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

Napovedi prihoda

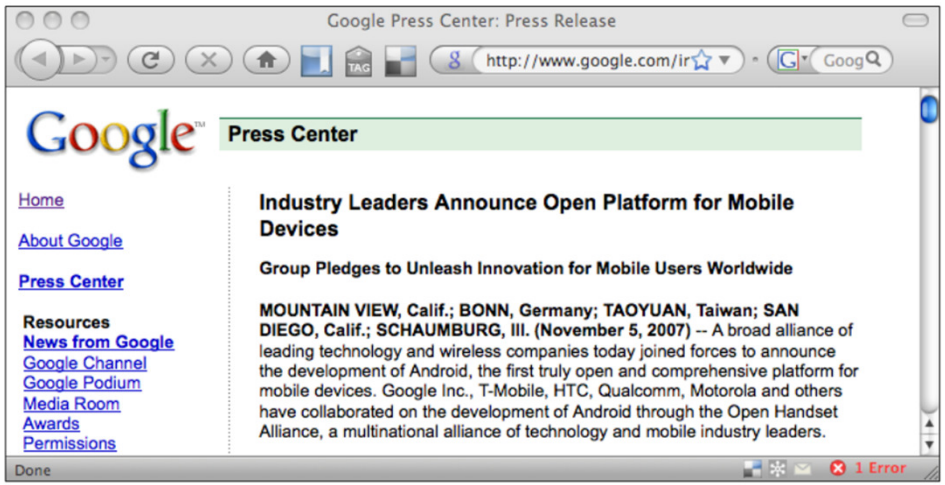
August 2005

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

- **Galaxy Nexus**
  - Dual-core 1200 MHz
  - 1 GB RAM
  - 1750 mAh battery
  - 4G (HSPA+)
  - Touchscreen AMOLED 720p x 1280p
  - 5 megapixel camera
  - GPS, compass, accelerometer, gyroscope
  - NFC

- **Wearable Android (WIMM)**
  - Bi-modal display
  - Sensors (accelerometer, magnetometer)
  - 32x36x12.5mm
  - 22g
  - Wi-fi
  - Bluetooth

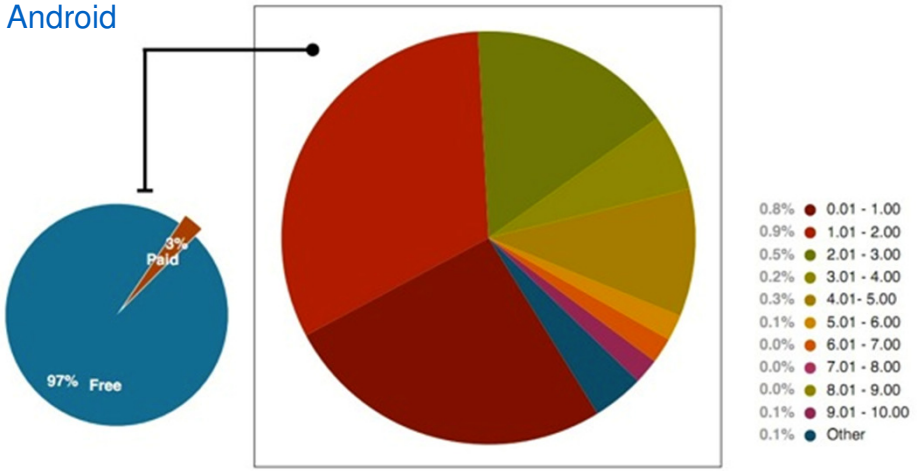
Splošno o Androidu

- Nastal 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

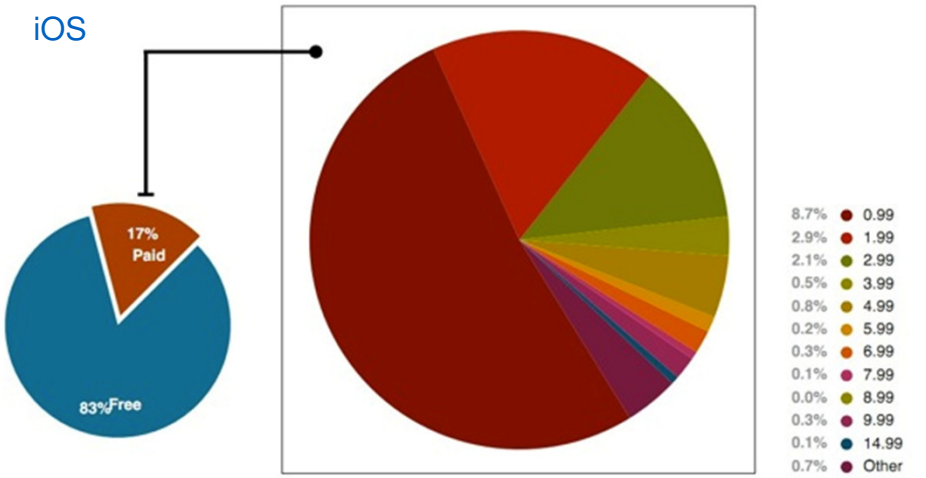
97% naloženih aplikacij je zastonj!

- **Vir: Chomp – april 2011 (USA)**

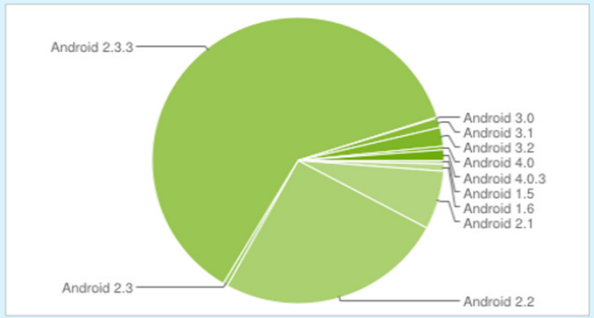
Android



iOS

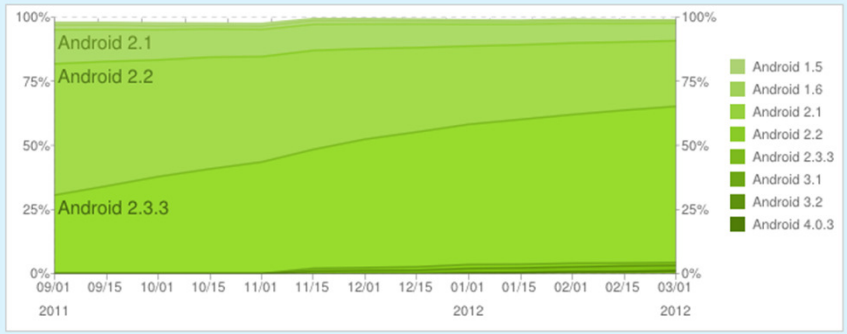


Različice Androida - dashboard



Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.4%
Android 1.6	Donut	4	0.8%
Android 2.1	Eclair	7	6.6%
Android 2.2	Froyo	8	25.3%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.3.3 - Android 2.3.7		10	61.5%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.1%
Android 3.2		13	2.1%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.4%
Android 4.0.3		15	1.2%

Data collected during a 14-day period ending on March 5, 2012



Last historical dataset collected during a 14-day period ending on March 5, 2012

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	VERSION_CODE
Android 4.0.3	15	ICE_CREAM_SANDWICH_MR1
Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

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 ☺

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)

Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

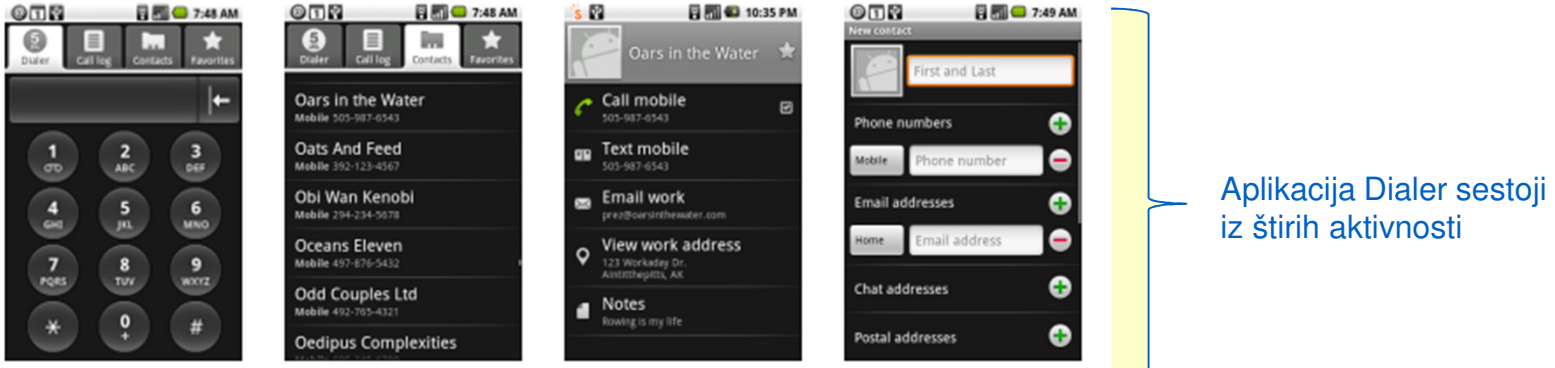
Glavne komponente Android aplikacij

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Aktivnost (Activity)

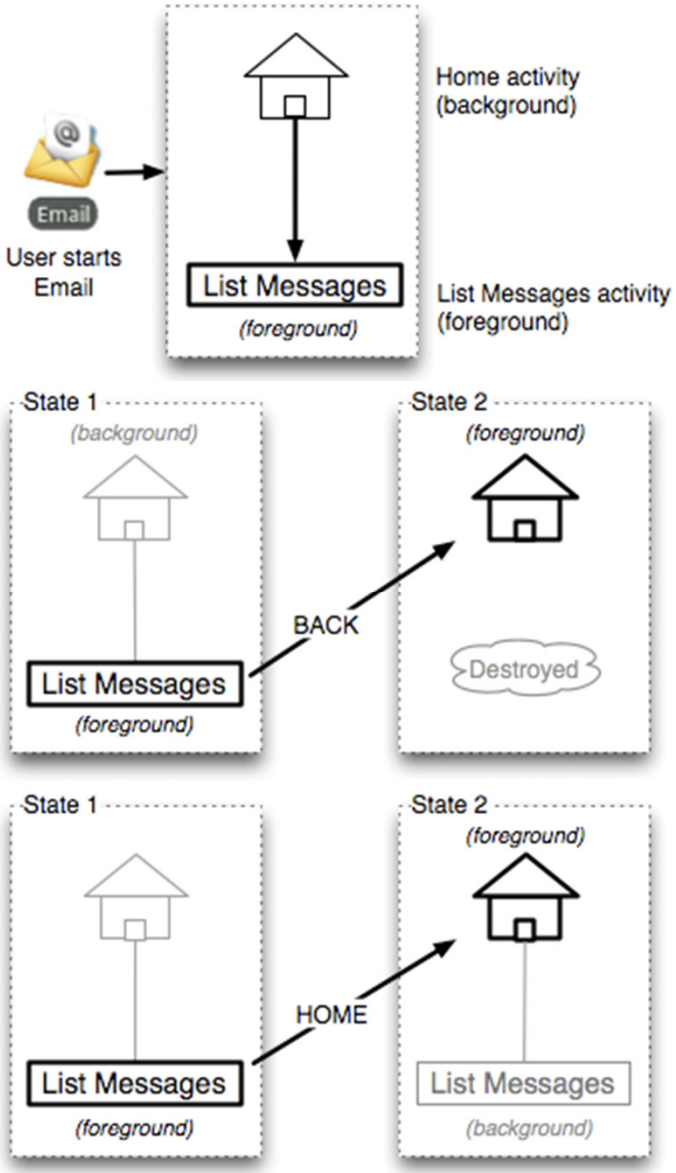
- **Aplikacija (Application)**
  - Običajno sestoji iz več aktivnosti, ki so ohlapno spojene
- **Aktivnost (Activity)**
  - Glavna komponenta aplikacije
  - Aktivnost ima svoj izgled
  - Namenjena je izvedbi ene, zaključene naloge
  - Vsaka aktivnost ima svoj življenjski cikel
  - Aktivnost je neodvisna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

