
- ◉ Množina intentů
- ◉ Množina práv, které aplikace může žádat

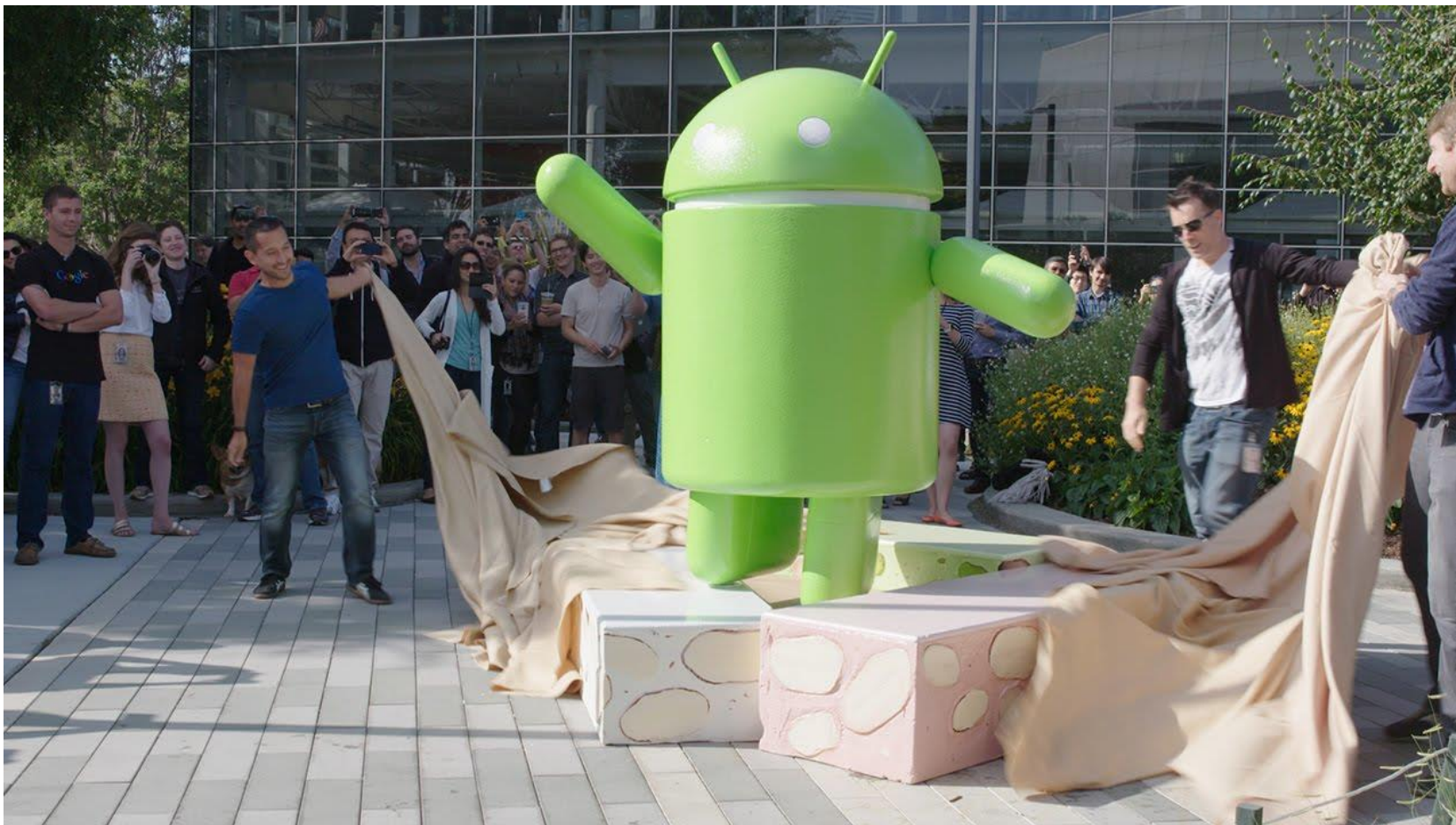
API LEVEL

verze	název	API level
Android 3.0	Honeycomb	11
Android 2.3.3		10
Android 2.3	Gingerbread	9
Android 2.2	Froyo	8
Android 2.1	Eclair	7
Android 2.0.1	Eclair	6
Android 2.0	Eclair	5
Android 1.6	Donut	4
Android 1.5	Cupcake	3
Android 1.1		2
Android 1.0		1

API LEVEL POKRAČOVÁNÍ

verze	název	API level
Android 7.1	Nougat	25
Android 7.0	Nougat	24
Android 6.0	Marshmallow	23
Android 5.1	Lollipop	22
Android 5.0-5.0.2	Lollipop	21
Android 4.4 wear.ext.	KitKat	20
Android 4.4	KitKat	19
Android 4.3	Jelly Bean	18
Android 4.2.x	Jelly Bean	17
Android 4.1.x	Jelly Bean	16
Android 4.0.3	Ice Cream Sandwich	15
Android 4.0-4.0.2	Ice Cream Sandwich	14
Android 3.2		13
Android 3.1		12

odkaz: http://en.wikipedia.org/wiki/Android_version_history



Zdroj: vyhledávání Google





Zdroje obrázků:

www.androidcentral.com



Zdroj obrázku: androidheadlines.com

POUŽITÍ API LEVEL V MANIFESTU

- V build.gradle - přidá se do manifestu
- Element v manifestu **<uses-sdk>**
- Tři základní atributy:

Použití i pro filtrování
v Android Marketu

- **android:minSdkVersion**
 - Minimální API, kterou aplikace vyžaduje
- **android:targetSdkVersion**
 - API pro které je aplikace navržena, **testována**
 - info že nemusí využívat „compatibility behavior“
- **android:maxSdkVersion**
 - Max. API na kterém je aplikace schopna běžet
 - Nedoporučuje se příliš využívat
v novějších verzích není vynucováno

PRO JAKÉ API VYVÍJET?

