Univerzitet u Sarajevu

Elektrotehnički fakultet
Predmet: Razvoj mobilnih aplikacija
Akademska godina: 2019/2020.

Korištenje kamere mobilnog Android uređaja

Kandidatkinja:

Amina Babić

Mentor:

dr.sc.Vensada Okanović

Sarajevo, 29.8.2020.

SADRŽAJ

UVOD

U sklopu ovog rada će biti obrađeno na koji način omogućiti da aplikacija koristi kameru mobilnog Android uređaja. Razlog za izbor ove teme je velika popularnost korištenja kamere u sklopu aplikacije. Danas, veliki broj aplikacija ili u potpunosti ovisan od kamere, ili za svoj rad traži dopuštenje za korištenje kamere, ili može funkcionisati i bez kamere, ali je upotreba iste “logična” s obzirom na prirodu aplikacije.

Npr. na facebook profil-u imamo opciju “Take a new profile photo” koja podrazumijeva korištenje kamere. Možemo zaključiti da bi aplikacija radila i bez te opcije, ali je očekivano od strane korisnika da ima tu opciju.

Slikanje, snimanje videa, prepoznavanje lica, u sklopu svake od navedenih akcija potrebno je omogućiti pristup i rad s kamerom Android uređaja.

Sve je veća zastupljenost korištenja kamere kroz aplikacije, što čini ovu temu jako atraktivnom.

Kroz cjeline ovog rada će biti pokrivena dva načina korištenja kamere, definisanje permisije, pristup spremljenim podacima na telefonu, te i usputno objašnjenje onoga što se koristi. Način na koji će biti prezentovana tema je način na koji se obično razrađuju problemi.

Korak po korak, uz teorijska objašnjenja, upoznavanje s logikom i načinom rada pojedinih komponenti, te nakon navedenog će biti sumiran i predstavljen krajnji kod.

1. NAČINI KORIŠTENJA KAMERE

U nastavku će biti obrađena dva načina korištenja kamere.

Prvo je potrebno, kao i prije rješavanja svakog problema, razmotriti sam problem i definisati šta se zapravo želi, te čime se koristiti do cilja. Potrebno je dosta više upućenosti i poznavanja logike i metodologije da bi jedan “klik” spremio željenu sliku korisnika.

1.1. Permisije i karakteristike

- Ukoliko je kamera neophodna za rad aplikacije, potrebno je i definisati da je aplikacija za uređaje koji imaju kameru.

  To se relizuje tako što se unutar manifest fajl-a doda tag <uses-feature> na sljedeći način :

  <uses-feature android:name="android.hardware.camera" />

- Ukoliko aplikacija koristi kameru, ali nije i neophodna za rad kamere onda postavljamo android : required = “false”, odnosno android : required = “true”, ako jeste neophodna.

  <uses-feature android:name="android.hardware.camera"
  android:required="true" />

- Ukoliko je potrebno da aplikacija koristi kameru, onda ona treba tražiti dopuštenje za korištenje kamere. Ovo se realizuje na sljedeći način :

  <uses-permission android:name="android.permission.CAMERA" />

1.2. Memorisanje

Kada treba snimiti rezultat korištenja kamere (slika, video), taj rezultat može biti vidljiv / dostupan samo našoj aplikaciji, a može se omogućiti da i druge aplikacije imaju pristup.

1.3 Korištenje kamere preko Intent – a

Jedinica za komunikaciju između dvije aktivnosti je Intent klasa (Mednieks, Dornin, Meike, Nakamura,2012).

Da bi jedna aktivnost pozvala drugu i poslala joj informaciju o tome šta korisnik zapravo želi da uradi, koristi se Intent. To je opis operacije koju jedna aktivnost zahtjeva od druge.