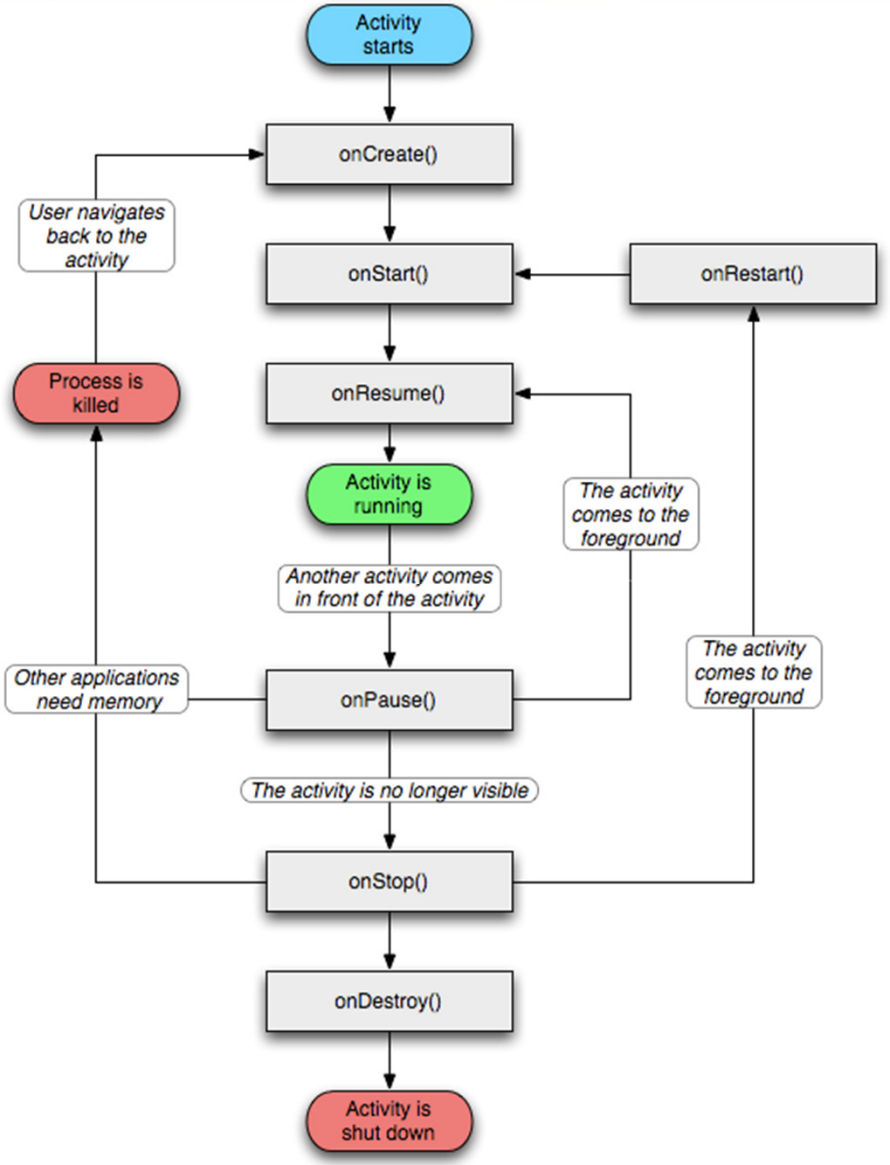


Povezovanje aktivnosti

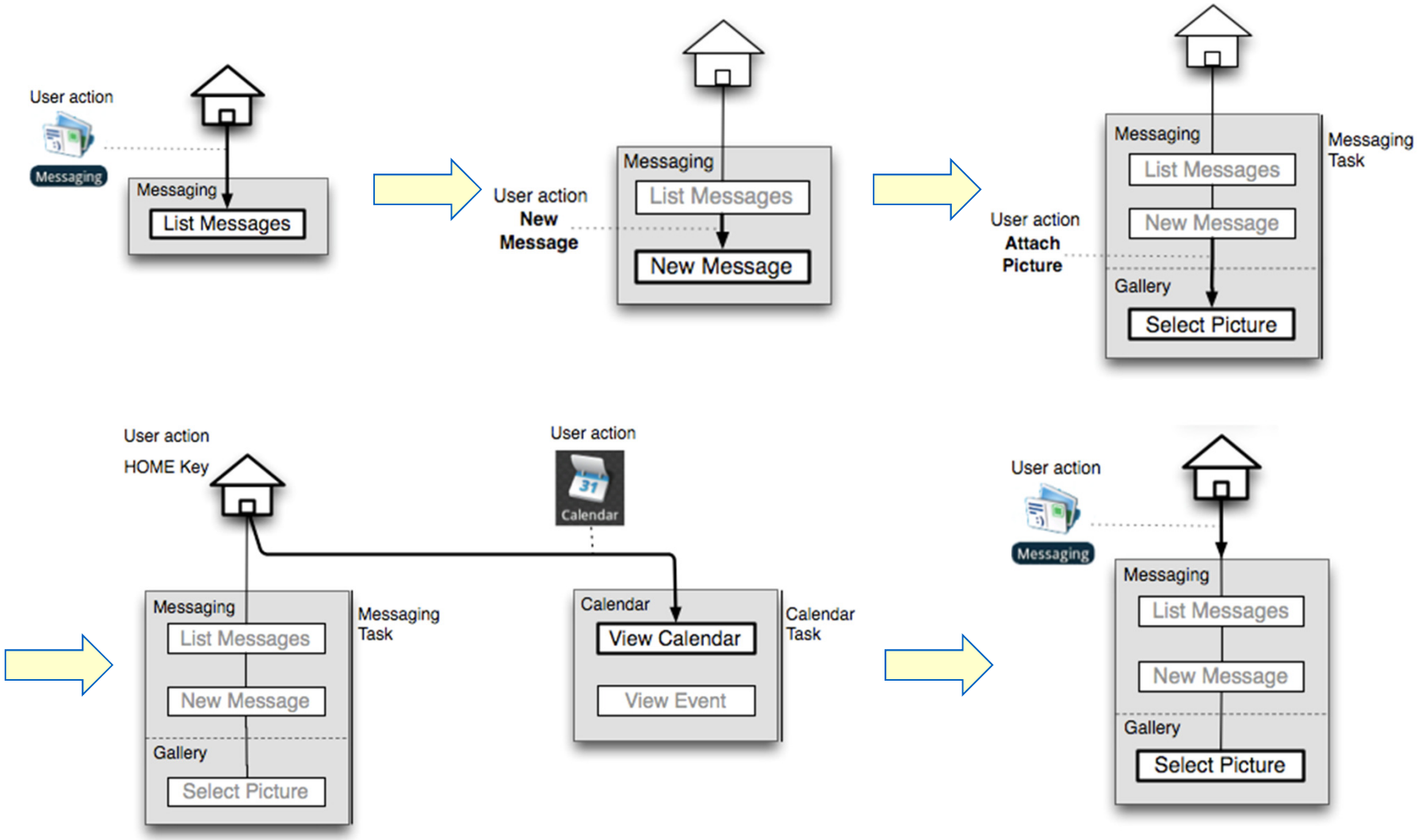
- Sklad aktivnosti (Activity Stack)
  - Android vzdržuje linearno navigacijsko sled prehodov med aktivnostmi
  - Takšno sled imenujemo sklad aktivnosti (Activity Stack)
  - Pomikanje nazaj po skladu je možno s tipko BACK (najdlje do Home)
  - Aktivnost je edina komponenta, ki se dodaja v sklad
- Naloga (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)

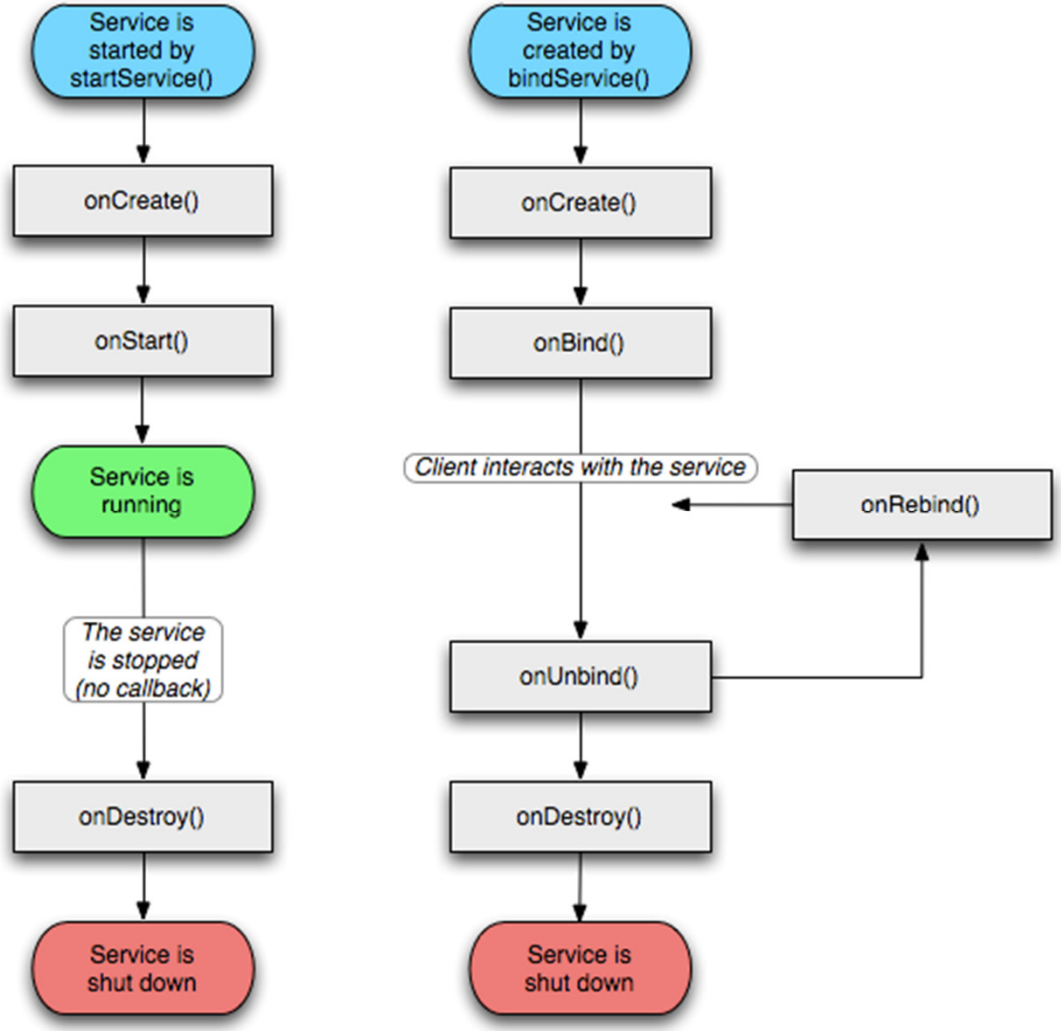


Storitev (Service)

- 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

Življenjski cikel storitve

- Dva cikla
  - Zagnane storitve
  - Pripete storitve



Komunikacija s storitvijo

- **Komunikacija s storitvami znotraj lastne aplikacije**
  - Razširitev `Binder` razreda –> instanco vrne metoda `onBind()` –> omogoča dostop do public metod storitve
  - To je priporočen način komunikacije s storitvami, ki opravljajo delo znotraj lastne aplikacije
- **Komunikacija s storitvami, ki se izvajajo v drugih procesih**
  - Uporaba mehanizma `Messenger`
  - Odjemalec in storitev si izmenjavata objekt tipa `Message`
  - Za sprejem sporočil se uporablja razred `Handler`, ki obravnava različne tipe sporočil
  - Dostava sporočil poteka znotraj ene niti (FIFO), možna pa je tudi implementacija večnitnega sprejema

