Razvoj aplikacij na mobilni platformi Android

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

O Androidu

Klemen Peternel

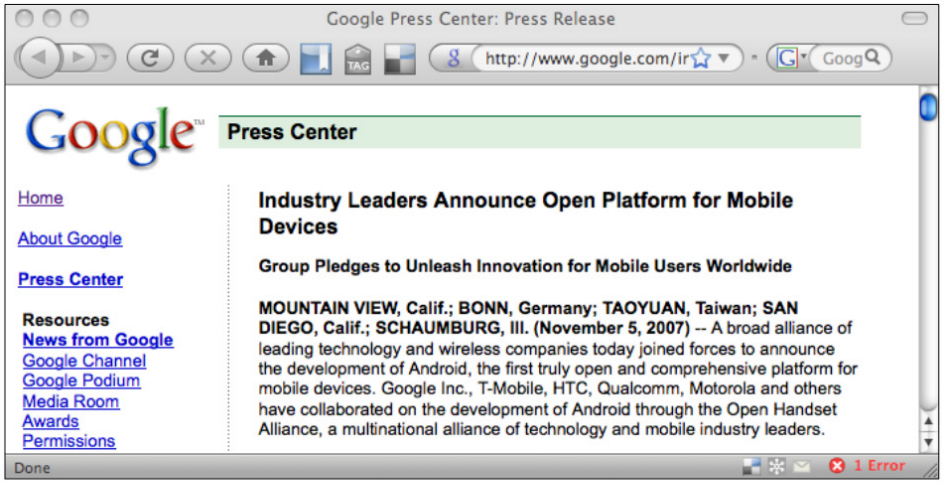
Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Google

Napovedi prihoda

August 2005

Google htc T-Mobile docomo 中国移动通信 CHINA MOBILE SAMSUNG LG QUALCOMM Sprint intel



November 2007

Prvi rezultati

November 2007

September 2008

G1 technical specs

- Qualcomm MSM7201A, 528 MHz
- ROM 256 MB
- RAM 192 MB
- 4.60 in x 2.16 in x 0.62 in
- 158 grams
- Lithium Ion battery, 1150 mAh
- 3G (HSDPA)
- touch screen, HVGA 320x480
- QWERTY keyboard
- 3.2 megapixel camera
- microSD expansion slot
- GPS, compass, accelerometer

Sedanjost

- **Samsung Galaxy S II**
  - **Dual-core 1200 MHz**
  - **1024 MB RAM**
  - **4.93 x 2.60 x 0.33 in**
  - **116 g**
  - **1650 mAh battery**
  - **4G (HSPA+)**
  - **Touchscreen 480 x 800**
  - **8 megapixel camera**
  - **GPS, compass, accelerometer, gyroscope**
  - **NFC**

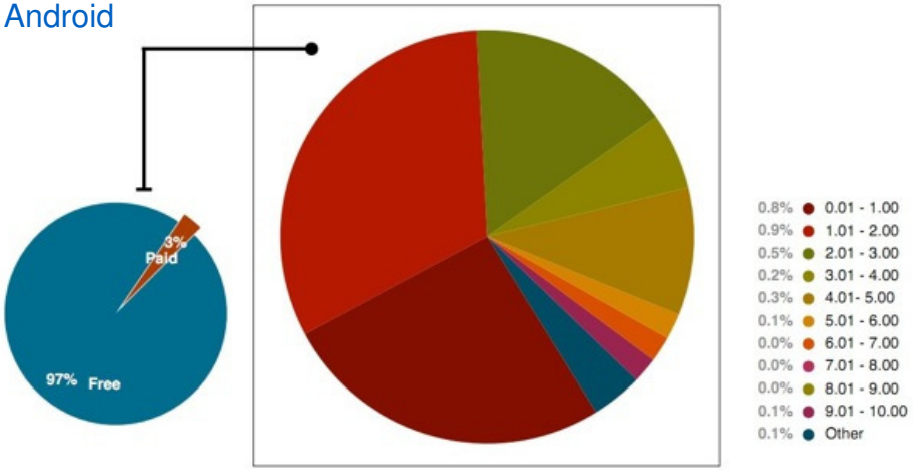
Splošno o Androidu

- Celotna rešitev nastala pod okriljem Open Handset Alliance
  - Danes skupina 80 tehnoloških podjetij
- Google je Android, ki je obstajal od leta 2003 priključil k sebi dve leti kasneje
- Android je celoten programski sklad za mobilne naprave
  - OS (osnovan na okleščenem Linux jedru)
  - Middleware
  - Osnovne aplikacije
- Android je trenutno najbolj prodajana platforma za pametne telefone
  - Podatki iz Google I/O 2011: 400K novih naprav vsak dan; več kot 100M trenutno aktivnih
- Android je odprtokoden (zaščiten z Apache licenco)
- Viri na temo razvoja na Androidu:
  - http://developer.android.com/index.html
  - http://appinventor.googlelabs.com/about/
  - http://www.vogella.de/android.html

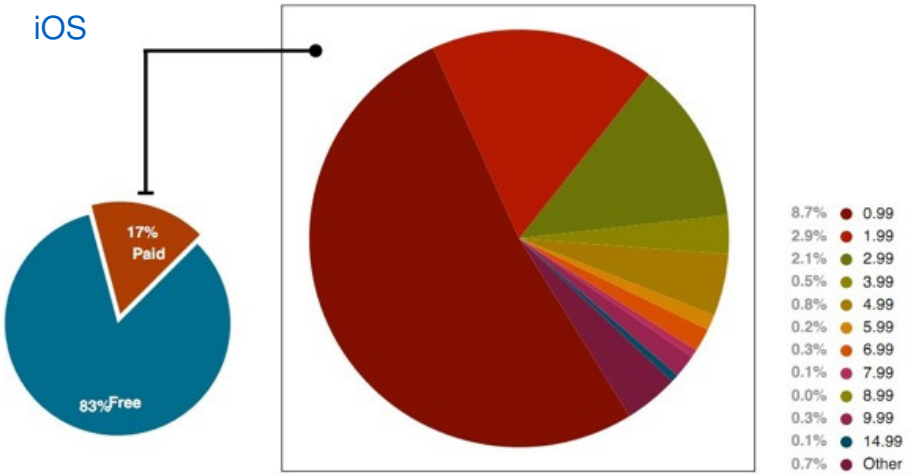
97% naloženih aplikacij je zastonj!