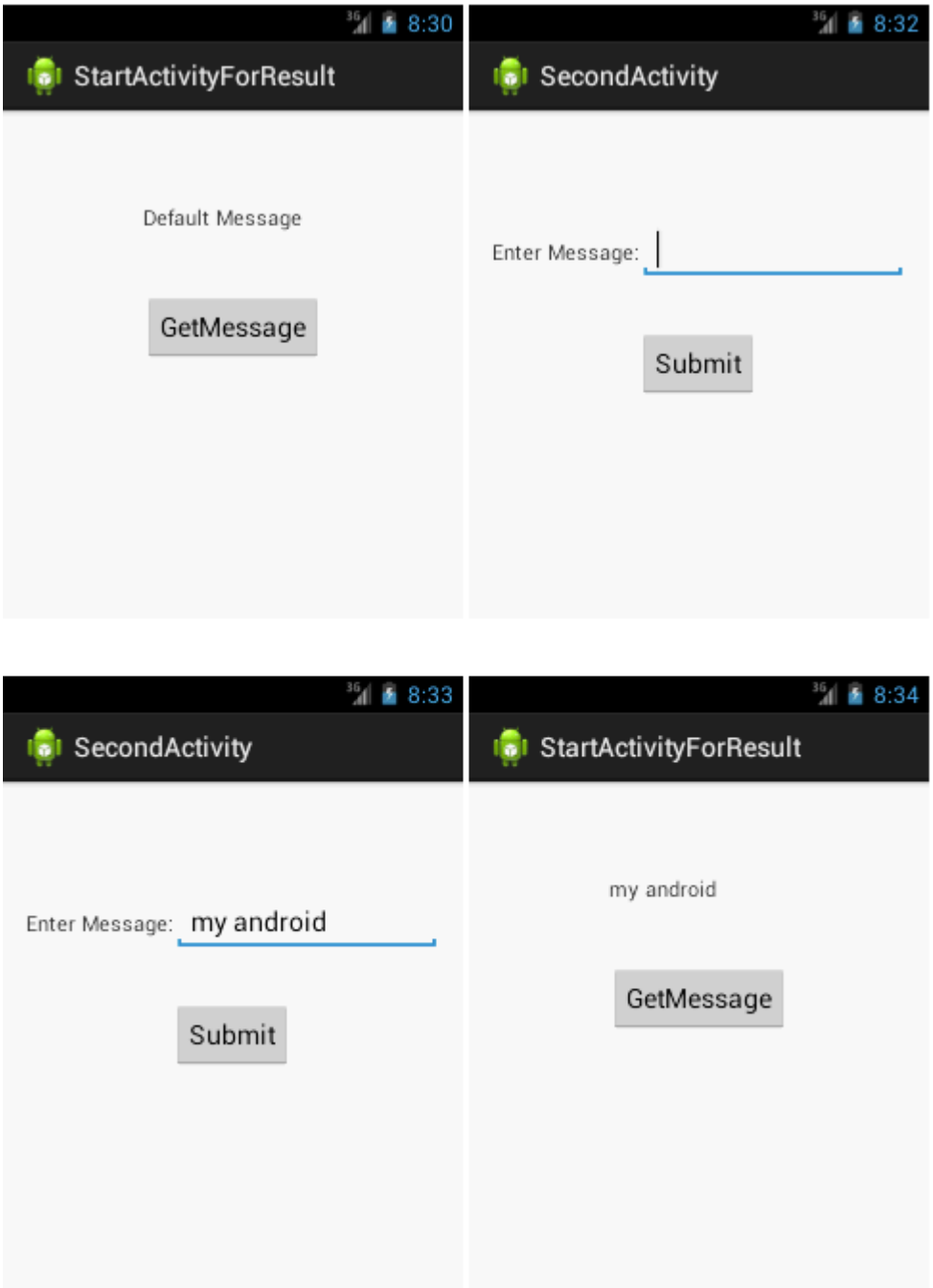
Upravo je korištenje Intent-a jedan od načina korištenja kamere, tj. korištenje Intent-a kako bi se pokrenula već postojeća aplikacija koja koristi kameru.Ova metoda ne zahtjeva puno koda od strane programera. Koraci ove metode su :

- kreirati Intent za MediaStore.ACTION_IMAGE_CAPTURE
  MediaStore je klasa koja je zapravo ugovor između provajdera i aplikacije.
  Media provajder nam zapravo pruža kolekciju media tipova kao što su Audio, Video, Slike (Images).
  ACTION_IMAGE_CAPTURE je standardnai intent koji se šalje kako bi aplikacija vratila odgovarajuću sliku.
- pozvati startActivityForResult()
  Pomoću ove metode se šalju informacije iz jedne aktivnosti u drugu.
  Postoje dvije varijante ove metode :
  public void startActivityForResult (Intent intent, int requestCode)

  public void startActivityForResult (Intent intent, int requestCode, Bundle options)

- pozvati onActivityResult()
  Ovo je metoda kojom se dobiju informacije iz druge aktivnosti i da bi se znalo kada je korisnik završio s kamerom.