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)

Življenjski cikel sprejemnika obvestil

- Sprejemnik obvestil živi samo za časa sprejema in obdelave obvestila!
  - To je dokler se ne zaključi delo metode `onReceive`
- Asinhrono delovanje je preprečeno
  - Vse kar naredimo na podlagi obvestila, naredimo takoj
- Iz sprejemnika ne smemo prikazovati grafičnih elementov ali se povezovati na storitve
- Aplikacije, ki imajo samo komponento sprejemnika obvestil ne živijo dolgo
  - Sistem ima namreč možnost proces ubiti vsakič, ko se zaključi izvajanje metode `onReceive`
  - Za takšne aplikacije se priporoča skupno delovanje sprejemnika in storitve

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

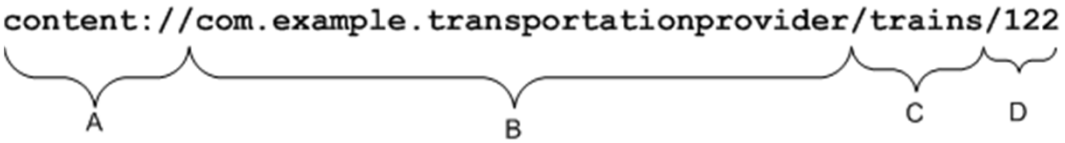
Delovanje ponudnika vsebine

- 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

Kreiranje ponudnika vsebine

- Najprej je potrebno nastaviti sistem, da shranjuje podatke
  - SQLLite podatkovna baza
  - Shranjevanje v datoteke
- Razširimo razred ContentProvider da zagotovimo dostop do podatkov
  - Implementiramo šest metod
    - `Query()`
    - `Insert()`
    - `Update()`
    - `Delete()`
    - `getType()`
    - `onCreate()`
- Deklariramo ponudnika vsebine znotraj manifest datoteke

Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

Viri znotraj aplikacije

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Organizacija virov

- Viri so organizirani znotraj projektne mape `res/`
  - V ustreznih podmapah
- Podmape:
  - `anim/` - XML datoteke za animacijo
  - `color/` - XML datoteke, kjer so opisane barve ob različnih stanjih
  - `drawable/` - bitmap datoteke (png, jpg, gif)
  - `layout/` - XML datoteke, ki opisujejo postavitve UI-ja
  - `menu/` - XML datoteke, ki opisujejo aplikacijske menije
  - `raw/` - poljubne datoteke shanjene v raw formatu
  - `values/` - XML datoteke, ki opisujejo preproste vrednosti (npr. nizi, barve, števila)
  - `xml/` - poljubne XML datoteke

Osnoven izgled organizacije virov znotraj projekta

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            icon.png
        layout/
            main.xml
            info.xml
        values/
            strings.xml
```

/color

- **Definiramo kako posamezni objekti tipa `View` spreminjajo svojo barvo, glede na stanje**

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:color="hex_color"
        android:state_pressed=["true" | "false"]
        android:state_focused=["true" | "false"]
        android:state_selected=["true" | "false"]
        android:state_checkable=["true" | "false"]
        android:state_checked=["true" | "false"]
        android:state_enabled=["true" | "false"]
        android:state_window_focused=["true" | "false"] />
</selector>
```

Sintaksa

XML file saved at `res/color/button_text.xml`:

Primer na gumbu

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:color="#ffff0000"/> <!-- pressed -->
    <item android:state_focused="true"
          android:color="#ff0000ff"/> <!-- focused -->
    <item android:color="#ff000000"/> <!-- default -->
</selector>
```

This layout XML will apply the color list to a View:

```
<Button
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/button_text"
    android:textColor="@color/button_text" />
```

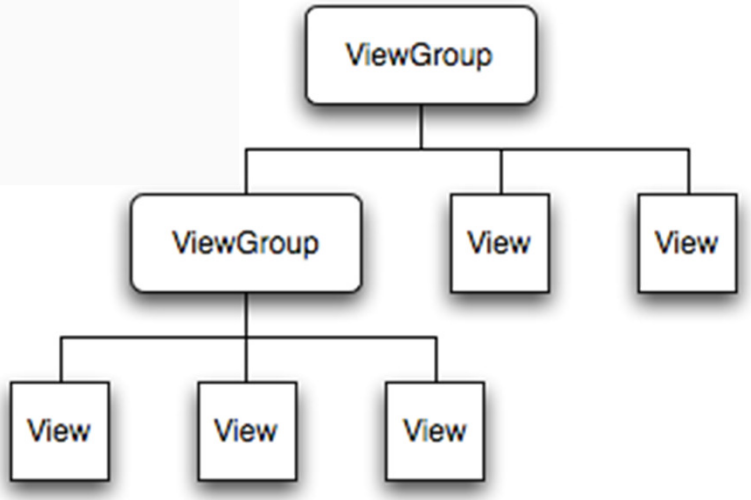
/layout

- **Layout vir definira arhitekturo UI znotraj aktivnosti**
  - **Glavni komponenti sta `View` in `ViewGroup`**

```
<?xml version="1.0" encoding="utf-8"?>
<ViewGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@[+][package:]id/resource_name"
    android:layout_height=["dimension" | "fill_parent" | "wrap_content"]
    android:layout_width=["dimension" | "fill_parent" | "wrap_content"]
    [ViewGroup-specific attributes] >
    <View
        android:id="@[+][package:]id/resource_name"
        android:layout_height=["dimension" | "fill_parent" | "wrap_content"]
        android:layout_width=["dimension" | "fill_parent" | "wrap_content"]
        [View-specific attributes] >
        <requestFocus/>
    </View>
    <ViewGroup >
        <View />
    </ViewGroup>
    <include layout="@layout/layout_resource"/>
</ViewGroup>
```

Sintaksa

ViewGroup je zbiralnik za posamezne objekte tipa View. UI sestavljamo po principu gnezdenja

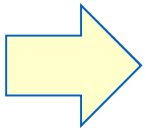


Vrste objektov `ViewGroup`

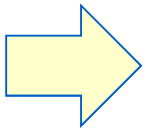
- `LinearLayout`
  - Organizacija objektov `View` v eni vrstici ali stolpcu
- `AbsoluteLayout`
  - Organizacija objektov `View` glede na koordinate (X,Y)
- `TableLayout`
  - Organizacija objektov `View` v vrstice in stolpce
- `RelativeLayout`
  - Organizacija objektov `View` relativno glede na sosednjega
- `FrameLayout`
  - Definira t.i. placeholder za prikaz enega objekta `View`;
  - Če je objektov `View` več, se prekrivajo
  - Velikost je določena z največjim objektom
- `ScrollView`
  - Poseben tip `FrameLayout`-a, ki omogoča implementacijo scroll-erja
  - Gosti lahko samo en objekt (običajno kar drug `ViewGroup`)

LinearLayout (primer I)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:layout_width="105px"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Button"
        />
</LinearLayout>
```

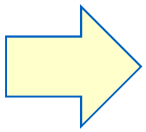


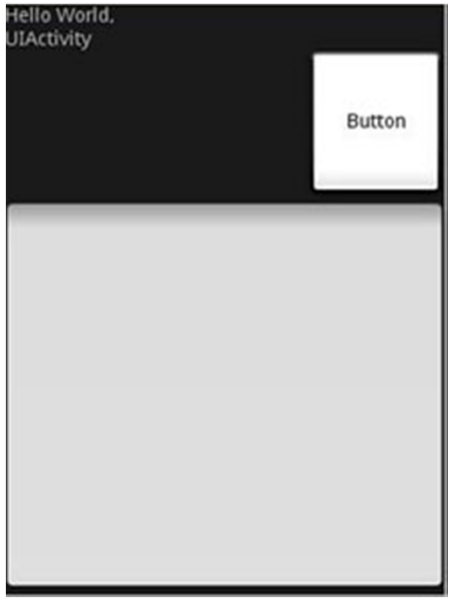

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
```



LinearLayout (primer II)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
    >
    <TextView
        android:layout_width="105px"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_gravity="right"
        android:layout_weight="0.2"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_weight="0.8"
        />
</LinearLayout>
```