- Vir: Chomp – april 2011 (USA)

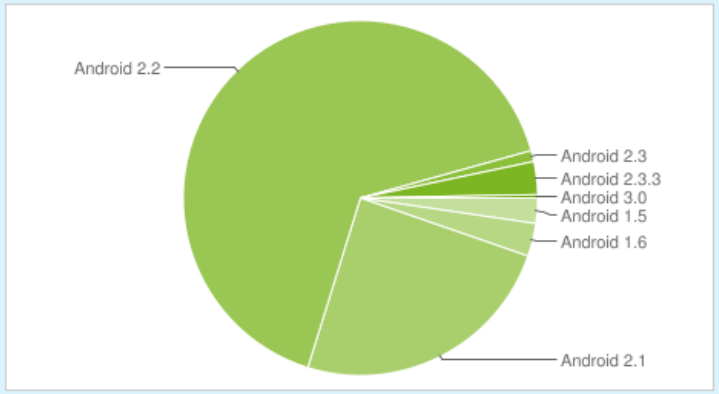
Android



iOS

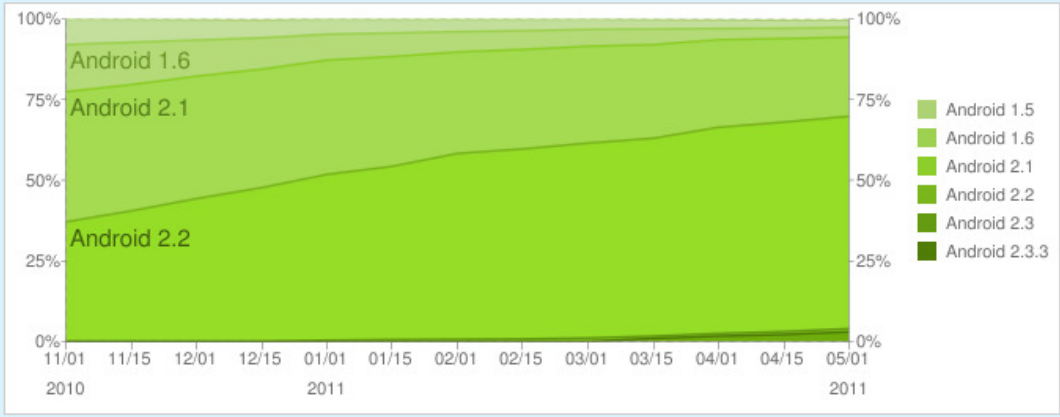


Različice Androida - dashboard



Platform	API Level	Distribution
Android 1.5	3	2.3%
Android 1.6	4	3.0%
Android 2.1	7	24.5%
Android 2.2	8	65.9%
Android 2.3	9	1.0%
Android 2.3.3	10	3.0%
Android 3.0	11	0.3%

Data collected during two weeks ending on May 2, 2011



Last historical dataset collected during two weeks ending on May 2, 2011

Kaj različice prinašajo v praksi?

- Vsaka različica platforme ima svojo oznako API-ja
  - API Level – integer, ki unikatno označuje t.i. Framework API
- Framework API sestavljajo
  - Jedrni paketi in razredi
  - XML elementi in atributi za deklaracijo manifest datoteke
  - XML elementi in atributi za deklaracijo in dostop do virov
  - Namere (Intent)
  - Dovoljenja (permissions)
- Aplikacije z uporabo parametra API Level določajo različico ciljne platforme
  - `Android:minSdkVersion` -> Minimalen API Level
  - `Android:targetSdkVersion` -> Ciljni API Level
  - `Android:maxSdkVersion` -> Maksimalen API Level (!!!)

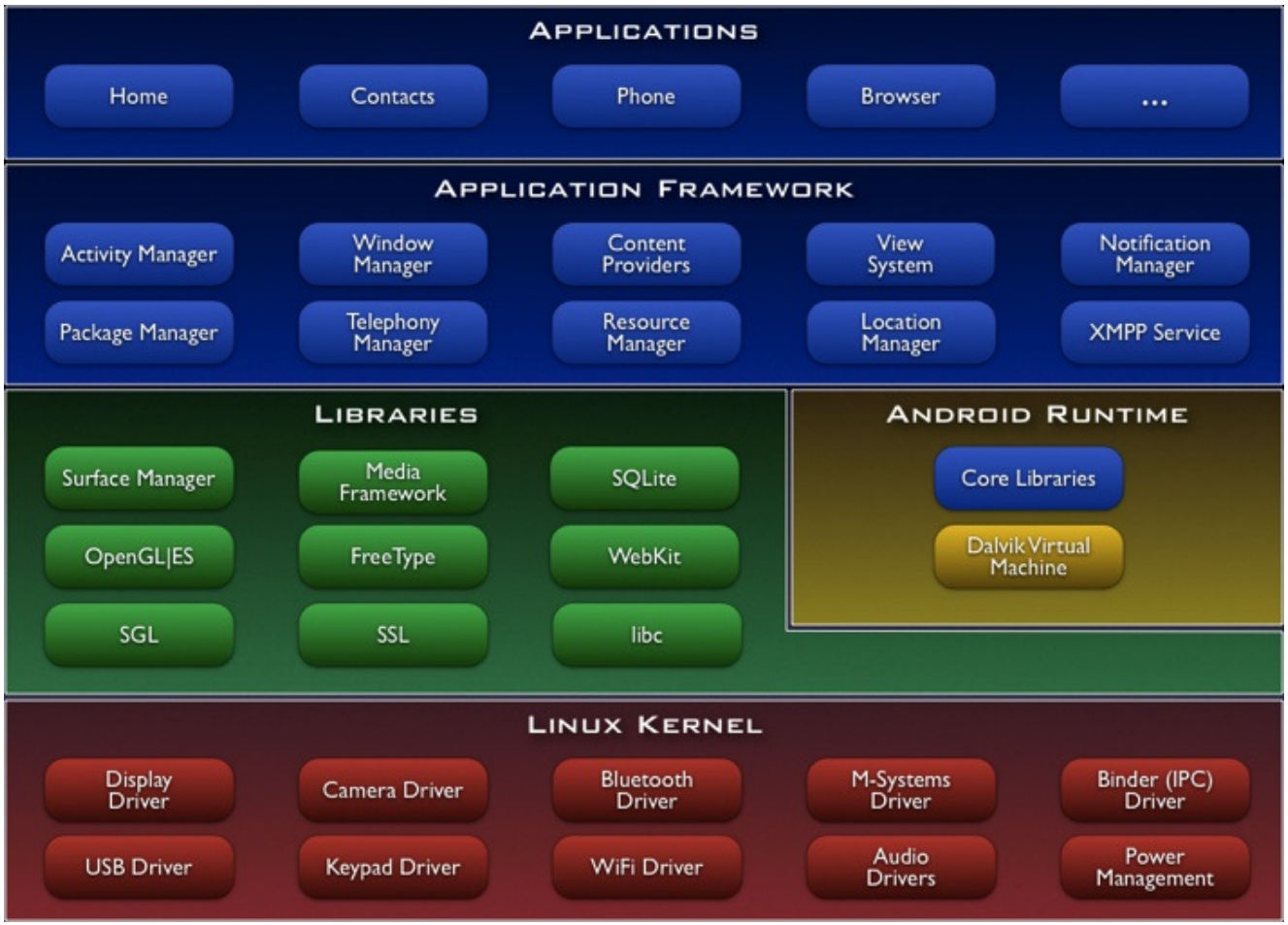
Platform Version	API Level
Android 3.1	12
Android 3.0	11
Android 2.3.4	10
Android 2.3.3	
Android 2.3	9
Android 2.2	8
Android 2.1	7
Android 2.0.1	6
Android 2.0	5
Android 1.6	4
Android 1.5	3
Android 1.1	2
Android 1.0	1