Nový projekt v Android Studio:

☒ Phone and Tablet

Minimum SDK **API 14: Android 4.0 (IceCreamSandwich)**

Lower API levels target more devices, but have fewer features available. By targeting API 14 and later, your app will run on approximately **90,4%** of the devices that are active on the Google Play Store. [Help me choose.](#)

☒ Phone and Tablet

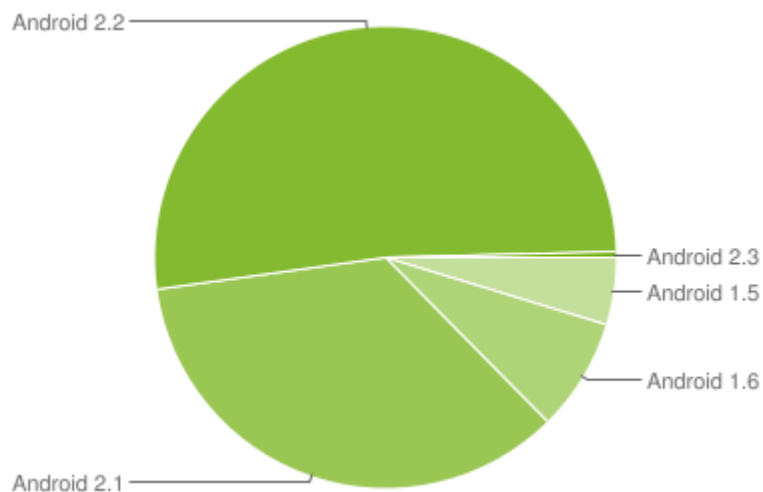
Minimum SDK **API 19: Android 4.4 (KitKat)**

Lower API levels target more devices, but have fewer features available. By targeting API 19 and later, your app will run on approximately **33,9%** of the devices that are active on the Google Play Store. [Help me choose.](#)

Odkaz - Help me choose




ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
2.3 Gingerbread	10	
4.0 Ice Cream Sandwich	15	97,4%
4.1 Jelly Bean	16	95,2%
4.2 Jelly Bean	17	87,4%
4.3 Jelly Bean	18	76,9%
4.4 KitKat	19	73,9%
5.0 Lollipop	21	40,5%
5.1 Lollipop	22	24,1%
6.0 Marshmallow	23	4,7%

ZASTOUPENÍ JEDNOTLIVÝCH VERZÍ - DŘÍVE

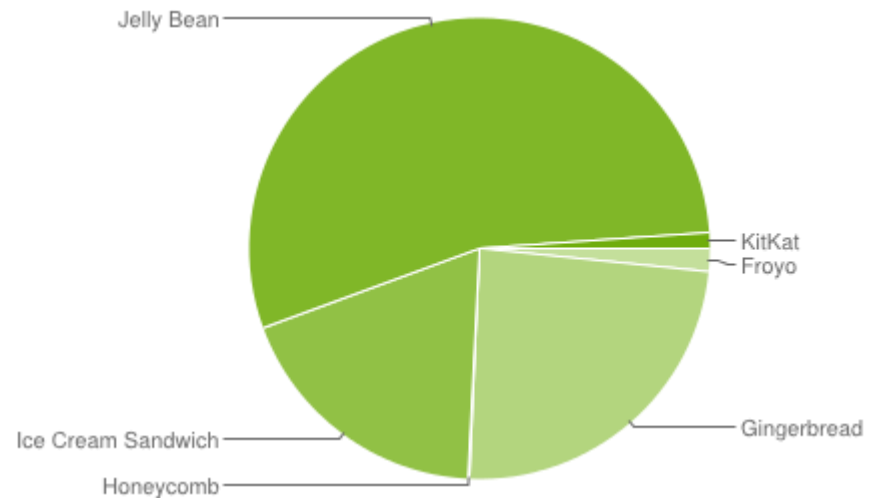


K únoru 2011

Zdroj: wikipedia

Platform 	API Level 	Distribution 
Android 2.3 (Gingerbread)	9/10	0.8%
Android 2.2 (Froyo)	8	57.6%
Android 2.0/2.1 (Eclair)	7	31.4%
Android 1.6 (Donut)	4	6.3%
Android 1.5 (Cupcake)	3	3.9%

ZASTOUPENÍ VERZÍ - 2014



Version ↕	Code name ↕	Release date ↕	API level ↕	Distribution ↕
4.4	KitKat	October 31, 2013	19	1.4%
4.3.x	Jelly Bean	July 24, 2013	18	7.8%
4.2.x		November 13, 2012	17	15.4%
4.1.x		July 9, 2012	16	35.9%
4.0.3–4.0.4	Ice Cream Sandwich	December 16, 2011	15	16.9%
3.2	Honeycomb	July 15, 2011	13	0.1%
2.3.3–2.3.7	Gingerbread	February 9, 2011	10	21.2%
2.2	Froyo	May 20, 2010	8	1.3%

K lednu 2014

Zdroj: wikipedia

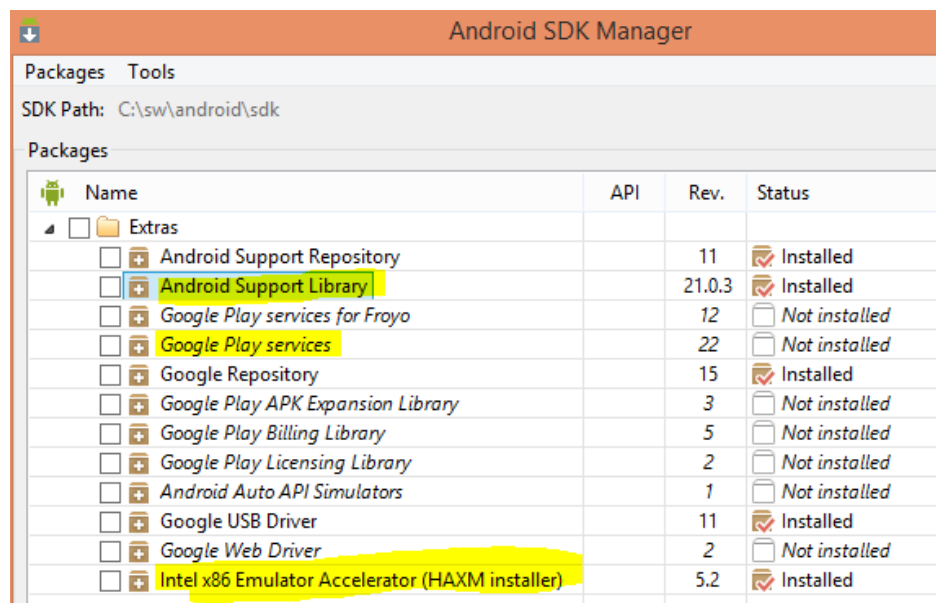
PŘ. ANDROID 2.3.3 PLATFORM

- ◉ API level: **10**
- ◉ přidává např:
NFC, Bluetooth non-secure socket connection
- ◉ vestavěné aplikace:
 - Browser, Calculator, Camera, Clock, Contacts
 - Custom Locale, Dev Tools, Downloads, Email
 - Gallery, Messaging, Music
 - IME Japonstina, Cinstina, Latin
 - Phone, Search, Settings, Speech Recorder
 - Spare Parts (developer app)

