UNIVERZITET U NIŠU

ELEKTRONSKI FAKULTET

Katedra za računarstvo

# Forenzika mobilnih uredjaja

## - kreiranje aplikacije koja ekstraktuje podatke o SMS porukama-

Mentor: prof. dr Bratislav Predić

Student: Mila Rančić 1370

Niš, 2023

# Sadržaj

# Uvod

U današnje vreme, razvoj tehnologije doprineo je povećanoj upotrebi mobilnih telefona. Gotovo da ne postoji čovek koji ne poseduje bar jedan telefon. Prvobitna namena ovih uređaja bila je obavljanje komunikacije putem poziva ili slanjem/primanjem SMS (**Short Message Service**) poruka. Danas, oni poseduju mnoštvo drugih funkcionalnosti. Koriste se za pristup internet sadržajima, imaju različite multimedijalne funkcije kao što su digitalna kamera, snimanje zvuka, navigacija i sl, skladištenje podataka. Svi naši podaci počev od datuma rođenja, mesta stanovanja, obrazovanja, medicinske dokumentacije do dnevnika i bankovnih evidencija računa, praktično ceo naš život je uskladišten negde u digitalnom obliku. Sve ovo obezbeđuje veliku količinu informacija o vlasniku, njegovim navikama i interesovanjima. Zbog toga su ovi uređaji često meta za zlonamerne aktivnosti (digitalne zločine i napade). Pored toga, pomenuti podaci mogu se iskoristiti za razne procedure i istrage, od privatnih, korporativnih do krivičnih. Upravo je to, ono čime se bavi digitalna forenzika.

Digitalna forenzika se definiše kao prikupljanje, zaštita i analiza dokaza u digitalni oblik i njihovo predstavljanje u eventualnim sudskim postupcima. Sam postupak dobijanja informacija se naziva istraga. Digitalna forenzika se ne koristi samo u slučaju kriminalnih radnji, već u bilo kojim slučajevima kada je potrebna manipulacija digitalnim podacima.

Sa porastom značaja mobilnih uređaja, kao posebna grana digitalne forenzike izdvojila se forenzika mobilnih uređaja. Ona svoje podatke dobija upravo iz mobilnih uređaja, ali i tableta, pametnih satova, kamera, GPS uređaja ili dronova. Forenzika mobilnih uređaja obezbeđuje niz alata koji omogućavaju pribavljanje informacija iz oštećenih uređaja, greškom ili namerno obrisanih podataka. Podaci mogu da budu npr. SMS poruke, kontakti, poruke elektronske pošte, dokumenti, fotografije, video zapisi itd. Oni se zapravo ne brišu iz memorije, samo nisu vidljivi korisnicima.

Jedan od najvećih problema koji otežava prikupljanje podataka je zaštita koja postoji na uređaju. Različiti softver i različiti operativni sistemi imaju različita šifrovanja, bilo da je u pitanju šifrovanje celog diska, šifrovanje zasnovano na datotekama ili uređaji zaštićeni lozinkom.

U ovom radu biće obrađena forenzika mobilnih uređaja sa Android operativnim sistemom. U okviru prvog dela upoznaćemo se sa strukturom datoteka Android sistema, kao i načinom na koji se podaci skladište na uređaju.

Zatim će biti obrađeno šta je ADB, odnosno kako se uspostavlja komunikacija sa mobilnim telefonom i objašnjeno, korak po korak postupak povezivanja.

Pošto će u praktičnom delu biti korišćen Content Provider za pribavljanje podataka, u trećem poglavlju biće prikazano šta je on, kako funkcioniše, kako mu se pristupa.

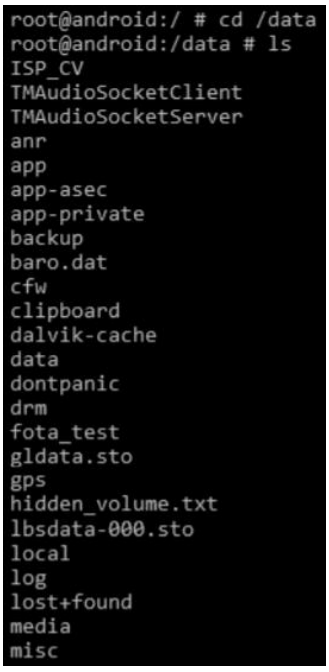
U okviru praktičnog dela, kreirana je android aplikacija koja pristupa content provider-u mobilnog telefona. Ona prikuplja podatke i omogućava korisniku aplikacije da vrši analizu i manipulaciju nad tim podacima.

# Struktura datoteka Android sistema

U cilju što boljeg obavljanja forenzičke analize, odnosno dobijanja željenih podataka sa uređaja, potrebno je znati sledeće: koji podaci se čuvaju na uređaju, gde se čuvaju, kako oni čuvaju detalje o sistemima datoteka na kojima su podaci uskladišteni. Odgovore na ova pitanja je bitno znati pre vršenja analize, jer se na taj način sužava potraga i određuje tehniku koju je najbolje primeniti za pribavljanje podataka. Zato je najbolje prvo se upoznati sa izgledom Android particija. Particije su način logičke organizacije podataka u memoriji uređaja i omogućavaju logičku alokaciju prostora za skladištenje na delove kojima se može pristupiti nezavisno jedan od drugog. Izgled particije varira između dobavljača i verzije.

Najčešće Android particije koje se nalaze na većini mobilnih uređaja su:

1. /boot: Kao što ime govori, ova particija sadrži potrebne podatke i datoteke za pokretanje operativnog sistema mobilnog uređaja. Sadrži jezgro i RAM (engl. Memorija sa slučajnim pristupom), a bez ove particije mobilni uređaj ne može da pokrene procese.
2. /recovery: Particija za oporavak, dozvoljava pokretanje operativnog sistema mobilnog uređaja u konzoli za oporavak, a uz pomoć konzole obavljaju se aktivnosti kao što su ažuriranja telefona i druge vrste održavanja.
3. /system: Sve glavne komponente osim kernela i RAM-a su sačuvana na ovoj particiji. Sadrži Android biblioteke, sistemske binarne datoteke i unapred instalirane aplikacije. Bez ove particije, uređaj ne može da se pokrene u normalnom režimu.
4. /data: Ova particija se obično naziva particijom podataka i tu su uskladišteni podaci svih aplikacija. Upravo zbog toga je ona od najvećeg značaja prilikom forenzičke analize.




