1. MOBILNE APLIKACIJE – UVOD

MOBILNI UREĐAJI

Karakteristike:

- mobilnost (prenosiv i koristi servise bazirane na lokaciji)
- ograničeni hardverski resursi (centralni i grafički procesor, operativna i spoljna memorija, izvor el. energije)
- korisnički interfejs (ekran osetljiv na dodir, prepoznavanje govora, senzori)
- mobilne komunikacije (mobilna i WI-FI mreža, Bluetooth, NFC)
- senzori (a. pozicije (GPS, mob i WI-FI mreža),
  b. kretanja (akcelerometar, žiroskop),
  c. okoline (magnetno polje, temp., pritisak, osvetljenost, blizina objekta))

Vrste:

- pametni telefoni,
- tablet računari,
- wearables (satovi, naočare, itd.) ,
- namenski mobilni uredaji (fotoaparati, kamere, GPS, itd.)

Uredaji sličnih karakteristika

- TV/STB ,
- uredaji u automobilima,
- uredaji u kući (frižider, mikrotalasna pećnica, itd.)

Native vs. Web Apps

- Android/iOS
- HTML5/CSS/JavaScript

Native vs. Hybrid Apps

- UI/UX - portabilnost
- performanse - brži razvoj

Hybrid Frameworks

- Xamarin, React Native, Flutter, NativeScript, Cordova (Ionic)

ANDROID PLATFORMA

Sta je Android?

- OS
- Application framework
- Applications

Istorija

- Android, Inc.
- Google
- Open Handset Alliance (OHA)

Verzije

- 1.0(2008), Cupcake, Donut, Lolipop, Ice Cream sandwich, Marshmallow, 1.14(2024)

- Android 11 Red Velvet Cake, Android API level 30

Karakteristike Androida

- FLOSS (OS, VM, API, Apps, SDK)
- multitasking (više procesa, više niti)
- sandboxing (izolovanje aplikacija)

Arhitektura Androida

- Linux kernel – Drivers(Camera, Wifi, USB, Audio, Display), Power Managment
- Hardware abstraction layer (HAL) - Audio, Bluetooth, Camera, Sensors
- Android runtime – ART, Core Libraries
- Native C/C++ libraries – Webkit, Media Framework
- Java API framework – Content providers, View System, Managers(Activity, Location, Package, Notification)
- System apps – Dialer , Email, Calendar, Camera

Android platforma

- GNU/Linux
    - Bionic C biblioteka
- JVM
    - Dalvik do Androida 4.4
    - Android Run Time (ART) od Androida 5.0
- standardna Java biblioteka
    - Apache Harmony do Androida 6.0
    - OpenJDK od Androida 7.0

KOMPONENTE ANDROID APLIKACIJE

Softverska komponenta je jedinica kompozicije softvera sa specifciranim interfejsom i eksplicitnom kontekstnom zavisnosnoscu

- **1. Aktivnost (activity)**
  - Predstavlja pojedinacni ekran Android aplikacije
  - Aplikacija se sastoji iz vise slabo povezanih aktivnosti
  - Glavna aktivnost je aktivnost koja ce se prikazati korisniku kada pokrene aplikaciju
  - Pravljenje aktivnosti :
    - Definisati klasu koja nasleduje Activity klasu (ili neku od njenih naslednica)
    - Dodati activity element u AndroidManifest.xml
  - Zivotni ciklus aktivnosti :
    - Aktivnost moze da se nalazi u jednom od tri stanja:
      - resumed (aktivnost se izvrsava - ako se nalazi u prvom planu i ima fokus)
      - paused (aktivnost je pauzirana - ako se druga aktivnost nalazi u prvom planu i ima fokus, ali je prva aktivnost jos uvek vidljiva (zato sto je druga aktivnost transparentna ili ne pokriva ceo ekran).
      - stopped (aktivnost je zaustavljena - ako se nalazi u pozadini (potpuno je prekrivena drugom aktivnoscu).
- Zivotni vek aktivnost:
  - Ceo zivotni vek
  - Zivotni vek u kome je vidljiva
  - Zivotni vek u kome je u prvom planu

- onCreate
    - Sistem poziva kada startuje aktivnost.
    - Ova metoda treba da zauzme resurse i inicijalizuje komponente neophodne za pravilno funkcionisanje aktivnosti.
    - Pozivom setContentView metode iscrtava se korisnicki interfejs.
- onStart
    - Sistem poziva neposredno pre nego sto aktivnost postane vidljiva korisniku.
- onResume
    - Poziva se neposredno pre nego sto aktivnost pocne interakciju sa korisnikom. U ovom trenutku aktivnost se nalazi na vrhu steka aktivnosti.
- onPause
    - Sistem poziva neposredno pre nego sto pauzira izvrsavanje aktivnosti.
    - Ova metoda se obicno koristi za snimanje perzistentnih podataka i zaustavljanje procesa koji zauzimaju procesor.
    - Mora biti vrlo brza zato sto sledeca aktivnost ne moze da pocne da se izvrsava sve dok se ova metoda ne zavrsi.
- onStop
    - Poziva se kada aktivnost vise nije vidljiva korisniku.
- onRestart
    - metoda se poziva nakon sto je aktivnost zaustavljena, a pre nego sto je ponovo startovana.
- onDestroy
    - Poslednja metoda koja se poziva pre nego sto se aktivnost unisti.
    - Ova metoda oslobada zauzete resurse pre nego sto se aktivnost unisti

- Snimanje stanja aktivnosti:
    - Kada se aktivnost pauzira ili zaustavi, njeno stanje je sacuvano u memoriji.

    Medutim, da bi se sacuvalo stanje aktivnosti ako se ona unisti, potrebno je implementirati dodatnu metodu. Oprez: Android moze u bilo kom trenutku ubiti aktivnost koja se ne nalazi u prvom planu!!!
    - onSaveInstanceState
        - Poziva se pre nego sto se aktivnost unisti da bi se snimilo njeno stanje koje se ponovo inicijalizuje u onCreate ili onRestoreInstanceState metodi.
    - onRestoreInstanceState
        - Poziva se posle onStart metode da bi se aktivnost ponovo inicijalizovala iz prethodno snimljenog stanja.
    - Klasa Bundle sadrzi metode oblika:
        - T getT(String key);
        - void putT(String key, T value);
    - Podrazumevana implementacija ovih metoda poziva onSaveInstanceState nad svakim elementom korisnickog interfejsa sto za rezultat ima cinjenicu da se stanje korisnickog interfejsa automatski snima.
    - Rukovanje promenom konfiguracije
        - Ako se konfiguracija uredaja promeni (orijentacija ekrana, jezik, itd.), korisnicki interfejs se mora osveziti da bi odgovarao konfiguraciji. Promena konfiguracije prouzrokuje unistenje i ponovno stvaranje aktivnosti
- Zadatak (task)
    - Aplikacija se obicno sastoji iz vise aktivnosti.
    - Zadatak je skup aktivnosti sa kojima korisnik intereaguje da bi izvrsio odreden posao.
- Povratni stek (back stack)
    - Aktivnosti su uredene u povratni stek u redosledu u kome su startovane.
    - Kada se aktivnost startuje, stavlja se na vrh steka, prelazi u prvi plan i dobija fokus.
    - Pritiskom na Back dugme, tekuca aktivnost se skida sa vrha steka i unistava.
    - Svakom zadatku odgovara jedan povratni stek.

- Samo jedan zadatak se moze nalaziti u prvom planu i imati fokus u datom trenutku.
- Zadatak koji je u pozadini ima ocuvan svoj povratni stek.

- **2. Servis (service)**
  - Servis se izvrsava u pozadini
  - Ne zahteva interakciju sa korisnikom
  - Servis je komponenta koja izvrsava duge operacije. Postoje tri vrste servisa:
    servis u prvom planu (foreground service) - izvrsava uocljivu operaciju (npr. reprodukcija muzike)
    servis u pozadini (background service) - nije potrebna interakcija sa korisnikom (npr. file download)
    vezan servis (bound service) - za implemantaciju klijent-server arhitekture
  - Servis u prvom planu i u pozadini se nazivaju i startovani servisi (started service)
  - Izvrsava se u istoj niti u kojoj se izvrsavala komponenta koja ga je startovala (cak i ako komponenta nije aktivna)
  - Druga komponenta moze da se veze za servis i da sa njime komunicira (cak i ako se nalazi u drugom procesu)
  - Moze biti startovan ili vezan (moze istovremeno biti oba, ali se retko koristi)
  - Startovan servis se izvrsava neodredeno vreme (servis treba da se sam zaustavi kada izvrsi operaciju)
  - Vezan servis se izvrsava samo dok je neka komponenta vezana za njega (nudi interfejs koji omogucava komponentama da komuniciraju sa njim saljuci zahteve i dobijajuci odgovore)
  - Zivotni ciklus : onCreate (poziva se prilikom stvaranja servisa)
    onStartCommand (poziva se posle poziva startService metode)
    onBind (poziva se posle poziva bindService metode)
    onUnbind (poziva se posle poziva unbindService metode)
    onRebind (poziva se posle poziva bindService ako je prethodno izvrsena onUnbind metoda)
    onDestroy (poziva se prilikom unistavanja servisa)
  - Razlikuje se ceo zivotni vek servisa (izmedu poziva onCreate i onDestroy metoda) i aktivni (pocinje pozivom onStartCommand ili onBind metode, a zavrsava se pozivom onDestroy ili onUnbind metode)
  - Moze da se napravi nasledivanjem klasa:
    - Service (u ovom slucaju je vazno startovati pozadinsku nit u kojoj ce se izvrsiti operacije i voditi racuna u sinhronizaciji ukoliko vise komponenti istovremeno koriste isti servis)
    - IntentService (u ovom slucaju ce se operacije automatski izvrsiti u pozadinskoj niti i pozivi metoda servisa ce se automatski sinhronizovati). Od verzije Android 11 (API level 30) koristi se JobIntentService
  - Flags: If the system kills the service after onStartCommand() returns,
    START_NOT_STICKY- do not recreate the service.
    START_STICKY - recreate the service and call onStartCommand().
    START_REDELIVER_INTENT - recreate the service and call onStartCommand() with the last intent delivered.
  - Zaustavljanje servisa
    -Servis se moze zaustaviti sam pozivom stopSelf metode, moze ga zaustaviti druga komponenta pozivom stopService metode ili ga moze zaustaviti Android platforma (da bi oslobodila memoriju)
    - Aplikacije bi trebalo da zaustave svoje servise cim izvrse operaciju da se ne bi trosili resursi (npr. baterija)
  - Pokretanje servisa u prvom planu
    -Moze se pokrenuti pozivom startForeground metode, a ukloniti iz pozivom stopForeground metode
    - Trebalo bi da se nalazi u prvom planu ukoliko je korisnik svestan servisa (sto znaci da ne treba da se ubije u nedostatku memorije) .
    - Mora obezbediti obavestenje u statusnoj liniji