SUPPORT LIBRARY

- ◉ umožní využívat API, které není dostupné na starší platformě
- ◉ menší starost o verzi platformy
min. verze: API 4, API 13
- ◉ např. použití Fragmentu
v dvojkových androidech

<http://developer.android.com/tools/extras/support-library.html>

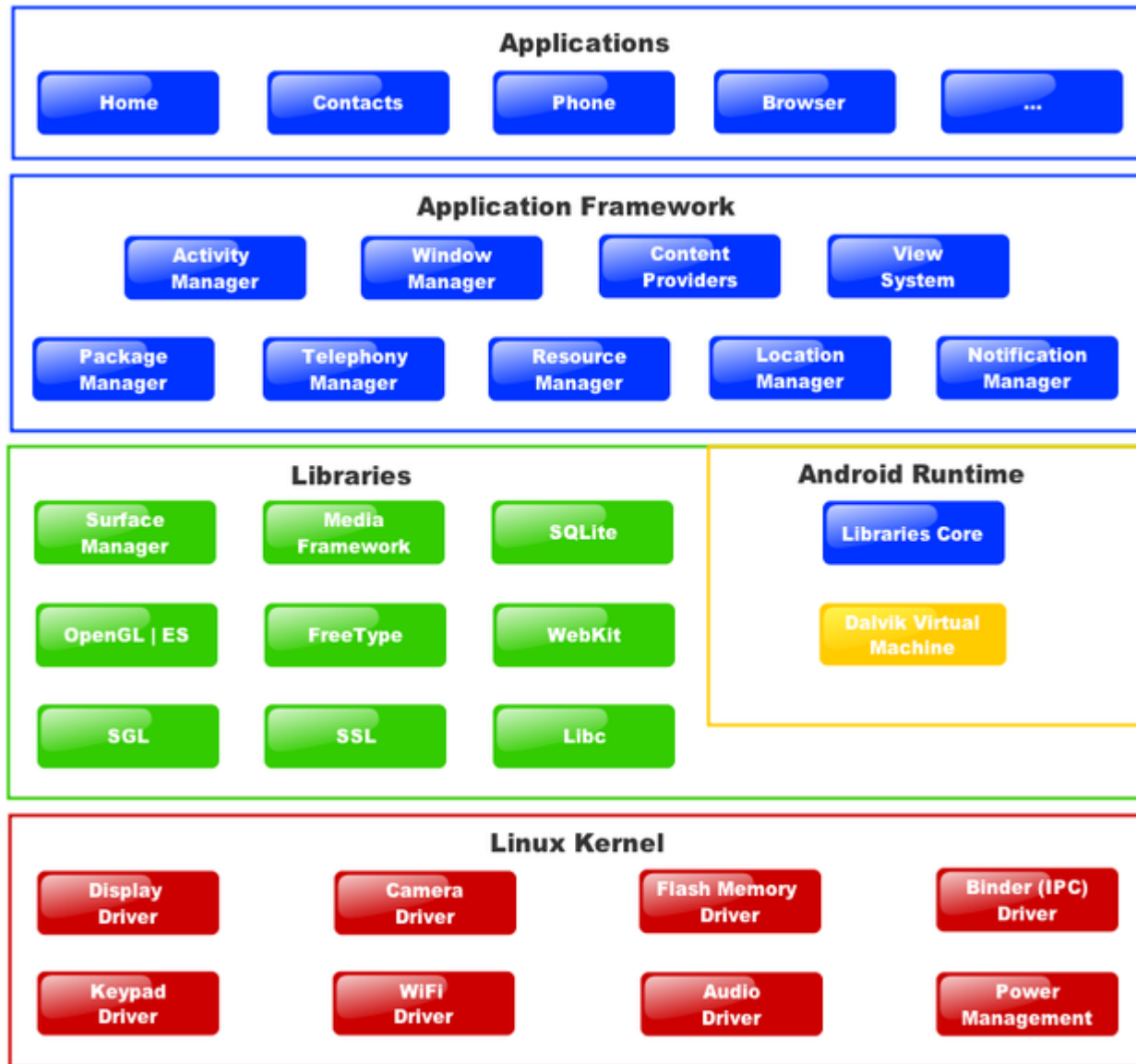


SUPPORT LIBRARY

Revisions

This section provides details about the Support Library package releases.

- ▶ [Android Support Library, revision 23.2.0](#) (*February 2016*)
- ▶ [Android Support Library, revision 23.1.1](#) (*November 2015*)
- ▶ [Android Support Library, revision 23.1.0](#) (*October 2015*)
- ▶ [Android Support Library, revision 23.0.1](#) (*September 2015*)
- ▶ [Android Support Library, revision 23](#) (*August 2015*)
- ▶ [Android Support Library, revision 22.2.1](#) (*July 2015*)
- ▶ [Android Support Library, revision 22.2.0](#) (*May 2015*)
- ▶ [Android Support Library, revision 22.1.0](#) (*April 2015*)
- ▶ [Android Support Library, revision 22](#) (*March 2015*)
- ▶ [Android Support Library, revision 21.0.3](#) (*December 2014*)
- ▶ [Android Support Library, revision 21.0.2](#) (*November 2014*)



ZÁKLADNÍ STAVEBNÍ BLOKY

◉ Activity

- jedna obrazovka UI, nezávislé na sobě
- můžeme pustit aktivitu jiné aplikace
- Do aktivity můžeme „umístit fragment“