*Slika 1: Sadržaj /data particije*

5. /cache: Koristi se za skladištenje podataka kojima se često pristupa i raznim podacima drugih datoteka, kao što su evidencije oporavka i paketi ažuriranja koji se preuzimaju preko mobilnih mreža. Podaci koji se ovde nalaze su takođe od velikog značaja za forenzičare, jer su možda izbrisani iz *data* particije.
6. /misc: Particija koja sadrži informacije o raznim podešavanjima. Ova podešavanja uglavnom definišu stanje uređaja, uključeno/isključeno, zatim informacije o podešavanjima hardvera, USB podešavanjima...
7. /sdcard: Sadrži sve informacije prisutne na Secure Digital (SD) kartici. Podaci koji se ovde obično čuvaju su npr slike, video zapisi, datoteke, dokumenti itd.

Sada kada smo razumeli stukturu Android datoteka, potrebno je da se upoznamo i sa načinom na koji Android sistem čuva podatke na uređajima.

## Čuvanje podataka na Android uređajima

Android koristi sistem datoteka koji pruža nekoliko opcija za čuvanje podataka aplikacije. Različite aplikacije smestaju podatke na različitim lokacijama na Android uređajima. Prilikom forenzičke analize bitno je znati gde sve aplikacije mogu da smeste svoje podatke. Oni često sadrže obilje korisnickih podataka koji mogu biti relevantni za istragu.

Podaci aplikacije mogu da se čuvaju na jednoj od sledećih lokacija:

- **Preferences**: Ovde se podaci čuvaju u parovima ključ/vrednost u XML formatu i privatnog su tipa. Mogu se naći u folderu ***shared_pref*** u ***data*** direktoriju.
- **App-specific storage**: Podaci se smeštaju u internoj memoriji uređaja. Privatni su i sadrže osetljive informacije kojima druge aplikacije ne smeju da pristupaju.
- **Deljena memorija**: Ovde se čuvaju podaci koji su javni (medijske datoteke, dokumenti i slike) i nalaze se u eksternoj memoriji uređaja.
- Razlikujemo dva tipa eksterne memorije:
    - Ugrađena memorija koja je poznata kao primarna eksterna memorija - to je kada ne postoji spoljni fizički uređaj za skladištenje i
    - SD kartica.
- **SQLite baza podataka**: Podaci smešteni u SQLite bazama mogu se naći na lokaciji /data/data/PackageName/database. Obično se čuvaju sa ekstenzijom datoteke .db. Mogu se videti pomoću SQLite pretraživača (https://sourceforge.net/projects/sqlitebrowser/) ili izvršavanjem potrebne SQLite komande za odgovarajuće datoteke.

Svaka Android aplikacija skladišti podatke na uređaju koristeći jedan ili više prethodno pomenutih opcija skladištenja podataka. Aplikacije rade u zaštićenom okruženju sa ograničenim pristupom, pa su potrebne posebne dozvole za pristup podacima.

## Dozvole

Ako aplikacija treba da koristi resurse ili informacije izvan sopstvenog zaštićenog okruženja, moguće je zatražiti od korisnika dozvolu koja obezbeđuje ovaj pristup, u toku izvršavanja.

Mnoge dozvole za vreme izvršavanja pristupaju privatnim korisničkim podacima, posebnoj vrsti ograničenih podataka koji uključuju potencijalno osetljive informacije. Primeri privatnih korisničkih podataka uključuju lokaciju i kontakt, zatim korišćenje mikrofona i kamere...

U datoteci manifest aplikacije potrebno je da se navedu sve dozvole koje će aplikacija možda morati da zatraži od koisnika. Pre traženja dozvole, vrši se provera da li je korisnik već dao toj aplikaciji određenu dozvolu pomoću funkcije ***checkSelfPermission***. Ovaj metod vraća ili PERMISSION_GRANTED ili PERMISSION_DENIED, u zavisnosti od toga da li aplikacija ima dozvolu. Ako nema, poziva se metoda ***requestPermissions*** čime se zahteva odobrenje dozvola.

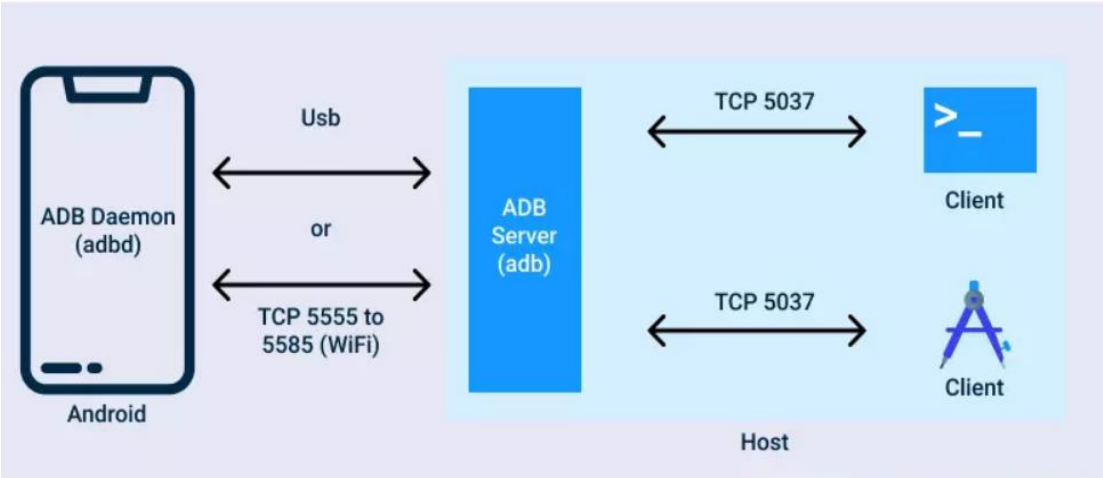
# ADB

## Šta je ADB?

Android Debug Bridge (ADB) je alat koji olakšava komunikaciju između razvojnog okruženja i uređaja. Služi za pokretanje i otklanjanje grešaka u aplikacijama, slanje i primanje datoteka između dva povezana uređaja...