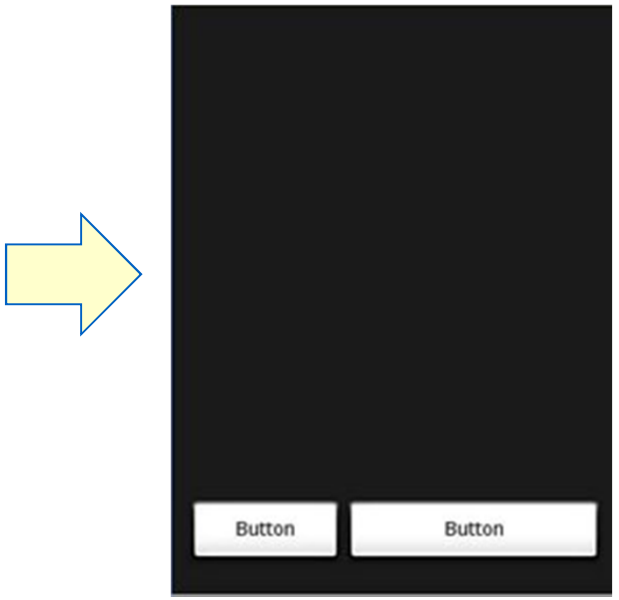




AbsoluteLayout (primer)

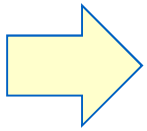
- **Z Android 1.5 (API Level 3) je uporaba odsvetovana – (ne)podpora različnim dimenzijam zaslona!**
  - **Android dokumentacija nasplošno odsvetuje uporabo enote *px (pixels)***

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <Button
        android:layout_width="188px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="126px"
        android:layout_y="361px"
        />
    <Button
        android:layout_width="113px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="12px"
        android:layout_y="361px"
        />
</AbsoluteLayout>
```



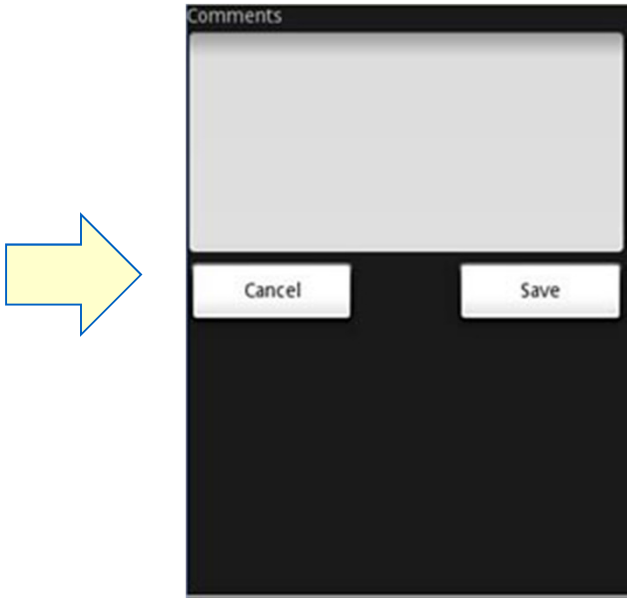
TableLayout (primer)

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/and
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:background="#000044">
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120px"
            />
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
            />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
            />
    </TableRow>
    <TableRow>
        <TextView />
        <CheckBox android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
            />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In" />
    </TableRow>
</TableLayout>
```



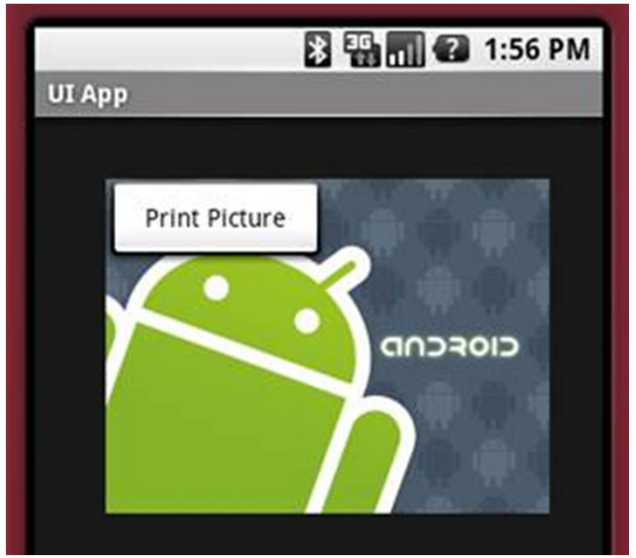
RelativeLayout (primer)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <TextView
        android:id="@+id/lblComments"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Save"
        android:layout_below="@+id/txtComments"
        android:layout_alignRight="@+id/txtComments"
        />
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignLeft="@+id/txtComments"
        />
</RelativeLayout>
```



FrameLayout (primer)

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget68"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="40px"
        android:layout_y="35px"
        >
        <ImageView
            android:src = "@drawable/androidlogo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
        <Button
            android:layout_width="124px"
            android:layout_height="wrap_content"
            android:text="Print Picture"
            />
    </FrameLayout>
</AbsoluteLayout>
```



ScrollView (primer)

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    android:id="@+id/widget54"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
    <LinearLayout
        android:layout_width="310px"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1"
            />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2"
            />
        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 3"
            />
        <EditText
            android:id="@+id/txt"
            android:layout_width="fill_parent"
            android:layout_height="300px"
            />
        <Button
            android:id="@+id/button4"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 4"
            />
        <Button
            android:id="@+id/button5"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 5"
            />
    </LinearLayout>
</ScrollView>
```

/values

- XML datoteke, ki definirajo preproste vrednosti, kot so nizi, števila, barve itd.
- Najbolj pogosto videvamo datoteki strings.xml in colors.xml
  - Definirani so nizi, ki se izpisujejo znotraj aplikacije in barve posameznih gradnikov

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <color name="opaque_red">#f00</color>
   <color name="translucent_red">#80ff0000</color>
</resources>
```

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="@color/translucent_red"
    android:text="Hello"/>
```

Primer uporabe colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello!</string>
</resources>
```

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello" />
```

Primer uporabe strings.xml

/drawable

- Vsebuje vire, ki so povezani s prikazom slik na zaslonu
- Obstaja več tipov virov, ki se lahko nahajajo v tej mapi
  - Bitmap datoteke (.png, .jpg, .gif)
  - Različne XML datoteke, ki definirajo lastnosti slik (skupni prikazi, tranzicije, prioritete pri prikazovanju, skaliranje, stopnja prikaza (clip) itd.)

/anim

- Vir, kjer so definirane nekatere animacije
- Možni sta dve vrsti animacij
  - Tween animacija – različne transformacije na isti sliki
    - Skaliranje
    - Fade-in, fade-out
    - Vertikalno ali horizontalno premikanje
    - Rotacija
  - Frame animacija – sekvenčno prikazovanje slik

/menu

- Tukaj definiramo aplikacijske menije
  - Options Menu
    - Se pojavi ob pritisku na tipko *Menu*
  - Context Menu
    - Se pojavi ob dolgem pritisku na določenih objektih `View`
    - Podobno desnemu kliku na PC-ju
  - Podmeniji
    - Definirani znotraj menijev

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/file"
          android:icon="@drawable/file"
          android:title="@string/file" >
        <!-- "file" submenu -->
        <menu>
            <item android:id="@+id/create_new"
                  android:title="@string/create_new" />
            <item android:id="@+id/open"
                  android:title="@string/open" />
        </menu>
    </item>
</menu>
```

Primer menija s podmenijem

Privzeti in alternativni viri

- Vire, kot so slike in različni nizi, je potrebno vedno izločiti iz programske kode aplikacije
  - S tem omogočimo neodvisnost vzdrževanja
- Definiramo lahko tudi alternativne vire
  - Podpora specifičnim konfiguracijam naprav (npr. jezik, velikost zaslona, orientacija zaslona itd.)
  - Pomen alternativnih virov postaja vedno bolj pomemben – vedno več različnih naprav

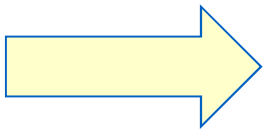
Dve različni napravi, ki uporabljata privzete vire

Dve različni napravi, ena uporablja alternativne vire

Lokalizacija

Ko razvijamo aplikacijo

Definiramo privzete vire ter dodatno t.i. alternativne vire, ki se uporabijo ob posameznih lokalnih nastavitvah



Ko uporabnik zažene aplikacijo

Sistem izbere katere vire bo uporabil glede na lokalne nastavitve.