◉ Service

- běží na pozadí, nemá UI
př. přehrávání hudby, stahování dat na pozadí

◉ Content Provider

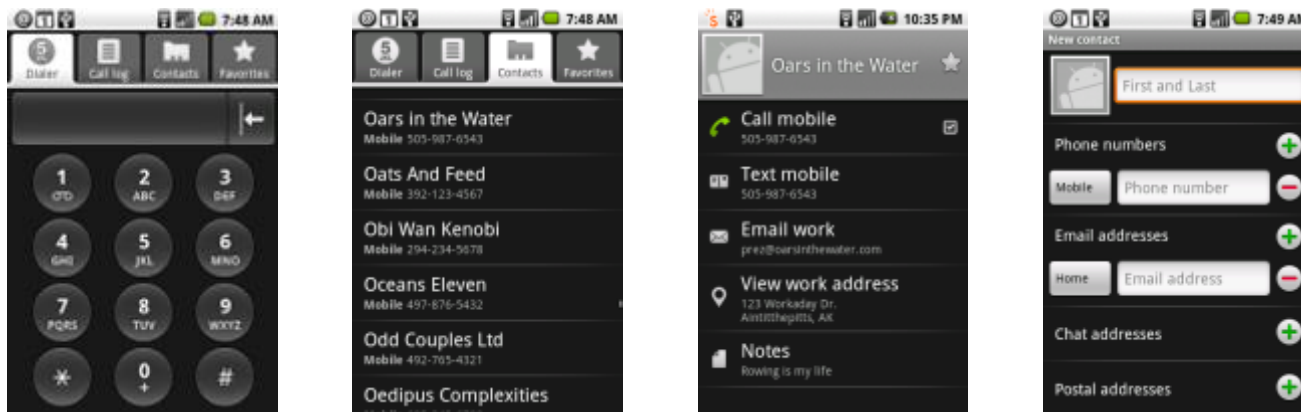
- správa sdílených dat (fs, databáze, web,...)
- aplikace se může dotazovat na data nebo je měnit
- jak sdílet data ven z aplikace (sms, kontakty,...)

◉ Broadcast Receiver

- reaguje na zprávy vně i zevnitř aplikace

ACTIVITY

- Hlavní stavební blok aplikací
- aplikace se může skládat z 1 nebo více aktivit
- Představuje vizuální UI, např. výběr položky z menu, zobrazení fotky atd.



Příklad 4 aktivit - dialer, contacts, view contact, new contact

PŘ AKTIVIT: KALENDÁŘ, HRA

◉ Kalendář - aktivity

- Zobrazit den
- Zobrazit týden
- Zobrazit měsíc
- Zobrazit agendu
- Editace události
- Editace preferencí
- Zobrazit alert

◉ Hra - aktivity

- Hraní hry
- Nastavení hry

ACTIVITY STACK

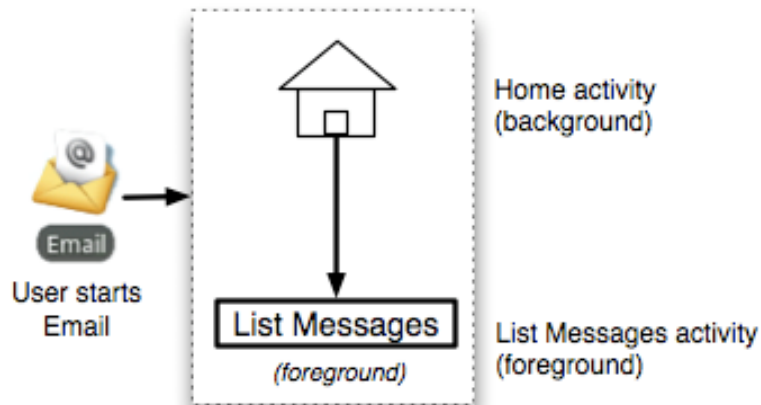
- Systém si pamatuje historii aktivit, jak se uživatel přesune k nové aktivitě
 - Lineární navigační historie aktivit
 - Tlačítkem BACK se lze vrátit k předchozí aktivitě
-
1. pustí se nová aktivita
 2. předchozí aktivitu pozastaví, zásobník LIFO
 3. uživatel stiskne BACK, ze zásobníku vybere poslední aktivitu
právě vykonávaná aktivita se zruší

TASK

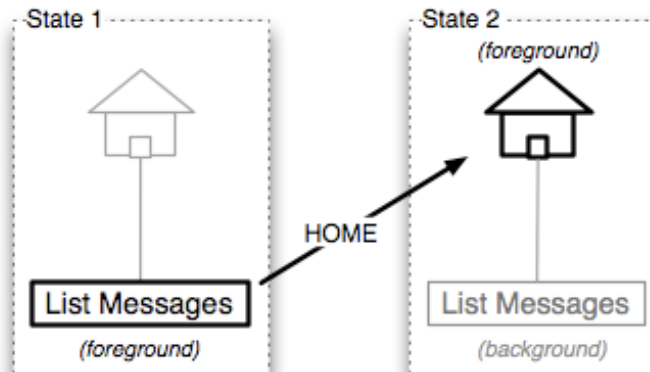
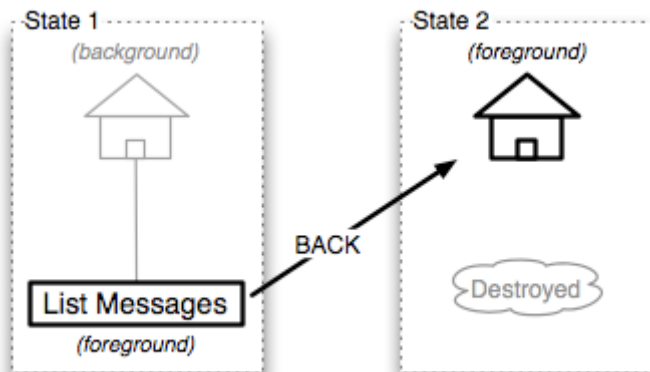
- ◉ Posloupnost aktivit, kterou uživatel vykoná pro splnění nějakého cíle
- ◉ Př.: posílání textové zprávy s přílohou
 - přepínáme se postupně mezi více obrazovkami