ADB je klijent-server program koji uključuje tri komponente:

1. Klijent, šalje komande. Radi na razvojnoj mašini. Može mu se pristupiti iz terminala, izdavanjem adb komande.
2. Demon (adbd), pokreće komande na uređaju. Radi kao pozadinski proces na svakom uređaju.
3. Server, upravlja komunikacijom između klijenta i demona. Radi kao pozadinski proces na mašini za razvoj.




*Slika 2: Komponente ABD klijent-server programa*

Kada se pokrene adb klijent, on prvo proverava da li je već pokrenut proces adb servera. Izvršavanjem komande ***adb devices***, klijent može videti sve povezane uređaje.

```
Interactive Session
$ adb devices
List of devices attached
emulator-5554      device
$
```

*Slika 3: Pokretanje komande adb devices (server je aktivan)*

Ako ne postoji, pokreće se proces servera. Kada se server pokrene, on se vezuje za lokalni TCP port 5037 i osluškuje komande koje šalju klijenti. Server zatim postavlja veze sa svim pokrenutim uređajima.

```
Interactive Session
$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
emulator-5554      device
$
```

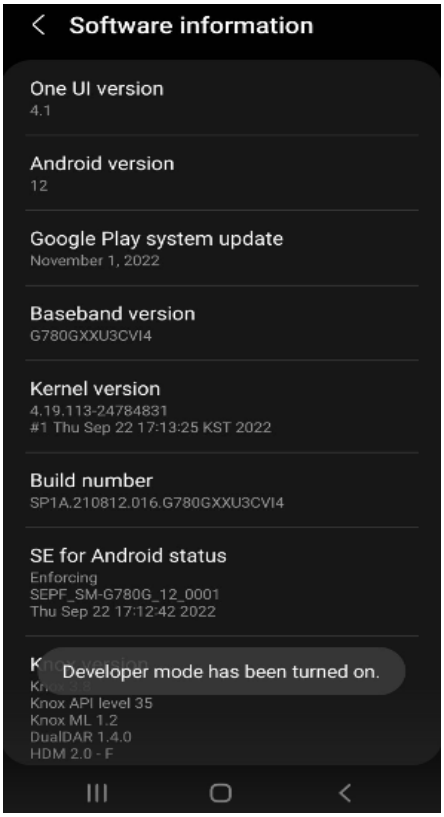
*Slika 4: Pokretanje komande adb devices (server nije aktivan)*

Nakon toga mogu se koristiti adb komande za pristup tim uređajima, uspostavljanjem komunikacije sa odgovarajućim demonom.

U ovom radu komunikacija je uspostavljena između Android Studio-a i fizičkog Android uređaja. ADB se koristi i uključen je u paket alata za Android SDK platformu. Preuzimanjem paketa pomoću SDK Manager-a, instalira se na lokaciji android_sdk/platform-tools/.

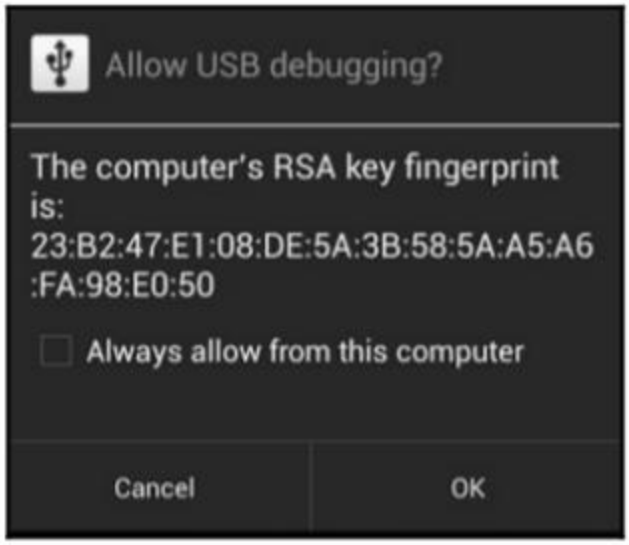
## Podešavanje Android uređaja za rad sa ADB-om

Kako bi se omogućila ekstrakcija podataka sa mobilnog uređaja, potrebno je prvo uspostaviti komunikaciju između Android Studia-a i uređaja. Prvi korak je uključivanje opcije USB Debugging. U ranijim Android verzijama, ova opcija se nalazila u ***Podešavanjima***. Međutim, počevši od Android verzije 4.2, meni sa opcijama za programere je skriven da bi se onemogućilo da mu korisnici slučajno pristupe. Lokacija pomenute opcije varira od proizvođača do proizvođača, kao što variraju i njihovi interfejsi. U slučaju Samsung telefona, koji je korišćen u praktičnom delu, opcija se nalaze na sledećoj lokaciji ***podešavanja/o telefonu/informacije o softveru***. Tu je potrebno pronaći ***Broj verzije (Build version)***. Nakon što se sedam puta pritisne ova opcija, pojaviće se poruka da je uključen ***Developer mode***.




*Slika 5: Uključivanje developer mod-a*

Nakon toga, u podesavanjima se pojavljuje dodatan meni ***Developer options***. Ovde je moguće uključiti USB Debugging. Kada se uređaj poveže sa novom radnom stanicom preko USB-a da bi mu pristupali, potrebno je prvo da se uređaj otključa i odobri pristup.




*Slika 6: Prozor koji traži odobravanje za USB Debugging*

Pristup se odobrava pritiskom na OK u prozoru koje se otvorio, slika 6. Ako se označi *Uvek dozvoli sa ovog računara*, uređaj neće tražiti autorizaciju u budućnosti.