Funkcionalnosti

Handset layouts	The platform is adaptable to larger, VGA, 2D graphics library, 3D graphics library based on OpenGL ES 2.0 specifications, and traditional smartphone layouts.
Storage	SQLite, a lightweight relational database, is used for data storage purposes
Connectivity	Android supports connectivity technologies including GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi (no connections through Proxy server[61] and no Ad hoc wireless network[62]), LTE, NFC and WiMAX.
Messaging	SMS and MMS are available forms of messaging, including threaded text messaging and now Android Cloud to Device Messaging Framework (C2DM) is also a part of Android Push Messaging service.
Multiple Language Support	Multiple languages are available on Android. The number of languages more than doubled for the platform 2.3 (Gingerbread). Yet Android lacks font rendering of several languages even after official announcements of added support (e.g. Hindi).
Web browser	The web browser available in Android is based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine. The browser scores a 93/100 on the Acid3 Test.
Java support	While most Android applications are written in Java, there is no Java Virtual Machine in the platform and Java byte code is not executed. Java classes are compiled into Dalvik executables and run on the Dalvik virtual machine. Dalvik is a specialized virtual machine designed specifically for Android and optimized for battery-powered mobile devices with limited memory and CPU. J2ME support can be provided via third-party-applications.
Media support	Android supports the following audio/video/still media formats: WebM, H.263, H.264 (in 3GP or MP4 container), MPEG-4 SP, AMR, AMR-WB (in 3GP container), AAC, HE-AAC (in MP4 or 3GP container), MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, BMP.[60]
Streaming media support	RTP/RTSP streaming (3GPP PSS, ISMA), HTML progressive download (HTML5 <video> tag). Adobe Flash Streaming (RTMP) and HTTP Dynamic Streaming are supported by the Flash 10.1 plugin.[63] Apple HTTP Live Streaming is supported by RealPlayer for Mobile,[64] and by the operating system in Android 3.0 (Honeycomb).[49] Microsoft Smooth Streaming is planned to be supported through the awaited port of Silverlight plugin to Android.
Additional hardware support	Android can use video/still cameras, touchscreens, GPS, accelerometers, gyroscopes, magnetometers, dedicated gaming controls, proximity and pressure sensors, thermometers, accelerated 2D bit blits (with hardware orientation, scaling, pixel format conversion) and accelerated 3D graphics.
Multi-touch	Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero. The feature was originally disabled at the kernel level (possibly to avoid infringing Apple's patents on touch-screen technology at the time).[65] Google has since released an update for the Nexus One and the Motorola Droid which enables multi-touch natively.[66]
Bluetooth	Supports A2DP, AVRCP, sending files (OPP), accessing the phone book (PBAP), voice dialing and sending contacts between phones. Keyboard, mouse and joystick (HID) support is available through manufacturer customizations and third-party applications. Full HID support is planned for Android 3.0 (Honeycomb).[49]
Video calling	The mainstream Android version does not support video calling, but some handsets have a customized version of the operating system which supports it, either via UMTS network (like the Samsung Galaxy S) or over IP. Video calling through Google Talk is planned for Android 3.0 (Honeycomb). Android 2.3.4 has added Video calling through Google Talk.
Multitasking	Multitasking of applications is available.[67]
Voice based features	Google search through voice has been available since initial release.[68] Voice actions for calling, texting, navigation, etc. are supported on Android 2.2 onwards.[69]
Tethering	Android supports tethering, which allows a phone to be used as a wireless/wired hotspot. Prior to Android 2.2 this was supported by third-party applications or manufacturer customizations.[70]

Arhitektura platforme Android

- Vsaka aplikacija dobi svojo instanco Dalvik VM
  - Vsak javanski razred se pretvori v Dalvik Executable (.dex) format



Application Framework - AF

- Razvijalec ima poln dostop do vseh API-jev, ki jih izkoriščajo tudi bazične aplikacije
  - Arhitektura aplikacij je takšna, da omogoča pouporabo komponent
  - Vsaka aplikacija lahko objavi svoje zmožnosti, ki jih druge lahko izkoriščajo
- Preko AF razvijalec dostopa do C/C++ knjižnic Android platforme
- Aplikacije izkoriščajo različne storitve in sisteme znotraj AF
  - View System -> grafični elementi
  - Content Providers -> dostop do podatkov drugih aplikacij
  - Resource Manager -> dostop do zunanjih virov (grafika, lokalizacija)
  - Notification Manager -> prikaz sporočil v statusni vrstici
  - Activity Manager -> nadzor nad življenjskim ciklom aktivnosti

Android aplikacije

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Osnove

- Aplikacije so pisane v programskem jeziku Java
  - Aplikacija je pakirana v datoteko s končnico .apk
- Dele aplikacije možno napisati tudi v C/C++ (NDK)
- Na Android napravi vsaka aplikacija živi v svojem “peskovniku”
  - Vsaka aplikacija je svoj uporabnik znotraj OS-a in dobi lasten Linux user ID
  - Vsaka aplikacija je svoj Linux proces
  - Vsak proces dobi svojo instanco VM
- Aplikacija lahko dostopa samo do tistih komponent sistema za katere ima dodeljene pravice
- Aplikacija lahko deli lastne podatke med druge aplikacije in dostopa do sistemskih podatkov/storitev
  - Dve aplikaciji imata lahko isti user ID za dostop do podatkov druge (posledično sta lahko del istega procesa in uporabljata isto instanco VM)
  - Aplikacija lahko zahteva pravice za dostop do podatkov/storitev naprave (kontakti, SMS-i, kamera, BT, SD, ...)

Komponente

- Komponente so osnovni gradniki Android aplikacij
- Vsaka ima svoj življenjski cikel
- Aktivnosti (Activity)
  - Predstavlja del uporabniškega vmesnika (en prikaz na zaslonu)
  - Vsaka aplikacija običajno sestoji iz več aktivnosti
- Storitve (Service)
  - Storitev se izvaja v ozadju in je namenjena izvedbi dolgotrajnih operacij
  - Storitev nima uporabniškega vmesnika
- Ponudniki vsebine (Content providers)
  - Omogočajo delo s podatki (poizvedovanje, shranjevanje)
- Sprejemniki obvestil (Broadcast receivers)
  - Aplikacija se lahko prijavi na sprejemanje sporočil s strani drugih aplikacij oz. sistemskih storitev
  - Vsako sporočilo je dostavljeno v obliki objekta tipa `Intent`

Aktiviranje komponent

- Komponente Activity, Service in Broadcast receiver aktiviramo s pomočjo asinhronega sporočila Intent (objekt Intent)
  - Sporočilo je dostavljeno sistemu, ki ima ustrezne pravice za aktiviranje komponente
- Aktiviranje komponente Activity
  - Kreiramo Intent, kjer navedemo tip akcije in podatke, ki se pošiljajo
    - `startActivity()`, `startActivityForResult()` -> Intent kot parameter
- Aktiviranje komponente Service
  - Velja isto kot za Activity
    - `startService()`, `bindService()`-> Intent kot parameter
- Aktiviranje komponente Broadcast receiver
  - Intent vsebuje sporočilo, ki se pošilja drugi komponenti
  - Metoda za aktivacijo:
    - `sendBroadcast()` -> Intent kot parameter
- Aktiviranje komponente Content provider
  - Z uporabo objekta ContentResolver (metoda `query()`)

Manifest datoteka

- AndroidManifest.xml
  - Primarna naloga manifest datoteke je informiranje sistema o komponentah aplikacije