- **3. Prijemnik poruka (broadcast receiver)**
  - Prijemnici poruka obradjuju asinhrone dogadjaje (opisane objektima klase Intent)
  - Dogadjaje mogu da izazovu operativni sistem(Android platforma) ili druga komponenta
  - Moguce ga je registrovati staticki (manifest fajla) ili dinamicki (iz programkog koda).

- Parametri onReceive metode su:
    - context (kontekst u kome se izvrsava onReceive metoda)
    - intent (namera koja opisuje dogadaj koji treba obraditi)
- Namera prosledena startActivity ili startService metodi nece prouzrokovati dogadaj koji ce obraditi onReceive metoda (vazi i obrnuto)
- Akcije: ACTION_BATTERY_LOW - A warning that the battery is low.
    ACTION_POWER_CONNECTED - External power has been connected to the device.
    ACTION_HEADSET_PLUG - A headset has been plugged into the device, or unplugged from it.
    ACTION_SCREEN_ON - The screen has been turned on.
    ACTION_SHUTDOWN - Device is shutting down
- Metoda onReceive se poziva iz glavne niti (to znaci da dugacke operacije treba izvrsavati u posebnom servisu koji startuje posebnu nit)
- Prijemnik poruka postoji samo u toku izvrsavanja onReceive metode (zato se u ovoj metodi ne mogu izvrsiti asinhrone operacije kao sto su prikazivanje dijaloga ili vezivanje za servis)
- Proces u kome se izvrsava onReceive metoda ima foreground prioritet (nakon toga, prioritet procesa odreduju ostale komponente koje se u njemu nalaze)
- Informacije o dogadajima se do prijemnika poruka prenose putem namera.
- Broadcast poruke se dele na eksplicitne (namenjene konkretnom prijemniku poruka) i implicitne (prijemnik poruka se pronalazi na osnovu filtera namera).
- Sistemski dogadaji koriste implicitne namere kako bi razlicite komponente mogle primiti poruke o tim dogadajima.
- Prijavljivanje prijemnika poruka na odredenu vrstu implicitno definisanih poruka moze dovesti do preopterecenja resursa uredaja ukoliko postoji veci broj prijemnika poruka koji ocekuju istu vrstu poruke.
- Tako je za neke implicitne broadcast poruke koje generise sistem dovoljno koristiti <intent-filter>. Poruke koje se ne desavaju cesto. Primeri su opisani akcijama: ACTION_BOOT_COMPLETED, ACTION_TIMEZONE_CHANGED, ACTION_USB_DEVICE_ATTACHED/DETACHED, ACTION_MEDIA_MOUNTED/UNMOUNTED
- Za ostale sistemske dogadaje treba koristi dinamicku registraciju. npr: CONNECTIVITY_ACTION
- Pored dinamicke, moze biti neophodan i dinamicki (runtime) zahtev za odgovarajucim permisijama potrebnim za obradu te vrste dogadaja.
- Za neke sistemske dogadaje u novijim verzijama Androida se ne emituju broadcast poruke. To je slucaj sa: ACTION_NEW_PICTURE, ACTION_NEW_VIDEO. Tada se preporucuje koriscenje JobScheduler klase.
- Postoje dva vrste dogadaja koje prijemnici poruka mogu obraditi:
    - normalni dogadaji (asinhroni, prijemnici ih obraduju nedefinisanim redosledom i obrada je efikasnija)
    - uredeni dogadaji (obraduje ih vise prijemnika redom, a svaki prijemnik moze da prosledi dogadaj sledecem prijemniku ili da potpuno obustavi njegovu obradu)
- Metode: abortBroadcast() - Sets the flag indicating that this receiver should abort the current broadcast.
    getResultCode() - Retrieve the current result code, as set by the previous receiver.
    getResultData() - Retrieve the current result data, as set by the previous receiver.
    setResultCode(int code) - Change the current result code of this broadcast.
    setResultData(String data) - Change the current result data of this broadcast
- Prijemnici omogucavaju razmenu poruka izmedu komponenti razlicitih aplikacija (paziti na zloupotrebe)
- Moguce je primeniti prava pristupa prilikom slanja poruke (prosledujuci permission parametar sendBroadcast metodi) ili prilikom prijema poruke (postavljajuci permission atribut receiver elementa)
- Komponenta koja salje/prima poruku mora imati odgovarajuce pravo pristupa (zatrazeno <uses-permission> u AndroidManifest.xml)

- **4. Dobavljavljac sadrzaja (content provider)**
    - Dobavljaci sadrzaja upravljaju podacima
    - Omogucavaju skladistenje podataka i razmenu podataka izmedu aplikacija (tj. komunikaciju izmedu procesa)
    - Enkapsuliraju podatke omogucavajuci da im se pristupi na standardizovan nacin

- Omogucava razmenu podataka izmedu komponenti koje se nalaze u razlicitim procesima i obezbeduju bezbedan (a obicno i perzistentan) pristup podacima
- Pruzaju podatke drugim komponentama u formi jedne ili vise tabela (slicno relacionalnom modelu podataka)
- Vrsta u tabeli predstavlja instancu entiteta, a kolona njegovo svojstvo
- Podatke opisuje URI i MIME tip URI (content://user_dictionary/words) se sastoji iz seme (content), imena dobavljaca sadrzaja (user_dictionary) i imena tabele (words)
- Pojedinacnoj vrsti se moze pristupiti dodavanjem njenog identifikatora na URI (content://user_dictionary/words/1)
- MIME tip specificira tip sadrzaja (text/plain, text/pdf, image/jpeg, itd.)
- Za pristup podacima koje pruza dobavljac sadrzaja koristi se ContentResolver klasa
- Ova klasa omogucava izvrsavanje CRUD operacija nad (perzistentnim) skladistem podataka
- Pri tome se dobavljac sadrzaja ne mora nalaziti u istom procesu, a automatski je obezbeden pristup podacima
- Podacima se pristupa u dva koraka:
    - Zatraze se odgovarajuca prava pristupa
    - Izvrsi se upit nad dobavljacem sadrzaja
- Da bi mogla da pristupi podacima, komponenta mora da ima read access permission nad odgovarajucim provajderom u AndroidManifest.xml
- Upit se postavlja na slican nacin na koji se postavlja SQL upit
- Sadrzi URI, spisak kolona koje treba vratiti (projekciju), uslov koji vracene vrste treba da zadovolje (selekciju) i nacin sortiranja rezultata
- Na slican nacin na koji je moguce pristupiti podacima, moguce ih je i promeniti.
- Razlikuju se:
    - Sistemski - koji su ukljuceni u Android (Browser, Calendar, CallLog, Contacts, Settings, UserDictionary)
    - Aplikacioni - koje pisu programeri koji pisu i ostale komponente aplikacije

        Pravljenje se sastoji od nekoliko koraka:

        - Prvo treba odrediti na koji nacin ce se skladistiti podaci
        - Zatim treba naslediti ContentProvider klasu i deklarisati provajder u AndroidManifest.xml
        - Na kraju treba definisati odvojenu Contract klasu u kojoj se nalazi ime provajdera, tabela i kolona, kao i prava pristupa
        - Podaci se mogu skladistiti u bilo kojoj vrsti skladista. Ali trebalo bi: Strukturirane podatke u SQLite,

            Slike (i ostale podatke u binarnom obliku koji zauzimaju dosta prostora) u sistemu datoteka ,

            Takode je moguce koristiti klase iz java.net paketa za skladistenje podataka na mrezi
- Prilikom nasledivanja ContentProvider klase obavezno treba implementirati query(), insert(), update(), delete(), getType() i onCreate() metode
- Ove metode imaju iste potpise kao i odgovarajuce metode ContentResolver klase
- Sve metode osim metode onCreate() moraju biti thread-safe!
- Izbegavati izvrsavanje dugackih operacija u onCreate() metodi!
- Ne postoji onDestroy metoda (dobavljaci sadrzaja postoje od pocetka do kraja procesa)
- Contract klasa predstavlja “ugovor” izmedu dobavljaca sadrzaja i aplikacija koje ga koriste
- Omogucava pristup dobavljacu sadrzaja, cak i ako se njegova implementacija promeni (URI, imena tabela, imena kolona, itd.)
- To je javna finalna klasa koja sadrzi konstante koje odgovaraju URI-u koji identifikuje podatke, njihovom MIME tipu, imenima tabela i imenima kolona
- Contract klasa takode olaksava posao programerima zato sto obicno sadrzi konstante koja se lako pamte