Kada je povezivanje završeno, može se izvršiti otkrivanje povezanih uređaja, a ujedno i provera da li je okruženje lepo podešeno. Ako je uređaj povezan, a komanda ne vrati ni jednu vrednost, to znači da nije i potrebno je ponovo ispratiti korake i instalirati potrebne drajvere. Ukoliko jeste računar detektuje telefon. Nakon toga, moguća je komunikacija sa telefonom.

```
C:\android-sdk\platform-tools>adb.exe devices
List of devices attached
4df16ac3115e5f05        device
```

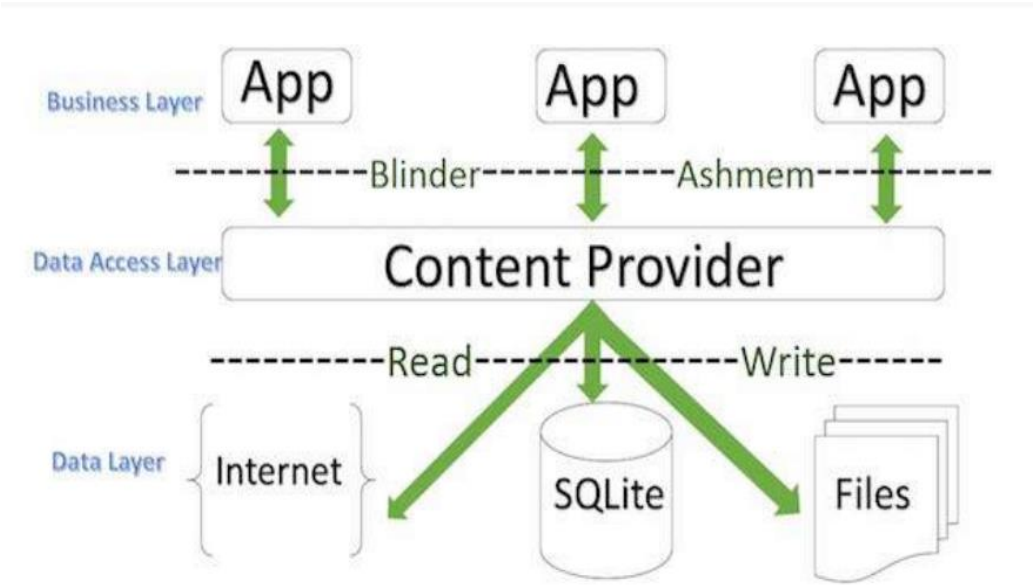
*Slika 7: Otkrivanje povezanih uređaja*

## Razmena podataka

Ekstraktovanje podataka može se vršiti root-ovanjem telefona ili pomoću Content Provider-a. Rutovanje je proces kojim korisnici Android uređaja mogu da ostvare privilegovanu kontrolu (poznatu kao root pristup) nad različitim podsistemima uređaja, obično pametnim telefonima. Rutovanje se često izvodi sa ciljem da se prevaziđu ograničenja koja nosioci i proizvođači hardvera postavljaju na neke uređaje. Dakle, daje mogućnost (ili dozvolu) da se izmene ili zamene sistemske aplikacije i podešavanja, pokreću specijalizovane aplikacije koje zahtevaju dozvole na nivou administratora ili obavljaju druge operacije koje su inače nedostupne normalnom Android korisniku. Drugi metod, koji je korišćen u praktičnom delu biće detaljnije objašnjen u narednom poglavlju.

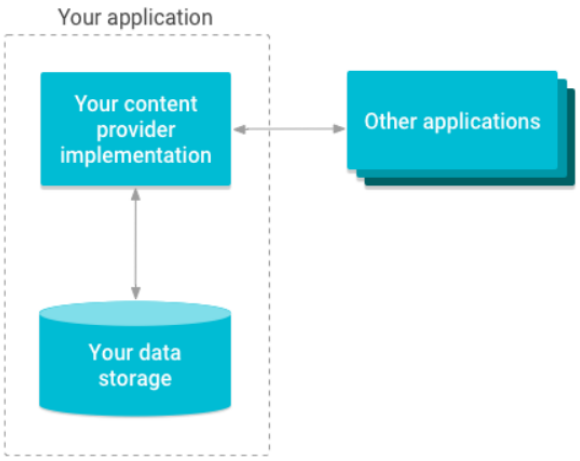
# Content Provider

U Androidu, Content Provideri su veoma važne komponente aplikacija, kako za nju, tako i za druge. Uloga Provider-a je poput centralnog skladišta u kome se čuvaju podaci aplikacije i omogućava drugim aplikacijama da bezbedno pristupe i menjaju te podatke na osnovu zahteva korisnika. Android sistem omogućava Content Provider-u da skladišti podatke aplikacije na nekoliko načina. Korisnici mogu da upravljaju skladištenjem podataka aplikacije kao što su slike, audio, video zapisi i lične kontakt informacije tako što će ih čuvati u SQL bazi podataka, u datotekama ili čak na mreži.



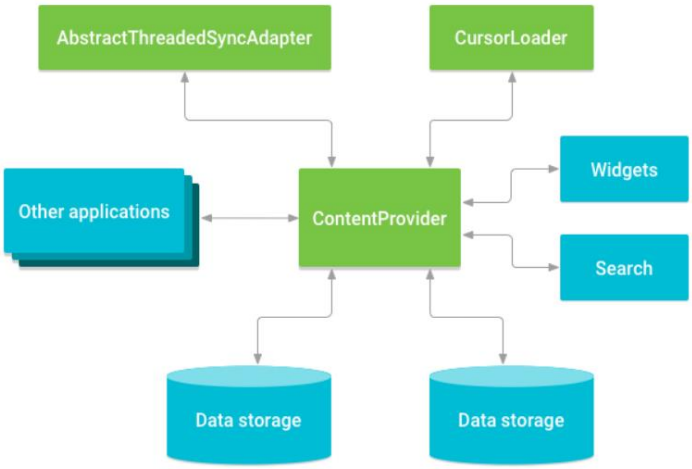

*Slika 8: Prikaz Content Providera kao sloja između aplikacija i podataka*

Content Provider omogućava aplikaciji da upravlja pristupom podacima koje sama čuva, koje čuvaju druge aplikacije i obezbeđuje način za deljenje podataka sa drugim aplikacijama. Takođe, inkapsulira podatke i obezbeđuje mehanizme za definisanje bezbednosti podataka. Predstavljaja standardni interfejs koji povezuje podatke u jednom procesu sa kodom koji se pokreće u drugom procesu. Upotreba content provider-a ima mnogo prednosti. Najvažnija je mogućnost konfigurisanja provider-a tako da omogući drugim aplikacijama da bezbedno pristupaju i menjaju podatke aplikacije kao što je ilustrovano na slici.