SPUŠTĚNÍ AKTIVITY

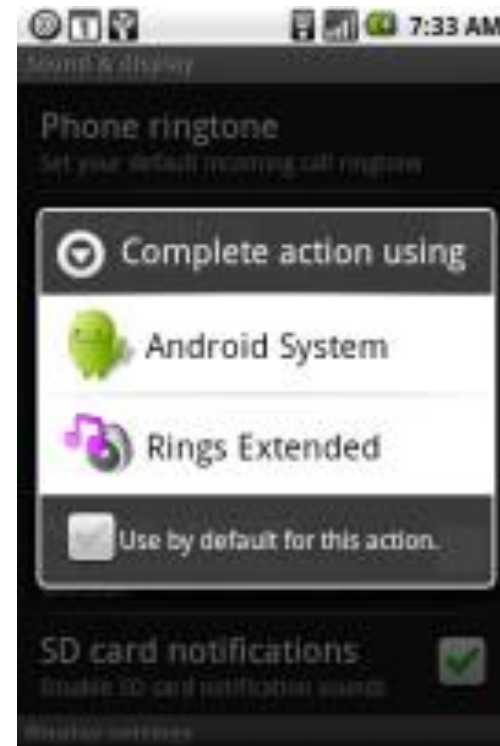
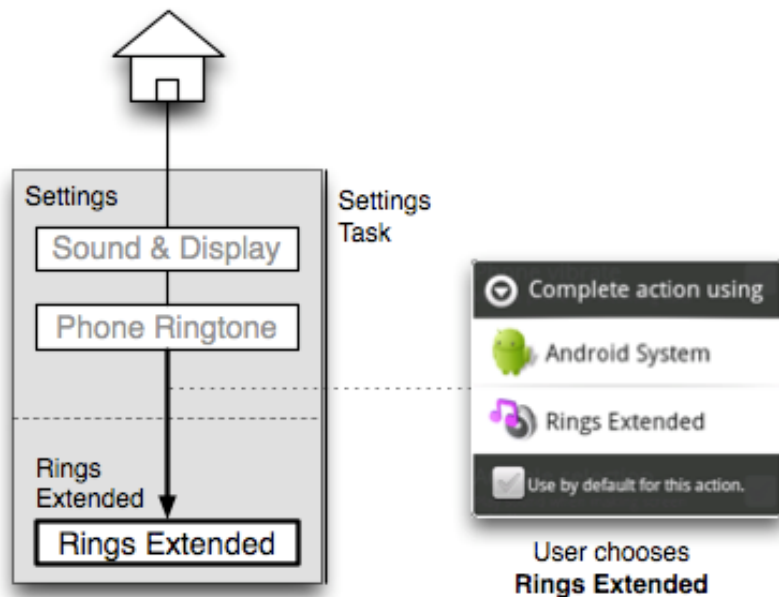
VLIV TLAČÍTEK BACK, HOME