- **Vsem dosedaj omenjenim virom lahko določimo tudi lokalne**
  - **Definiramo nove mape z uporabo različnih kvalifikatorjev**
    - `res/<resource_name>-<config_qualifier>`
    - `<resource_name>` **- ime privzetega vira (npr. values)**
    - `<config_qualifier>` **- ime konfiguracije za katero se alternativni vir uporablja**
  - **Za lokalizacijo uporabimo nacionalne kvalifikatorje (ISO-639-1)**
    - **http://www.loc.gov/standards/iso639-2/php/code_list.php**

Alternativni viri

- Seznam kvalifikatorjev je dolg in ni omejen samo na lokalizacijo
- Alternativni viri rešujejo različne situacije
  - Jeziki
  - Velikosti zaslonov
  - Orientacija zaslona
  - Ali je naprava v nosilcu?
  - Čas dneva (dan/noč)
  - Tip zaslona na dotik
  - Tip tipkovnice
  - Uporaba navigacijskih tipk
  - API Level
  - ...
- Privzeti viri so uporabljeni vsakič, ko alternativni za lokalne sistemske nastavitve ne obstajajo
  - Privzeti viri morajo biti vedno definirano 100%

Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

Shranjevanje podatkov

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Možnosti

- Android ponuja več načinov za persistenčno shranjevanje podatkov
  - Uporabljeni način je odvisen predvsem od potreb (npr. ali želimo, da so podatki privatni oz. so dostopni drugim aplikacijam, ter kakšna je količina podatkov)
- Možnosti shranjevanja podatkov so:
  - Shared Preferences
    - Shranjevanje privatnih primitivnih podatkov po principu ključ-vrednost
  - Internal Storage
    - Shranjevanje privatnih podatkov v pomnilnik naprave
  - External Storage
    - Shranjevanje javnih podatkov v zunanji pomnilnik (SD kartica)
  - SQLite DB
    - Shranjevanje strukturiranih podatkov v privatno podatkovno bazo
  - Network Connection
    - Shranjevanje javnih podatkov na spletni strežnik
- Privatne podatke lahko naredimo javne z uporabo ponudnikov vsebine

Shared Preferences

- Shranjevanje primitivnih podatkovnih tipov v obliki ključ-vrednost
  - Boolean, Float, Integer, Long, String
- Aktivnost `PreferenceActivity` za shranjevanje podatkov uporablja Shared Preferences

Primer uporabe Shared Preferences

```
public class Calc extends Activity {
    public static final String PREFS_NAME = "MyPrefsFile";

    @Override
    protected void onCreate(Bundle state){
       super.onCreate(state);
       . . .

       // Restore preferences
       SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
       boolean silent = settings.getBoolean("silentMode", false);
       setSilent(silent);
    }

    @Override
    protected void onStop(){
       super.onStop();

      // We need an Editor object to make preference changes.
      // All objects are from android.context.Context
      SharedPreferences settings = getSharedPreferences(PREFS_NAME, 0);
      SharedPreferences.Editor editor = settings.edit();
      editor.putBoolean("silentMode", mSilentMode);

      // Commit the edits!
      editor.commit();
    }
}
```

Internal Storage

- Datoteke lahko shranjujemo neposredno v pomnilnik naprave
- Podatki so privatni in do njih lahko dostopa samo prvotna aplikacija

```
String FILENAME = "hello_file";
String string = "hello world!";

FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
fos.write(string.getBytes());
fos.close();
```

Primer shranjevanja v datoteko

- Včasih želimo, da aplikacija takoj po namestitvi lahko dostopa do nekaterih podatkov
  - Datoteko s podatki shranimo v `/res/raw`
  - Metoda `openRawResource()` vrača objekt tipa `InputStream` za branje iz datoteke (pisanje ni možno)

External Storage

- Vsaka Android naprava vsebuje tudi zunanji pomnilnik
- Preden dostopamo do zunanjega pomnilnika iz aplikacije moramo preveriti njegovo razpoložljivost

```
boolean mExternalStorageAvailable = false;
boolean mExternalStorageWriteable = false;
String state = Environment.getExternalStorageState();

if (Environment.MEDIA_MOUNTED.equals(state)) {
    // We can read and write the media
    mExternalStorageAvailable = mExternalStorageWriteable = true;
} else if (Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
    // We can only read the media
    mExternalStorageAvailable = true;
    mExternalStorageWriteable = false;
} else {
    // Something else is wrong. It may be one of many other states, but all we need
    //  to know is we can neither read nor write
    mExternalStorageAvailable = mExternalStorageWriteable = false;
}
```

- Za dostop do direktorijske strukture uporabimo metodi
  - `getExternalStorageDirectory()`, `getExternalStoragePublicDirectory()`
  - Kot parameter podamo tip direktorija (npr. DIRECTORY_MUSIC)
  - Druga metoda dostopa do javnih map – podatki v njih se ne brišejo, če je aplikacija odstranjena iz sistema

SQLite DB

- Android ponuja polno podporo za SQLite DB
- Pri delu z bazo razširjujemo razred `SQLiteOpenHelper`

```
public class DictionaryOpenHelper extends SQLiteOpenHelper {

    private static final int DATABASE_VERSION = 2;
    private static final String DICTIONARY_TABLE_NAME = "dictionary";
    private static final String DICTIONARY_TABLE_CREATE =
                "CREATE TABLE " + DICTIONARY_TABLE_NAME + " (" +
                KEY_WORD + " TEXT, " +
                KEY_DEFINITION + " TEXT);";

    DictionaryOpenHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DICTIONARY_TABLE_CREATE);
    }
}
```

- Za pisanje in branje uporabimo metodi `getWritableDatabase()` in `getReadableDatabase()`
  - Obe metodi vračata objekt tipa `SQLiteDatabase`, ki predstavlja podatkovno bazo in prinaša metode za delo z njo
  - Ob poizvedbi (`query()`) je rezultat objekt tipa `Cursor`, ki kaže na vse vrstice v rezultatu poizvedbe

Telekomunikacije in informacijske tehnologije
Laboratorij za telekomunikacije
Fakulteta za elektrotehniko

Senzorji

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Senzorji v Androidu

- Android omogoča dostop do množice različnih senzorjev
  - Pospeškometer (Accelerometer)
  - Težnostni senzor (Gravity sensor)
  - Žiroskop (Gyroscope)
  - Svetlobni senzor (Light sensor)
  - Senzor magnetnega polja (Magnetic field sensor)
  - Senzor orientacije (Orientation sensor)
  - Senzor pritiska (Pressure sensor)
  - Senzor bližine (Proximity senzor)
  - Senzor temperature (Temperature sensor)
- Do posameznih senzorjev lahko dostopamo z uporabo razreda `SensorManager`
  - Do razreda dostopamo posredno – `Context.getSystemService(SENSOR_SERVICE)`
- Priporoča se, da izklapljamo senzorje vsakič, ko jih ne potrebujemo -> varčevanje s porabo baterije