*Slika 9: Pristup jedne aplikacije podacima druge pomoću Content provider*

Koordinira pristup sloju za skladištenje podataka u aplikaciji za brojne različite API-je i komponente kao što je ilustrovano na slici 10, a one uključuju:




*Slika 10: Odnos content providera sa ostalim komponentama i API*

- Other applications - Deljenje pristupa podacima aplikacije sa drugim aplikacijama
- Widgets - Slanje podataka u widget
- Search – Vraćanje predloga prilikom pretrage pomoću SearchRecentSuggestionsProvider
- Sinhronizacija podataka aplikacije sa serverom korišćenjem AbstractThreadedSyncAdapter
- Učitavanje podataka u korisnički interfejs pomoću CursorLoader-a

## Pristup provajderu

Content Provider isporučuje podatke iz jedne aplikacije u druge na zahtev. Predstavlja podatke spoljnim aplikacijama kao jednu ili više tabela koje su slične tabelama koje se nalaze u relacionoj bazi podataka. Red predstavlja instancu neke vrste podataka koje provajder prikuplja, a svaka kolona u redu predstavlja pojedinačni deo podataka prikupljenih za instancu.

Za pristup podacima content provider-a, koristi se objekat ContentResolver-a koji se ponaša kao klijent. Metode ContentResolver obezbeđuju osnovne funkcije „CRUD“ (kreiranje, preuzimanje, ažuriranje i brisanje) trajnog skladištenja.

ContentResolver objekat komunicira sa objektom provider-a, koji je instanca klase koju implementira ContentProvider. Ovaj objekat prima zahtev za podacima od kijenata, zatim izvodi zahtevanu akciju i vraca nazad dobijene rezultate. Pozivanjem njegovih metoda zapravo dolazi do poziva identicno nazvanih metoda instance jedne od konkretnih podklasa ContentProvider-a, cime se generisu trazeni rezultati.

Funkcija za slanje zahteva Content Provideru je sledeća:

*getContentResolver().query(Uri, projection, selection, selectionArgs, sortOrder)*

1. Uri – obuhvata simboličko ime provider-a (njegovo ovlašćenje) i ime koje ukazuje na tabelu. Objekat ContentResolver-a analizira ovlašćenje i koristi ga za upoređivanje sa sistemskom tabelom poznatih provider-a. Nakon pronalaska odgovarajućeg, može da pošalje argumente upita ispravnom provider-u. ContentProvider koristi deo putanje URI sadržaja da bi izabrao tabelu kojoj treba da pristupi.
   ***primer: content://user_dictionary/words***
   - String content:// (šema) - je uvek prisutan i identifikuje ga kao URI sadržaja.
   - string user_dictionary - ovlašćenje provider-a,
   - a words - je putanja tabele
2. projection je niz kolona koje treba uključiti za svaki preuzet red.
3. selection određuje kriterijume za izbor redova.
4. sortOrder specificira redosled kojim se redovi pojavljuju u vraćenom Cursor-u.

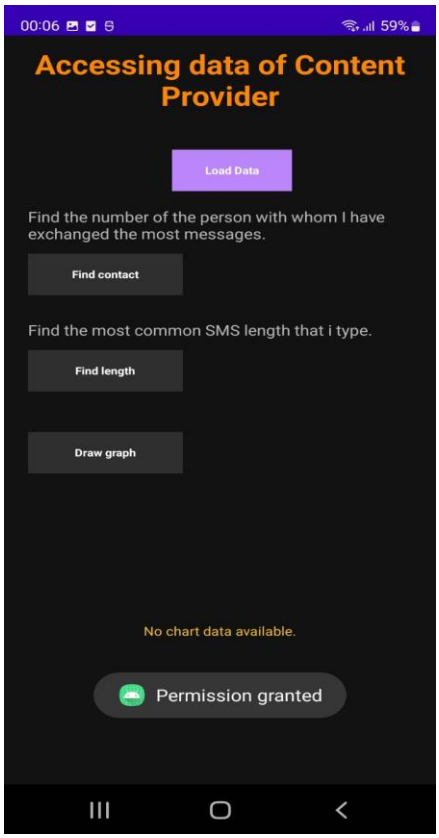
Prilikom podele podatka, Content Provideri imaju određene dozvole koje se koriste za davanje ili ograničavanje prava drugim aplikacijama. Zbog toga je prilikom preuzimanja podataka od provider-a, u okviru aplikacije neophodno da se navedu zahtevi za potrebne dozvole, u manifest fajlu, koristeći element <uses-permission> i tačno ime dozvole koje je definisao provider.

## Prikaz rezultata upita

Klijentski metod ContentResolver.query() uvek vraća Cursor koji sadrži kolone navedene u projekciji upita za redove koji odgovaraju kriterijumima za izbor upita. Cursor objekat pruža nasumični pristup za čitanje redovima i kolonama koje sadrži. Koristeći metode Cursor-a, moguće je iterirati po redovima u rezultatima, odrediti tip podataka svake kolone, izvaditi podatke iz kolone i ispitati druga svojstva rezultata. Neke implementacije Cursor-a automatski ažuriraju objekat kada se podaci provider-a promene.

# Praktični deo

U okviru praktičnog dela kreirana je android aplikacija koja će da komunicira sa content provider-om android mobilog telefona, preuzima poruke i vrši manipulaciju nad njima, kao i prikaz rezultata.