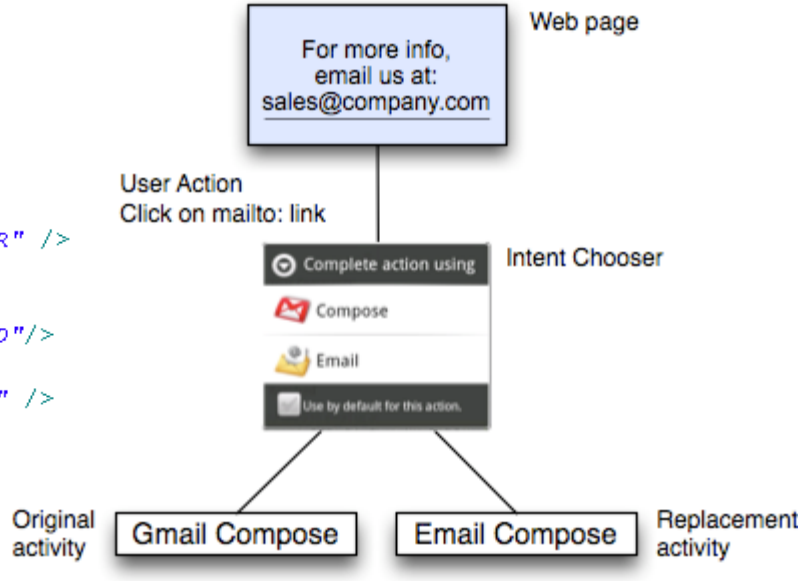
```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

- Poleg tega Ima manifest datoteka številne druge vloge, kot so npr.
  - Identificiranje uporabniških pravic, ki jih aplikacija zahteva (npr. Internet dostop)
  - Določanje API Level parametra
  - Določanje strojnih in programskih zmožnosti (npr. kamera, BT, ...)
  - Knjižnice, ki jih aplikacija uporablja poleg Framework API-ja
- Manifest je prstni odtis aplikacije
  - apktool je super orodje za učenje na izkušnjah drugih razvijalcev ☺

Deklariranje zmožnosti komponent

- S pomočjo namer (Intent) lahko aktiviramo komponente
  - Glavna moč mehanizma Intent je predvsem v konceptu definiranja akcije -> sistem nato sam ugotovi kateri app je najbolj primeren
- Sistem določi ustreznost komponente za izvedbo akcije glede na intent filtre v manifest datoteki
  - `<intent-filter>`
  - http://developer.android.com/reference/android/content/Intent.html

```
<activity android:name="QuickTag"
          android:label="@string/app_name"
          android:clearTaskOnLaunch="true"
          android:launchMode="singleTask">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <data android:scheme="http"/>
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```



Viri znotraj aplikacije

- Vsaka aplikacija poleg kode vsebuje dodatne vire (resources)
  - (multi)medijske datoteke
  - Datoteke XML za vizualizacijo
- Z uporabo virov lahko spremenimo videz aplikacije brez posega v kodo
- Za vsak vir SDK definira unikaten ID,
  - Uporabimo za sklicevanje na vir iz kode ali drugih datotek XML
  - Primer: logo.png -> shranimo v res/drawable/ -> dostopamo iz kode z identifikatorjem `R.drawable.logo`
- Lokalizacija
  - Viri omogočajo implementacijo lokalizacije -> definiramo različne mape za različne jezike (npr. res/values-fr), layoute, grafiko itd.
- Alternativni viri
  - Vedno lahko poleg osnovnega vira definiramo alternativni vir, ki ga postavimo v ustrezno poimenovano mapo (npr. različni layout-i za vertikalno/horizontalno postavitev)

LTFE.org
Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

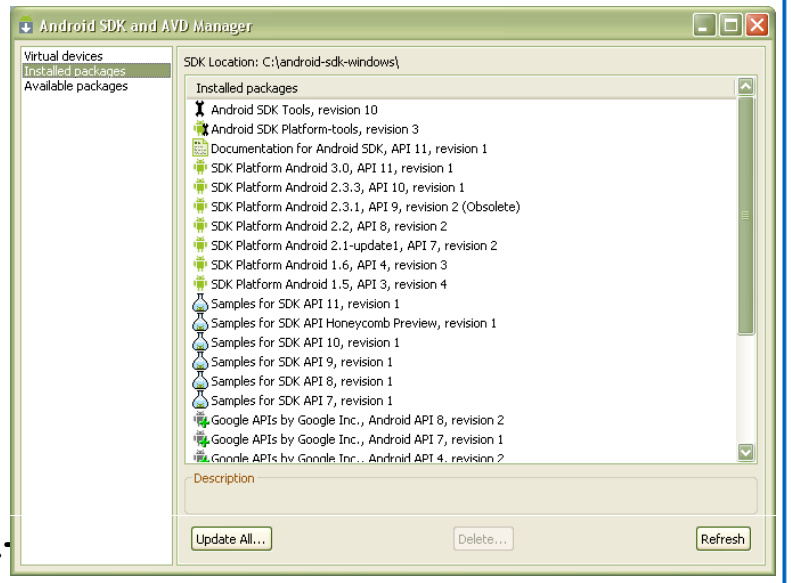
Razvojno okolje

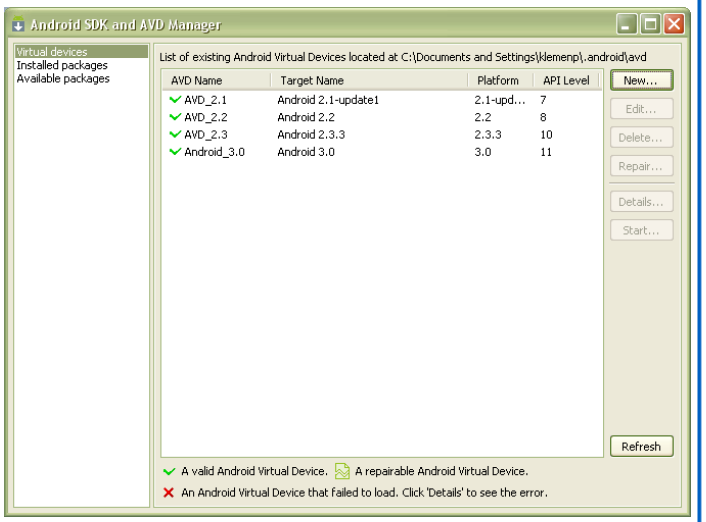
Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Postavitev razvojnega okolja

- 1. korak: Namestimo Eclipse razvojno okolje
  - Eclipse 3.5 ali novejši
  - JDK 5 & JDK 6
- 2. korak: Namestimo Android SDK Starter Package
  - Vsebuje jedrna SDK orodja
  - SDK se namesti v mapo `android-sdk-<machine-platform>`
- 3. korak: Namestimo ADT (Android Development Tools) vtičnik za Eclipse
  - Omogoča učinkovito delo z Android projekti znotraj Eclipse
  - https://dl-ssl.google.com/android/eclipse
- 4. korak: SDK nadgradimo z dodatnimi komponentami
  - Orodje Android SDK and AVD Manager





Android Development Tools

- Kreiranje novega projekta
- Upravljanje z zunanjimi viri
  - Različni namenski urejevalniki
- Dostop do emulatorjev
- Razhroščevanje
- Dostop do dnevniških datotek
- Emulacija zunanjih dogodkov (telefonija, lokacija, ...)
- Dostop do naprave
- Kreiranje posnetkov zaslona naprave
- Kreiranje testnega projekta (uporaba JUnit)
- Dostop do podatkovne baze SQLite
- ...