- **5. Namera (intent)**
    - Poruka koje se salje izmedu komponenti

- Apstraktni opis operacije koja treba da se izvrsi
- Sluzi za povezivanje komponenti Android aplikacije
- Sadrzi svojstva potrebna komponenti koja obraduje nameru (akcija, podaci, dodatne informacije) i sistemu (komponenta, kategorije i oznake).
- Eksplicitne i implicitne namere
    - Eksplicitne namere eksplicitno opisuju komptonentu koja treba da izvrsi akciju.

```
Intent intent = new Intent();
intent.setClassName("com.example", "ExampleActivity");
startActivity(intent);
```

    - Implicitne namere implicitno opisuju akciju koja treba da se izvrsi.

```
Intent intent = new Intent();
intent.setAction(Intent.ACTION_SEND);
intent.putExtra(Intent.EXTRA_EMAIL, recipients);
startActivity(intent);
```

- Startovanje aktivnosti
  - pozivom startActivity ili startActivityForResult metode.
  - Ove metode omogucavaju startovanje navedene aktivnosti (prosledivanjem eksplicitne namere) ili neke aktivnosti koja je opisana odredenim svojstvima (prosledivanjem implicitne namere).
- Zaustavljanje aktivnosti
  - Aktivnost se moze zaustaviti pozivom finish() metode, medutim zaustavljanje aktivnosti treba prepustiti sistemu.
- - Komponenta
    - Opisuje komponentu koja treba da obradi nameru.
    Postavlja se u konstruktoru ili Intent setClassName(String packageName, String className) metodom.
    Ukoliko je ovo svojstvo postavljeno, namera je eksplicitna.
  - Akcija
    - Svojstvo akcija (action) opisuje akciju koja treba da se izvrsi.
    Akcija u najvecoj meri odreduje kako je strukturiran ostatak namere (podaci i dodatne informacije).
    Postavlja se u konstruktoru ili Intent setAction(String action) metodom.
    Preporucuje se koriscenje predenisanih akcija. :
      ACTION_MAIN - Start up as the initial activity of a task.
      ACTION_CALL - Initiate a phone call.
      ACTION_EDIT - Display data for the user to edit.
      ACTION_SYNC - Synchronize data on a server with data on the mobile device.
- Podaci (data) i tip (type)
    - Opisuju podatke koji treba da se obrade i MIME tip tih podataka.
    Postavljaju se u konstruktoru ili setData(URI data), setType(String type) i setDataAndType(Uri data, String type) metodama.
    Zavise od akcije koja treba da se izvrsi.
- Dodatne informacije (extra)
  - potrebne komponenti koja obraduje nameru opisane su uredenim parovima (kljuc, vrednost).
  Postavljaju se metodama oblika putExtra(String key, T value). :
    EXTRA_PHONE_NUMBER - A String holding the phone number to call.
    EXTRA_EMAIL - A String array holding e-mail addresses that should be deliveered to.
    EXTRA_TEXT - A String used to supply the literal data to be sent.
- Kategorije
  - Svojstvo kategorije (categories) opisuje vrstu komponente koja obraduje nameru.
  Postavlja se addCategory(String category) metodom.
  Jedna namera moze sadrzati vise kategorija :

CATEGORY_DEFAULT - Set if the activity should be an option for the default action to perform on a piece of data.
CATEGORY_LAUNCHER - The activity can be the initial activity of a task and is listed in the top-level application launcher.
CATEGORY_GADGET The activity can be embedded inside of another activity that hosts gadgets.
CATEGORY_PREFERENCE The target activity is a preference panel.

- Oznake (flags)

  - sugerisu sistemu kako da startuje aktivnost (npr. kom zadatku treba da pripada) i kako da je tretira nakon sto je startuje (npr. da li treba da se prikaze u spisku nedavnih aktivnosti).
  Postavljaju se metodom setFlags(int flags).
  Jedna namera moze sadrzati vise oznaka (onda se oznake postavljaju disjunkcijom predefnisanih vrednosti). :
  NEW_TASK - If set, this activity will become the start of a new task on this history stack.
  EXCLUDE_FROM_RECENTS - If set, the new activity is not kept in the list of recently launched activities.

- Namera na cekanju (pending intent)

  - je namera koja omogucava jednoj komponenti da izvrsi operaciju koristeci identitet i prava pristupa druge komponente.

- Filter namera (intent filter)

  - Opisuje mogucnost komponente (namere koje komponenta moze da obradi)
  - Sadrzi polja koja odgovaraju svojstvima namere (akcija, podaci, i kategorija)
  - Kada primi implicitnu nameru da startuje aktivnost, sistem pronalazi odgovarajuce aktivnosti tako sto poredi nameru i filtere namera na osnovu:
    - akcije (akcija specicirana u nameri mora da odgovara jednoj od akcija speciciranih u filteru),
    - podataka (URI i MIME tip specicirani u nameri moraju da odgovaraju jednom URI-u i MIME tipu specificiranim u filteru),
    - kategorije (svaka kategorija specificirana u nameri mora da odgovara jednoj od kategorija specificiranih u filteru; ne mora da vazi obrnuto).
  - Namera mora proci sva tri testa da bi bila prosledena komponenti.
  - Jedna komponenta moze sadrzati vise filtera.

- **6. Resursi**

  - Android aplikacije pored izvornog koda (najcesce pisanog u Javi) sadrze i resurse
  - Resursi mogu bili tekst, rasterska i vektorska grafika, audio i video klipovi, itd.

Koraci u razvoju Android aplikacija

- design
- develop (JDK, Android SDK, Android Studio)
- distribute (Google Play)

Alati

- Java Development Kit (JDK)
- Android SDK (Software Development Kit)
  - SDK tools (alati koji ne zavise od platforme)
  - SDK platform tools (backward compatible alati koji zavise od platforme)
    - Android SDK Manager (android sdk)
    - AVD Manager (android avd)
    - Emulator (emulator)
    - Dalvik VM Bytecode Compiler (dx)
    - Android Asset Packaging Tool (aapt)
    - Android Debug Bridge (adb)
    - Dalvik Debug Monitor Service (ddms)
    - SQLite (sqlite3)

- SDK platform (Android API)
- sistemske slike
- dokumentacija
- primeri
- izvorni kod

- Android Studio (Android IDE)
  - Android Studio je razvojno okruzenje za Android aplikacije (zasnovano IntelliJ)
  - interno koristi SDK (platform) tools
  - za izgradnju projekta koristi Gradle
  - za upravljanje izvornim kodom koristi Git

FRAGMENTI I PERMISIJE

- **1. Fragmenti**
  - predstavljaju deo ponasanja ili GUI-a aktivnosti (mogu se posmatrati kao podaktivnosti).
  - Jedna aktivnost moze da sadrzi vise fragmenata i jedan fragment moze da bude sadrzan u vise aktivnosti (ali ne ista instanca fragmenta).
  - Fragmenti imaju zivotni ciklus (koji zavisi od zivotnog ciklusa aktivnosti u kojoj se nalaze) i mogu da obraduju dogadaje koje stvara GUI.
  - U toku izvrsavanja aplikacije se mogu izvrsavati transakcije nad fragmentima (mogu se dodavati, uklanjati, zamenjivati, itd.)
- Pravljenje fragmenta
  - Napisati klasu koja nasleduje Fragment klasu
  - Dodati fragment element u XML datoteku koja deklarise korisnicki interfejs
- Zivotni ciklus fragmenta
  - Slican je zivotnom ciklusu aktivnosti, ali oni sadrze dodatne metode koji omogucavaju interakciju sa aktivnoscu koja ih sadrzi:
    - onAttach (poziva se kada se fragment povezuje sa aktivnoscu)
    - onCreateView (poziva se da bi se iscrtao korisnicki interfejs fragmenta)
    - onActivityCreated (poziva se kada se onCreate metoda aktivnosti izvrsi)
    - onDestroyView (poziva se da bi se unistio korisnicki interfejs fragmenta)
    - onDetach (poziva se kada se fragment odvezuje od aktivnosti)
- Transakcije nad fragmentima
  - U okviru transakcije je moguce izvrsiti sledece operacije:
    - add (dodavanje fragmenta u aktivnost)
    - remove (uklanjanje fragmenta iz aktivnosti)
    - replace (zamena jednog fragmenta drugim fragmentom)
    - hide (skrivanje prikazanog fragmenta)
    - show (prikazivanje skrivenog fragmenta)
    - detach (odvajanje fragmenta od GUI)
    - attach (spajanje fragmenta nakon sto je odvojen od GUI)
- **2. Prava pristupa**
  - Operativni sistem izoluje aplikacije (kako aplikacije medusobno tako i operativni sistem od aplikacija).
  - Dodatne funkcije bezbednosti su implementirane mehanizmom prava pristupa
  - Aplikacija ne moze da izvrsi ni jednu operaciju koja moze da negativno utice na druge aplikacije, operativni sistem ili korisnike ukoliko joj to nije dozvoljeno

    CALL_PHONE - Allows an application to initiate a phone call.
    SEND_SMS - to send SMS messages.
    RECORD_AUDIOS - to record audio.
    CAMERA - to access the camera device.