Tlačítko BACK - zrušení aktivity
Tlačítko HOME - přesun aktivity do pozadí



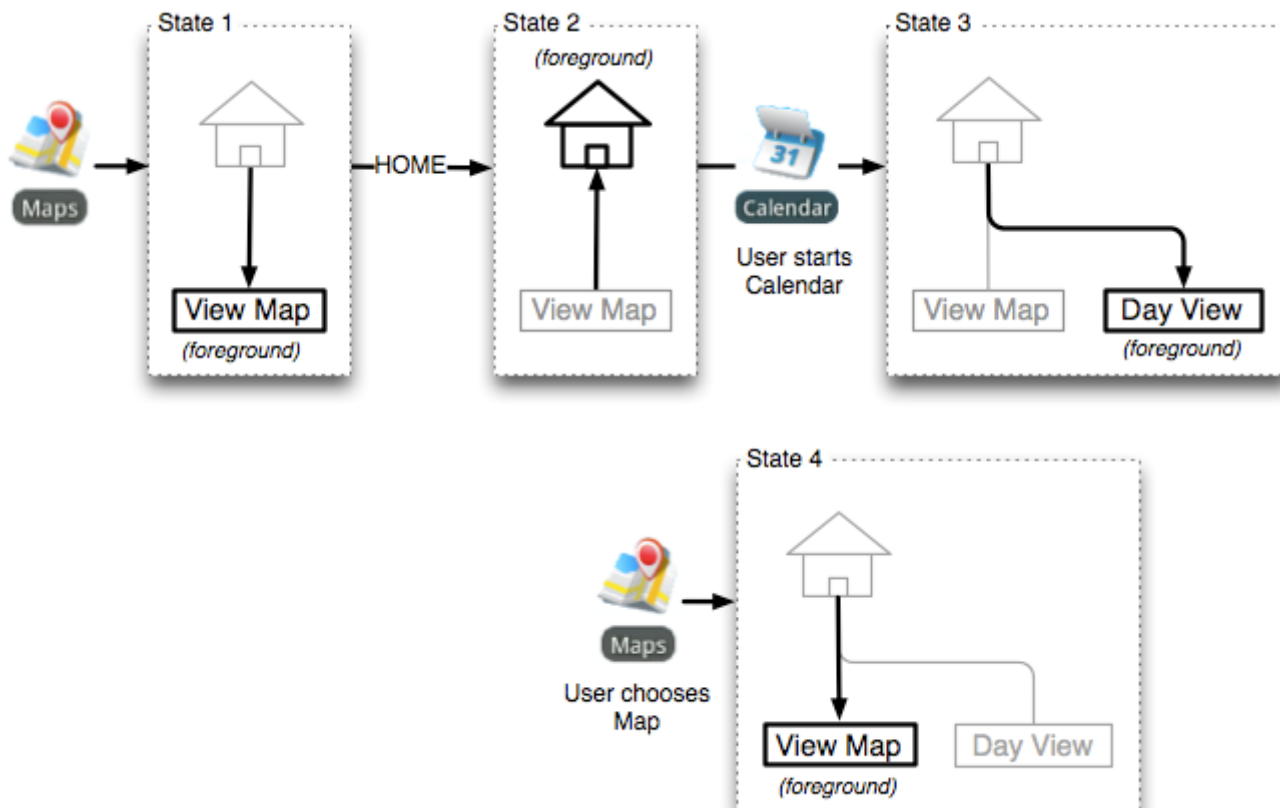
NAHRAZENÍ AKTIVITY



jednomu záměru může vyhovovat více aktivit

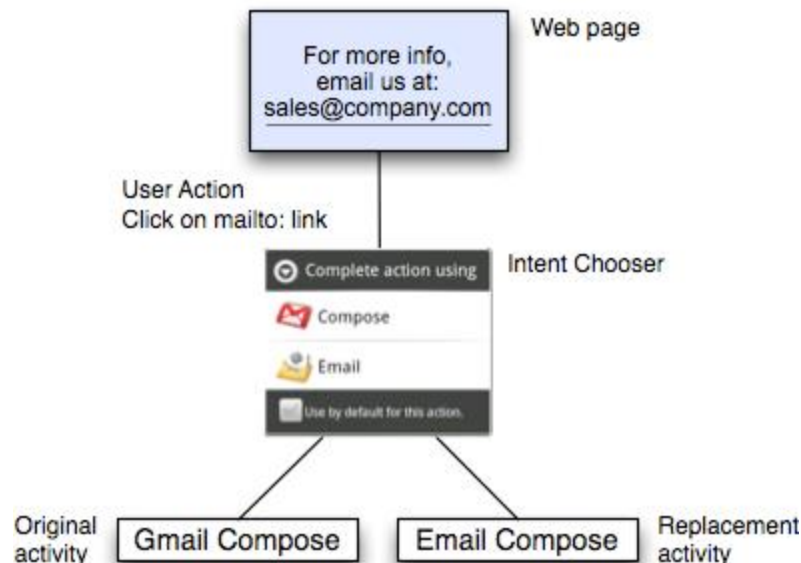
MULTITASKING

př.: zatímco se načítá mapa, prohlédnu si kalendář
a pak se vrátím k mapě



INTENT (ZÁMĚR)

Když uživatel vykoná určitou akci nad nějakými daty, např. touching <mailto:info@nekde.neco.cz> inicializuje se **intent** (intent objekt) a výsledkem může pak být spuštění aktivity



abstrakce
operace,
kterou je
třeba
provést

INTENTY

- ⊙ generované systémem
 - přišla SMS, změna polohy, ...
- ⊙ generované aplikací
 - otevři URL, vyfoť snímek, přehraj hudbu
- ⊙ struktura intentu
 - akce - ACTION_VIEW, ACTION_EDIT
 - data - ve formě Uri

<http://developer.android.com/reference/android/content/Intent.html>

INTENTY

◉ reakce na Intent

- kombinace akce a dat (např. URL)
akce:
 - `android.intent.action.VIEW`
 - `android.intent.action.SEND`
- na Intent může reagovat více aplikací

INTENTY

◉ explicitní

- specifikují exaktní třídu
- spuštění vnitřní aktivity uvnitř naší aplikace

◉ implicitní

- nespecifikují komponentu
- systém rozhodne, jaká komponenta bude vhodná

INTENTY

```
public void dalsiAktivita(View v) {  
    Intent i = new Intent(this, MainActivity2.class);  
    startActivity(i);  
}
```

```
public void call (View v) {  
    Intent callIntent = new Intent(Intent.ACTION_DIAL);  
    callIntent.setData(Uri.parse("tel:5556"));  
    startActivity(callIntent);  
}
```

INTENTY - PŘEDÁNÍ DAT

- ⊙ Aktivita předávající data:

```
Intent i = new Intent(this, Main2Activity.class);  
i.putExtra("ZPRAVA", vysledek);  
startActivity(i);
```

- ⊙ Spuštěná aktivita využije data:

```
Intent i = getIntent();  
String message = i.getStringExtra("ZPRAVA");
```

INTENT - PUTEXTRA

Předává data:

```
int intArray[] = {1,2,3,4};  
Intent in = new Intent(this, B.class);  
in.putExtra("my_array", intArray);  
startActivity(in);
```

Přijíma data:

```
Bundle extras = getIntent().getExtras();  
int[] arrayInB = extras.getIntArray("my_array");
```

<http://stackoverflow.com/questions/5219788/put-extra-in-android>

INTENTY

- ◉ ACTION_VIEW content://contacts/people/1
- ◉ ACTION_DIAL content://contacts/people/1
- ◉ ACTION_DIAL tel:123
- ◉ ACTION_EDIT content://contacts/people/1

- ◉ Category
CATEGORY_LAUNCHER, CATEGORY_ALTERNATIVE
- ◉ Type
explicitně definovat MIME type
- ◉ Extras
Bundle obsahující další informace

INTENTY

- ◉ ACTION_MAIN s kategorií LAUNCHER
 - Bude uvedena v seznamu aplikací pro spuštění
- ◉ ACTION_GET_CONTENT s MIME typem `vnd.android.cursor.item/phone`
 - Zobrazení seznamu čísel

INTENTY, KTERÉ UMÍ NAŠE AKTIVITA OBSLUHOVAT

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <action android:name="android.intent.action.EDIT" />
    <action android:name="android.intent.action.PICK" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

Aktivita umí zpracovat adresář poznámek

Cursor žádné nebo více položek - `vnd.android.cursor.dir`

Data (obsah jedné položky) - `vnd.google.note`

Uživatel si může prohlídnout nebo editovat adresář dat (`VIEW`, `EDIT`),

Může vybrat určitou poznámku a vrátit ji volající aktivitě (`PICK`)

Kategorie `DEFAULT` umožní zavolat aktivitu přes `Context.startActivity` bez explicitního uvedení celého jména.