*Slika 11: Izgled android aplikacije*

Na slici 11 prikazan je izgled aplikacije pre učitavanja podataka. Prvo se od korisnika zahteva da odobri, da dozvolu, za pristup sms porukama, a nakon toga mu se prikazuje ***Permission granted***.

```
final int REQUEST_CODE_ASK_PERMISSIONS = 123;
ActivityCompat.requestPermissions( activity: MainActivity.this, new String[]{"android.permission.READ_SMS"}, REQUEST_CODE_ASK_PERMISSIONS);
if(ContextCompat.checkSelfPermission(getBaseContext(),  permission: "android.permission.READ_SMS") == PackageManager.PERMISSION_GRANTED) {
    Toast.makeText(getBaseContext(),  text: "Permission granted",Toast.LENGTH_SHORT ).show();
}
```

*Slika 12: Deo koda koji traži dozvolu od korisnika*

Aplikacija sadrži 4 dugmeta. Dugme ***Load Data***, poziva funkciju *loadAllSms* i koristi se za učitavanje poruka (primljenih i poslatih). Ovi podaci će u nastavku biti korišćeni za daje analiziranje.

```
public void loadAllSms() {
    List<Sms> inbox = new ArrayList<Sms>();
    List<Sms> sent = new ArrayList<Sms>();

    Sms objSms;
    Uri message = Uri.parse( uriString: "content://sms/");
    ContentResolver cr = getContentResolver();

    String[] projection = new String[]{"_id","address","body","date","type"};
    String searchQuery = "date >= " +1641042001000L+ " and date < "+ 1672534800000L;

    Cursor c = cr.query(message, projection,  searchQuery,  selectionArgs: null,  sortOrder: "date ASC");

    int totalSMS = c.getCount();
    if (c.moveToFirst()) {
        for (int i = 0; i < totalSMS; i++) {

            objSms = new Sms();
            objSms.setId(c.getString(c.getColumnIndexOrThrow( s: "_id")));
            objSms.setAddress(c.getString(c.getColumnIndexOrThrow( s: "address")));
            objSms.setMsg(c.getString(c.getColumnIndexOrThrow( s: "body")));

            objSms.setTime(c.getString(c.getColumnIndexOrThrow( s: "date")));
            if (c.getString(c.getColumnIndexOrThrow( s: "type")).contains("1")) {
                objSms.setFolderName("inbox");
                inbox.add(objSms);
             } else {
                objSms.setFolderName("sent");
                sent.add(objSms);
             }
             c.moveToNext();
        }
        allSms.put( k: "inbox",inbox);
        allSms.put( k: "sent",sent);
        findContactButton.setEnabled(true);
        drawGraphButton.setEnabled(true);
        findMostCommonSmsLengthButton.setEnabled(true);
        Toast.makeText(getBaseContext(),  text: "All sms loaded",Toast.LENGTH_SHORT ).show();
    }
    // Closes the Cursor, releasing all of its resources and making it completely invalid.
    c.close();
}
```

*Slika 13: Prikaz loadAllSms funkcije za učitavanje poruka*

Na slici 13, možemo da vidimo da se prvo šalje zahtev content provider-u android mobilnog telefona i prosledjuju mu se parametri.

Na sledećoj slici date su kolone koje poseduje tabela sa SMS porukama.

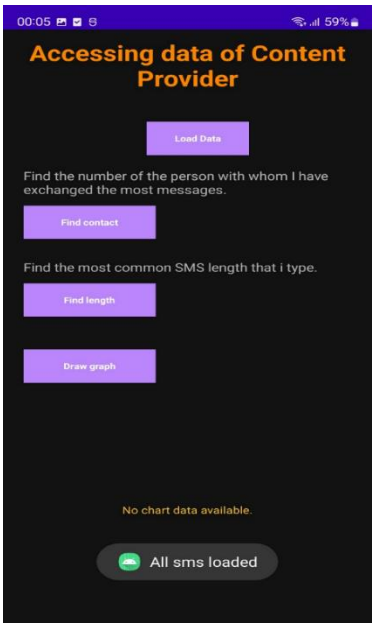
```
"service_center_address",
"address",
"message_class",
"body",
"date",
"status",
"index_on_icc",
"is_status_report",
"transport_type",
"type",
"locked",
"error_code",
"_id"
```

*Slika 14: Kolone SMS tabele*

Prilikom ekstrakcije podataka nije neophodno pribaviti sve kolone, ukoliko nam nisu sve potrebne, već je moguće specificirati samo one čiji podaci su nam od značaja. Zbog toga je pored URL-a, definisan i projection gde se zahtevaju pojedine kolone.

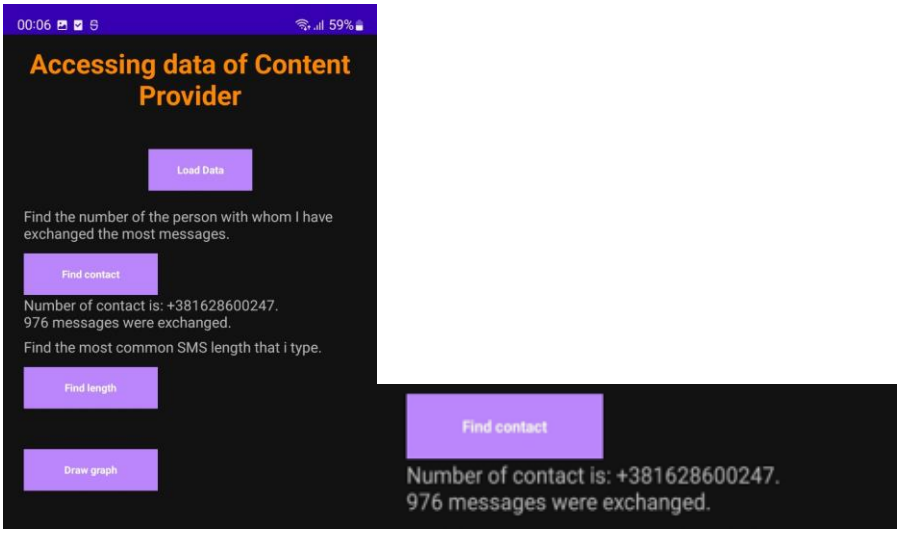
Definisani su i searchQuery kao i sortOrder. SearchQuery zahteva samo one poruke koje su primljene i poslate u 2022. pošto će se analizirati podaci u okviru jedne godine. SortOrder vratiće rezultate poređane po datumu, počev od najstarijih poruka.