VIBRATE - the vibrator.
ACCESS_COARSE_LOCATION - to access approximate location derived from network location sources such as cell towers and Wi-Fi.
ACCESS_FINE_LOCATION - precise location from location sources such as GPS, cell towers, and Wi-Fi.
INTERNET - to open network sockets.
BLUETOOTH - to connect to paired bluetooth devices

- Staticka prava pristupa
  - Do Androida 5.1 prava pristupa koja su potrebna za izvrsavanje aplikacije staticki se deklarisu u AndroidManifest.xml.
  - Korisnik moze da aplikaciji prilikom instalacije dodeli prava pristupa koja trazi ili da odustane od instalacije aplikacije
  - Svaki pokusaj da aplikacija izvrsi nedozvoljene operacije bice sprecen
- Dinamicka prava pristupa
  - Od Androida 6.0 aplikacija dinamicki trazi prava pristupa koja su joj potrebna
  - To znaci da aplikacija mora da svaki put pre nego sto izvrsi operaciju koja zahteva pravo pristupa proveri da li ima to pravo pristupa
  - Android moze automatski odobriti aplikaciji pravo pristupa ili moze zatraziti od korisnika da joj odobri pravo pristupa (u zavisnosti od osetljivosti operacije i resursa)
  - Korisnik ima mogucnost da aplikaciji u svakom trenutku oduzme pravo pristupa
- Permisije i nivoi rizika

  U zavisnosti od toga kojim podacima i akcijama ogranicavaju pristup, razlikuju se permisije niskog nivoa rizika i permisije visiokog nivoa rizika.
  - Permisije niskog nivoa rizika se automatski dozvoljavaju. Dovoljno ih je zatraziti u Manifest fajlu. Neke od permisija niskog nivoa rizika su INTERNET, BLUETOOTH, VIBRATE
  - Permisije visokog nivoa rizika se moraju dinamicki zatraziti tj. korisnik ih mora eksplicitno dozvoliti. Primeri permisija visokog nivoa su ACCESS_FINE_LOCATION, ACCESS_COARSE_LOCATION, CALL_PHONE, SEND_SMS, ...

GUI

- **Resursi i konguracije uredaja**
  - Android aplikacija je skup slabo povezanih komponenti
  - Komponente pored klasa mogu da sadrze i resurse (deklaracije GUI, tekst, rastersku i vektorsku grafiku, audio i video klipove, itd.)
  - Resurse treba eksternalizovati da bi se omogucilo prilagodavanje aplikacije razlicitim konfiguracijama uredaja (dimenzije, rezolucija i orijentacija ekrana, jezik, region..) i laksa sinhronizacija izmedu programera i grafickih dizajnera.
- Tipovi resursa

  anim - animacije
  drawable - vektorska ili rasterska grafika
  layout - deklaracije grafickog korisnickog interfejsa
  raw - sirovi podaci (audio i video klipovi)
  values - proste vrednosti (nizovi, boje, eksternalizovani stringovi, stilovi, itd.)
  xml - XML dokumenti
- Eksternalizovanje stringova - navodenje stringa kao resursa u strings.xml
- Svaki resurs identifikovan je tipom i nazivom
- Android generise jedinstveni identifikator svakog resursa (nalazi se u R klasi)
- Resursima se moze pristupiti iz Java koda (R.layout.main, R.string.hello_world) ili iz XML koda (@layout/main, @string/hello_world)
- Konfiguracije uredaja
  - Postoji veliki broj uredaja (sa razlicitim hardverskim karakteristikama) koji koriste Android platformu i veliki

broj verzija Android platforme
- Resursi se mogu definisati za razlicite konfiguracije uredaja (npr. ekran niske, srednje i visoke rezolucije)
- Razlicitim konfiguracijama uredaja odgovaraju resursi koji se nalaze u direktorijumima sa razlicitim sufiksima (ldpi, mdpi, hdpi)
- Moguce je istovremeno definisati resurse za vise tipova konfiguracija (ekran visoke rezolucije u nocnom modu)
language and region - en, fr, en-rUS, fr-rFR, fr-rCA, itd.
screen size - small, normal, large, xlarge
screen orientation - port, land
screen pixel density - ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi, nodpi, tvdpi
UI mode - car, desk, television, appliance, watch
touchscreen type - notouch, finger
night mode - night, notnight
platform version - (API level) v1, v2, v3, itd

- Preporuke
  - Aplikacija bi uvek trebalo da sadrzi podrazumevane resurse odn. resurse koji su nezavisni od konfiguracije
  - Za razlicite rezolucije ekrana bi trebalo pripremiti slike razlicitih rezolucija
  - Za razlicite velicine ili orijentacije ekrana bi trebalo pripremiti razlicite rasporede GUI-a
  - Potrebno je i internacionalizovati stringove da bi se omogucila lokalizacija aplikacije na razlicite jezike
- **GUI :**
  - Home Screen
  - All Apps
  - Overview Space (Recents Screen)
  - Notifications(obavestenja)
    - Poruka koja se prikazuje van korisnickog interfejsa aplikacije (u povrsini za obavestenja)
    - Ne prekida korisnika u izvrsavanju tekuceg zadatka
    - Obicno se prikazuju obavestenja o vremenski kriticnim dogadajima ili dogadajima u kojima ucestvuju drugi ljudi
    - Moguce je i izvrsiti akciju iz obavestenja, za to je potrebno da joj pridruzimo odgovarajucu nameru.
    - Prikazuje se kao ikona u povrsini za obavestenja (notofication area)
    - Vise informacija o obavestenju prikazuje se u fioci za obavestenja (notification drawer)
    - Sadrzi malu ikonu, naslov i tekst
    - android.permission.POST_NOTIFICATIONS.
  - App Screen
  - Toolbar
    - Element GUI-a koji se (obicno) nalazi na vrhu ekrana i obezbeduje:
      branding aplikacije , navigacijuS , promenu pogleda i izvrsavanje akcija
    - Sadrzi: title area (identifikuje aplikaciju)
      navigation area (omogucava navigaciju)
      action area (omogucava izvrsavanje akcija, redje koriscene akcije su prelivene u meni)
    - Moze se podeliti u vise delova: glavni deo (prikazuje ikonu i omogucava navigaciju)
      gornji deo (omogucava promenu pogleda)
      donji deo (omogucava izvrsavanje akcija)
    - Pravljenje toolbar-a:
      - Dodati appcompat i Material Design zavisnost u projekat
      - Dodati toolbar u raspored
      - Definisati klasu koja nasleduje AppCompatActivity klasu i u onCreate() pozvati setSupportActionBar()
      - Koristiti jednu od AppCompat.NoActionBar tema (sprecava koriscenje ugradene ActionBar klase)

- build.gradle, layout_main.xml, ExampleActivity.java, AndroidManifest.java
- Izvrsavanje akcija :
- Za implementaciju dugmadi treba deklarisati meni kao resurs i prikazati ga u onCreateOptionsMenu metodi i reagovati na akcije korisnika u onOptionsItemSelected metodi aktivnosti
- res -> New -> Android Resource File -> Resource Type: Menu
- Up dugme - se prikazuje u toolbar-u kao strelica na levo i sluzi za povratak na prethodnu aktivnost
- Za prikazivanje je potrebno:
U AndroidManifest.xml povezati (child) aktivnost sa drugom (parent) aktivnoscu
U (child) aktivnosti pozvati setDisplayHomeAsUpEnabled metodu i proslediti joj argument true
- Navigation Drawer
- Deklarisati DrawerLayout raspored kao koreni raspored
- Dodati jedan pogled koji sadrzi glavni sadrzaj aktivnosti i drugi pogled (NavigationView) koji predstavlja sadrzaj fioke za navigaciju
- Kreirati resurs (menu) sa stavkama fioke za navigaciju
- Reagovati na akcije korisnika
- Content Area.

- Gestovi
Touch
Long press
Swipe/Drag
Long press drag
Double touch
Double touch drag
Pinch open
Pinch close

- **Pogledi i rasporedi**
- Graficki korisnicki interfejs bilo koje aktivnosti moze se predstaviti hijerarhijom pogleda i rasporeda
- 1. **Pogledi (view)**
- Predstavljaju elemente GUI-a
- Predefinisani tipovi - labele, tekstualna polja, dugmad, itd.
- Moguce je definisati nove tipove pogleda
- Stanje pogleda odredeno je njegovim svojstvima (atributima)
- Postoje svojstva koja su zajednicka za sve tipove pogleda (npr. vidljivost, transparentnost, itd.), a neki tipovi pogleda mogu da sadrze i posebna svojstva
- Pogledi mogu obradivati razlicite dogadaje (dodir, pritisak tastera, promenu fokusa, itd.)
onTouch() - the user performs an action qualified as a touch event (e.g. ACTION_DOWN, ACTION_UP)
onClick() - the user clicks the item
onLongClick() - the user either touches and holds the item
onFocusChange() the user navigates onto or away from the item
- Pogledi se mogu definisati instanciranjem objekata u Java kodu ili dodavanjem elemenata u XML kodu
- Na slican nacin mogu se postaviti svojstva i obradivaci dogadaja pogleda
- Tipovi:
TextView - prikazuje tekst i omogucava njegovo kopiranje
ImageView - prikazuje proizvoljnu sliku iz razlicitih izvora, omogucava i skaliranje, odsecanje, primenu filtera
EditText - omogucava unos teksta, oznacavanje, isecanje, kopiranje, moguce je specificirati i tip tastature
Button - prikazuje tekst ili sliku koja simbolizuju akciju, pritiskom generise se click dogadaj sa onClick

atrib.