  Kako bi se bolje prikazao način rada ovih metoda, u nastavku će biti prikazane slike.



Slika 1. Komunikacija između dvije aktivnosti [1]

1.3.1 Kreiranje Intenta i pozivanje metode startOnActivity(Intent,int)

Funkcija koja poziva intent za snimanje fotografije bi bila:

private void dispatchTakePictureIntent(int actionCode) {

1 Slika preuzeta sa stranice https://www.javatpoint.com/android-startactivityforresult-example

Intent takePictureIntent = new Intent (MediaStore.ACTION_IMAGE_CAPTURE);

startActivityForResult(takePictureIntent, actionCode);

}

Ovdje actionCode možemo definisati kao npr.

public static int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 1034;

Drugi parametar ove metode zapravo identificira poziv.

Kako se ovom metodom očekuje od neke druge aktivnosti da snimi fotografiju, ako ne postoji kompatibilna aktivnost koja je spremna da uhvati intent, onda će aplikacija pasti. (https://developer.android.com/)

Rješenje ovog problema je funkcija koja provjerava da li postoji aktivnost za određeni intent.

public static boolean isIntentAvailable(Context context, String action) {

final PackageManager pMan = context.getPackageManager();

final Intent intent = new Intent(action);

List<ResolveInfo> list =

pMan.queryIntentActivities(intent,PackageManager.MATCH_DEFAULT_ONLY);

return list.size() > 0;

}

U nastavku će biti objašnjen PacketManager i metoda queryIntentActivities() kako bi bilo jasnije šta prethodna funkcija radi.

public abstract List<ResolveInfo> queryIntentActivities (Intent intent, int flags);

Ova metoda vraća sve aktivnosti koje se mogu izvršiti za dati intent. (https://developer.android.com/)

Parametri su :

- intent što predstavlja određeni intent u ovom slučaju takePictureIntent
- flags za modifikaciju rezultata. Pored ostalih flagova, u ovom slučaju je iskorišten MATCH_DEFAULT_ONLY koji ograničava rezultat na samo one aktivnosti koje podržavaju Intent.
- “Rezultat je lista ResolveInfo objekata koji sadrže odgovarajuće aktivnosti, poredane od najbolje do najgore. Drugim riječima, prvi element liste je jednak onom koji bi bio vraćen iz metode resolveActivity(Intent, int), koja inače vraća aktivnost koja podržava određeni intent. Ako nema odgovarajućih aktivnosti, vraća se prazna lista.”

(https://developer.android.com/)

Pored ove metode, koristi se i PackageManager.

PackageManager je klasa za preuzimanje različitih informacija koje su povezane s paketima aplikacije, a koji su instalirani na uređaju.

1.3.2 Dohvaćanje rezultata

Za dohvaćanje rezultata poziva se metoda getExtra() nad intentom. Ova metoda je definisana u Intent klasi. Međutim, korištenje Bundle -a može olakšati posao,

Bundle koristimo za mapiranje String ključ – Parcelable[2]vrijednost.

U biti, u Bundle pohranjujemo kljul-vrijednosti parove i takav objekat šaljemo kroz intent.

Rezultat ćemo dobiti metodom :

void onActivityResult(int requestCode, int resultCode, Intent data) koja se poziva kada se zatvori aktivnost koju smo pozvali / pokrenuli.

Aplikacija Android Camera kodira fotografiju i dostavlja je metodi *onActivityResult(int requestCode, int resultCode, Intent data)* kao Bitmap[3] -u koju ćemo dohvatiti koristeći upravo prethodno spomenutu metodu getExtra(), s tim da je Bitmapa pod ključem “data”.

Sada se može definisati određena metodu koja će se pozivati unutar metode onActivityResult(int, int, Intent) kako bi se određena slika prikazala na ImageView.

```
private void handleCameraPhoto(Intent intent) {
    Bundle bundle = intent.getExtras();
    bitmapImg = (Bitmap) bundle.get("data");
    imageView.setImageBitmap(bitmapImg);
}
```

Nakon navedenih objašnjenja krajnji kod za ovaj metod korištenja kamere bi bio [4]:

2 Parcel je kontejner za poruke, reference.
Parcelable je interfejs za klase čije instance mogu biti vraćene/obnpvljenje iz Parcel.

