- **XML (eXtensible Markup Language)**
- Način za organizaciju i predstavljanje podataka, nezavisno od platforme i aplikacije
- Bazirano na **tagovima**, slično kao HTML
- Za razliku od HTML-a tagove definiše korisnik, a nisu unaprijed definisani – biraju se da budu deskriptivni
- Dizajniran da ga može razumjeti i čovjek i mašina
- Primjer:

```
<?xml version = "1.0"?>
<igrac>
        <ime>Petar</ime>
        <prezime>Petrovic</prezime>
        <golovaPoMecu>0.5</golovaPoMecu>
</igrac >
<igrac >
        <ime>Marko</ime>
        <prezime>Markovic</prezime>
        <golovaPoMecu>0.75</golovaPoMecu>
</igrac >
```

Komponente Android aplikacije

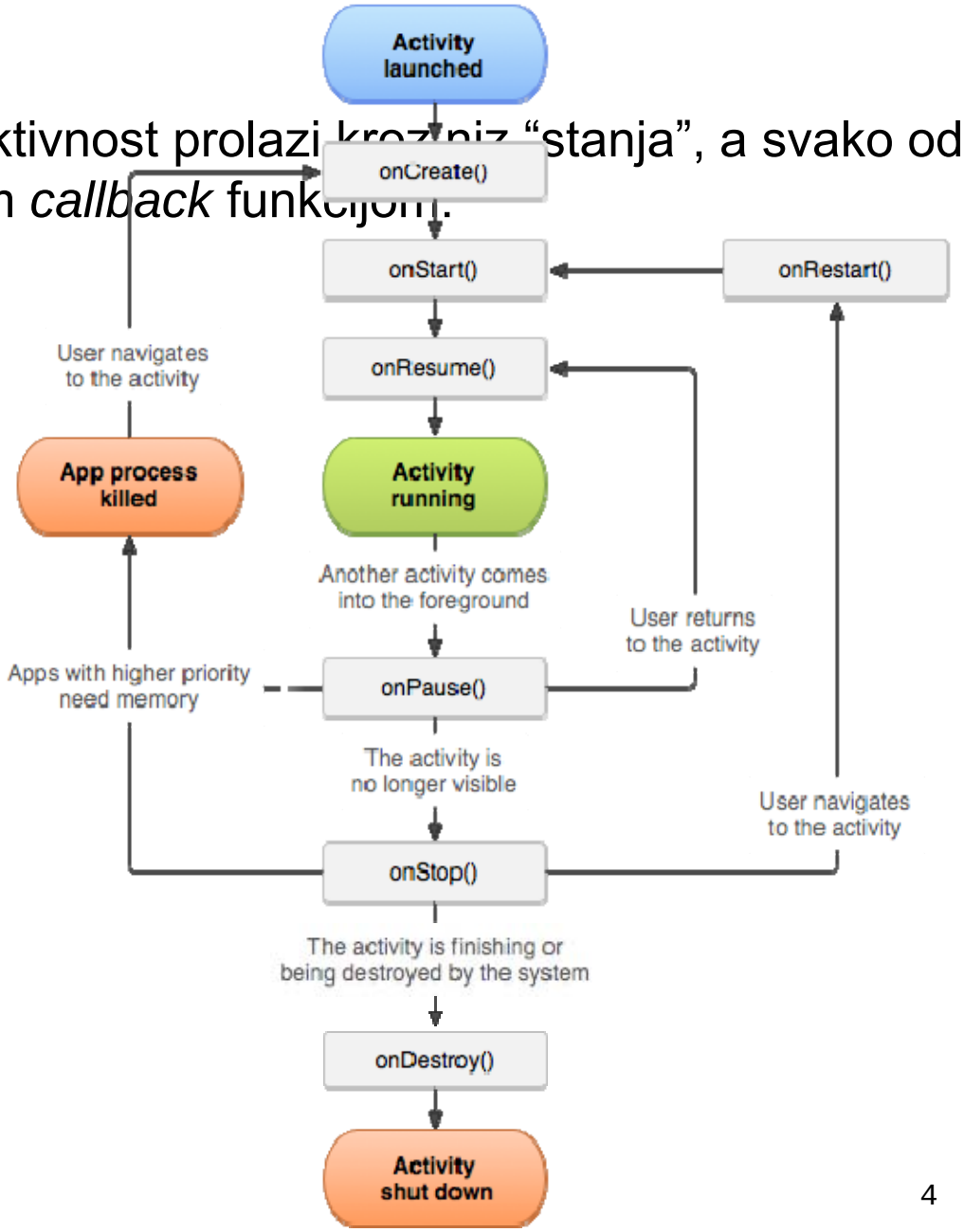
- Aktivnosti (*Activities*)
- Servisi (*Services*)
- *Broadcast receiver*-i
- *Content provider*-i