RadioButton – izbor jedne opcije iz skupa vise opcija, grupisani su objektom klase RadioGroup

ToggleButton - promena podesavanja izmedu dva stanja

Checkbox – izbor jedne ili vise opcija iz skupa opcija obicno se prikazuju u vertikalnoj listi.

- **2. <u>Rasporedi (layout)</u>**

- Su pogledi koji sadrze druge poglede i odreduju kako se oni rasporeduju na ekranu
- Kao i svaki drugi pogled, moze se definisati proceduralno u Java kodu (instanciranjem klase) ili deklarativno u XML kodu (dodavanjem elementa)
- Svojstva pogleda
  - Slicno CSS box modelu, svaki pogled ima geometriju pravougaonika - Poziciju i dimenzije pogleda odreduje vrsta rasporeda koji ga sadrzi i svojstva pogleda (koja mogu da zavise od vrste rasporeda)
  - Neka svojstva (npr. padding i margin) ne zavise od vrste rasporeda
  - Neka svojstva (npr. layout_width i layout_height) zavise od vrste rasporeda
- Merne jedinice : density-independent pixels, scale-independent pixels, tacka, piksel, milimetar, inc
- Iscrtavanje pogleda
  - Svaki pogled iscrtava sebe i svoju decu
  - Izvrsava se u dva prolaza:
    - prolazu merenja (measure pass)
    - prolazu rasporedivanja (layout pass)
- Vrste :

<u>AbsoluteLayout</u>

<u>GridLayout</u>

<u>FrameLayout</u> - prikazuje vise pogleda koji ce biti rasporedeni jedan na drugom (poslednji koji je dodat bice rasporeden na vrhu)

Velicina rasporeda odgovara velicini najveceg pogleda koga sadrzi (ako to dozvoljava roditelj)

<u>LinearLayout</u> - rasporeduje decu u jednom pravcu (vertikalno ili horizontalno), deca linearnog rasporedena su jedno pored drugog, tako da vertikalni ima jedno dete po vrsti (horizontalni jedno po koloni)

Svojstva:
- layout_weight - assigns value of how much space it should occupy on the screen
- gravity - specifies how an object should position its content, on both the X and Y axis
- orientation - use horizontal for a row, vertical for a column orientation

<u>RelativeLayout</u> - rasporeduje decu relativno u odnosu na sebe i jedno na drugo, pozicija moze se specificirati u odnosu na elemente istog hijerarhijskog nivoa (levo, desno, iznad ili ispod drugog pogleda) ili u odnosu na roditelja (poravnat sa levom, desnom, gornjom ili donjom ivicom)

Svojstva:
- layout_alignParentTop - makes the top edge of this view match the top edge of the parent
- layout_centerVertical - centers this child vertically within its parent
- layout_below - positions the top edge of this view below the view specified with a resource ID
- layout_toRightOf - positions the left edge of this view to the right of the view specified with resource ID

<u>DrawerLayout</u>

<u>ConstraintLayout</u>(ogranicavajuci)
- Raspored koji omogucava odredivanje pozicije i velicine na fleksibilan nacin.
- Odreduju se na osnovu ogranicenja u odnosu na druge poglede, roditeljski raspored ili nevidljive vodice.
- Za ovaj raspored je u build.gradle skripti potrebna zavisnost

<u>CoordinatorLayout</u> -upravlja interakcijom izmedu pogleda koje sadrzi

Obicno se koristi kao koreni raspored aktivnosti ili fragmenta

- **<u>Stilovi</u>**

- Stil je skup svojstava koja specificiraju izgled pogleda

- visina (height), sirina (width), popuna (padding), margina (margin), font (typeface), boja teksta (font color), velicina teksta (font size), boja pozadine (background color), itd.
- definisu se kao poseban resurs (odgovara im XML dokument u res/values/themes direktorijumu) i moguce ih je nasledivati
- style.xml, main_activity.xml

- Boje
  - Bojama je moguce dodeljivati imena i definisati ih kao zaseban resurs.
  - U datoteci res/values/colors.xml se svaka boja zasebno konfigurise preko osnovnih komponenti (RGB ili ARGB).
  - Jednom definisanu boju je moguce koristiti u stilovima, temama ili direktno na pogledima.
  - colors.xml
- Teme
  - Tema je stil primenjen na celu aktivnost ili celu aplikaciju umesto na pojedinacan pogled
  - Tada se na svaki pogled aktivnosti ili aplikacije primenjuje svako svojstvo teme koje pogled podrzava
  - Stilovi su lokalni, teme su globalne
  - res/themes/themes.xml, AndroidManifest.xml,
- Nasledivanje stilova / tema
  - Izmedu stilova se moze uspostaviti hijerarhijski odnos navodenjem roditeljskog stila iz kojeg se preuzimaju sve definicije, koje se mogu dopunjavati ili modifikovati. Isto vazi i za teme.
  - Roditeljski stil/tema se navodi u atributu parent ili se povezivanje vrsi put prostora imena. Na primer, teme "MyApp.MyTheme.Light" i "MyApp.MyTheme.Dark" nasleduju temu "MyApp.MyTheme".
- **Material Design**
  - Skup principa za vizuelni dizajn, dizajn pokreta i dizajn interakcija
  - Aplikacije dizajnirane po ovim principima pruzaju korisnicima konzistentno iskustvo na razlicitim platformama (mobilnim, web i desktop) i u razlicitim aplikacijama
  - Material Design koristi metafore da bi korisnicko iskustvo bilo intuitivno
  - Android podrzava tako sto pruza:
    - nove teme
    - nove poglede (npr. RecyclerView, CardView, itd.)
    - novi API za senke i animacije
  - Dodaje se u build.gradle (module), AndroidManifest.xml
  - Teme : - MaterialComponents tamna verzija
    - MaterialComponents.Light svetla verzija
    - MaterialComponents.Light.DarkActionBar svetla verzija sa toolbar-om
  - Principi preporuka mogu se grupisati u tri kategorije:
    - 1. opipljive povrsine (tengable sufraces) :

      Senke simuliraju visinu listova papira koja odreduje njihov medusobni odnos:
      - seam (dva lista papira koji dele zajednicku ivicu se krecu zajedno
      - step (dva lista papira koji se preklapaju se krecu nezavisno)
      - floating action button (dugme odvojeno od toolbar-a)
    - 2. smeo graficki dizajn (bold graphic design)

      Na listovima se prikazuje:
      - tekst (Roboto i Noto)
      - fotografije, ilustracije i ikonografija (predefinisane ikone za uobicajene akcije)
      - boje (primarna, sekundarna i akcentovana)
    - 3. smisleni pokreti (meaningful motion)
      - autenticni pokreti (pokreti treba da budu uskladeni sa masom, zapreminom i fleksibilnoscu objekta
      - interakcija sa kratkim odzivom (aplikacije reaguju na akcije korisnika i obezbeduju vizuelnu potvrdu)
      - smisleni prelazi (prelazi treba da usmere paznju korisnika i da budu glatki)

- **Adapteri**
  - Povezuju poglede (naslednice AdapterView pogleda) i izvore podataka
  - Postoje predefinisani adapteri koji povezuju razlicite poglede (ListView, GridView, Spinner, itd.) i razlicite izvore podataka (nizove, kolekcije, kursore, itd.)
  - Moguce je napraviti adaptere koji povezuju proizvoljan pogled i proizvoljni izvor podataka
  - ArrayAdapter
    - Povezuje TextView pogled (ili pogled koji sadrzi TextView pogled) i niz ili kolekciju
    - Automatski se poziva toString() metoda svakog objekta u nizu ili kolekciji i njena povrana vrednost se prikazuje u pogledu
    - list_item.xml, activity_main.xml, MainActivity.java
  - CursorAdapter
    - Koristi kursor kao izvor podataka
    - Kursor sadrzi rezultat upita nad bazom podataka (vise o kurzorima na jednom od narednih casova)
  - CustomAdapter
    - Moguce je definisati adapter koji koristi proizvoljan izvor podataka
    - Potrebno je definisati klasu koja nasleduje Adapter ili BaseAdapter i redefinisati njene metode
    - example_adapter.xml , ExampleAdapter.java
  - Spinner
    - Prikazuje stavke u meniju (korisnik moze da izabere jednu stavku iz menija)
    - Stavke se preuzimaju iz adaptera koji je pridruzen pogledu
    - spinner.xml, SpinnerActivity.java
  - ListView
    - Prikazuje listu stavki (koja moze da se skroluje)
    - Stavke se preuzimaju iz adaptera koji je pridruzen pogledu
    - list_view.xml, ListViewActivity.java
  - GridView
    - Prikazuje tabelu stavki (koja moze da se skroluje)
    - Stavke se preuzimaju iz adaptera koji je pridruzen pogledu
    - grid_view.xml, gridview_item.xml, GridViewActivity.java
- **Toasts**
  - Toast je pop-up poruka koja automatski nestaje posle odredenog vremena
  - Korisniku daje povratnu informaciju da je akcija izvrsena Aktivnost na vrhu povratnog steka ostaje vidljiva i u fokusu
  - U novijim verzijama je bolje koristiti Snackbar jer dozvoljava interakciju sa korisnikom