3 Bitmap je reprezentacija bitmap slike.

4 Kod se može pronaći i pokrenuti na sljedećem linku : https://github.com/ababic2/Kori-tenje-kamere-mobilnog-Android-ure-aja.git

```
package com.example.myapplication;
import ...
import java.util.List;

public class MainActivity extends AppCompatActivity {
    private static final int ACTION_TAKE_PHOTO = 2;
    private ImageView imageView;
    private Bitmap bitmap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        imageView = (ImageView) findViewById(R.id.imageView1);
        bitmap = null;
        Button picSBtn = (Button) findViewById(R.id.btnIntendS);
        setBtnListenerOrDisable(
                picSBtn,
                mTakePicSOnClickListener,
                MediaStore.ACTION_IMAGE_CAPTURE
        );
    }

    private void setBtnListenerOrDisable(Button btn, Button.OnClickListener onClickListener,
            String intentName) {
        if (isIntentAvailable(this, intentName)) {
            btn.setOnClickListener(onClickListener);
        } else {
            btn.setText(
                    getText(R.string.cannot).toString() + " " + btn.getText());
            btn.setClickable(false);
```

```
    }
  }

public static boolean isIntentAvailable(Context context, String action) {
    final PackageManager packageManager = context.getPackageManager();
    final Intent intent = new Intent(action);
    List<ResolveInfo> list =
            packageManager.queryIntentActivities(intent,
                    PackageManager.MATCH_DEFAULT_ONLY);
    return list.size() > 0;
  }

  Button.OnClickListener mTakePicSOnClickListener =
        new Button.OnClickListener() {
          @Override
          public void onClick(View v) {
            dispatchTakePictureIntent(ACTION_TAKE_PHOTO);
          }
        };

  private void dispatchTakePictureIntent(int actionCode) {
    Intent takeImageIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(takeImageIntent, ACTION_TAKE_PHOTO);
  }
  @Override
  protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
      case ACTION_TAKE_PHOTO: {
        if (resultCode == RESULT_OK) {
          handleSmallCameraPhoto(data);
```

```
            }
            break;
        }
    }
}
private void handleSmallCameraPhoto(Intent intent) {
    Bundle extras = intent.getExtras();
    bitmap = (Bitmap) extras.get("data");
    imageView.setImageBitmap(bitmap);
    imageView.setVisibility(View.VISIBLE);
}
}
```

U metodi *onActivityResult()* iskorišten je switch. Razlog za to će biti logičniji u nastavku kada se bude razmatralo korištenje kamere za snimanje videa.

Ono što je još karakteristično je spremanje slike / videa u galeriju. Pošto ovo nije konretno vezano za rad s kamerom, već za smještanje već gotovog rezultata nakon korištenja kamere, u nastavku će biti predstavljen kod koji bi se trebao uklopiti u prethodni kako bi se slika / video smjestili u galeriju, te na taj način bili dostupni i ostalim aplikacijama. Poziva me sistemski media skener.[5]

```
private void addPicToGallery() {
        Intent mediaScanIntent = new
        Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);
        File f = new File(currentPhotoPath);
        Uri contentUri = Uri.fromFile(f);
        mediaScanIntent.setData(contentUri);
        this.sendBroadcast(mediaScanIntent);
}
```

5 Za detaljniji opis posjetiti stranicu: https://developer.android.com/training/camera/photobasics#TaskCaptureIntent

1.3.3 Snimanje videa

Kada je poznato korištenje kamere za slike, sličnom logikom i načinom se koristi kamera i za snimanje vide-a. Naravno, koristi se Intent.

Funkcija koja poziva intent za snimanje videa, na osnovu prethodnih objašnjenja:

```
private void dispatchTakeVideoIntent() {
   Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
   startActivityForResult(takeVideoIntent, ACTION_TAKE_VIDEO);
}
```

Kako aplikacija ne bi pala, provjerava se da li postoji dostupna aktivnost i koristi se ista metoda kao u prethodnom rješenju za slike.