Primer dostopa do podatkov senzorja

- Za dostop do senzorja je potrebno v manifest datoteki zahtevati ustrezne pravice!

```
public class SensorActivity extends Activity, implements SensorEventListener {
    private final SensorManager mSensorManager;
    private final Sensor mAccelerometer;

    public SensorActivity() {
        mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
        mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    protected void onResume() {
        super.onResume();
        mSensorManager.registerListener(this, mAccelerometer, SensorManager.SENSOR_DELAY_NORMAL);
    }

    protected void onPause() {
        super.onPause();
        mSensorManager.unregisterListener(this);
    }

    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    public void onSensorChanged(SensorEvent event) {
    }
}
```

- Na podoben način lahko dostopamo tudi do nekaterih drugih storitev sistema

Fünf – Open Sensing Framework

- http://funf.media.mit.edu
- Odprtokodna rešitev, ki omogoča enostaven dostop do podatkov različnih senzorjev
  - Enostaven razvoj lastnih sond

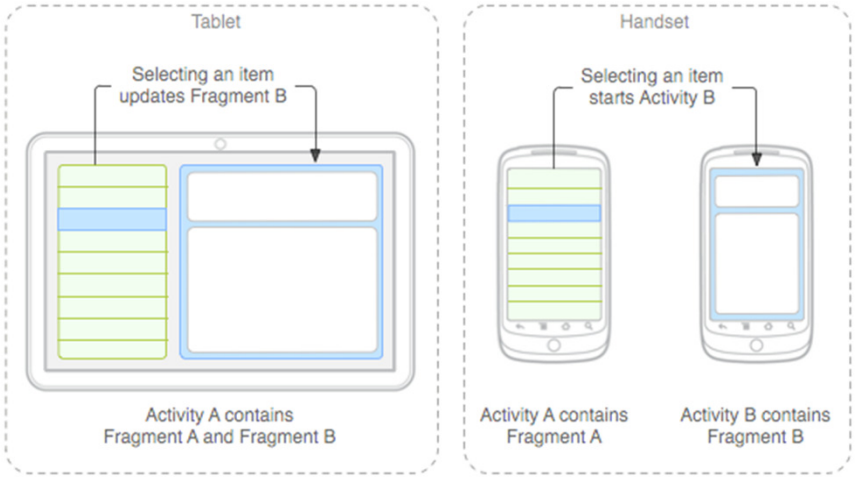
Glavne novosti različice ICS (4.0)

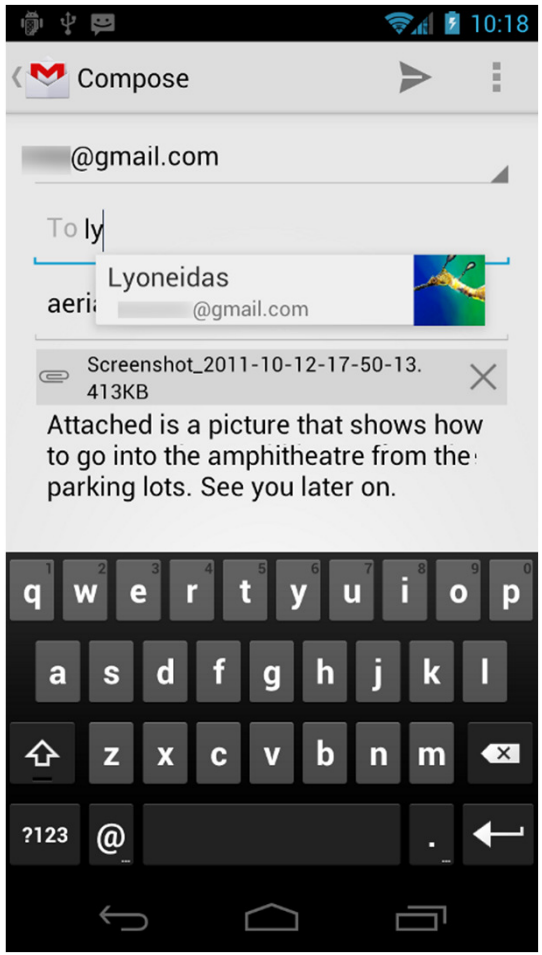
Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Android Market

- **Novi UI koncepti**
  - **Klasičnega sistemskega „menu“ gumba ni več – vlogo prevzema „Action bar“**
  - **Uporaba fragmentov:**





- **Android Beam**
  - **Omogoča deljenje vsebin z uporabo NFC**
- **Delo z medijo in povezljivost**
  - **HTTP live streaming, RTP, DRM framework, MTP/PTP file transfer, Bluetooth HDP…**

App Inventor

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

O App Inventorju

- http://appinventor.mit.edu/
- 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