LTFE.org
Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

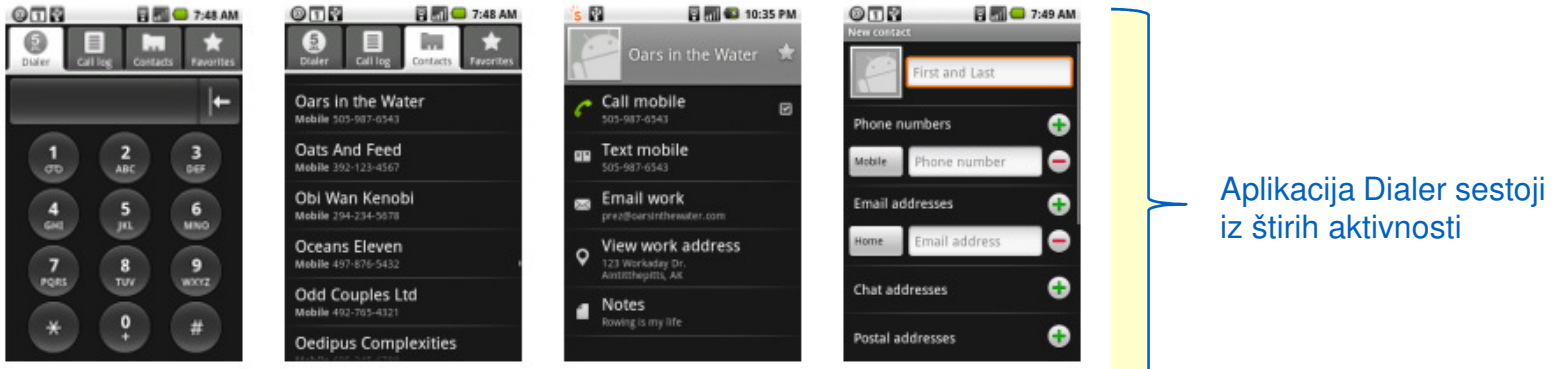
Glavne komponente Android aplikacij

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Aktivnost (Activity)

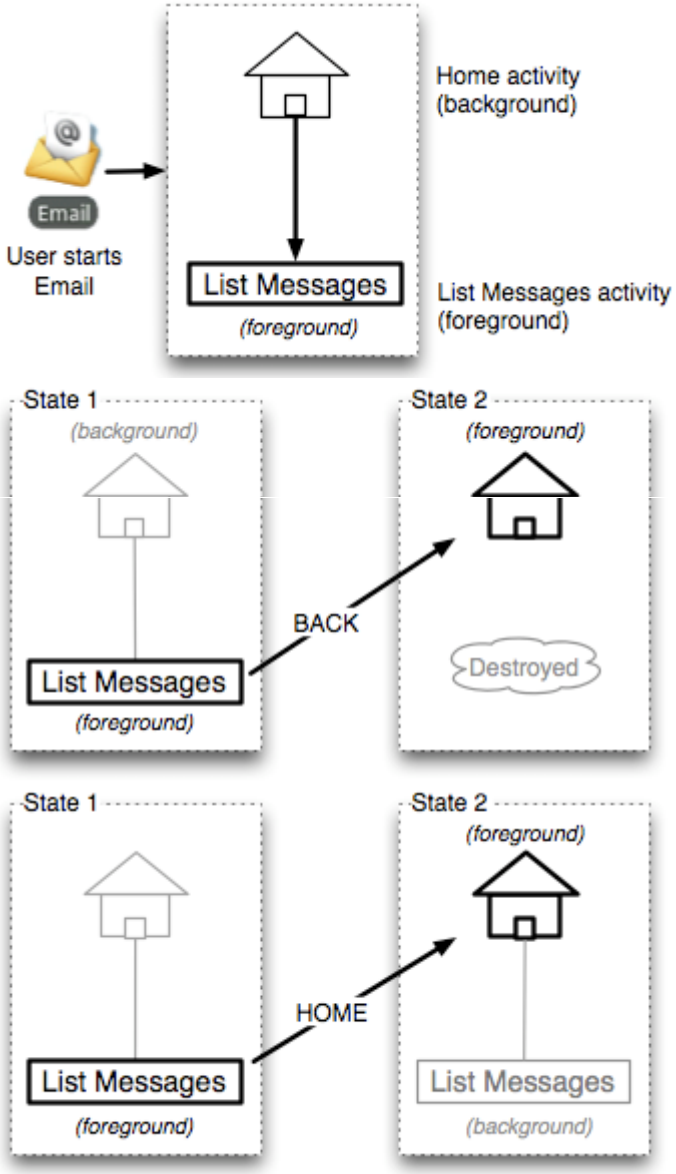
- **Aplikacija (Application)**
  - **Običajno sestoji iz več aktivnosti, ki so ohlapno spojene**
- **Aktivnost (Activity)**
  - **Glavna komponenta aplikacije**
  - **Aktivnost ima svoj izgled**
  - **Namenjena je izvedbi ene, zaključene naloge**
  - **Vsaka aktivnost ima svoj življenjski cikel**
  - **Aktivnost je neodvisna komponenta**
  - **Vsaka aplikacija, ki predstavlja karkoli na zaslonu, ima vsaj eno aktivnost**