Viz: <https://developer.android.com/reference/android/content/Intent.html>

ANDROID LAUNCHER

- ◉ Home screen (desktop telefonu)
 - ◉ App drawer („šuplík na aplikace“)
 - ◉ Spuštění mobilních aplikací
 - ◉ Telefonování, aj.
-
- ◉ Vestavěný Launcher (defaultní androidí)
 - ◉ Výrobci zařízení mají svoje (HTC Sensa UI, Samsung TouchWiz)
 - ◉ Další ke stažení z Google Play

VSTUPNÍ BODY APLIKACE

- ◉ tzv. **LAUNCHER**, viz manifest
- ◉ seznam všech těchto launcher záznamů v telefonu -> seznam aplikací na domovské obrazovce

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="toast.pesi.cz"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Toastik"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>

</manifest>
```

MANIFEST

- ◉ **AndroidManifest.xml**
- ◉ XML soubor v kořenovém adresáři
- ◉ unikátní **jméno balíčku aplikace**
- ◉ deklarace **komponent (aktivit, služeb ...)**
- ◉ deklarace **oprávnění**, které aplikace požaduje (přístup k síti, čtení kontaktů,...)
- ◉ deklarace **minimum API level (doplní Gradle)**
- ◉ deklarace **používaného hw, sw** (kamera, bluetooth služby, multitouch...)
- ◉ API **library**, které chce využívat (např. Google Maps library - klíč)

MANIFEST - OBECNÁ PRAVIDLA

- ◉ elementy `<manifest>` a `<application>` v manifestu **jen jednou**
- ◉ atributy elementů typicky začínají **android:**
- ◉ více variant jednoho atributu (např. oprávnění) - opakuje se daný element
- ◉ podtřídy Aktivit, Služeb aj. deklarovány pomocí atributu `android:name`
- ◉ podpora lokalizace
@drawable/icon, ...

MANIFEST - KOMPONENTY

- ⦿ komponenty je potřeba deklarovat:
 - <activity> .. pro aktivitu
 - <service> .. pro služby
 - <receiver> .. pro broadcast receivery
 - <provider> .. pro content provider
- ⦿ pokud jsou komponenty ve zdrojovém kódu, ale nejsou v manifestu - nemohou být spuštěny (výjimkou broadcast receiver)

MANIFEST - CAPABILITY

- ◉ jaké intenty může naše aplikace poskytnout ostatním
- ◉ přidání elementu **<intent-filter>**
- ◉ hlavní aktivita naší aplikace -
action MAIN, kategorie LAUNCHER

system identifikuje komponenty, které mohou reagovat na intent tím, že prohledá přijatý intent s **intent filters** v manifestech aplikací

MANIFEST S JEDNOU AKTIVITOU

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cz.pesi.pok000"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.cz.pesi.pok000.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```


RECEIVER - PŘÍJEM SMS

...

```
<receiver android:name="SMSPrijem">
```

```
<intent-filter>
```

```
<action android:name=  
    "android.provider.Telephony.SMS_RECEIVED">
```

```
</action>
```

```
</intent-filter>
```

```
</receiver>
```

...

MANIFEST - POŽADAVKY APLIKACE

- ◉ např: „*chci kameru a API Level 7 a výše*“
- ◉ **screen size and density** (<supports-screens>)
- ◉ **input configuration** <uses-configuration>
 - hw klávesnice, trackball,
- ◉ **device features** <uses-feature>
 - kamera, světelný senzor, bluetooth
- ◉ **platform version** <uses-sdk>

mějte na paměti rozmanitost zařízení, kde všude Android běží

MANIFEST - PERMISSIONS

- ◉ chránit kritická data a služby přes zneužitím
- ◉ každé právo jednoznačný název

`android.permission.CALL_EMERGENCY_NUMBERS`

`android.permission.SET_WALLPAPER`

`android.permission.READ_CONTACTS`

`android.permission.SEND_SMS`

aplikace deklaruje v `<uses-permission>`

`<uses-permission android:name="android.permission.INTERNET">`

Vkládáme před `application` v manifestu

PERMISSIONS - PŘÍKLAD

```
<uses-permission  
    android:name="android.permission.ACCESS_WIFI_STATE" />
```

```
<uses-permission  
    android:name="android.permission.CHANGE_WIFI_STATE" />
```

seznam:

<http://developer.android.com/reference/android/Manifest.permission.html>

Struktura manifestu:

<http://developer.android.com/guide/topics/manifest/manifest-intro.html>

UMÍSTĚNÍ V MANIFESTU

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.pesicka.myapplication" >

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="My Application" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name=".MainActivity2Activity"
            android:label="MainActivity2Activity"
            android:parentActivityName=".MainActivity" >
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="com.example.pesicka.myapplication.MainActivity" />
        </activity>

    </application>

</manifest>
```

USER INTERFACE - LAYOUT

- ◉ Programově
- ◉ XML soubor

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, I am a TextView" />

    <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello, I am a Button" />
  </LinearLayout>
```

DEFINICE UI

- Definice v layout xml souboru, unikátní id:

```
<Button android:id="@+id/my_button"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="@string/my_button_text"/>
```

R.id.my_button

- Vytvoření instance s využitím layoutu:

Pro usnadnění
internacionalizace
nemusíme text
vkládat přímo

```
Button myButton = (Button)  
    findViewById(R.id.my_button);
```

<http://developer.android.com/reference/android/Manifest.permission.html>

ALTERNATIVA FINDVIEWBYID

Data binding

From an Activity, instead of:

```
setContentView(R.layout.hello_world);  
TextView hello = (TextView) findViewById(R.id.hello);  
hello.setText("Hello World"); // for example, but you'd use  
                               // resources, right?
```

You load it like this:

```
HelloWorldBinding binding =  
    DataBindingUtil.setContentView(this, R.layout.hello_world);  
binding.hello.setText("Hello World"); // you should use resources!
```


DATA BINDING

Podrobný popis a zdroj zde:

<https://medium.com/google-developers/no-more-findviewbyid-457457644885#.t849zeu6l>



George Mount

Jun 21, 2016 · 1 min read

This mechanism doesn't change anything in that regard. Normally, after you inflate, you find the views and set the data. This mechanism **is just finding the views** for you and you can **set the data immediately** so that the user doesn't see any blank data.



yukuku

Jun 22, 2016

How about Instant Run? Is it still incompatible?



George Mount

Jun 22, 2016

Yes!

ŽIVOTNÍ CYKLUS AKTIVITY

- ◉ Aktivní, running
 - Na popředí obrazovky, zaměřena pozornost uživatele
- ◉ Paused
 - Ztracen focus, stále viditelná (částečně) uživateli
 - Např. leží pod jinou aktivitou, která nezakrývá celou plochu
- ◉ Stopped
 - Překryta jinou aktivitou

FCE VOLANÉ PŘI PŘECHODU MEZI STAVY AKTIVITY