Modificira se metoda onActivityResult() tako da se sada može dohvatiti i video kao rezultat.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        switch (requestCode) {
                case ACTION_TAKE_PHOTO: {
                if (resultCode == RESULT_OK) {
                        handleCameraPhoto(data);
                }
                break;
        }
        case ACTION_TAKE_VIDEO: {
                if (resultCode == RESULT_OK) {
                        handleCameraVideo(data);
                }
                break;
        }
}
```

Sada se može definisati određena metoda koja će se pozivati unutar metode onActivityResult(int, int, Intent) kako bi se određeni video prikazao.

```
private void handleCameraVideo(Intent intent) {
        videoUri = intent.getData();
        videoView.setVideoURI(videoUri);
}
```

Android kamera aplikacija vraća video u intentu i dostavlja to metodi onActivityResult() kao Uri koji pokazuje na video lokaciju u memoriji.

Nakon navedenih modifikacija, kod koji podržava rad kamere u smislu da se mogu snimati i videi bi bio:

```
public class MainActivity extends AppCompatActivity {
   private static final int ACTION_TAKE_PHOTO = 1;
   private static final int ACTION_TAKE_VIDEO = 2;
   private ImageView imageView;
   private Bitmap bitmap;
   private VideoView videoView;
   private Uri videoURI;
   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      imageView = (ImageView) findViewById(R.id.imageView1);
      bitmap = null;
      Button picSBtn = (Button) findViewById(R.id.btnIntendS);
      setBtnListenerOrDisable(
            picSBtn,
            takePhotoOnClickListener,
            MediaStore.ACTION_IMAGE_CAPTURE
      );
      Button vidBtn = (Button) findViewById(R.id.btnIntendV);
      setBtnListenerOrDisable(
            vidBtn,
```

```
            takeVideoOnClickListener,
            MediaStore.ACTION_VIDEO_CAPTURE
        );
    }

    private void setBtnListenerOrDisable(Button btn, Button.OnClickListener onClickListener,
            String intentName) {
        if (isIntentAvailable(this, intentName)) {
            btn.setOnClickListener(onClickListener);
        } else {
            btn.setText(
                    getText(R.string.cannot).toString() + " " + btn.getText());
            btn.setClickable(false);
        }
    }
    public static boolean isIntentAvailable(Context context, String action) {
        final PackageManager packageManager = context.getPackageManager();
        final Intent intent = new Intent(action);
        List<ResolveInfo> list = packageManager.queryIntentActivities(intent,
PackageManager.MATCH_DEFAULT_ONLY);
        return list.size() > 0;
    }
    Button.OnClickListener takePhotoOnClickListener =
            new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                    dispatchTakePictureIntent(ACTION_TAKE_PHOTO);
                }
            };
    Button.OnClickListener takeVideoOnClickListener =
            new Button.OnClickListener() {
```

```
        @Override
        public void onClick(View v) {
            dispatchTakeVideoIntent();
        }
    };

private void dispatchTakeVideoIntent() {
    Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
    startActivityForResult(takeVideoIntent, ACTION_TAKE_VIDEO);
}
private void dispatchTakePictureIntent(int actionCode) {
    Intent takeImageIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    startActivityForResult(takeImageIntent, ACTION_TAKE_PHOTO);
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case ACTION_TAKE_PHOTO: {
            if (resultCode == RESULT_OK) {
                handleSmallCameraPhoto(data);
            }
            break;
        }
        case ACTION_TAKE_VIDEO: {
            if (resultCode == RESULT_OK) {
                handleCameraVideo(data);
            }
            break;
        }
    }
```

```
    }
    private void handleSmallCameraPhoto(Intent intent) {
        Bundle extras = intent.getExtras();
        bitmap = (Bitmap) extras.get("data");
        imageView.setImageBitmap(bitmap);
        imageView.setVisibility(View.VISIBLE);
    }
    private void handleCameraVideo(Intent intent) {
        videoURI = intent.getData();
        videoView.setVideoURI(videoURI);
        bitmap = null;
        videoView.setVisibility(View.VISIBLE);
        imageView.setVisibility(View.INVISIBLE);
    }
}
```

1.3 Kontrola kamere i kreiranje specijalne aplikacije za kameru

Ovaj način je nešto zahjtevniji od korištenja postojeće kamera aplikacije.

Prvo se kreira instanca objekta Camera i kamera bi se trebala otvarati na posebnoj niti / thread-u. Kako ne bi došlo do izuzetka, ukoliko je kamera u upotrebi, otvaranje kamere postavlja se u try-catch blok i ona se oslobađa.