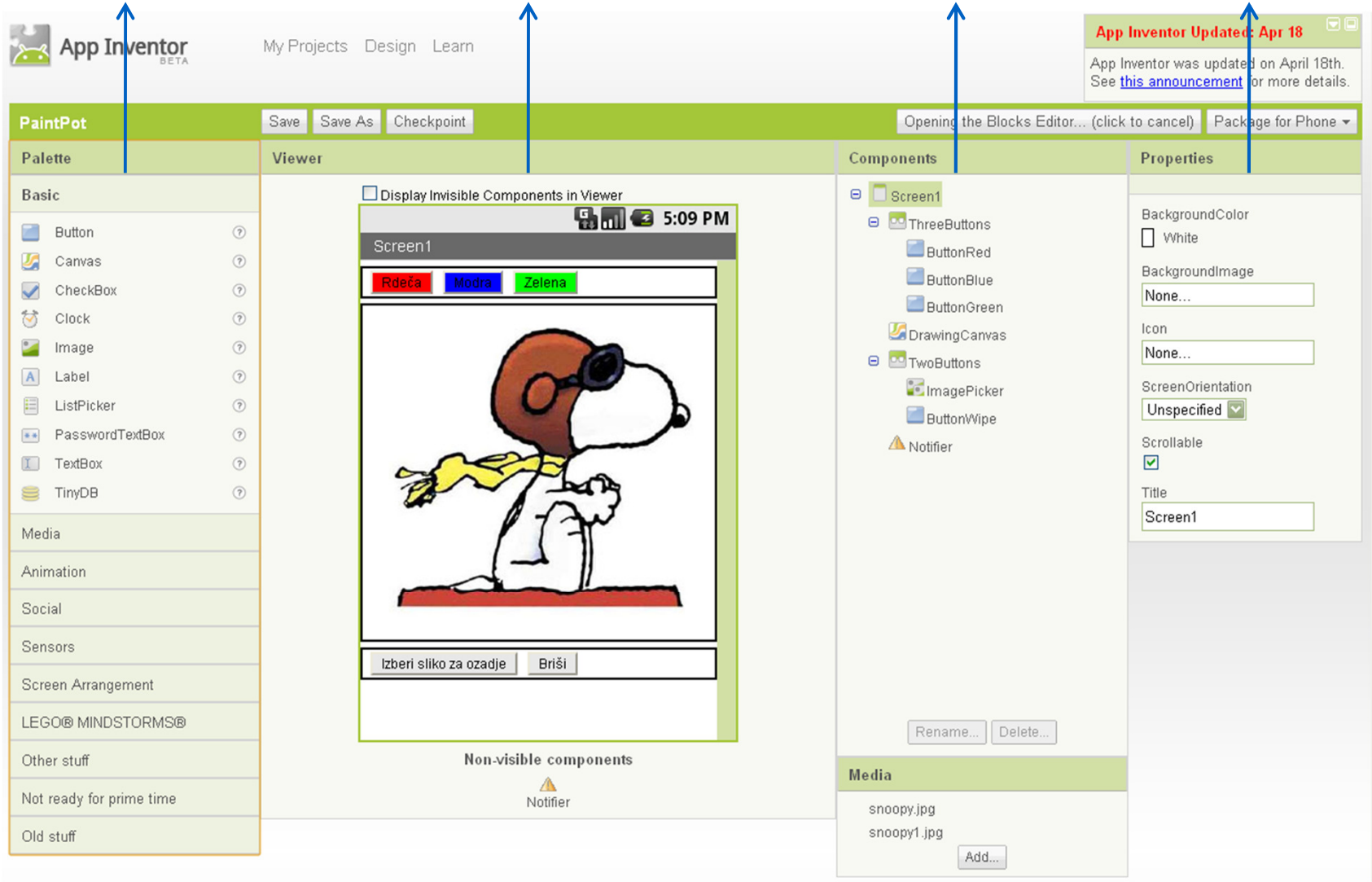
Designer

Seznam gradnikov

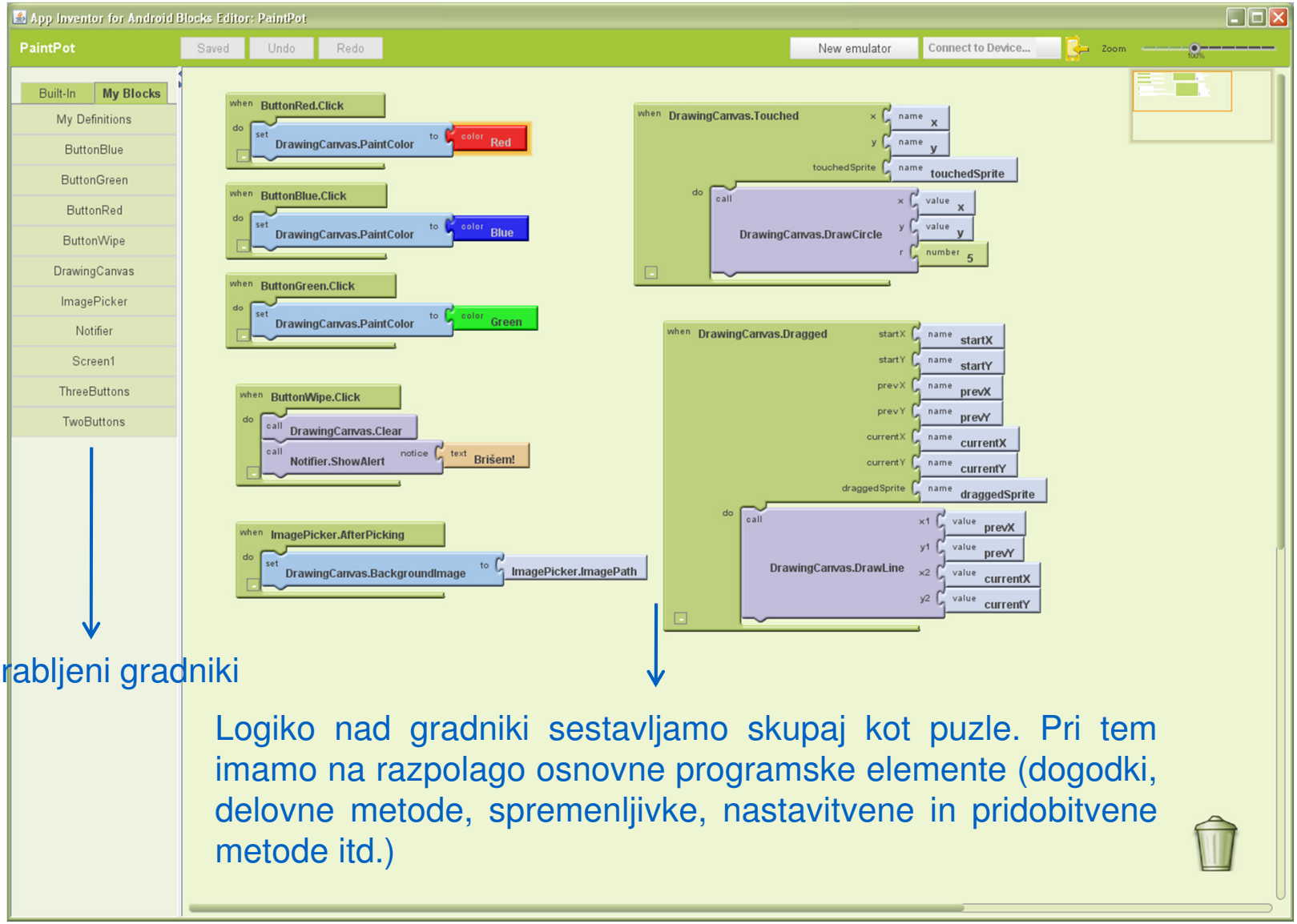
Delovna površina

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.)

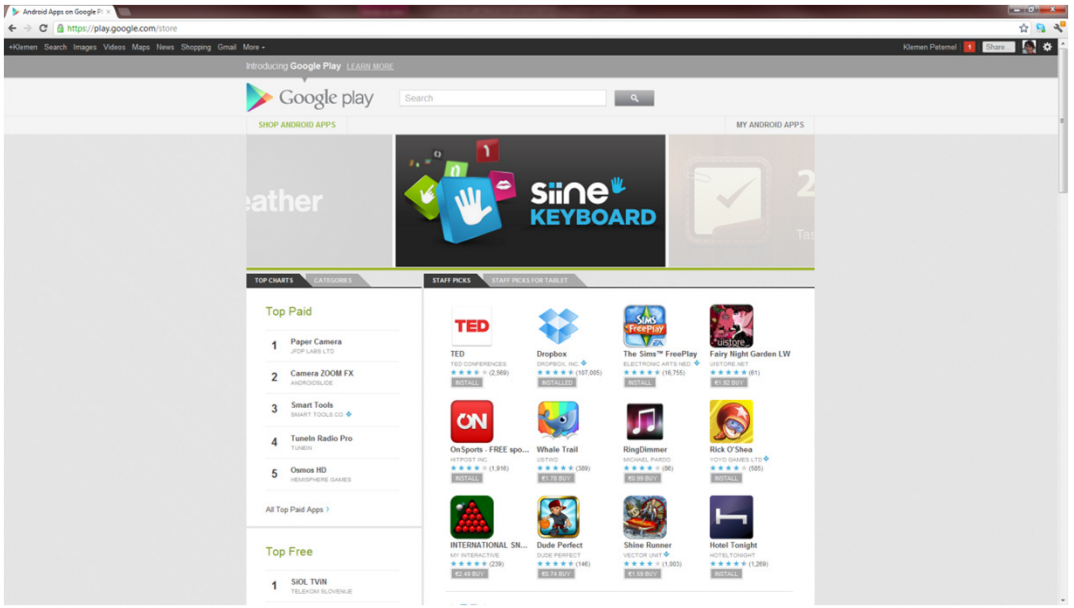
Objava aplikacije na Google Play (Android Market)

Klemen Peternel

Univerza v Ljubljani
Fakulteta za elektrotehniko
Laboratorij za telekomunikacije

Google Play

- Online trgovina za Android aplikacije
  - Na večini Android naprav je Play Store aplikacija že vključena
  - Do konca 2011 je bilo v storu okrog 380K različnih aplikacij
  - https://play.google.com/store
  - V Sloveniji že nekaj časa možno dostopati tudi do plačljivih aplikacij



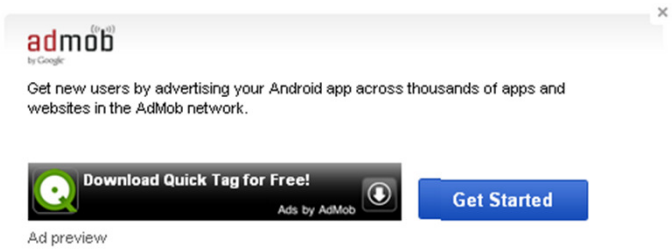
- Obstajajo tudi alternativne trgovine (Google to dopušča)
  - SlideMe, AndAppStore, Handango, AndroidGear, Phoload, Mobihand, AppsLib, ...

Objavljanje lastne aplikacije

- Objavljanje je možno na https://play.google.com/apps/publish
- 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