Na slici 15 prikazan je izgled aplikacije nakon učitavanja poruka, kao i odgovarajuća poruka koja se pojavljuje korisniku.




*Slika 15: Izgled aplikacije nakon učitavanja poruka*

Dugme ***Find Contact***, poziva funkciju *findContact* i vraća broj telefona osobe sa kojom je razmenjeno najviše poruka, kao i njihov broj.




*Slika 16: Prikaz vraćenih rezultata nakon pritiska na dugme Find Contact*

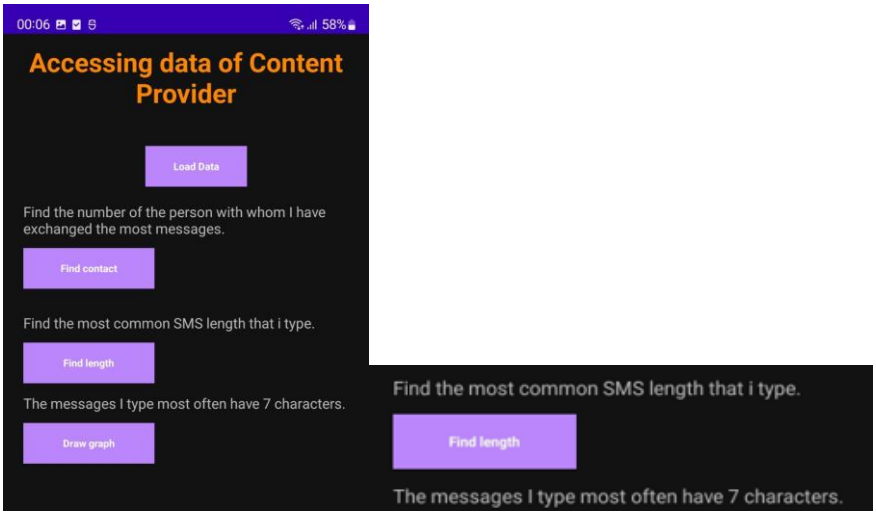
```
public String findContact(Map<String, List<Sms>> allSms){
    HashMap<String,Integer> contactDictionary=new HashMap<String,Integer>();
    List<Sms> inbox = allSms.get("inbox");
    List<Sms> sent = allSms.get("sent");

    for(int i=0; i < inbox.size(); i++){
        String contact = inbox.get(i).getAddress();
        boolean isKeyPresent = contactDictionary.containsKey(contact);
        int value =isKeyPresent ? (contactDictionary.get(contact) + 1) : 1;
        contactDictionary.put(contact,value);
    }
    for(int i=0; i < sent.size(); i++){
        String contact = sent.get(i).getAddress();
        boolean isKeyPresent = contactDictionary.containsKey(contact);
        int value =isKeyPresent ? (contactDictionary.get(contact) + 1) : 1;
        contactDictionary.put(contact,value);
    }
    int maxValueInMap=(Collections.max(contactDictionary.values()));  // This will return max value in the HashMap
    numOfExchangedessages=maxValueInMap;
    String key = Collections.max(contactDictionary.entrySet(), Map.Entry.comparingByValue()).getKey();

    return key;
}
```

*Slika 17: Implementacija funkcije findContact*

Treće dugme ***Find Length***, poziva *findMostCommonSmsLength*, koja vraća dužinu poruke koju najčešće kucam.




*Slika 18: Prikaz vraćenih rezultata nakon pritiska na dugme Find Length*

I poslednje dugme, ***Draw graph***, poziva funkciju za iscrtavanje grafika. Na njemu je prikazano, za svaki mesec, koliko sam poruka poslala.

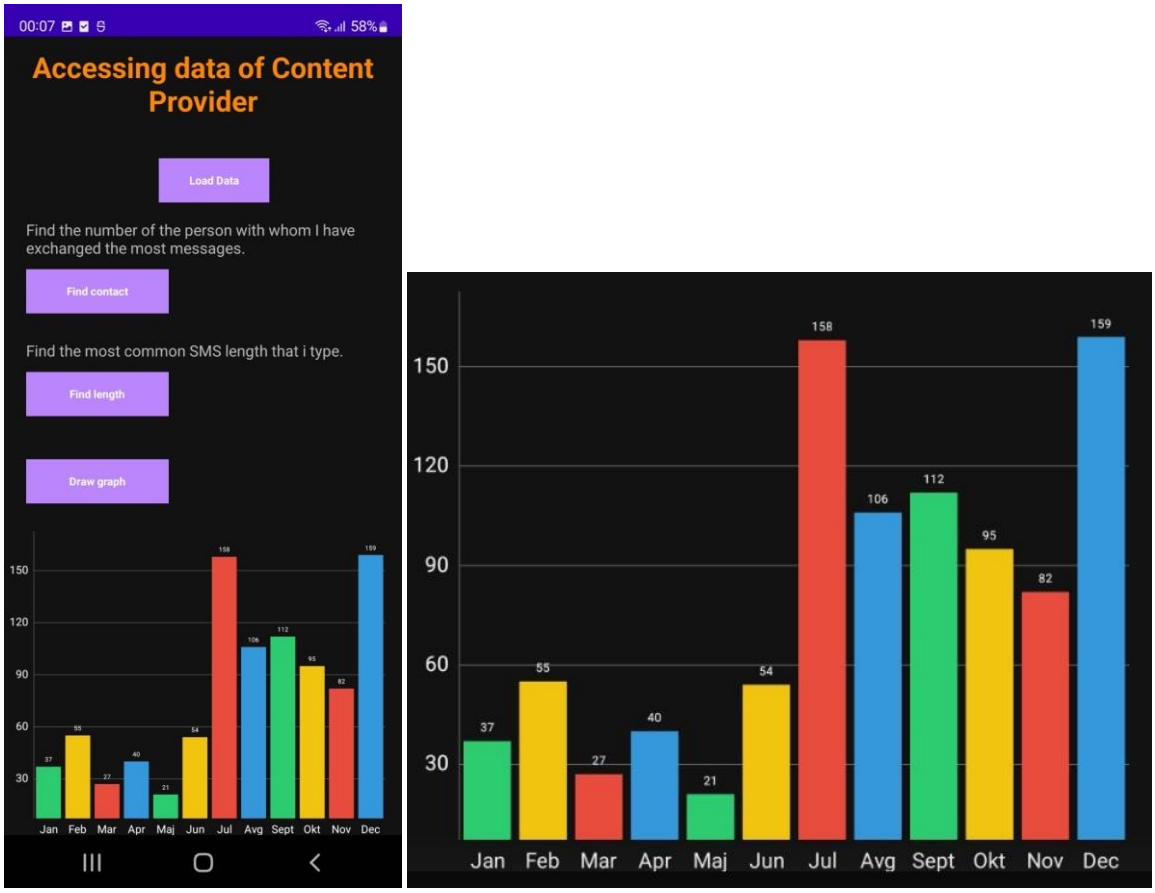
```
private ArrayList getEntriesForNumSms(List<Sms> sms) {
    // creating a new array list
    ArrayList barEntriesArrayList = new ArrayList<>();

    HashMap<Integer,Integer> months=new HashMap<Integer,Integer>();
    for(int i = 0; i < 12; i++) {
        months.put(i, 0);
    }
    for(int i = 0; i < sms.size(); i++){
        long time =Long.parseLong(sms.get(i).getTime());

            Calendar calendar = Calendar.getInstance();
            calendar.setTimeInMillis((long)time);
            SimpleDateFormat df1 = new SimpleDateFormat( pattern: "MM");
            String monthString =df1.format(calendar.getTime());
            int month = Integer.parseInt(monthString);
            month = month-1;
            Integer value = months.get(month) + 1;
            months.put(month , value);
    }
    for(Map.Entry<Integer, Integer> map  :  months.entrySet() ) {
        barEntriesArrayList.add(new BarEntry(new Float(map.getKey()), new Float(map.getValue())));
    }
    return barEntriesArrayList;
}
```