```
private boolean safeCameraOpen(int id) {
        boolean opened = false;
        try {
                releaseCameraAndPreview();
                camera = Camera.open(id);
                opened = (camera != null);
        } catch (Exception e) {}
        return opened;
```

```
}
private void releaseCamera() {
        if (camera != null) {
        camera.release();
        camera = null;
        }
}
```

Kako bi se prikazalo korisniku na šta senzor kamere pokazuje, koristi se SurfaceView i kako bi se proslijedila slika od hardvera(kamere) do aplikacije, koristi se SurfaceHolder.

```
class Preview extends ViewGroup implements SurfaceHolder.Callback {[6]
        SurfaceView surfaceView;
        SurfaceHolder holder;
        Preview(Context context) {
        super(context);
        surfaceView = new SurfaceView(context);
         addView(surfaceView);
        holder = surfaceView.getHolder();
        holder.addCallback(this);
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
         }
}
```

Ukoliko se koristi za ovaj metod, sljedeći koraci bi bili da se postavi i pokrene pregled / preview. Ono što se može još podešavati jeste orijentacija metodom *setCameraDisplayOrientation()* kako bi se promijenio prikazani pregled, ali to neće uticati na snimanje fotografije.

Kako bi se pokrenulo slikanje, koristi se metoda takePicture(), tj. Camera.takePicture() i nakon što se izvrši slikanje, potrebno je restartovati pregled kako bi korisnik mogao bez problema snimiti sljedeću fotografiju.

Nakon što aplikacija završi sa korištenjem kamere, potrebno je osloboditi Camera objekat kako se ostale aplikacije ne bi krahirale.

6 Kod preuzet sa linka: https://developer.android.com/training/camera/cameradirect#TaskOpenCamera

```
@Override[7]
public void surfaceDestroyed(SurfaceHolder holder) {
        if (mCamera != null) {
                //Poziva se stopPreview() kako bi se zaustavilo ažuriranje surface-a
                mCamera.stopPreview();
        }
}
private void stopPreviewAndFreeCamera() {
        if (mCamera != null) {
                mCamera.stopPreview();
                mCamera.release();
                mCamera = null;
        }
}
```

Baš kao i prilikom korištenja Intent-a. Nakon što shvatimo kako snimiti sliku, analogno će biti i snimanje videa. To isto vrijedi i za ovaj način korištenja kamere. Više uputa i informacija se može pronaći u dokumentaciji.[8]

1.4 Detekcija lica

Android 4.0(API Level 14) framework pruža API za prepoznavanje lica.

Koraci za korištenje prepoznavanja lica u aplikaciji su:

- Provjera da li uređaj to podržava. Poziva se metoda *getMaxNumDetectedFaces().*
- Kreiranje listener klase koja će implementirati odgovarajući interfejs
- Postavljanje listenera na Camera objekat

  camera.setFaceDetectionListener(new MyFaceDetectionListener());
- Započinjanje detekcije:

7 Kod preuzet sa sljedećeg linka: https://developer.android.com/training/camera/cameradirect#TaskOpenCamera

8 https://developer.android.com/guide/topics/media/camera#capture-video

```
public void startFaceDetection(){
    Camera.Parameters params = mCamera.getParameters();
    // start face detection only *after* preview has started
    if (params.getMaxNumDetectedFaces() > 0){
    // camera supports face detection, so can start it:
    mCamera.startFaceDetection();
   }
}
```

Literatura

- Mednieks, Z., Dornin, L., Meike, G., Nakamura, M., Oram, A., (2012), *Programming Android* , Sebastopol, O'Reilly Media, Inc.
- internet stranica: https://developer.android.com/training/camera
- internet stranica: https://guides.codepath.com/android/Accessing-the-Camera-and-Stored-Media#create-a-file-reference
- Felker, D. Android Application Development for Dummies

  Dostupno na: https://endangcahyapermana.files.wordpress.com/2016/05/android-application-development-for-for-dummies.pdf

  Očitano: 30.08.2020
- Hardy, B., Philips, B., (2013), *Android Programming: The Big Nerd Ranch Guide*,

  Indianapolis, IN 46240 USA, Pearson Technology Group