  Snackbar moze sadrzati neku akciju (najvise jednu).
  - Trajanje prikazivanja moze biti Snackbar.LENGTH_SHORT, Snackbar.LENGTH_LONG, Snackbar.LENGTH_INDEFINITE.
  - Snackbar se prikazuje preko korenskog rasporeda aktivnosti. Snackbar se uklanja protekom vremena, klikom na ponudenu akciju ili pozivom dismiss metode
- **Dijalog**
  - Modalni prozor koji prikazuje poruku i (opciono) omogucava unos podataka i potvrdu izvrsavanja akcije
  - Ne zauzima ceo ekran (aktivnost koja prikazuje dijalog se pauzira) Postoje predenisani dijalozi kao sto su: -
    - AlertDialog – sadrzi naslov
      poruku, listu, radio buttons, checkboxes ili proizvoljan raspored
      do tri dugmeta (negativno, neutralno i pozitivno)
    - U projektu se mogu defnisati kao resuris: -> Android Resource File -> Resource Type: Values
    - DatePicker - Za unos datuma
    - TimePicker - Za unos vremena

- Moguce je definisati sopstvene dijaloge (preporucljivo je da se umesto klase Dialog koristi klasa DialogFragment zato sto ona vodi racuna o zivotnom ciklusu dijaloga i omogucava ponovno koriscenje dijaloga)

- **Podesavanja**
  - Za podesavanje aplikacije koristi se Preferece API (da bi bilo konzistentno)
  - Razlicitim tipovima parametrima odgovaraju razliciti tipovi kontrola koje nasleduju Preference klasu
  - Kontrole se mogu grupisati u kategorije ili u podekrane
  - Vrednosti parametara se automatski ucitavaju i snimaju
  - res/xml/preferences.xmlToo

- **Process**
  - Je jedna instanca nekog programa koji se izvrsava
  - Karakterisu ga:
    - angazovanje procesora na izvrsavanju programa
    - upotreba dela operativne memorije koji sadrzi naredbe u masinskom jeziku i podatke na stack-u i heap-u
    - atributi kao sto su: ID, stanje, prioritet, itd.
  - Rasporedivanje niti
    - Razlicite niti mogu da se izvrsavaju na jednom procesoru (konkurentno) ili na vise (paralelno)
    - Kako jedan procesor ne moze istovremeno da izvrsava vise niti, one se moraju izvrsavati naizmenicno
    - S obzirom da razlicite niti mogu da pristupaju istom resursu, potrebno je voditi racuna o sinhronizaciji niti
  - Android i procesi :
    - Kada Android startuje prvu komponentu neke aplikacije, startuje je u novom procesu sa jednom niti
    - Svaka sledeca komponenta iste aplikacije startuje se u istom procesu i u istoj niti kao i prva komponenta - Moguce je startovati razlicite komponente iste aplikacije u razlicitim procesima ili razlicite komponente razlicitih aplikacija u istom procesu (nije preporucljivo)
    - Android zadrzava procese u operativnoj memoriji sto je duze moguce
    - Da bi se oslobodila memorija za procese viseg prioriteta, nekada je potrebno ubiti proces nizeg prioriteta - Prioritet procesa se odreduje na osnovu vrste i stanja komponenti koje sadrzi kao i prioriteta drugih procesa koji od njega zavise
    - Zato bi aktivnosti i prijemnici poruka koji izvrsavaju dugacke operacije trebalo da startuju servis a ne niti
    - Prioritet procesa (opadajuce) : foreground (proces sadrzi aktivnost koja se nalazi u prvom planu)
      visible (proces sadrzi vidljivu ali pauziranu aktivnost)
      service (proces sadrzi servis)
      background (proces sadrzi zaustavljenu aktivnost koja se nalazi u pozadini)
      empty (proces ne sadrzi komponente)

- **Thread(Nit)**
  - Je redosled izvrsavanja naredbi u procesu
  - Jedan proces moze da sadrzi vise niti (svaka nit sadrzi stack, stanje i prioritet i izvrsava relativno nezavisnu sekvencu naredbi)

- Razlika izmedu procesa i niti
  - Niti se koriste za male zadatke, a procesi za velike zadatke (izvrsavanje aplikacije)
  - Niti koje pripadaju istom procesu dele isti adresni prostor (to znaci da mogu da komuniciraju direktno preko operativne memorije)
  - Procesi ne dele isti adresni prostor (to znaci da je komunikacija izmedu procesa slozenija i sporija od komunikacije izmedu niti)
  - Android i niti:

- Android izvrsava aplikaciju (tj. njene komponente) u glavnoj niti
- Ova nit je zaduzena za slanje i primanje poruka od komponenti korisnickog interfejsa (zato se zove i UI nit)
- Stoga nije preporucljivo blokirati UI nit (application isn't responding dijalog) i pristupati komponentama korisnickog interfejsa iz drugih niti (nisu thread-safe)
- Metode zivotnog ciklusa servisa i dobavljaca sadrzaja moraju biti thread-safe

- **Rukovaoci**
  - Red poruka (MessageQueue) je red koji sadrzi poruke koje je potrebno obraditi (zadatke koje treba izvrsiti)
  - Rukovaoc (Handler) obraduje poruke (izvrsava zadatke) koje se nalaze u redu poruka
  - Looper odrzava nit u zivotu i prosleduje poruke (zadatke) iz reda poruka rukovaocu na obradu
  - Za obradu poruka potrebno je implementirati void handleMessage(Message msg) i pozvati:
    boolean sendEmptyMessage(int)
    boolean sendMessage(Message)
    boolean sendMessageAtTime(Message, long)
    boolean sendMessageDelayed(Message, long)

  Za izvrsavanje proizvoljnog koda potrebno je pozvati:
    boolean post(Runnable)
    boolean postAtTime(Runnable, long)
    boolean postDelayed(Runnable, long)

- **Asinhroni zadatak**
  - Do API level 30 za ainhrono izvrsavanje operacija se koristio AsyncTask.
  - Na taj nacin je automatizovano izvrsavanje blokirajuce operacije u pozadinskoj niti, vracanje rezultata u UI nit i neke dodatne funkcije (npr obavestavanje o progresu operacije).
  - Svi asinhroni zadaci jedne aplikacije izvrsavaju se u jednoj niti (oni se serijalizuju)
  - AsyncTask je genericka klasa koja koristi tri tipa:
    - params (tip parametara koji se prosleduju pozadinskoj niti)
    - progress (tip jedinice u kojoj se meri progres operacije)
    - result (tip povratne vrednosti koju vraca pozadinska nit)
  - Od API level 30 se koriste ExecutorService i Executors iz paketa java.util.concurrent.
  - Interfejs ExecutorService definise mehanizme za izvrsavanje zasebnih niti, a klasa Executors kreira instance ovih servisa.
  - Za potrebe komunikacije sa UI niti koristi se klasa Handler.
  - ExecutorService koji koristi jednu nit : Executors.newSingleThreadExecutor();
  - ExecutorService koji koristi vise niti: Executors.newFixedThreadPool(n)

- **Zakazivanje**
  - Tajmer (Timer) zakazuje izvrsavanje jednokratnih zadataka (u apsolutnom trenutku ili posle relativnog kasnjenja) ili zadataka koji se ponavljaju (sa fiksnim periodom ili fiksnom frekvencijom)
  - Svaki tajmer ima jednu nit koja zadatke izvrsava sekvencijalno (to znaci da moze da dode do kasnjenja u izvrsavanju zadatka ukoliko je ta nit zauzeta)
  - Komponenta koja je zakazala izvrsavanje zadatka ne mora biti aktivna u trenutku u kome zadatak treba da se izvrsi (moze da se ne izvrsi)
  - Metode:

schedule(TimerTask task, Date when) - Schedule a task for single execution when a specific time has been reached.
cancel() - Cancels the Timer and all scheduled tasks
scheduleAtFixedRate (TimerTask task, long delay, long period) - Schedule a task for repeated fixed-rate execution after a specific delay has passed.

  - Klasa AlarmManager omogucuje pristup sistemskom alarmu i startovanje aplikacije u nekom trenutku u buducnosti

- Kada se alarm aktivira, sistem emituje objekat klase Intent, sto kao posledicu ima automatsko startovanje aplikacije (ukoliko vec nije)

Deljena podesavanja (SharedPreferences)
- Olaksavaju perzistentno skladistenje prostih tipova podataka i skladiste se u datoteci kao parovi (kljuc, vrednost)
- Moze im se pristupiti metodom SharedPreferences getSharedPreferences(String name, int mode)
- Ova metoda je definisana u klasi Context, pa je samim tim dostupna i u okviru njenih naslednica, kao sto su Activity, Service i IntentService.
- Moguce je koristiti vise skupova deljenih podesavanja cija imena se navode kao parametar (name) :
    MODE_PRIVATE - The created file can only be accessed by the calling application
- U klasi Activity je definisana metoda SharedPreferences getPreferences(int mode) koja omogucava pristup skupu podesavanja te aktivnosti. Za razliku od metode getSharedPreferences, ova metoda koristi podrazumevan naziv skupa podesavanja koji cine naziv paketa i naziv klase u kojoj je aktivnost implementirana.
- Uz koriscenje paketa androidx.preferences na raspolaganju je i pristup podrazumevanom skupu podesavanja: SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
- Upis vrednosti:
    - Vrednosti prostog tipa T mogu se zapisati u tri koraka:
        1 Pozvati edit() metodu koja zapocinje transakciju
        2 Dodati vrednost(i) tipa T metodama oblika SharedPreference.Editor putT(String key, T value)
        3 Pozvati commit() metodu koja zavrsava transakciju u sinhronom modu (i vraca rezultat o uspehu zapisivanja podataka) ili apply() koja to radi u asinhronom modu.
    - U zavisnosti od tipa T, imamo metode: putBoolean, putFloat, putInt, putLong, putString, putStringSet.
    - Ove metode vracaju tip SharedPreference.Editor tako da je olaksano ulancavanje visestrukih upisa vrednost
    - Zapisane vrednosti mogu se procitati metodama oblika T getT(String key, T defaultValue)
    - U zavisnosti od tipa T, na raspolaganju su metode: getBoolean, getFloat, getInt, getLong, getString, getStringSet
    - S obzirom na to da su nazivi skupova podesavanja i nazivi vrednosti predstavljeni stringovima, moze biti korisno njihovo eksternalizovanje (R.string.xyz) jer se u projektu mogu koristiti po vise puta.
- PreferenceActivity -
- Mnoge aplikacije omogucavaju korisnicima konfigurisanje
- U tu svrhu treba koristiti PreferenceFragmentCompat kako bi korisnici imali konzistentan graficki korisnicki interfejs (i da bi sebi olaksali posao) , cijim nasledivanjem definisemo sopstvene fragmente za rad sa podesavanjima.
- Definisan je u paketu koji je potrebno dodati u Gradle skriptu projekta kao zavisnost: 'androidx.preference:preference:'
- Ovakav fragment dinamicki kreira svoj izgled ako pri njegovom kreiranju pozovemo metodu setPreferencesFromResource kojom se ucitava XML resurs sa definisanim podesavanjima
- Postupak kreiranja aktivnosti i fragmenta pomocu wizard-a: File -> New -> Activity -> Settings Activity
- Datoteka sa vrednostima podesavanja: data/data/package_name/shared_prefs
- Alat ADB omogucava pristup fajl sistemima uredaja (fizickim i virtuelnim) koji su povezani na racunar(adb devices)