Aktivnosti (*Activities*)

- Ulazna tačka za interakciju sa korisnikom. Predstavlja jedan ekran sa korisničkim interfejsom (čitav ekran ili *pop-up*).
- Omogućava interakciju između sistema i aplikacije:
  - Prati šta korisnik radi na trenutnom ekranu, da bi se osiguralo izvršavanje procesa koji upravlja **aktivnošću**
  - Brine o prioritetu zaustavljenih **aktivnosti** kako bi se lakše pokrenuli njihovi procesi ako se korisnik vrati na njih (samo jedna je u potpunosti aktivna u jednom trenutku)
  - Brine da se prilikom povratka u prethodnu **aktivnost** vrati i njeno pređašnje stanje
- Implementira se kao potklasa klase **Activity**
- „Glavna“ aktivnost je ona koja se pokreće kad se startuje aplikacija (dodirom ikonice), ali se može direktno pozvati i neka druga aktivnost – sa nekog drugog mjesta (iz notifikacija ili druge aplikacije)
- Da bi koristili aktivnosti u svojoj aplikaciji, moraju se registrovati podaci o njima u manifestu aplikacije i mora se pravilno upravljati njihovim životnim ciklusima

Aktivnosti – životni ciklus

- U toku svog životnog ciklusa aktivnost prolazi kroz niz “stanja”, a svako od njih se servisira odgovarajućom *callback* funkcijom:
  - onCreate()
  - onStart()
  - onResume()
  - onPause()
  - onStop()
  - onRestart()
  - onDestroy()



Aktivnosti – životni ciklus - onCreate()

- Obavezna funkcija!
- Pokreće se kad OS kreira aktivnost, ali prije nego što ona postane vidljiva korisniku
- U njoj se inicijalizuju najvažnije komponente aktivnosti
- Mora se pozvati funkcija setContentView() da definiše izgled korisničkog interfejsa aktivnosti
- Kada se onCreate() završi, izvršava se onStart()

Aktivnosti – životni ciklus - onStart()

- Aktivnost postaje vidljiva korisniku
- Finalne pripreme za stavljanje aktivnosti u prvi plan i omogućavanje interaktivnosti

Aktivnosti – životni ciklus - onResume()

- Izvršava se prije nego aktivnost započne interakciju sa korisnikom
- U tom trenutku aktivnost prihvata unos od strane korisnika
- Najveći dio funkcionalnosti aplikacije se implementira u ovoj funkciji
- Nakon onResume() uvijek slijedi onPause()

Aktivnosti – životni ciklus - onPause()

- Izvršava se kada aktivnost više nije u fokusu, odnosno pauzira se (npr. aktivirano dugme ***back***)
- Aktivnost je još djelimično vidljiva, ali je korisnik (vjerovatno) napušta – uskoro će ući u stanje Stop ili Resume
- Nakon onPause() slijedi ili onStop() (ako se napušta aktivnost) ili onResume() (ako se vraćamo u aktivnost)

Aktivnosti – životni ciklus - onStop()

- Izvršava se kad aktivnost više nije vidljiva korisniku:
  - Napušta se
  - Nova aktivnost je pokrenuta
  - Neka aktivnost je ušla u Resume stanje i “prekrila” tekuću aktivnost
- Nakon onStop() slijedi ili onRestart() (ako će aktivnost ponovo da komunicira sa korisnikom) ili onDestroy() (ako se aktivnost definitivno završava)

Aktivnosti – životni ciklus - onRestart()

- Izvršava se kad aktivnost u stanju Stop treba da se ponovo pokrene
- onRestart() obnavlja stanje aktivnosti kakvo je bilo kada je zaustavljena
- Nakon onRestart() slijedi onStart()

Aktivnosti – životni ciklus - onDestroy()

- Izvršava se prije nego je aktivnost “uništena”
- Ovo je poslednja *callback* metoda
- Uobičajeno se koristi da osigura da su svi resursi koje je aktivnost zauzela oslobođeni kada aktivnost, ili process kojemu je aktivnost pripadala, više ne postoji