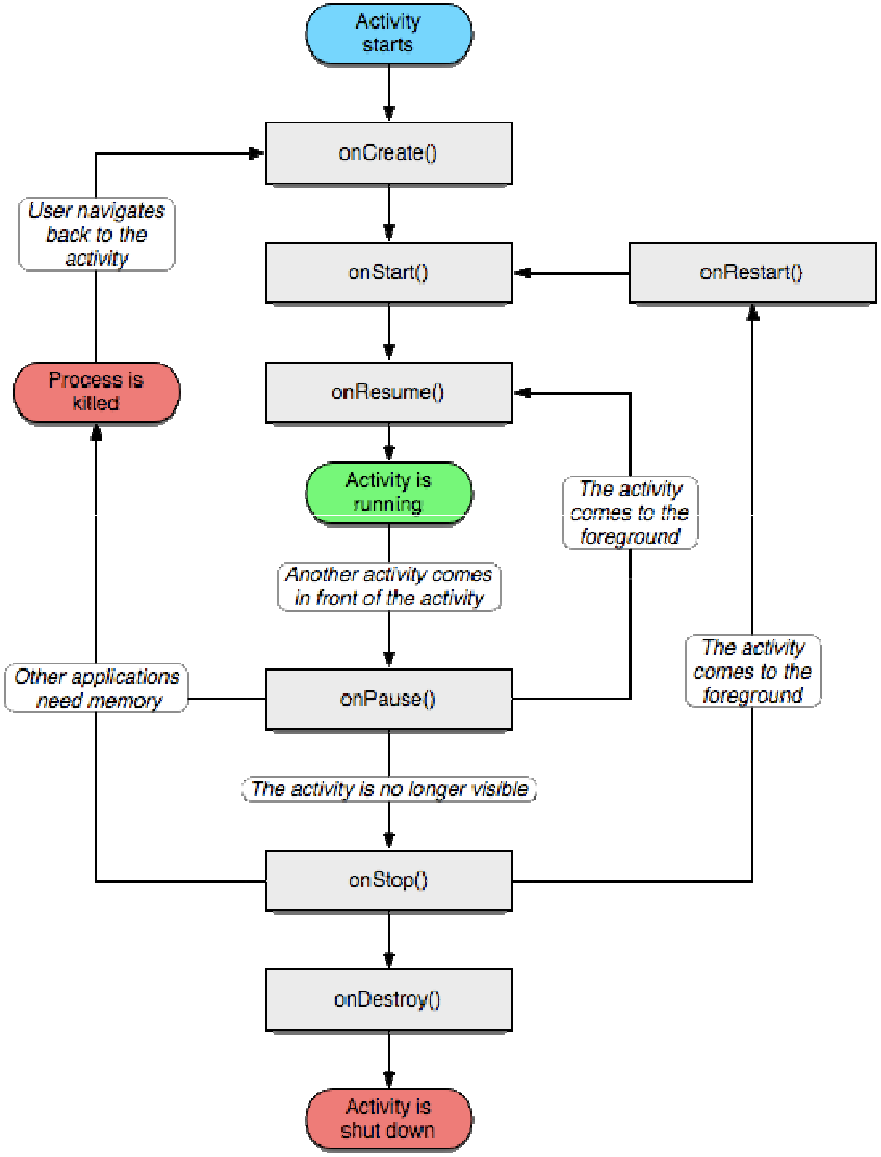


Povezovanje aktivnosti

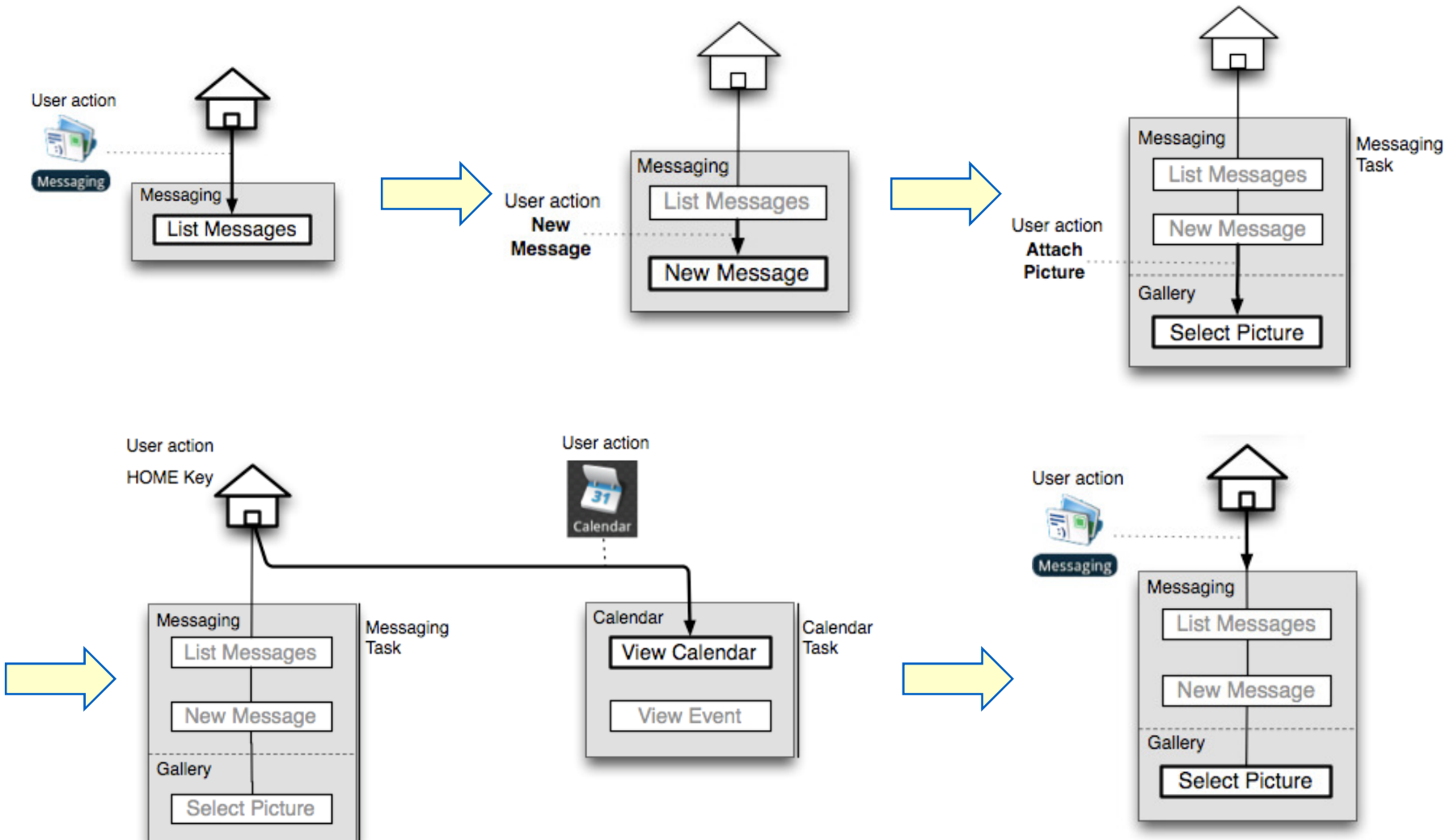
- Sklad aktivnosti (Activity Stack)
  - Android vzdržuje linearno navigacijsko sled prehodov med aktivnostmi
  - Takšno sled imenujemo sklad aktivnosti (Activity Stack)
  - Pomikanje nazaj po skladu je možno s tipko BACK (najdlje do Home)
  - Aktivnost je edina komponenta, ki se dodaja v sklad
- Naloga (Task)
  - Sklop več aktivnosti (lahko iz različnih aplikacij), ki skupaj zaključujejo neko nalogo
  - Primer: Ogled YouTube videa in pošiljanje linka prijatelju z uporabo elektronske pošte
  - V primeru prekinitve izvajanja naloge (npr. zaradi telefonskega klica), se uporabnik lahko kasneje vrne nazaj tja, kjer je bil med izvajanjem naloge prekinjen

Življenjski cikel aktivnosti





Preklaplanje med nalogami (taski)



Storitev (Service)

- Storitev ni ločen proces
  - Teče v istem procesu, kot celotna aplikacija
- Storitev ni nit
  - Storitev ni način na katerega bi izvajali opravilo v ločeni niti
- Storitev je način, kako aplikacija izvaja določeno opravilo v ozadju
  - To sovpada s klicem metode `Context.startService()`
  - N klicev je ena instanca storitve!
- Storitev je tudi način, kako aplikacija izpostavi del svojih funkcionalnosti drugim aplikacijam
  - To sovpada s klicem metode `Context.bindService()`
- V praksi sistem storitev ubije samo v primeru, ko je izredno malo spomina na razpolago
  - Storitev ima višjo prioriteto kot aktivnosti, ki živijo v ozadju in manjšo ali enako kot tista, ki je v ospredju

Sprejemnik sporočil (Broadcast receiver)

- Mehanizem za sprejem sporočil
  - S strani notranjih ali zunanjih komponent
- Implementiran znotraj razreda `BroadcastReceiver`
- Sporočila delimo na dve vrsti
  - Normal broadcasts
    - poslano z uporabo metode `Context.sendBroadcast`
    - Sporočilo dostavljeno vsem sprejemnikom (lahko tudi hkrati)
  - Ordered broadcasts
    - Poslano z uporabo metode `Context.sendOrderedBroadcast`
    - Sporočilo poslano zaporedno posameznim sprejemnikom
    - Posamezen sprejemnik lahko posreduje naslednjim svoje rezultate ali celo prekine razpošiljanje sporočila
    - Vrstni red sprejemnikov je določen s prioriteto (`android:priority`)
- Sporočila se prenašajo v obliki namer (Intent)