Datoteke
- Podaci koji se nalaze u operativnoj memoriji se ne cuvaju kada se unisti proces
- Komponente koje se nalaze u razlicitim procesima ne mogu da razmenjuju podatke koji se nalaze u operativnoj memoriji (ne dele adresni prostor)
- Najjednostavniji nacin da se prevazidu ova ogranicenja je koriscenje datoteka
- Za rad sa datotekama koriste se klase iz java.io paketa
- Medutim, mogu se koristiti metode klase Context koje olaksavaju pristup internom i/ili eksternom skladistu podataka, rad sa privremenim datotekama i upravljanje pravima pristupa
    - FileInputStream openFileInput(String name)
    - FileOutputStream openFileOutput(String name, int mode)
    - String[] fileList() // vraca listu datoteka u folderu kojem samo tekuca aplikacija ima pristup (privatan folder)
    - boolean deleteFile(String name) // brise navedenu datoteku u privatnom folderu

- File getDir(String name, int mode) // kreira ili pristupa folderu unutar privatnog foldera
- File getCacheDir() //folder za privremene fajlove
- File getExternalCacheDir() //folder za privremene fajlove na eksternom skladistu
- File getFilesDir() //privatan folder - kome samo tekuca aplikacija ima pristup
- File getExternalFilesDir(String type) // folder kome i druge aplikacije mogu pristupiti

- Vrednosti parametra mode :
  MODE_PRIVATE - file can only be accessed by the calling application
  MODE_APPEND if the file already exists then write data to the end of the existing file instead of erasing it
  MODE_WORLD_READABLE Allow all other applications to have read access to the file
  MODE_WORLD_WRITEABLE Allow all other applications to have write access to the created file
- Interno skladiste podataka se nalazi u mobilnom uredaju
  - Uvek je dostupno
  - Obicno manjeg kapaciteta (nije ga moguce prosiriti)
  - Privatno je
- Eksterno se (obicno) nalazi na SD kartici
  - Nije uvek dostupno
  - Obicno veceg kapaciteta (moguce ga je prosiriti)
  - Javno je
- Privremene datoteke treba skladistiti u cache direktorijumu (automatski se brise kada ponestane slobodnog prostora) :
  File getCacheDir(), File getExternalCacheDir()
- Datoteke koje deli vise aplikacija treba snimiti u javni eksterni direktorijum :
  File getExternalFilesDir(String type), File[] getExternalFilesDirs(String type)
- Pri koriscenju eksternog skladista potrebno je obezbediti staticke (manifest fajl) i dinamicke (putem dijaloga) permisije. Permisije nisu neophodne kada se koriste direktorijumi namenjeni tekucoj aplikaciji odn. direktorijumi koje vracaju metode getExternalFilesDir() i getExternalCacheDir().
- Tip javnog eksternog direktorijuma: DIRECTORY_ALARMS, DIRECTORY_DCIM, DIRECTORY_DCIM, DIRECTORY_DOWNLOADS, DIRECTORY_MOVIES, DIRECTORY_MUSIC, DIRECTORY_NOTIFICATIONS, DIRECTORY_PICTURES, DIRECTORY_PODCASTS, DIRECTORY_RINGTONES

SQLite

- Android aplikacije mogu da koriste ugraden sistem za upravljanje bazama podataka (SQLite)
- Za razliku od vecine SUBP, SQLite se izvrsava u istom procesu kao i aplikacija koja koristi njegove usluge
- Obezbeduje referencijalni integritet i omogucava rad u transakcijama
- sqlite3 : .databases - List attached databases .tables [TABLE] - List tables
  .dump [TABLE] - Dump database to SQL .schema [TABLE] - Show CREATE statements
  .backup [DB] FILE - Backup database to FILE .restore [DB] FILE - Restore database from FILE
  .read FILENAME - Execute SQL from FILENAME .import FILE TABLE - Import data from FILE into TABLE
  .headers on|off - Toggle display of headers .nullvalue STRING - Use STRING for NULL values
  .mode MODE [TABLE] - Set output mode (csv, column, html, insert, line, list) <sql statement> - proizvoljna komanda
- Za pravljenje, izmenu i otvaranje baze podataka koristi se SQLiteOpenHelper klasa
- Potrebno je implementirati neke od sledecih metoda:
  - void onCreate(SQLiteDatabase database)
  - void onOpen(SQLiteDatabase database)
  - void onUpgrade(SQLiteDatabase database, int old_ver, int new_ver)
  - void onDowngrade(SQLiteDatabase database, int old_ver, int new_ver)
- Baza podataka predstavljena je klasom SQLiteDatabase.
- CRUD operacije nad bazom podataka izvrsavaju se pozivom insert, query, update i delete metoda
- Kursori - Relacija koja je rezultat SQL upita predstavljena je kursorom.
  - Koriste za navigaciju kroz rezultat upita: boolean move(int offset), boolean moveToFirst(), moveToLast() , moveToNext() , moveToPrevious()
  - kao i za citanje rezultata upita: int getCount(), int getColumnIndex(String column_name), String getColumnName(int column_index), String getString(int column_index), int getInt(int column_index), long getLong(int column_index) , float getFloat(int column_index), double getDouble(int column_index)

MOBILNE KOMUNIKACIJE

- Radio komunikacije

- Radio talasi - su oscilacije elektromagnetnog polja u vremenu i prostoru. Slabljenje radio talasa zavisi od njegove frekvencije i karakteristika medija kroz koji se prostire
- Radio uradaji : Posiljalac, Predajnik, Antena, Medij, Antena, Prijemnik, Primalac
- Analogni signal - kontinualni signal koji informacije prenosi kao promena amplitude, frekvencije i faze.
- Digitalni signal - diskretan signal koji informacije prenosi kao niz znakova (logickih nula i jedinica).
- Modulacija - proces kojim se signal prilagodava karakteristikama prenosnog medija (transponuje se u podrucje frekvencija pogodnih za prenos radio talasima).
    - Analogni postupci koriste se za molulisanje analognog signala
        - Amplitudna modulacija
        - Frekventna modulacija
        - Fazna modulacija
    - Digitalni postupci koriste se za modulisanje digitalnog signala
        - Pulsna kodna modulacija
        - Delta modulacija

- Celularna mreza

- Celijski sistem - Odredena teritorija je podeljena na celije
- Svakoj celiji je dodeljen skup frekvencija koje su izabrane tako da minimizuju interferenciju sa susednim celijama.
- Skup frekvencija moze se koristiti i u drugim celijama, sve dok te celije nisu susedne.
- Kada mobilni telefon prede iz jedne celije u drugu dok je poziv u toku, mreza ce izdati naredbu mobilnom telefonu da promeni kanal (frekvenciju) i u isto vreme preusmeriti poziv na novi kanal.
- Usluge GSM-a
- Telefonija (prenos govora)
- Short Message Service (SMS)
- Multimedia Messaging Service (MMS)
- Prenos podataka
- SIM (Subscriber Identity Module)
- Pametna kartica koja omogucava mobilnom telefonu da promeni pretplatnika i pretplatniku da promeni telefon.
- Implementira Java Card specifikaciju da bi omogucio interoperabilnost aplikacija.
- Pruza sigurno skladistenje:
    - Integrated Circuit Card Identifier (ICCID) - identifikator kartice
    - International Mobile Subscriber Identity (IMSI) - identifikator pretplatnika
    - Kriptografskog kljuca (Ki) koji se koristi za autentifikaciju pretplatnika.
- Takode skladisti:
    - Personal Identification Number (PIN) - lozinku za uobicajenu upotrebu
    - Personal Unblocking Code (PUK) - lozinku za otkljucavanje PIN-a
    - Service Provider Name (SPN)
    - Local Area Identity (LAI)
    - broj SMSC-a
    - broj za hitne slucajeve
    - spisak uluga kojima pretplatnik moze da pristupi.
- Nudi i dodatne funkcije kao sto je skladistenje telefonskog imenika i tekstualnih poruka.
- **Telefonija**
- Osnovna usluga mobilne mreze je mobilna telefonija (prenos govora). Medutim, iz bezbednosnih razloga nije moguce napraviti in call aktivnost.