Servisi (*Services*)

- Ulazna tačka opšte namjene za aplikaciju koja se pokreće u pozadini, za obavljanje dugotrajnih operacija ili za obavljanje poslova za udaljene procese
  - Na primjer, **servis** može da pušta muziku u pozadini dok je korisnik u nekoj drugoj aplikaciji ili može prenositi podatke preko mreže bez blokiranja interakcije korisnika sa nekom aktivnošću
- Ne pruža korisnički interfejs
- Pokrenuti (*started*) servisi:
  - Ako je korisnik svjestan njihovog izvršavanja (npr. muzika) onda sistem treba da ih drži aktivnim sve do njihovog završetka
  - Ako korisnik ne prati izvršavanje procesa (npr. sinhronizacija podataka) onda sistem može da ih privremeno suspenduje da bi oslobodio resurse za prioritetnije poslove
- Povezani (*bound*) servisi: pokreće ih neka aplikacija ili sistem i zavise od izvršavanja te aplikacije
- Implementira se kao potklasa klase **Service**

Broadcast receiver

- Omogućava sistemu da pošalje obavještenje o nekom događaju, a aplikaciji da odgovori na takvo obavještenje
- Sistem može isporučiti obavještenje i aplikaciji koja nije trenutno aktivna (na primjer zadavanje alarma)
- Primjeri sistemskih obavještenja: o isključenju ekrana, o statusu baterije, …
- Primjeri obavještenja koje generiše aplikacija: završen prenos podataka, podaci ili resursi su dostupni za korišćenje, …
- Ne pruža korisnički interfejs, ali može kreirati obavještenje u *status bar*-u kako bi korisnik bio upozoren o događaju
- Nije predviđen da radi veliki posao, već da bude veza između komponenti
- Implementira se kao potklasa klase **BroadcastReceiver**, a obavještenja (*broadcast*) se isporučuju u obliku **Intent** objekata

Content provider

- Upravlja setom podataka koji se dijeli između aplikacija – omogućava da se podacima iz jednog procesa može pristupiti iz koda drugog procesa
- Podaci se mogu nalaziti u fajlovima, bazi podataka, na *web*-u, ili bilo kojoj lokaciji za čuvanje podataka kojoj aplikacija može pristupiti
- *Content provider* je, gledano iz ugla sistema, ulazna tačka u aplikaciju za pristup objektima (koji imaju ime definisano sa **URI**) u kojima se nalaze podaci
- Pogodan je i za rad sa podacima koji se ne dijele, već su privatni za pojedinu aplikaciju

Šta je URI?

- **U**niform **R**esource **I**dentifier je sekvenca karaktera koja označava resurs – fizički ili apstraktni
- Jedna od uobičajenih formi URI je adresa *web* stranice - *Uniform Resource Locator* (URL)
- Android koristi URI string kao osnovu za pristup podacima u *content provider*-u ili za pokretanje neke akcije (npr. otvaranje *web* stranice u *browser*-u)

- **Android studio – važni fajlovi**:
- **app > java > com.example.naziv_projekta > MainActivity**

  U ovom fajlu se nalazi glavna aktivnost (*Activity*), koja predstavlja ulaznu tačku prilikom izvršavanja aplikacije. Dakle, prilikom pokretanja aplikacije biće pokrenuta instanca ovog *Activity*-a.
- **app > res > layout > activity_main.xml**

  U ovom XML fajlu se definiše izgled i raspored komponenti *Activity*-a. Prilikom kreiranja projekta Android studio sam ubacuje *TextView* element sa tekstom „Hello world!“.
- **app > manifests > AndroidManifest.xml**

  U ovom fajlu se opisuju osnovne karakteristike aplikacije i definiše svaka njena komponenta.

Aktivnosti (*Activities*) – konfiguracija manifesta

- Da bi deklarisali aktivnost, u fajl manifesta treba dodati <activity> element, kao podređeni elementu <application>:

```
<manifest ... >
        <application ... >
                <activity android:name=".ExampleActivity" />
                ...
        </application ... >
        ...
</manifest >
```

- Jedini zahtijevani atribut za ovaj element je android:name, koji određuje naziv klase aktivnosti
- Mogu se dodati i drugi atributi koji definišu labele, ikone, UI teme, ...