Ponudnik vsebine (Content provider)

- Ponudniki vsebin shranjujejo in pridobivajo podatke ter omogočajo dostop do njih drugim aplikacijam
  - To je edini način deljenja podatkov med aplikacije
- Android prinaša številne ponudnike vsebin za pogoste podatkovne tipe
  - Audio, video, slike, osebni kontaktni podatki itd.
- Razvijalec lahko kreira svojega ponudnika vsebin, ali pa svoje podatke streže preko obstoječega (če ustreza podatkovnemu tipu)
- Kako ponudnik vsebin dejansko shranjuje podatke je stvar implementacije
  - Vsi ponudniki vsebin implementirajo isti vmesnik
  - Odjemalci do vmesnika dostopajo posredno z uporabo razreda `ContentResolver`
  - `ContentResolver` omogoča dostop do metod za interakcijo s ponudniki vsebin

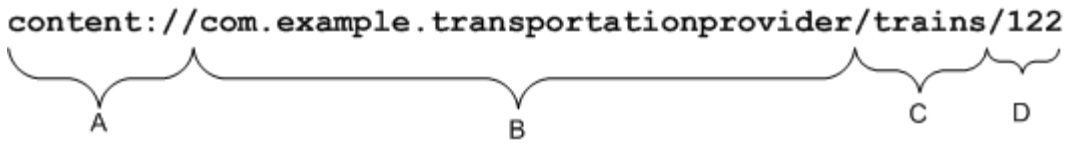
Delovanje ponudnika vsebine

- Ko se poizvedovanje za podatki začne, sistem identificira ustreznega ponudnika vsebine
- Ponudniki vsebin predstavljajo podatke v obliki tabel
  - Po principu relacijskih podatkovnih baz

_ID	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
13	(425) 555 6677	425 555 6677	Kirkland office	Bully Pulpit	TYPE_WORK
44	(212) 555-1234	212 555 1234	NY apartment	Alan Vain	TYPE_HOME
45	(212) 555-6657	212 555 6657	Downtown office	Alan Vain	TYPE_MOBILE
53	201.555.4433	201 555 4433	Love Nest	Rex Cars	TYPE_HOME

  - Poizvedba vrača objekt tipa `Cursor` s katerim se premikamo po tabeli
- Vsaka tabela je naslovljena z ustreznim URI-jem (content://)
- Za pridobivanje podatkov s strani ponudnika storitve je potrebno
  - Poznati URI ustrezne tabele (ponudnika storitve)
  - Poznati imena podatkovnih polj znotraj tabele
  - Poznavanje podatkovnih tipov podatkov znotraj polj

Naslavljanje podatkovnih tabel (URI)



- A - shema, ki sporoča, da so podatki kontrolirani s strani ponudnika vsebine
- B – unikatno definira ponudnika vsebine
  - Polno ime razreda ponudnika vsebine (pisano z malo)
  - Ime je deklarirano znotraj manifest datoteke – `android:authorities`

```
<provider android:name=".TransportationProvider"
          android:authorities="com.example.transportationprovider"
          . . .  >
```

- C – pot znotraj URI-ja definira vrsto podatkov, ki jih zahtevamo
  - Na nivoju tabele
- D – ID vrstice

Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

App Inventor

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

O App Inventorju

- http://appinventor.googlelabs.com
- Omogoča enostaven razvoj preprostih Android aplikacij
  - Z omejenim naborom funkcionalnosti
  - Možno testiranje na emulatorju ali mobilnem terminalu
  - Ni možno izvoziti izvorne kode
- Razvoj ne zahteva znanja programiranja
  - Drag&Drop
- Razvojno okolje sestavljajo tri glavne komponente
  - Designer (določimo izgled)
  - Blocks Editor (določimo logiko delovanja)
  - Emulator (testiranje)
- Trenutno ni možno sestavljati kompleksnih aplikacij z več komponentami