```
void onCreate(Bundle savedInstanceState)  
void onStart()  
void onRestart()  
void onResume()  
void onPause()  
void onStop()  
void onDestroy()
```

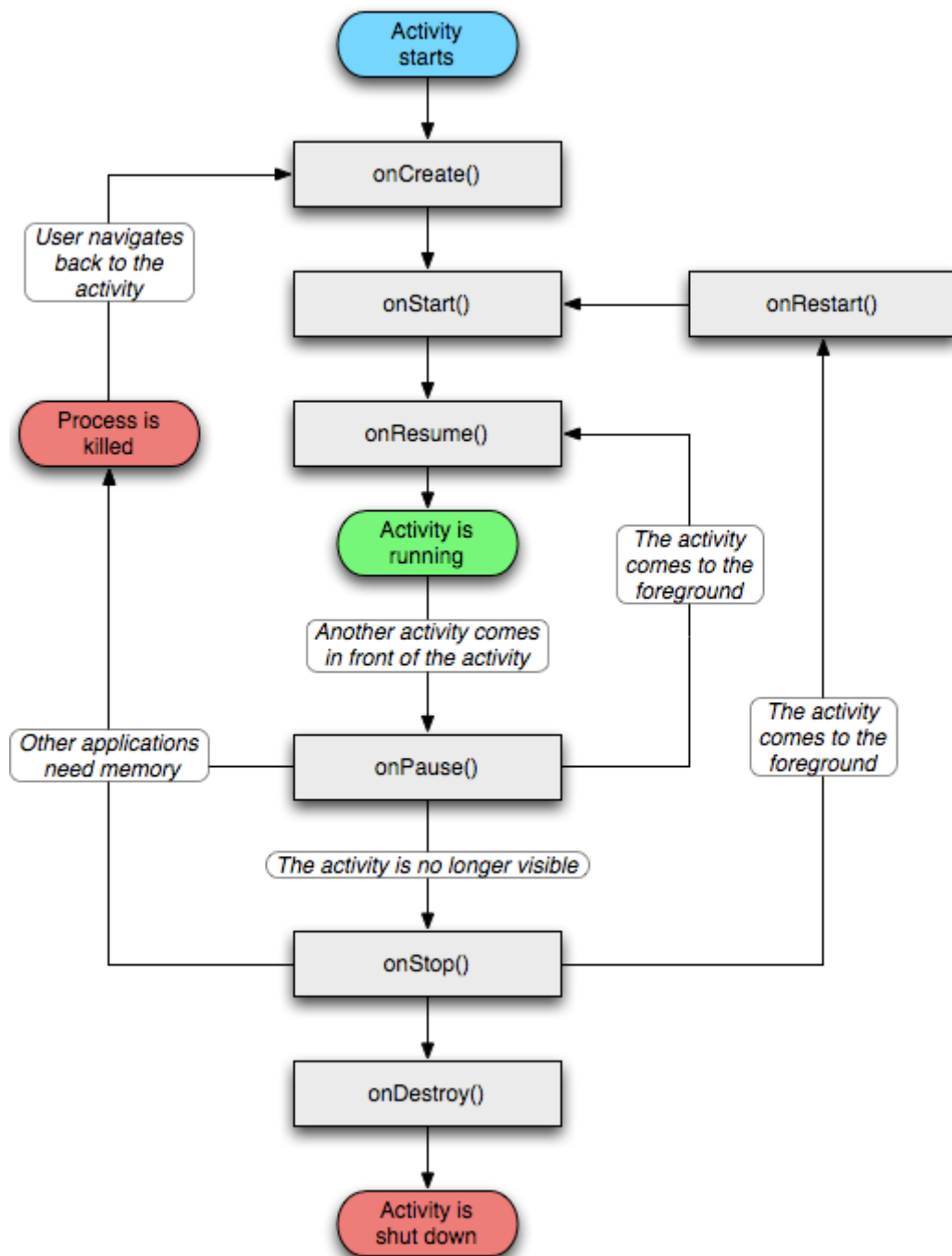
kdy pustit a ukončit naslouchání pozice z GPS?

.. úspora baterie - důležité !

kdy přerušit akci hry?

.. nepřítel zaútočí, když nemůžu reagovat

kdy uložit data, aby o ně uživatel nepřišel?



ROZDÍL ONPAUSE, ONSTOP

- ◉ Zdají se být podobné
- ◉ Jedna volána vždy

That leaves a question which activity is considered fully opaque and covering the whole screen and which isn't. This decision is based on the window containing the activity. If the window has a flag `windowIsFloating` OR `windowIsTranslucent`, then it is considered that the activity doesn't make the underlying stuff invisible, otherwise it does and will cause `onStop()` to be called. The relevant code can be found in `com.android.server.am.ActivityRecord` :

<http://stackoverflow.com/questions/9266417/difference-between-onpause-and-onstop>

OBSLUHA UDÁLOSTI - TLAČÍTKO

Listener (posluchač)

- ◉ Programově
- ◉ Deklarativně

Každé tlačítko vlastní posluchač (typicky anonymní třída)

Lze i společný listener

DEKLARACE LISTENERU

V souboru main.xml:

```
android:id="@+id/button1"  
    android:text="Konec"  
    android:onClick="StisknutoTlacitko"
```



Deklarace
události

Obslužná funkce:

```
public void StisknutoTlacitko(View v) {  
    finish();  
}
```

LISTENER PROGRAMOVĚ

Do metody *onCreate()* přidáme:

```
final Button button3 = (Button)
    findViewById(R.id.button3);
```

```
button3.setOnClickListener(new
    View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Toast.makeText(v.getContext(),
                "ahoj", Toast.LENGTH_LONG).show();
        }
    });
```

Použití
anonymní třídy

Zobrazí info
zprávu (toast)

EDITTEXT - POUŽITÍ

```
final EditText edit1 = (EditText)  
    findViewById(R.id.editText1);
```

// získání číselné hodnoty

```
int cislo =  
    Integer.parseInt(edit1.getText().toString());
```

// změna obsahu

```
edit1.setText(String.valueOf(cislo));
```

POUŽITÁ LITERATURA

Zpracováno s využitím materiálů a obrázků z:

<http://developer.android.com/>

<http://www.root.cz/clanky/programovani-pro-android-v-prikladech/>

<http://zdrojak.root.cz/clanky/vyvoj-pro-android-ii/>

<http://www.onlinecomputerbooks.com/free-android-books.php>