*Slika 19: Funkcija za definisanje podataka koji će biti korišćeni za iscrtavanje grafika*

Za iscrtavanje je korišćena MPAndroidChart biblioteka.




*Slika 20: Prikaz broja poslatih poruka za svaki mesec*

# Zaključak

Forenzika mobilnih uređaja vrlo je važna grana digitalne forenzike. Prateći savremene trendove u svetu, razvoj mobilnih uređaja nameće se kao trend i predstavlja jednu od najbrže rastućih industrija. Sve širi razvoj brojnih funkcionalnosti i mogućnosti koje su dostupne na pametnim mobilnim uređajima dovodi po jačanja potrebe uređaja kako za poslovne, tako i za privatne svrhe. Sve veća količina podatka nalazi se u mobilnim uređajima i ukoliko dođe do bilo kojeg oblika oštećenja uređaja, gubitka podataka i/ili uređaj bio uključen u zlonamerne radnje, forenzika mobilnih uređaja pruža niz alata kojima je moguće napraviti povrat gotovo svih podataka.

Ukoliko poznajemo stukturu Android datoteka i znamo na koji način i gde se na uređaju čuvaju podaci, vrlo lako im možemo pristupiti. Sve što je potrebno da se obezbedi je alat, ADB, koji će da omogući komunikaciju i da se odabere način pristupa telefonu. Ekstrakcija se može vršiti rootovanjem telefona, pri čemu korisnici Android uređaja mogu da ostvare privilegovanu kontrolu (poznatu kao root pristup) nad različitim podsistemima uređaja, obično pametnim telefonima. Drugi način za razmenu podataka je korišćenjem Content Providera, koji je primenjen u ovom radu.

Komponenta content provider-a isporučuje podatke iz jedne aplikacije u druge na zahtev. Takvi zahtevi se obrađuju pomoću metoda klase ContentResolver. U okviru zahteva potrebno je uneti koji provider se poziva, koja tabela se koristi, zatim, moguće je specificirati i neke dodatne uslove. Neki od njih su recimo koje kolone tabele su potrebne, koji redovi, u kom redosledu želimo da rezultat bude vraćen...

Nad dobijenim podacima mogu se vršiti razne manipulacije. Mogu se izdvojiti pozivi, poruke primeljene od odredjene osobe, videti kad je primljena koja poruka, kao i njen sadržaj i izvući mnoge druge informacije.

Upravo zbog ovih mogućnosti, pridaje se tolika važnost digitalnoj forenzici mobilnih uređaja, kao i forenzičarima i ispitivačima koji se njome bave. Zbog toga je dobijanje podataka sa mobilnih telefona toliko značajno u kriminalističkim istragama. Forenzičari se služe svim sredstvima i metodama kako bi došli do podataka.

# Literatura


1. Explore the Android File System Hierarchy In-Depth- https://thesecmaster.com/explore-the-android-file-system-hierarchy-in-depth/, [poslednji pristup: 10.01.2023.]
2. Data and file storage overview - https://developer.android.com/training/data-storage, [poslednji pristup: 15.01.2023.]
3. What is the Difference Between External Storage and Internal Storage? - https://www.cashify.in/what-is-the-difference-between-external-storage-and-internal-storage-cashify-explains, [poslednji pristup: 20.01.2023.]
4. Request runtime permissions - https://developer.android.com/training/permissions/requesting, [poslednji pristup: 20.01.2023.]
5. Permissions on Android - https://developer.android.com/guide/topics/permissions/overview, [poslednji pristup: 20.01.2023.]
6. Using Android Debug Bridge - https://emteria.com/learn/android-debug-bridge, [poslednji pristup: 25.01.2023.]
7. Android Debug Bridge (adb) - https://developer.android.com/studio/command-line/adb, [poslednji pristup: 25.01.2023.]
8. Content provider basics - https://developer.android.com/guide/topics/providers/content-provider-basics, [poslednji pristup: 28.01.2023.]
9. Android - Content Providers - https://www.tutorialspoint.com/android/android_content_providers.htm, [poslednji pristup: 28.01.2023.]