-Android API omogucava:

- koriscenje podrazumevane in call aktivnosti za obavljanje telefonskih poziva
- pristup podacima o telefonu (tip, identifikator, verzija softvera, telefonski broj)

- pristup podacima o SIM kartici (stanje, drzava i ime operatora, serijski broj)
- pristup podacima o mrezi (drzava, identifikator operatora, ime mreze, tip mreze)
- pristup podacima o prenosu podataka (stanje i trenutna aktivnost)
- reagovanje na promenu stanja mreze, poziva, lokacija celije, snage signala, aktivnosti i prenosa podataka

- **SMS (Short Message Service)**
  - Je tehnologija koja omogucava slanje i primanje kratkih poruka izmedu mobilnih telefona.
  - Podrzavaju je svi mobilni telefoni.
  - Jedna SMS poruka moze da sadrzi najvise 140 bajtova (160 znakova ukoliko se koristi 7-bitno kodiranje, 70 znakova ukoliko se koristi 16-bitno kodiranje). Pored teksta, SMS poruke mogu da prenose i binarne podatke.
  - PDU mode (protocol data unit) SMS
    - Konkatenirane SMS poruke ili PDU mode SMS poruke mogu da sadrze vise od 140 bajtova.
    - Mobilni telefon posiljaoca deli dugacku poruku u manje delove i salje svaki deo kao pojedinacnu SMS
  - Mobilni telefon primaoca spaja pojedinacne SMS poruke u dugacku poruku.
  - Arhitektura SMS sistema:
    - Short Message Service Center (SMSC) :
      - Upravlja SMS operacijama celularne mreze (njegova glavna funkcija je rutiranje SMS poruka).
      - Kada mobilni telefon posalje SMS poruku, ona stize u SMS centar.
      - SMS centar prosleduje SMS poruku primaocu.
      - Ako je primalac nedostupan (telefon je iskljucen ili nema domet), SMS centar skladisti poruku i prosleduje je primaocu kada postane dostupan.
    - SMS Gateway:
      - Pre nego sto stigne do odredista, SMS moze da prode kroz SMS gateway i druge SMS centre.
      - On omogucava razlicitim operaterima telefonije da povezu SMS centre i razmenjuju SMS.
  - MMS (Multimedia Messaging Service) je prosirenje SMS-a.
    - Omogucava formatiranje teksta i slanje i primanje multimedijalnih poruka (fotografije, audio i video)
  - SMS i Android
    - Postoje dva nacina slanja SMS: putem podrazumevane SMS aplikacije ili putem SMSManager servisa
    - sendDataMessage (sends a data based SMS to a specific application port)
    - sendMultimediaMessage (sends an MMS message)
    - sendMultipartTextMessage (sends a multi-part text based SMS)

- **Networking**

- Umrezavanje mobilnih uredaja je slicno kao i kod racunara i racunarske opreme.
- Bez obzira na tip konekcije za razmenu podataka, na raspolaganju su nam protokoli sa visih OSI nivoa.
- Nekim lokacijama nije potrebno pristupati cesto (slika, audio zapis, video klip i sl.) dok se neke lokacije mogu intenzivno koristiti (npr. mail server, web servisi)

- Web servisi
  - Za udaljeno izvrsavanje operacija nad podacima se obicno koriste REST servisi.
  - Podaci se najcesce prenose u JSON formatu sto olaksava njihovu serijalizaciju/deserijalizaciju.
  - Iako je ovaj nacin komunikacije relativno jednostavan za implementiranje, u praksi se koriste gotove biblioteke koje dodatno olaksavaju koriscenje REST servisa

- Retrofit je HTTP klijent za Android i Javu.
  - Omogucava rad sa sinhronim i asinhronim pozivima, pri cemu je na Android platformi asinhrona varijanta prakticnija jer se na taj nacin ne blokira glavna UI nit.
  - Retrofit biblioteka ne vrsi serijalizaciju i deserijalizaciju JSON objekata, pa je u te svrhe potrebno ukljuciti i druge biblioteke u projekat.
  - Da bi smo koristili potrebno je da najpre instanciramo objekat klase Retrofit putem Builder-a.

  Builder-u smo setovali: osnovni URL (na koji ce se dodavati relativne putanje definisane u interfejsu endpoint-a)
  - Omogucava kreiranje objekta koji reprezentuje servis i pozivanje metoda koje smo deklarisali.

  Nad ovim objektom kreiramo poziv neke od metoda:

- Objekat tipa Call reprezentuje poziv ka servisu i dozvoljava sinhrono i asinhrono izvrsavanje.
- Za sinhrono izvrsavanje se koristi metoda execute().
- Za asinhrono izvrsavanje se koristi metoda enqueue(). Kada je potrebno da se neka graficka datoteka sa veba ucita u pozadini korisno je upotrebiti Picasso biblioteku.

- Model podataka
  - Za konverziju izmedu JSON i Java objekata potrebno je u projektu definisati model podataka.
  - Ovaj model predstavljaju POJO klase sa odgovarajucim anotacijama koje povezuju atribute ovih klasa sa odgovarajucim poljima u JSON objektima.
- Endpoint
  - Nakon sto smo povezali Java model sa JSON modelom potrebno je definisati endpoint pomocu metoda u Java interfejsu.
  - To se takode postize pomocu anotacija uz metode i njihove parametre tako sto oznacavaju na koji nacin ucestvuju u HTTP request-u i response-u.
  - Pri tome se putanje do endpointa navode u relativnom zapisu dok se osnovni deo URL-a setuje pri inicijalizaciji Retrofit instance.
  - Anotacije: @GET @POST @PUT @DELETE @Path @Query @Body @Header @Headers

SENZORI

- Fizicke velicine opisuju svojstva materije i fizickih pojava
  - Mogu biti skalarne (temperatura, vlaznost vazduha, vazdusni pritisak), vektorske (pozicija, brzina, ubrzanje), ..
- Merenje je proces uporedivanja nepoznate fizicke velicine sa poznatom fizickom velicinom
  - Postoji standardna merna jedinica za svaku fizicku velicinu Postoje osnovne merne jedinice (duzina, masa, vreme, elektricna struja, temperatura, kolicina supstance i jacina svetlosti) i izvedene merne jedinice
- Senzor je uredaj koji pretvara jednu fizicku velicinu u drugu fizicku velicinu koju covek moze neposredno da opazi (ili koju racunar moze da ocita)
- Digitalizacija
  - uzorkovanje (ocitavanje vrednosti analognog signala (obicno sa konstantnom frekvencijom)
  - kvantizacija (aproksimacija ocitane vrednosti sa vrednostima iz konacnog skupa)
- Senzorski koordinatni sistem
  - x osa (horizontalna, od levo prema desno)
  - y osa (vertikalna, od dole prema gore)
  - z osa (od uredaja)
- Tipovi:
  - ACCELEROMETER Meri ubrzanje uredaja (sa g)
  - AMBIENT_TEMPERATURE Meri temperaturu vazduha
  - GRAVITY Meri g
  - GYROSCOPE Meri ugaonu brzinu uredaja
  - LIGHT Meri jacinu svetlosti
  - LINEAR_ACCELERATION Meri ubrzanje uredaja (bez Zemljinog g)
  - MAGNETIC_FIELD Meri jacinu magnetnog polja
  - PRESSURE Meri vazdusni pritisak
  - PROXIMITY Meri udaljenost objekta od ekrana
  - RELATIVE_HUMIDITY Meri relativnu vlaznost vazduha
  - ROTATION_VECTOR Meri orijantaciju uredaja
- Sensors API :
  - SensorManager Omogucava pristup senzorima
  - Sensor Sadrzi informacije o svojstvima odredenog senzora
  - SensorEvent Dogadaj koji sadrzi informacije o odredenom merenju
  - SensorEventListener Sadrzi obradivace SensorEvent dogadaja

1 Zatraziti prava pristupa (staticki ili dinamicki)

2 Odrediti koji senzori su dostupni na uredaju
3 Odrediti mogucnosti dostupnih senzora
4 Napisati obradivace dogadaja koji reaguju na promenu fizicke velicine ili tacnosti merenja
5 Registrovati i odregistrovati obradivace dogadaja
- Metode klase Sensor:
    - float getMaximumRange() - maksimalan raspon izmerenih vrednosti
    - int getMinDelay() - minimalan period izmedu dva merenja
    - float getResolution() - rezolucija senzora
    - float getPower() - potrosnja
    - String getName() - ime senzora
    -int getType() - genericki tip senzora
    - String getVendor() - proizvodac senzora
    - int getVersion() - verzija senzora
- Metode interfejsa SensorEventListener:
    - onSensorChanged(SensorEvent event) - obradjuje promenu fzicke velicine
    - onAccuracyChanged(Sensor sensor, int accuracy) - obraduje promenu tacnosti merenja
- Atributi klase SensorEvent:
    - float[] values - izmerena vrednost (skalar ili vektor)
    - long timestamp - vreme merenja [ns]
    - int accuracy - tacnost merenja
    - Sensor sensor - koriscen senzor
- Parametri metode registerListener:
    - listener - obradivac dogadaja
    - sensor = senzor
    - samplingPeriodUs - period uzorkovanja
- Dobra praksa: Koristiti Google Play filtere za izbor uredaja sa odgovarajucim tipovima senzora ili detektovati senzore u toku izvrsavanja aplikacije i po potrebi o(ne)moguciti odredene funkcije. Odregistrovati obradivac dogadaja kada senzor vise nije potreban (stedi bateriju)