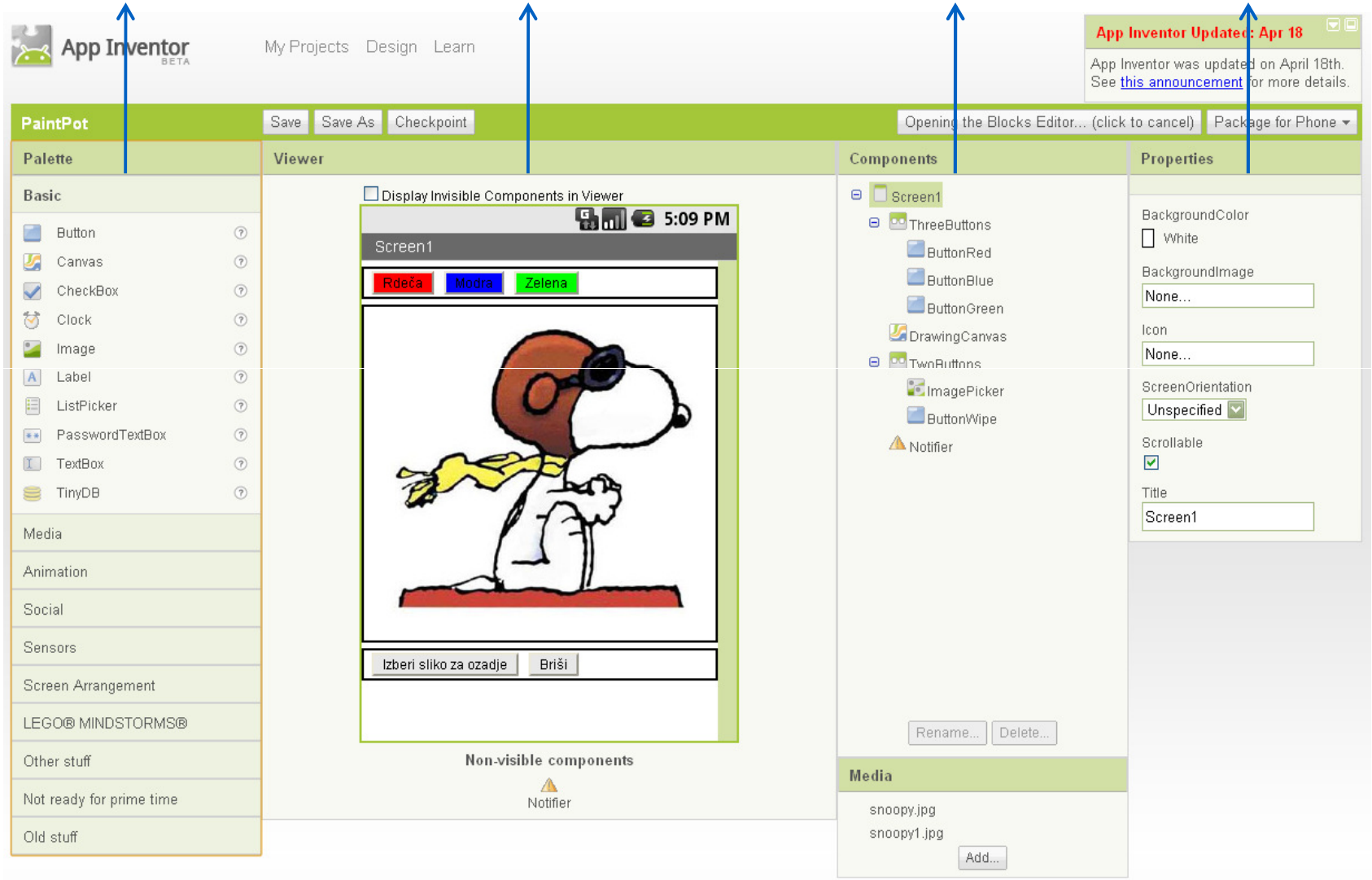
Designer

Seznam gradnikov

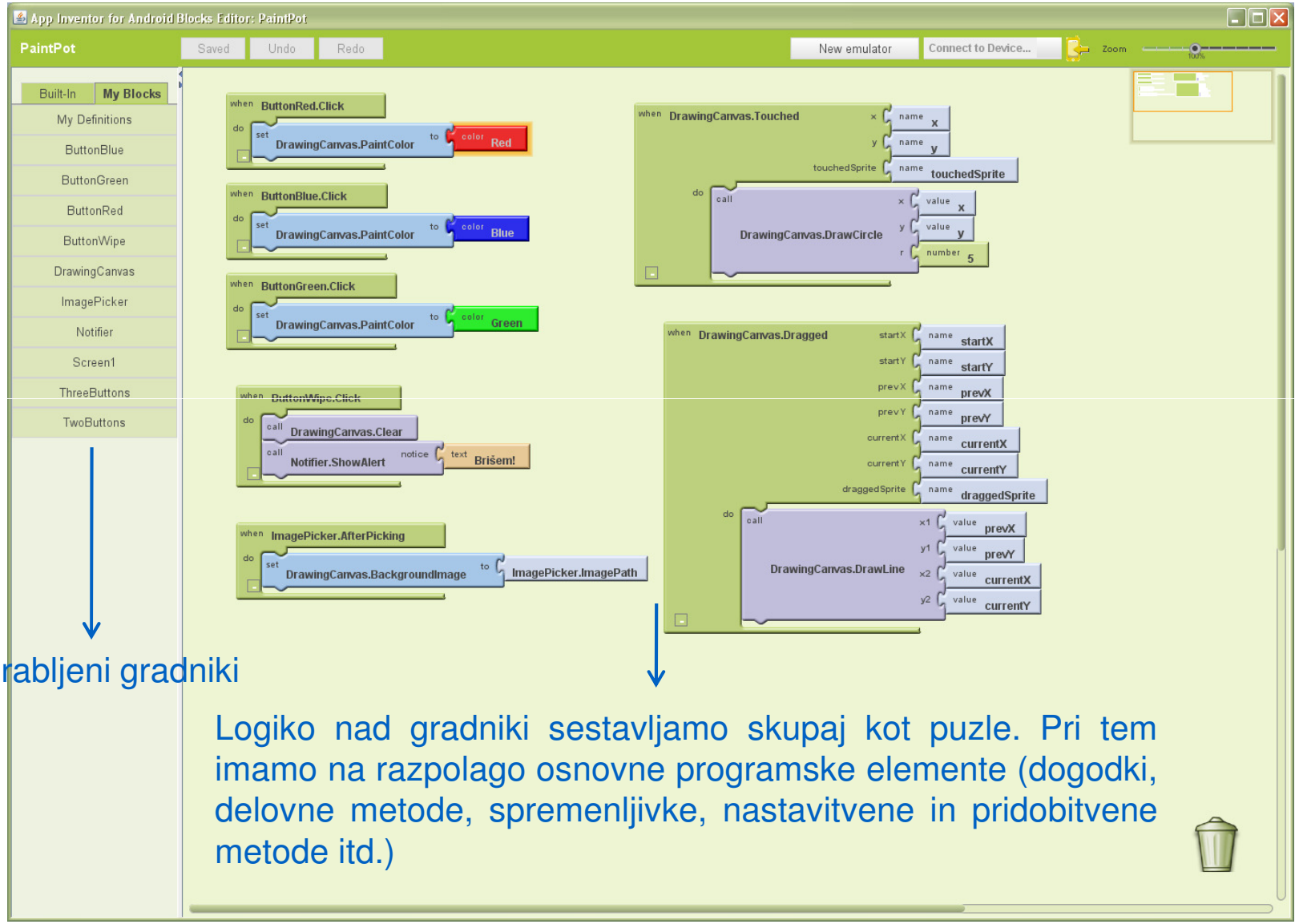
Delovna površina

Uporabljeni gradniki

Lastnosti gradnika



Blocks Editor



Uporabljeni gradniki

Logiko nad gradniki sestavljamo skupaj kot puzle. Pri tem imamo na razpolago osnovne programske elemente (dogodki, delovne metode, spremenljivke, nastavitvene in pridobitvene metode itd.)

LTFE.org
Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

Objava aplikacije na Android Marketu

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Android Market

- Online trgovina za Android aplikacije
  - Na večini Android naprav je Market aplikacija že vključena
  - Do konca 2010 je bilo v Marketu okrog 200K različnih aplikacij
  - https://market.android.com/
- V Sloveniji sedaj možno dostopati do plačljivih aplikacij

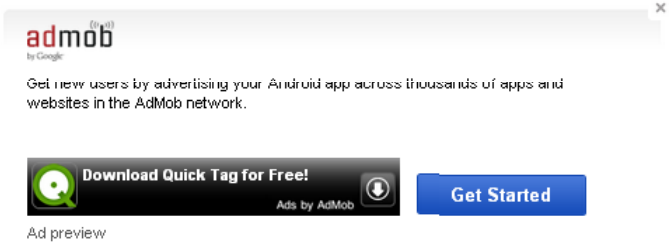
- Obstajajo tudi alternativne trgovine (Google to dopušča)
  - SlideMe, AndAppStore, Handango, AndroidGear, Phoload, Mobihand, AppsLib, ...

Objavljanje lastne aplikacije

- Objavljanje je možno na https://market.android.com/publish/Home
- Treba vplačati enkratno "članarino" - $25
- Oris postopka
  - Potrebno je naložiti .apk datoteko
  - Dodati je potrebno vsaj 2 posnetka izgleda aplikacije
  - Potrebno je naložiti ikono, ki predstavlja aplikacijo
  - Določimo ime, opis
  - Izberemo ustrezno kategorijo, kamor aplikacija spada
  - Vpišemo kontaktne podatke
- Zelo pomembno je dodeljevanje ustrezne različice aplikaciji
  - Znotraj manifest datoteke
  - `android:versionCode` - integer, ki označuje različico aplikacije glede na predhodne –> povečamo z vsako različico
  - `android:versionCode` - string, ki označuje različico aplikacije, kot jo vidi uporabnik (predlagan format - <major>.<minor>.<point>)
  - Pomembno za delovanje mehanizma nadgradnje

Nadzor nad dogajanjem

- Na voljo je obsežna statistika, ki prikazuje podatke o širjenju namestitev aplikacije
- Statistika se obnavlja dnevno
- Aplikacijo možno tudi reklamirati
  - Google AdMob



- Uporabniki nudijo feedback
  - Ocene
  - Komentarji